De briques et de blocs. La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Samuel Goyet

CELSA
UNIVERSITÉ PARIS-SORBONNE

ÉCOLE DOCTORALE nº 5 « Concepts et Langages »

Laboratoire de recherche GRIPIC (EA 1498)

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-SORBONNE

Discipline : Sciences de l'Information et de la Communication

Présentée et soutenue par :

Samuel GOYET

le 22 novembre 2017

De briques et de blocs.

La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Tome I

Sous la direction de :

M. Emmanuël SOUCHIER – Professeur des universités, CELSA Université Paris-Sorbonne.

Membres du jury :

M. Bruno BACHIMONT – Professeur des universités, Université de Technologie de Compiègne.

Mme Marie-Paule CANI – Professeure des université, École Polytechnique.

M. Jean DAVALLON – Professeur émérite, Université d'Avignon et des Pays de Vaucluse.

M. Milad DOUEIHI – Professeur des universités, Université Paris-Sorbonne.

Mme Joëlle LE MAREC – Professeure des universités , CELSA Université Paris-Sorbonne.

Mme Marie-Anne PAVEAU – Professeure des universités, Université Paris 13.

M. Emmanuël SOUCHIER – Professeur des universités, CELSA Université Paris-Sorbonne.

De briques et de blocs. La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Résumé

Boutons « J'aime », tweets ancrés… Toutes ces formes sont générées par des interfaces de programmation ou API, outils d'écriture informatique qui ont integré la chaîne de production des textes de réseau contemporains. Cette thèse interroge la fonction éditoriale des API, soit leur rôle dans la production, la standardisation et la circulation des « petites formes » des textes de réseau. Avec comme corpus les API de Facebook et de Twitter ainsi que les petites formes qu'elles permettent de produire, notre analyse techno-sémiotique s'articule en cinq chapitres. Le premier est une généalogie des API du point de vue de l'écriture combinatoire. Nous montrons que cette conception de l'écriture est un trait saillant de la programmation et de l'informatique. Le second chapitre interroge les imaginaires de l'écriture informatique, entre chiffre, combinatoire et méthode scientifique universelle. Le troisième chapitre est une analyse des conséquences sémiotiques de cet universalisme combinatoire, où nous montrons que les API proposent une conception du texte comme ensemble abstrait de blocs combinables. Abstraction du texte qui sert une « économie des passages », objet de notre quatrième chapitre, dans laquelle les API sont des lieux d'industrialisation d'une « pratique lettrée » : elles établissent des critères de lisibilité et de reproductibilité du texte. Parmi ces critères, nous notons une invisibilisation du rôle pourtant fondamental du calcul informatique. Nous proposons donc, dans un cinquième chapitre, des pistes pour développer une sémiotique qui prenne en compte le calcul comme mode d'expression propre aux médias numériques.

Mots-clés : Calcul ; numérique ; informatique ; combinatoire ; éditorialisation ; interfaces de programmation ; API ; écrits d'écran ; sémiotique ; écriture ; industrialisation de l'écriture ; techno-sémiotique ; code informatique.

Of Bits and Blocks. The publishing function of web Application Programming Interfaces (APIs): from a combinatorial science to an industry of text-processing.

Summary

« Like » buttons, embedded tweets… All of these visual forms are produced by Application Programming Interfaces (APIs). APIs are digital writing tools which have become part of the publishing process of contemporary web pages. This thesis aims at understanding the « publishing function » of APIs : their role in the production, standardization and circulation of the « little forms » of online texts. Focused on Facebook's and Twitter's APIs, our work is divided into five chapters. The first one is a genealogy of the APIs, starting from their combinatorial aspect, a conception of writing which trace back to early programming and the invention of computer science. The second chapter is an inquiry about the imaginaries of calculus as a kind of writing, torn between the imaginary of numbers, of combinatorics and the search for a universal scientific method. The third chapter is a study of the semiotic consequences of this combinatorial universalism. We show how APIs are based on an idea of text as an abstract, modular object. This abstraction of the text is beneficial to an « economy of passages ». In this economy where circulation produce value, APIs are a place of « literate practices » (chapter four). They establish visual standards for the readability, production and circulation of online texts. Among these standards, there's a systematic invizibilisation of the action of machines, although calculus is a necessary part of the production of digital texts. Therefore, in the fifth chapter, we give some epistemological elements towards non-anthropocentric semiotics, meaning : semiotics which would take into account computational machines as a part of the utterance of digital texts.

Keywords : Calculus ; digital studies ; computing ; combinatorics ; editorialization ; publishing process ; literary practices ; coding ; written screens ; API ; programming ; semiotics ; writing ; industrialization of writing.

Thèse en **Sciences de l'Information et de la Communication,** préparée au sein du **GRIPIC**
(Groupe de Recherche Interdisciplinaire sur les Processus d'Information et de Communication).

CELSA, UNIVERSITÉ PARIS-SORBONNE — 77 rue de Villiers – 92200 Neuilly-sur-Seine.
École doctorale nº 5 « Concepts et langages » — Maison de la Recherche – 28 rue Serpente, 75006 Paris.

REMERCIEMENTS

Ceux qui liront cette thèse savent combien un texte est éminemment polyphonique et combien chaque acteur ayant, un jour ou l'autre, intervenu sur sa fabrication fait partie intégrante du texte. Ils savent également que cette polyphonie est sans cesse aménagée, négociée et qu'elle n'échappe pas à certains jeux de pouvoir. Une voix finit toujours par l'emporter, par s'imposer sur les autres – par nécessité communicationnelle (il faut bien se faire entendre) – ou par coup de force éditorial.

Il en va du texte de recherche comme de n'importe quel autre : valorisant certains acteurs, en négligeant d'autres en les reléguant dans les marges ou les limites… Sa topologie dit quelque chose des rapports de pouvoir et d'auctoralités qui s'y jouent.

Qu'il me soit donc permis, en ces pages liminaires, en ces limites du textes, ces seuils, de ménager un lieu dédié à celles et ceux qui ont fait que ce texte est ce qu'il est matériellement, intellectuellement et institutionnellement. Quoi qu'en dise la couverture de cette thèse, vous êtes vous aussi auteurs de cette thèse.

Mes premières pensées vont à Milad Doueihi et Emmanuël Souchier, qui ont accepté de diriger mon travail. Non seulement avez-vous nourri mes réflexions, tous les deux à votre manière, mais vous avez aussi fait preuve de grandes qualités humaines dans votre accompagnement toujours bienveillant. J'ose croire que vos enseignements dépasserons le seuil de ces pages. Un merci particulier à Milad Doueihi, pour m'avoir permis de financer ma thèse et de rejoindre l'Université Laval, découvrant ainsi de nouveaux horizons de recherche.

Merci à Adeline Goyet pour la conception de la maquette de cette thèse ; merci à Martine Fournier pour avoir accepté d'en être les « petites mains » et plus largement pour m'avoir donné les conditions matérielles adéquates pour terminer cette thèse.

Merci à mon équipe de relecteurs : Laure et ses enveloppes improbables ; Guillaume pour sa célérité ; Thomas pour sa méticulosité ; Frank pour son œil impeccable et sa syntaxe exquise ; Pauline pour sa disponibilité infaillible ; Martine pour m'avoir (aussi) relu ; Marc enfin pour son *timing* impeccable malgré une invitation de dernière minute.

Une thèse n'existe que parce qu'elle a été lue, commentée et discutée par des spécialistes. Merci donc à Marie-Paule Cani, Joëlle Le Marec, Marie-Anne Paveau, Jean Davallon et Bruno Bachimont d'avoir accepté de faire partie de mon jury. Votre travail est aussi précieux qu'indispensable et j'espère que la lecture de cette thèse vous sera agréable.

Merci au GRIPIC, mon *alma mater* où j'ai commencé à goûter aux joies de la sémiotique des écrits d'écran auprès des meilleurs guides : Étienne Candel, Gustavo Gomez-Mejia, Yves Jeanneret, Caroline de Montéty… Emmanuël, ne change rien à ton cours sur les théâtres de mémoire, c'est lui qui m'a donné le goût de la recherche. Merci à l'Université Laval pour m'avoir accueilli pendant dix-huit mois et à Stéphane Roche pour le bureau de fortune.

J'ai une pensée particulière pour toute l'équipe du COSTECH, et notamment le groupe EPIN, qui m'ont fait confiance pendant l'année 2014-2015 en tant qu'ATER : Virginie Julliard, Serge Bouchardon, Jérôme Valluy, Isabelle Cailleau, Clément Mabi… Merci à vous. Cette thèse est en très grande partie le fruit des réflexions entamées lors de nos discussions collectives, et j'ai trouvé dans votre équipe une façon de faire de la recherche à la fois agréable et inspirante. Merci au CREM pour m'avoir recruté en tant qu'ATER pour l'année 2016-2017, particulièrement à Nathalie Pignard-Cheynel et Jean-François Diana pour leur accueil, à Marie Chagnoux pour son travail colossal.

Merci à tous ceux, amis, étudiants ou collègues, qui m'ont permis – parfois sans le savoir – de construire ma pensée. Merci à Liesbeth De Mol pour ses précieuses suggestions de lecture, à Christophe pour m'avoir rappelé l'importance du monde des images ; à tous les participants du séminaire *Chemin des écritures* qui ont eux aussi compris que nous pensons avec des images et par la rêverie. C'est tout le mérite d'Anne et d'Emmanuël de cultiver chez leurs auditeurs

cette précieuse faculté d'imagination. J'adresse également un franc remerciement à tous les Lursiens et Lursiennes qui ont illuminé mes fins d'été, mais qui m'ont aussi fait redécouvrir la profondeur du geste typographique, la beauté de cet attachement parfois maniaque à soigner les jambages des *p* et les yeux des *e*. Merci à cet étudiant de SI05 (était-ce vous Hugo ?) qui a fait planer, au détour d'une copie, l'ombre de Giordano Bruno sur la sémiotique des écrits d'écran. Une petite pensée pour Laure et à son initiation à la calligraphie. Merci à Gustavo, Étienne, Valérie et tout les membres du CRÉE. Vous m'avez aidé à comprendre ce qu'est ce métier et à mieux vivre certaines moments angoissants grâce à vos bons conseils. Thomas, Pauline, Guillaume… Merci d'avoir été là. Votre soutien a largement dépassé les enjeux strictement scientifiques de ce travail

Au cours d'une telle trajectoire, certaines personnes vous frappent parfois par leur bienveillance. Hélène, je ne sais pas si tu as fini ta thèse, mais je pense à toi en ce moment. Maroussia, merci pour ton intelligence – l'une des plus vives que je connaisse – les délicieux thés et ton immense générosité. Je suis désolé que cette thèse t'ait coûté notre collaboration, mais je sais que nous aurons de quoi faire dans le futur.

Je n'oublie pas non plus que cette thèse fut faite entre plusieurs canapés, au fil des déménagements et séjours parisiens. Que soient donc remerciés Laure, Thomas, Pauline, Adeline et François pour leur moelleux matelas, canapés et autres chaufferettes.

Ma dernière pensée est pour Cléo, auprès de qui l'expression « joies de la recherche » prend tout son sens. Sans cette grande joie, sans cette émulation rare et nourricière, cette thèse ne serait pas ce qu'elle est. Je suis extrêmement fier et heureux du travail mené avec toi. Merci pour tout ce que tu as fait. Merci d'avoir cassé mon jouet et d'avoir fait des beaux lundis. *We will make it through, somehow*.

Premier niveau

LES API DANS L'HISTOIRE DU LOGICIEL ET DE L'INFORMATIQUE

I

Dans ce premier chapitre, nous partons de la dimension combinatoire des { API } pour montrer comment elle s'inscrit dans une histoire de la { programmation }, de l'industrie du { logiciel } et de l'informatique. En quoi une { API } peut être dite combinatoire ? D'où vient cette part combinatoire et à quoi sert-elle selon les différents contextes où elle est mobilisée?

API ; codage, code, logiciel, programmation, programme, standardisation, subroutine

Voir glossaire pp.xxx-xxx

L'hypothèse principale de ce chapitre est que les { API } sont des outils d'écriture qui valorisent des principes combinatoires pour des raisons économiques, pour faciliter la { programmation } et pour des raisons techniques.

Chacune de ces étapes correspond à autant d'étapes dans l'histoire de notre objet. Histoire dont nous préciserons dans un premier temps le cadre général (I. Introduction) avant de montrer que la part combinatoire des { API } web contemporaines permet de répondre aux besoins d'une industrie de l'écriture fondée sur des enjeux de plasticité et de { standardisation } (I. 1). Cette combinatoire fut initialement exploitée pour faciliter l'écriture des premiers { programmes } (I. 2). C'est ce que nous montrons en nous plaçant à la naissance de la { programmation } à la fin des années 1940, à travers l'exemple de la { *subroutine* }, soit le { codage } de certaines opérations routinières sous forme de blocs de { code } préécrits et mobilisables dans différents contextes. De ce point de

Deuxième niveau

1 LES API : UNE ÉCRITURE COMBINATOIRE AU SERVICE D'UNE ÉCONOMIE DE L'ÉCRITURE LOGICIELLE

Nous avons jusqu'ici pris comme point de départ que les { API } sont des outils d'écriture combinatoire. Il s'agit maintenant de préciser cette idée, en commençant par la définition de ce qu'est la combinatoire (I. 1. A). Une { API } est un outil d'écriture combinatoire, au sens où elle met à disposition des internautes des modules élémentaires qui servent à écrire une page web ou une { application } (I. 1. B). La combinatoire est alors envisagée comme combinaison de modules standards, plus ou moins adaptables selon les contextes et la { littératie } de l'écrivant, ce que nous montrerons avec l'exemple des { *widgets* } (I. 1. C). Cette plasticité est plus largement une solution à un marché de l'informatique structuré autour de la différence entre le { matériel } (*hardware*) et le { logiciel } (*software*) (I. 1. D) et intègre à partir des années 1970 certains modèles de { programmation } (I. 1. E).

API, application, littératie, logiciel, matériel, programmation, *widget*

Voir glossaire pp.xxx-xxx

« Fil d'Ariane »

Troisième niveau

Quatrième niveau

1 A La combinatoire, une première définition

La question centrale qui nous occupe est de montrer en quoi les interfaces de programmation web sont des outils d'écriture à dimension combinatoire. Il faut pour cela définir ce que nous appelons combinatoire, autour de la notion de finitude (I. 1. A. a), définition qui se concentre sur la mécanique d'un système combinatoire plutôt que sur ses mises en pratique dans un contexte culturel et technique précis (I. 1. A. b). Cette définition large permettra dans la partie suivante de bien comprendre en quoi les { API } sont une certaine mise en application de principes combinatoires.

API

Voir glossaire pp.xxx-xxx

a Combinatoire et finitude

Par combinatoire, nous entendons un système réglé de permutations entre un nombre fini d'éléments. Par exemple, une boîte de Lego® est un système combinatoire. Il y a un certain nombre de briques élémentaires – qui ne peuvent être divisées en plus petites briques – et qui ne peuvent s'assembler qu'à condition qu'elles aient chacune au moins une zone de libre. C'est leur règle de permutation. Le nombre de briques, ainsi que leurs zones exploitables, définissent le nombre de combinaisons possibles. De la même façon, on peut dire du langage qu'il est une structure combinatoire, au moins partiellement. Notre alphabet est la liste des éléments combinables, tandis que la grammaire et l'orthographe donne la liste des permutations autorisées[11].

Pourquoi le langage est-il un système partiellement combinatoire ? Du point de vue discursif tout d'abord. Certains mots (entendre certaines combinaisons) n'ont pas de sens et ne sont donc pas autorisés. Mais cela tient surtout au nombre fini de permutations d'un système combinatoire. Le langage autorise à répéter une lettre *ad indefinitum*. Ce faisant, il y a un nombre de mots tendant vers l'infini, ne serait-ce qu'avec la lettre A – si par exemple elle est répétée x fois. Il est donc essentiel, pour définir un strict système combinatoire, d'avoir un ensemble de règles qui organise les permutations et permet d'en déterminer le nombre total. Si par exemple on postule que dans un système combinatoire à quatre éléments de base (A, B, C, D), il ne peut y avoir de redoublement d'éléments (la combinaison

11 Ce qui laisse de côté un nombre considérable de combinaisons.

AAAA, BB ou CCC est impossible) et que ce système est commutatif[12], alors cela restreint fortement le nombre de combinaisons. La règle que n'applique pas le langage et qui fait que le nombre de combinaisons tend vers l'infini, c'est que le nombre maximal d'éléments utilisables dans une combinaison n'est pas défini. Par exemple, si nous voulons trouver toutes les combinaisons possibles de A, B, C et D, il faut déterminer en amont si on autorise des combinaisons de plus de quatre lettres et jusqu'à quelle limite – six, sept, huit lettres, etc. Ce n'est qu'une fois défini ce nombre fini qu'on peut calculer le nombre total de combinaisons[13].

b Une conception mécaniste de la combinatoire

Pourquoi est-ce important d'avoir un nombre fini de combinaisons, un nombre fini d'éléments de base et un nombre fini déterminant la plus longue combinaison possible ? C'est à ces conditions qu'on peut savoir – par le calcul – qu'il existe un nombre donné de combinaisons possibles, même si ce nombre est colossal et difficilement appréhendable. Il n'y a pas besoin d'effectivement noter ou calculer toutes les combinaisons possibles. On peut déterminer en amont le nombre de combinaisons possibles, au moins de façon virtuelle, c'est-à-dire sans actualiser ces possibles par le calcul. Toute la force de la combinatoire réside dans ce rapport entre une grande économie de moyens – quelques règles et éléments de base – et un grand nombre possible de résultats. Il suffit d'appliquer méthodiquement deux ou trois principes pour arriver à épuiser toutes les combinaisons possibles, ce qui revient à accéder à une totalité finie. Cette définition fait donc la part belle à une conception mécaniste de la combinatoire.

Cette conception mécaniste permet d'interroger au mieux les imaginaires de la combinatoire, les discours totalisants qu'elle sert et justifie. Ce qui nous intéresse, c'est sur quelles bases fonctionnent un système combinatoire et ensuite dans quel contexte historique, philosophique, technique, voire esthétique il est utilisé. Si nous reprenons nos deux exemples ci-dessus, on peut supposer que dans le cas des Lego®, le nombre de combinaisons possibles soutient un discours sur la créativité. On peut, avec une boîte de Lego®, créer une foule de choses : il suffit de faire fonctionner son

12 Le principe de commutativité pour l'opérateur des combinaisons utilisant les mêmes éléments mais dans un ordre différent. Le langage, n'est signalé, est non-commutatif. Il peut s'abstenir n'est pas la même que le mot « abstenir », même s'il est composé des mêmes éléments de base. L'anagramme est un exemple de pratique du langage qui en explore le principe de non-commutativité.

13 Le nombre de combinaisons possibles pour un ensemble fini *n*, commutatif et sans redoublement, est de 2^n-1. En l'occurrence, il y 15 combinaisons possibles: $2^4-1 = (2x2x2x2) - 1 = 16-1=15$. Ce calcul prétoupose qu'il ne peut y avoir de combinaisons plus longue que *n*.

Quatrième niveau

REMARQUES FORMELLES

Cette thèse a fait l'objet d'un certain nombre de choix formels qu'il nous faut expliquer avant de rentrer dans le vif du sujet.

Mise en page

Un soin particulier a été apporté à la maquette de ce texte, conçue par Adeline Goyet et l'auteur de cette thèse ; mise en page par Martine Fournier. Nous avons voulu inciter à une lecture savante du texte, tout en facilitant l'orientation dans les quelques cinq cents pages de ces deux tomes.

Il existe quatre niveaux de titre. Le premier niveau – le niveau des chapitres – est désigné par un chiffre en capitale romaine (I). Le deuxième niveau est désigné par un chiffre en notation arabe **(1)** ; le troisième niveau par une lettre capitale **(A)** et le quatrième niveau par une lettre bas de casse (**a**). Par exemple, l'indication « III. 3. B. c. » renvoie à la troisième sous-sous partie (**c**) de la deuxième sous-partie **(B)** de la troisième partie **(3)** du troisième chapitre (III). Nous avons fait un saut de page entre chaque niveau de titre, sauf entre le troisième et quatrième niveau.

En haut de page, un « fil d'Ariane » permet de répérer à chaque page où le lecteur se situe dans la thèse. Sur la page de gauche, on trouve le numéro du chapitre et les premiers mots de son titre. Sur la page de droite, un trouve le numéro de la sous-partie et de la sous-sous partie, assortis des premiers mots de leurs titres respectifs.

Ce paragraphe est composé en Avenir

Entre chaque chapitre et partie (niveau de titre 1 et 2), nous avons ménagé des transitions qui synthétisent les résultats obtenus et annoncent le contenu de la partie à venir. Ces transitions sont composées en police de caractères Avenir, elles sont donc facilement repérables dans le cours du texte. Idéalement, la dynamique globale de cette thèse peut être lue en sautant de transitions en transitions. Toutes les annonces de partie à venir sont soulignées, et assortie d'un repère de titre entre parenthèses (IV. 3. B. a).

1 En sus de ce système d'annonce, nous avons crée des « drapeaux » dans la marge de gauche, composés d'un repère de titre assorti de quelques mots tirés de la thèse (soulignés dans le texte) et de numéros de page. Ce drapeau est un renvoi interne : il signifie que les mots soulignés renvoient à une autre partie de la thèse, celle indiquée par les repères de titre.

2 Compte tenu de notre objet, nous avons décidé de faire un glossaire des termes techniques et des concepts utilisés. Vous trouverez ce **glossaire** dans le second tome de la thèse, p. 3. Chaque entrée de glossaire est mise entre { accolades }. Toutes les entrées de glossaire présentes dans une double page sont rassemblées dans un bloc de texte situé dans la marge extérieure d'une des deux pages, selon là où se situe la première entrée. Nous avons également inclus dans le second tome un index. Nous avons choisi de ne pas créer d'entrée de glossaire lorsque le mot apparaît dans une citation. Il s'agit là d'une parole rapportée, le sens du mot n'est peut-être donc pas exactement le même que celui que nous utilisons dans notre glossaire.

Marche éditoriale

Le nous de modestie est utilisé dans l'essentiel de la thèse. Beaucoup de textes en anglais ont été traduits en français par nos soins. Lorsque c'est le cas, le texte anglais original est reproduit dans la note de bas de page où est donnée la référence bibliographique de la citation. Lorsque ce n'est pas le cas, le texte anglais original n'est pas reproduit. Le nom du traducteur n'est pas mentionné dans la note de bas de page mais dans la référence bibliographique complète en fin d'ouvrage (voir « Bibliographie »).

Pour la présentation générale, nous avons suivi la marche éditoriale de Paris-Sorbonne[1] ainsi que les recommandations du *Lexique des règles typographiques en usage à l'Imprimerie Nationale* (troisième édition, 1997, ISBN : 978-2-11081-075-5). Nous n'avons fait qu'une seule exception à ces recommandations. Dans le cas d'un article ou d'une partie de chapitre en anglais dans une revue ou un livre anglophone, nous avons choisi de composer en romain le titre de l'article, non en italique comme c'est le cas pour tout autre mot en anglais. Par exemple, si nous citons l'article « *On "Sourcery," or Code as a Fetish* » dans le texte courant, il sera en italique mais la référence apparaîtra de la façon suivante en note de bas de page : On "Sourcery," or Code as a Fetish. *Configurations*. 2008, vol. 16, nº 3, p. 299-324. Nous avons voulu ainsi privilégier la lisibilité des références bibliographiques, au prix d'une petite entorse aux règles de présentation. Tous les autres mots en langue étrangère sont en italique, sauf « tweet » et « retweet » ayant intégrés l'édition 2017 du *Petit Robert*. Les acronymes anglais (API) ne sont pas italisés et ne portent pas de marques de pluriel, pour améliorer la lisibilité du texte. Les noms de logiciels sont composés en romain.

1 *Guide pour la rédaction et la présentation des thèses à l'usage des doctorants*. 2007, Université Paris-Sorbonne. [En ligne] [consulté le 29 août 2017]. Disponible à l'adresse : http://www.paris-sorbonne.fr/IMG/pdf/guidoct.pdf.

`Tous les extraits de code sont composés en Courier New.`
Le mot « Web », lorsqu'il désigne le *World Wide Web*, commence par une capitale. Lorsqu'il est employé en tant qu'adjectif, nous l'écrivons « web », comme dans « page web » ou « API web ». « Écrits d'écran » est systématiquement mis sous cette forme (écrits au pluriel, écran au singulier[2]), ainsi que l'expression « textes de réseau » : nous parlons **des** textes sur **un seul réseau**, Internet et plus particulièrement le Web.

Les références bibliographiques suivent la norme de présentation ISO 690[3] et le *Rational Bibliographic : guide de rédactions des références bibliographiques (v. 0.91)*[4] de la Bibliothèque de l'École Polytechnique Fédérale de Lausanne lorsque la norme ISO ne suffisait pas.

Bonne lecture !

2 Nous avons jugé que la catégorie de « l'écran » renvoyait d'abord à la dimension anthropologique de la « pensée de l'écran » et en second plan à la diversité technnologique des écrans : tablettes, téléphones portables, ordinateurs, etc. Voir Christin, Anne-Marie. *L'image écrite ou la déraison graphique.* Paris : Flammarion, 1995.

3 Nous avons consulté la synthèse de cette norme faite par Véronique Pierre. *La norme ISO 690 (Z 44-005).* 2007. [En ligne] [mis en ligne le 23 mai 2007] [dernière modification le 14 avril 2009] [consulté le 29 août 2017]. Disponible à l'adresse : http://revues.refer.org/telechargement/fiche-bibliographie.pdf.

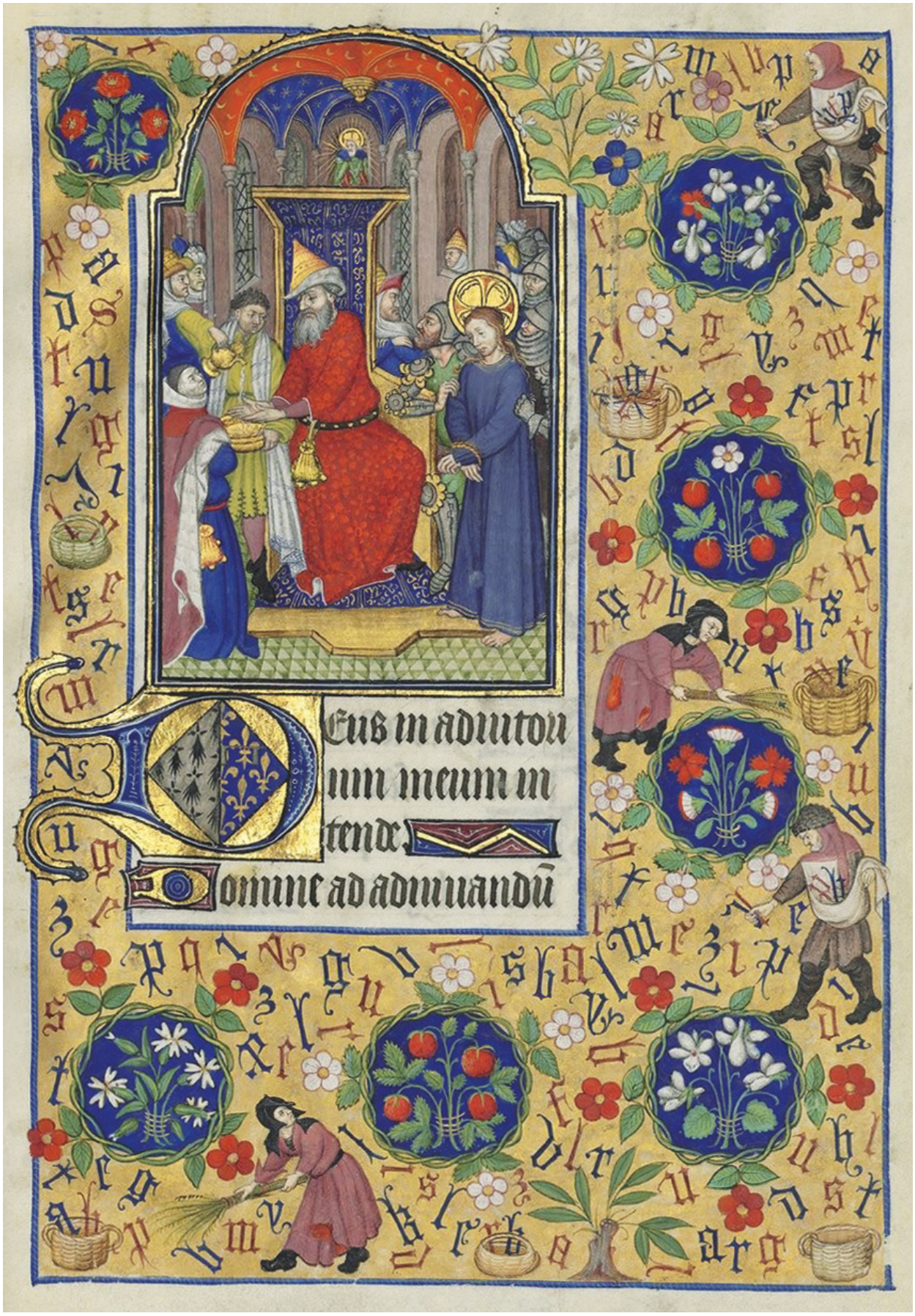
4 École Polytechnique Fédérale de Lausanne. *Rational Bibliographic : guide de rédactions des références bibliographiques (v. 0.91).* 2015. [En ligne] [mis en ligne le 2 mars 2015] [consulté le 29 août 2017]. Disponible à l'adresse : Disponible sur http://go.epfl.ch/guide-bibliographie.

TABLE DES MATIÈRES

TOME II

TABLE DES ANNEXES

TABLE DES ILLUSTRATIONS


Deus in adiutoriu
um meum in
tende
Domine ad adiuuandū

Un engendra Deux.
Deux engendra Trois.
Trois engendra les myriades,
et les myriades

retournent dans

l'Unique

Borel, Henri. *Wu Wei : le non-agir d'après Lao Tse et le taoïsme.* Nice : Nataraj, 2016 [1913], p. 23.

(Page précédente) : *Livre d'heures de Marguerite d'Orléans,* vers 1437. BNF Latin 1156b, fol. 135r. [En ligne] [consulté le 1er septembre 2017]. Disponible sur : http://gallica.bnf.fr/ark:/12148/btv1b52502614h/f279.image

INTRODUCTION

Le constat d'une présence

Allumez votre ordinateur, votre téléphone connecté, votre tablette. Connectez-vous à Internet, ouvrez votre navigateur, lisez une page au hasard. Vous y trouverez – dans une immense majorité des cas – des boutons « Partager », « J'aime », « +1 », des lecteurs vidéos, des { tweets }, des cartes Google Maps… Autant de « { petites formes[5] } » qui peuplent nos navigations ordinaires et à la présence desquelles nous sommes aujourd'hui habitués, tout du moins acclimatés. Elles composent notre paysage médiatique contemporain.

petites formes, standardisation, tweet, Web

Voir glossaire tome II, p. 3-25

Cette thèse a pour objectif d'interroger cette présence. D'où viennent ces formes ? Comment expliquer leur omniprésence ? Quels procédés techniques en permettent l'apparition ? Quels objectifs communicationnels, économiques, éventuellement idéologiques servent-elles ? Qu'est-ce que les entreprises qui en promeuvent la diffusion (Google, Twitter, Facebook…) en disent ? Pourquoi sont-elles partout identiques et pourtant à chaque fois légèrement différentes, selon la page sur laquelle elles apparaissent ? En quoi participent-elles à la { standardisation } visuelle du { Web } contemporain, à cet « air de famille » qu'entretiennent la plupart des pages web ?

5 Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *L'économie des écritures sur le Web. Volume 1 : traces d'usage dans un corpus de sites de tourisme.* Cachan : Hermès Science – Lavoisier, 2012, p. 165-201.

D'un constat à une question de recherche

Ce constat d'un air de famille fut le point de départ de notre parcours intellectuel. Notre projet initial de thèse en Sciences de l'Information et de la Communication (SIC) entamée en cotutelle entre le GRIPIC (Paris-Sorbonne) et la Chaire de Recherches sur les Cultures Numériques (Université Laval, Québec) portait sur les outils permettant de gérer plusieurs comptes issus de divers réseaux socionumériques comme Facebook ou Twitter. C'était déjà la question très générale de l'hétérogénéité qui nous animait alors, les rapports entre l'un et le multiple. Comment des { logiciels } permettent de manier des fragments textuels issus de plusieurs plateformes, comment les donnent-ils à lire dans un espace unifié et à quelles fins communicationnelles ?

API,
appel,
code,
développeur,
documentation,
HTML,
logiciel,
petites formes,
programme,
requête,
tweet,
Web
widget

Voir glossaire tome II, p. 3-25

Si la question de l'hétérogénéité ne nous a pas vraiment quitté, nous avons très vite resserré notre focale sur un acteur en particulier : les { API } web, ou *Application Programming Interface* (interface de programmation[6]). Les { API } sont des outils d'écriture informatique qui permettent d'écrire des { programmes } ou des pages web. Une { API } permet à un { développeur } – une personne qui écrit des { programmes } informatiques – d'accéder à des jeux de données ou à des blocs de { code informatique } préécrits pour remplir une certaine fonction. Pour accéder aux blocs préconstruits, il lui suffit de lire la { documentation } de l'{ API } où sont expliqués ces blocs, leur fonctionnement et comment y accéder. Pour récupérer ces jeux de données ou ces blocs, il doit bien souvent s'identifier, puis – selon les { API } – formuler un { appel } (ou { requête }) à l'{ API }, en suivant la syntaxe appropriée. L'{ API } lui renvoie alors une réponse avec idéalement les données demandées, ou un message d'erreur si la { requête } a été mal formulée ou si l'authentification ne permet pas d'accéder aux données demandées. Si, par exemple, un { développeur } veut se servir d'un fond de carte Google Maps pour écrire sa page web, il pourra s'identifier auprès de Google, formuler une { requête } à l'{ API } pour accéder à ce fond de carte et ainsi en disposer pour écrire son site.

L'{ API } est le procédé technique qui structure et permet cet accès. Ainsi, des { développeurs } externes peuvent utiliser des données (ici un fond de carte) auxquelles ils n'ont théoriquement pas accès, sauf de manière

6 Il n'existe pas d'acronyme pour la traduction française. Nous utiliserons donc dans cette thèse soit l'acronyme anglais API, soit le terme français « interface de programmation ».

très contraignante[7]. Une { API } se compose donc de deux éléments : l'infrastructure technique (serveurs, protocoles d'authentification, etc.) qui permet l'échange de données ; la { documentation }, soit la présentation de l'{ API } : de ses usages possibles, la syntaxe à respecter, les formats et type de données disponibles…

Pour que naisse le sujet de cette thèse, il fallait encore opérer un déplacement. Notre intérêt n'était pas tant la place des { API } dans les outils de gestion des réseaux socionumériques, outils souvent destinés à des professionnels, ce qui aurait pu nous amener vers la question de l'expertise ou en tout cas d'une forme de savoir professionnel. Or, nous avons toujours été plus intéressé par l'infra-ordinaire[8] que par l'extra-ordinaire. Plutôt que d'étudier le rôle des { API } dans un certain domaine professionnel, il nous semblait plus intéressant de voir comment cet outil d'écriture, qui nécessite certains savoirs informatiques spécialisés, avait intégré le { Web } que nous lisons quotidiennement.

Nous sommes alors revenu aux « { petites formes } » évoquées plus haut. Il s'avère en effet que, dans la liste que nous avons donnée, toutes ont pour point commun d'être générées par le biais d'une { API }. Sans { API }, sans accès structuré aux données de Google, Facebook ou Twitter, tous ces boutons, ces cartes, ces vidéos n'existeraient pas, ou en tout cas pas sous cette forme et – posons encore cela comme hypothèse – certainement pas en si grand nombre. En effet, ces plateformes web donnent accès, dans la { documentation } de leur { API }, à un lot de formes préconstruites, peu adaptables mais très facilement reproductibles. L'{ API } fournit le { code } { HTML[9] } pour générer ces formes, { code } qu'il suffit ensuite de copier / coller dans sa page web pour pouvoir y intégrer une carte, une vidéo, un { tweet }, un bouton *Like*, etc. C'est ce que les plateformes web nomment des « { *widgets*[10] } ». Ce sont sur ces { *widgets* }, unités élémentaires du texte numérique à venir, que s'est portée notre attention.

Précisons d'emblée que les { API } que nous étudions sont des { API } web publiques, c'est-à-dire celles dont le système de { requête }/réponse passe par le protocole { HTTP }, dont les données transitent par le ré-

7 On peut imaginer un programme qui récupère le fond de carte directement sur Google Maps, sans passer par l'API, mais cela prendrait beaucoup plus de temps et les données récupérées seraient alors sous un format beaucoup moins exploitable que celui fourni par l'API.

8 Perec, Georges. Approches de quoi ? Dans : *L'infra-ordinaire.* Paris : Éditions du Seuil, 1989, p. 9-33.

9 L'HTML, pour HyperText Markup Langage est un langage informatique, principalement utilisé dans le Web pour décrire à l'aide de balises le contenu d'une page.

10 *Widget* est un mot valise, composé de *windows* et *gadget*, qui désigne en informatique un élément visuel d'une interface graphique, une brique élémentaire de composition des interfaces. Une liste déroulante, une petite fenêtre isolée avec une horloge ou la météo, sont des *widgets*.

API,
code,
code source,
développeur,
documentation,
interopérabilité,
langage de programmation,
programmation,
standardisation,
Web

Voir glossaire tome II, p. 3-25

seau Internet et dont la { documentation } est rendue publique. Il y'a des { API } non-web, par exemple l'{ API } du { langage de programmation } Java ; il y a des { API } au degré de publicité très faible, par exemple une { API } qui servirait à structurer l'échange de données entre deux systèmes d'une même entreprise. Dans ce cas, la { documentation } n'est rédigée qu'à destination des ingénieurs internes à l'entreprise.

Les { API } web publiques constituent donc notre objet de recherche. Lorsque nous parlons, dans les pages de cette thèse, d'{ API }, nous parlons uniquement des { API } web publiques. Et notre question de recherche part d'un constat : ces { API } sont des outils d'écriture qui ont aujourd'hui intégré la chaîne éditoriale du { Web }, c'est-à-dire la façon d'écrire des pages web. Dès lors nous pouvons formuler la **question générale de notre thèse** : quels rôles jouent les { API } dans la production des textes de réseau contemporains ? Comment les { API } contribuent-elles à la fabrication de ces textes, à leur { standardisation } graphique, à leurs conditions de visibilité et de lisibilité ? Au nom de quels attendus et de quelles conceptions du texte ? Au profit de qui, de quoi, en vertu de quelles valeurs ? Sous quelles modalités techniques précises ?

L'API : un objet composite [11]

Jusqu'à maintenant, les { API } ont fait l'objet de peu d'analyses, du moins en sciences humaines et sociales. Leur rôle a été souligné dans l'essor d'une « culture collaborative[12] » ou d'un « expressivisme digital[13] », notamment parce qu'elles reposent sur un principe de délégation de l'écriture qui permet et encourage des formes comme les *mashups*[14] ou les initiatives de { développeurs } amateurs. On trouve des analyses d'ordre socio-économiques, qui analysent la structuration de communauté de { développeurs } autour des { API[15] }, le rôle des { API } dans les nouveaux modèles économiques[16], ou encore sur le statut juridique des { API } et leur rôle dans la législation sur l'{ interopérabilité } logicielle[17], mais la liste reste – pour l'instant – assez courte.

Un objet technique

Toute la difficulté de prendre comme objet les { API } web est que ce sont d'abord des objets techniques, au sens élémentaire du terme : ils demandent un certain savoir-faire. Ils nécessitent d'entrer dans une certaine logique, celles de l'informatique. Ce sont des outils d'écriture encore largement ancrés dans le { code informatique }, ce qui a deux conséquences. Tout d'abord, il faut – pour un doctorant en Sciences de l'Information et de la Communication comme nous – une acclimatation à cette culture technique. Sans prétendre savoir programmer, nous avons dû pour autant acquérir quelques notions de base en { programmation }, ne serait-ce que pour savoir lire un { code source } et comprendre comment une { API } fonctionne. Ensuite, faire une thèse sur les { API }, c'est bien souvent se retrouver confronté au mieux à un silence poli, au pire à une réticence,

11 Le Marec, Joëlle. *Ce que le « terrain » fait aux concepts : vers une théorie des composites.* Habilitation à diriger des recherches. Paris : Université Paris Diderot – Paris 7, 2002.

12 Rieder, Bernhard. Entre marché et communauté : une discussion de la culture participative à l'exemple de Google Maps. Dans : *Ludovia 2008 : Do It Yourself 2.0,* Ax-les-Thermes, 2008, p. 282-298.

13 Allard, Laurence. L'impossible politique des communautés à l'âge de l'expressivisme digital. *Cahiers Sens public*. 2010, nº 7-8, p. 105-126.

14 Benslimane, Djamal, Dustdar, Schahram et Sheth, Amit P. Services Mashups: The New Generation of Web Applications. *IEEE Internet Computing*. 2008, vol. 12, nº 5, p. 13-15 ; Plantin, Jean-Christophe. Les cartes numériques comme support de remédiation d'une catastrophe nucléaire. Le processus de cartographie de radiation suite à l'accident nucléaire de Fukushima Daiichi. *Réseaux*. 2014, nº 187, p. 163-193. Les *mashups* sont des façons de composer un logiciel, un site web et plus largement n'importe quel texte par collage et mélange d'éléments disparates.

15 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Computational Culture* [en ligne]. 2013, nº 3. [Mis en ligne le 16 novembre 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api.

16 Bodle, Robert. Regimes of Sharing. *Information, Communication & Society.* 2011, vol. 14, nº 3, p. 320-337 ; Giffard, Alain. Des lectures industrielles. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *Pour en finir avec la mécroissance : quelques réflexions d'Ars industrialis.* Paris : Flammarion, 2009, p. 117-216.

17 Samuelson, Pamela. Oracle v. Google: Are APIs Copyrightable? *Communications of the ACM*. 2012, vol. 55, nº 11, p. 25-27 ; Samuelson, Pamela. The Uneasy Case for Software Copyrights Revisited. *George Washington Law Review.* 2011, vol. 79, nº 6, p. 1746-1782.

voire à un rejet, lorsque nous essayions de présenter notre objet à des novices de l'informatique. Trop compliqué, trop abstrait, trop « technique »… l'aspect informatique de notre objet rebutait la plupart des gens à qui nous avions l'occasion de parler. Au contraire, les personnes plus familières de l'informatique voyaient rapidement ce à quoi nous faisions allusion, mais c'est alors notre regard qui posait problème. Peu voyaient en quoi les { API } n'étaient pas autre chose que des « tuyaux » à récupérer des données.

API, code, développeur, logiciel, petites formes, Web

Voir glossaire tome II, p. 3-25

En d'autres termes, nous étions face à deux impensés, certes différents dans leurs expressions mais qui tiennent au même objet : la médiation technique. D'un côté, la médiation est ignorée dans sa part technique (rejet du { code informatique }) ; de l'autre elle est niée dans sa part culturelle (les { API } ne sont que des tuyaux). Pourtant, la plupart des gens rebutés au premier abord par le { code informatique } ont eu affaire à une { API } (en cliquant sur un bouton « J'aime », en manipulant une carte Google…). Ceux dont c'est le métier reconnaissaient, eux, assez vite au fil de la discussion qu'une { API } était plus qu'un simple conduit à données. Notre objet demande donc un effort : ne pas se laisser piéger par la technicité – réelle – des { API }, pour en accepter la complexité. Si le nécessaire à été fait, dans cette thèse, pour rendre la technicité de nos objets abordables, le sujet réclame de nos lecteurs un certain effort.

Un objet social

Ce que montrent en partie les travaux existants, c'est que les { API } ne peuvent justement pas être réduites à leur fonction technique. Elles ont un rôle économique, culturel, politique… Elles sont éminemment « composites[18] ». Ce sont des objets techniques, qui font partie de nos lectures quotidiennes sans que l'ont ait à s'interroger plus en avant sur cette part technique. Les { API } font l'objet de puissants discours de valorisation de la part des industriels de l'informatique[19] et accompagnent les mutations contemporaines du { Web }. Historiquement, les { API } web se sont développées à partir des années 2000, d'abord autour d'Ebay qui publie son { API } en 2000, puis Amazon en 2002[20]. Le mouvement

18 Le Marec, Joëlle. *Op. cit.*

19 Nous en prenons pour exemple les conférences annuelles « APIDays », le blog de « l'évangéliste des API » Kin Lane ou encore l'annuaire des API Programmable Web. Voir Lane, Kin. *The Api Evangelist blog* [en ligne]. Blog personnel. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://apievangelist.com/blog/ ; Apidays. Site officiel [en ligne]. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.apidays.io/ ; Programmable Web. *ProgrammableWeb – APIs, Mashups, and the Web as Platform* [En ligne]. Site officiel. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.programmableweb.com/.

20 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

s'amplifie au milieu des années 2000, avec la publication de l'{ API } de Google Maps en 2005, puis celle de Twitter – quasi concomitante avec le lancement de la plateforme – et de Facebook, toutes deux en 2006[21]. On peut donc dire que les { API } ont accompagné l'évolution du { Web }, du { Web } marchand d'Ebay et d'Amazon jusqu'au { Web } de plateformes[22] des Google et consorts[23].

De façon similaire, les { API } font partie d'un mouvement contemporain de valorisation de la « donnée » comme vecteur de richesse[24]. En effet, si les données sont un enjeu économique et symbolique important, alors les moyens d'accès à ces données deviennent des points de passage névralgiques. Or, les { API } fournissent précisément de tels points d'accès. Amazon, Google, Facebook ou Twitter ont largement construit leur développement sur la publication d'une { API[25] }. En donnant accès à leurs données, elles ont permis la création de plusieurs centaines, voire de plusieurs milliers d'applications et de sites web connexes qui utilisent ces données – phénomène que les professionnels qualifient souvent d'« écosystème ». Grâce à leurs { API }, Amazon ou Google créent un « écosystème » de { développeurs }, qui dépendent des données de ces plateformes pour écrire leurs propres { logiciels }, mais qui en retour bénéficient de moyens d'accès facilités à ces données (grâce à l'{ API }) et peuvent donc les utiliser à leurs propres fins.

Un lieu de définition de la culture numérique

Plus largement, les { API } en tant qu'elles permettent la création de « { petites formes } » participent à un mouvement de transformation, d'hybridation d'objets culturels fondamentaux par le biais du numérique. La carte en est un bon exemple. L'une des premières { API } rendue pu-

21 Pour plus de détails sur l'histoire des API web, voir Bodle, Robert. Regimes of Sharing. *Op. cit.*, p. 325-327.

22 Alors que le Web des années 1990 s'est construit sur un ensemble de documents – les sites web – reliés entre eux, les années 2000 ont marqué une évolution avec la naissance de sites construits comme des logiciels spécialisés, dédiés à une tâche précise : réseau social (Facebook), achats en ligne (Amazon), suite bureautique (Google avec Gmail, Google Drive, Google Docs, etc.). Chacune de ces plateformes recueille et exploite à sa façon les données des utilisateurs. Cette fragmentation du navigateur – point central d'accès au Web – en une myriade d'applications est une dynamique importante du Web contemporain. Voir Doueihi, Milad. *Pour un humanisme numérique.* Paris : Éditions du Seuil, 2011, p. 17.

23 Cette dynamique du Web de plateformes n'étant pas achevée et cette thèse n'ayant pas de visée historique, nous avons décidé de ne pas lui donner de bornes historiques. Si notre objet concret est apparu au milieu des années 2000, notre objet de recherche (le texte numérique) puise dans un histoire bien plus longue et la dynamique que nous décrivons est encore vivace. Il nous semble donc peu pertinent de circonscrire temporellement notre objet.

24 Haupt, Michael. "Data is the New Oil" — A Ludicrous Proposition. Dans : *Twenty One Hundred* [en ligne]. Article du 2 mai 2016. [Mis en ligne le 2 mai 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://medium.com/twenty-one-hundred/data-is-the-new-oil-a-ludicrous-proposition-1d91bba4f294#.vjyvcwnp0 ; The world's most valuable resource is no longer oil, but data. *The Economist* [en ligne]. Article du 6 mai 2017. [Mis en ligne le 6 mai 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.economist.com/news/leaders/21721656-data-economy-demands-new-approach-antitrust-rules-worlds-most-valuable-resource.

25 Bodle, Robert. Regimes of Sharing. *Op. cit.*, p. 320-337 ; Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

blique fut celle de Google Maps, ouverte en novembre 2005[26]. Elle permit à cet objet particulièrement circulant de trouver des moyens inédits de reproduction, mais en rendant explicite dans le même temps l'hybridation de la carte et du { code }. Ce que Google donne à lire dans sa *Geo API*, c'est la façon dont Google construit une carte et cette construction passe par le { code informatique } et sa logique propre. En ce sens, les { API } sont un lieu privilégié d'observation de la façon dont « [...] le numérique interroge [...] nos objets premiers[27] ». Ce sont des lieux où l'hybridation se fait jour, entre la culture informatique de la { programmation } et la culture lettrée propre aux objets que l'{ API } permet de créer (en l'occurrence ici, les savoir-faire cartographiques). Une { API } se trouve donc à l'articulation du culturel et du technique, entre l'histoire longue de la façon dont les humains créent du sens en commun et de l'outillage technique nécessaire à cette création. À ce titre, elles sont un objet particulièrement intéressant à interroger pour qui s'intéresse à la culture numérique, entendue comme

API, code, programmation, Web

Voir glossaire tome II, p. 3-25

> *« [...] faite de modes de communication et d'échange d'informations qui déplacent, redéfinissent et remodèlent le savoir dans des formes et des formats nouveaux, et de méthodes pour l'acquérir et le transmettre[28] ».*

Si notre question, dans sa formulation la plus large, porte sur ce que le numérique fait à la culture, alors les { API } s'avèrent être un objet essentiel pour comprendre ces transformations.

26 Google. Introducing our Geo Developers Blog. Dans : *Google Geo Developers Blog* [en ligne]. Article du 28 mai 2008. [Mis en ligne le 28 mai 2008] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://maps-apis.googleblog.com/2008/05/.

27 Doueihi, Milad. *Op. cit.*, 2011, p. 12.

28 Doueihi, Milad. *La grande conversion numérique.* Paris : Éditions du Seuil, 2008, p. 37.

D'un objet de recherche à une problématique : la voie de l'écriture

Comment interroger cette hybridité et cet aspect composite des { API } ? Les Sciences de l'Information et de la Communication (SIC) le permettent, à deux titres. D'abord parce qu'un examen critique de la médiation technique est une des dimensions constitutives de cette discipline[29], sans pour autant se contenter de cette dimension. Il s'agit plutôt de prendre en compte le « lestage techno-sémiotique[30] » inhérent à tout acte de communication. Ensuite, parce que les SIC sont une discipline où « [...] le point de vue crée l'objet[31] ». Plutôt donc que de se contraindre à considérer la technique du seul point de vue technicien, on peut avec les SIC replacer cette médiation technique dans une perspective sémiotique, économique, culturelle, politique, idéologique... En d'autres termes, on peut interroger toute la complexité de nos objets.

Ce cadrage disciplinaire ne suffit toutefois pas, car il ne traduit pas quel est, dans la discipline, notre positionnement personnel. Le cadre très général de cette thèse est le rapport entre technique et culture[32], la façon dont l'une transforme l'autre et vice-versa. Plus précisément, nous étudions comment les { API }, en tant qu'outils d'écriture du { Web }, transforment un « objet premier[33] » de notre culture : le texte. Pour pouvoir au mieux problématiser les { API } et leur rôle, il convient donc de définir ce que nous entendons par écriture.

On peut définir l'écriture en première analyse comme l'inscription de signes sur un support. Cette inscription produit un texte pouvant être lu et interprété. À ce titre, l'écriture comporte un pan sémiotique – elle produit des signes observables, interprétables et analysables –, un pan technique – elle est la mise en œuvre d'un savoir-faire technique d'altération (l'inscription) d'une matière (le support) – et un pan communicationnel : elle produit un objet observable (le texte, nous y reviendrons) qui organise une situation de communication. Notons d'emblée que nous

29 Jeanneret, Yves et Ollivier, Bruno. L'invention problématique d'un champ. *Hermès*. 2004, nº 38, p. 27-29.

30 Davallon, Jean. Objet concret, objet scientifique, objet de recherche. *Hermès*. 2004, nº 38, p. 35.

31 Bruno Ollivier reprend cette formule de Saussure, synthétisant ainsi ce qui fait l'une des spécificités épistémologiques des SIC. Voir Jeanneret, Yves et Ollivier, Bruno. L'invention problématique d'un champ. *Op. cit.*, p. 27-29 ; Perret, Jean-Baptiste. Y a-t-il des objets plus communicationnels que d'autres ? *Hermès*. 2004, nº 38, p. 124 ; Gomez-Mejia, Gustavo. *De l'industrie culturelle aux fabriques de soi ? Enjeux identitaires des productions culturelles sur le Web contemporain*. Thèse de doctorat. Paris : Paris-Sorbonne, 2011, p. 9.

32 Rappelons que cette thèse fut commencée en cotutelle entre Paris-Sorbonne et la Chaire de Recherche sur les Cultures Numériques, à l'Université Laval, Québec. La codirection était assurée par Emmanuël Souchier côté Sorbonne et Milad Doueihi, alors à l'Université Laval.

33 Doueihi, Milad. *Op. cit.*, 2011, p. 12.

ne présupposons pas que l'écriture est la consignation volontaire d'un discours et qu'elle ne s'adresse qu'à d'autres humains. Il y a des pratiques d'écriture dédiées aux dieux[34], il y en a d'autres qui ont vocation à accompagner des rituels et ne transcrivent pas intégralement le discours[35]… En d'autres termes, l'écriture comme consignation volontaire de la pensée par un sujet rationnel, à destination d'autres sujets humains rationnels, est un cas particulier de ce phénomène qui recouvre bien d'autres modalités.

En l'état, cette définition de l'écriture est toutefois limitée car elle ne rend pas compte de toutes les structures sociales, politiques, économiques et intellectuelles nécessaires pour permettre cette « inscription de signes sur un support ». Il faut en effet pour cela avoir préalablement compris le support comme une surface inscriptible. Il faut une « pensée de l'écran[36] » commune à toute pratique d'écriture. La mise en application de cette technique qu'est l'écriture n'est ensuite pas neutre, car l'écriture est une « technologie de l'intellect[37] » : elle structure et conditionne des façons de penser le monde. Elle n'est pas l'application d'une structure de pensée préalable existant chez l'humain écrivant, mais c'est l'humain qui, écrivant et engageant son corps dans une situation d'énonciation[38], structure sa façon de se penser lui et le monde. L'écriture est donc en premier lieu un **fait anthropologique**.

Elle est en second lieu un **fait social**, inhérent à son statut de technique. Si l'écriture est en effet une technique d'inscription, encore faut-il apprendre cette technique. Il faut donc des institutions structurant son apprentissage, des maîtres, des savoirs canoniques à même d'être transmis… L'écriture nécessite des « lieux de savoirs[39] ». La maîtrise de cette technique n'est également pas neutre. Elle mobilise tout autant que produit une structuration sociale entre les sachants et les ignorants[40].

34 Qu'on pense aux plaques votives déposées à l'entrée des temples shinto au Japon. Ce sont là des pratiques d'écriture dont les destinataires sont des divinités.

35 Voir la notion d'« écriture attachée sélective » chez Déléage, Pierre. *Inventer l'écriture : rituels prophétiques et chamaniques des Indiens d'Amerique du Nord, XVII^e-XIX^e siècles.* Paris : Les Belles Lettres, 2013, p. 159-169.

36 Christin, Anne-Marie. *L'image écrite ou la déraison graphique.* Paris : Flammarion, 1995.

37 Goody, Jack. *La raison graphique. La domestication de la pensée sauvage.* Paris : Les Éditions de Minuit, 2007 [1979].

38 Klock-Fontanille, Isabelle. Penser l'écriture : corps, supports et pratiques. *Communication & langages.* 2014, n° 182, p. 29-43.

39 Jacob, Christian (dir.). *Lieux de savoir. 1 : Espaces et communautés.* Paris : Albin Michel, 2007 ; Jacob, Christian. *Lieux de savoir. 2 : Les mains de l'intellect.* Paris : Albin Michel, 2010.

40 Glassner, Jean-Jacques. *Ecrire à Sumer : l'invention du cunéiforme.* Paris : Éditions du Seuil, 2000.

Elle est parfois mise au service de luttes politiques, que ce soit pour renforcer le pouvoir en place[41], ou au contraire pour le combattre[42].

Troisièmement, l'écriture est une pratique **symbolique**. Elle mobilise des imaginaires, des représentations de cette pratique, de son rôle social, de celui (ou de ce à quoi) elle est destinée… Ces imaginaires peuvent être analysés sémiotiquement, soit dans les discours d'escorte, soit dans les outils mêmes, notamment dans les { interfaces[43] } si l'on prend comme objet des { logiciels } informatiques.

industrialisation, interface, logiciel, Web

Voir glossaire tome II, p. 3-25

Quatrièmement, l'écriture est un fait **économique**. Qui achète le papier, la tablette d'argile ou le papyrus ? Est-il cher ? Combien coûte l'outil d'écriture ? De quoi est-il fait, comment est-il fabriqué, quels sont ses circuits de production ? Ces questions sont importantes car l'écriture a entamé un mouvement d'{ **industrialisation** }, qui date au moins de l'usage en Europe de l'imprimerie à caractères mobiles et qui a été transformé par l'informatique, notamment avec la démocratisation du traitement de texte[44], puis par le { Web[45] }. Dans quel système économique se place une pratique d'écriture ? Dans quelle chaîne industrielle ? Quels sont les corps de métiers mobilisés et ceux valorisés tant économiquement que symboliquement ? Toutes ces questions font – aussi – partie intégrante de l'analyse d'une pratique d'écriture.

On le voit, si l'écriture est une technique, elle ne peut se comprendre qu'en contexte, à la croisée de logiques techniques et sémiotiques bien sûr, mais aussi anthropologiques, sociales, symboliques et économiques. C'est en cela que toute pratique d'écriture est composite[46] et donc une porte d'entrée privilégiée pour adopter un point de vue communicationnel. Elle permet de lier les multiples dimensions d'un phénomène, de le comprendre non pas dans son intégralité mais dans la façon dont ces différentes logiques se croisent et s'hybrident.

41 Névot, Aurélie. *Comme le sel, je suis le cours de l'eau : le chamanisme à écriture des Yi du Yunnan, Chine.* Nanterre : Société d'ethnologie, 2008.

42 Goody, Jack. *Pouvoirs et savoirs de l'écrit.* Paris : La Dispute, 2007, p. 129-161.

43 Candel, Etienne et Gomez-Mejia, Gustavo. Littératures de salon : Des « régimes sociaux » du littéraire dans les « réseaux en ligne ». Dans Saleh, Imad, Leleu-Merviel, Sylvie, Jeanneret Yves *et al.* : *H2PTM'09. Rétrospective et perspective 1989-2009.* Cachan : Hermès Science – Lavoisier, 2009, p. 205-128 ; Jeanne-Perrier, Valérie. Des outils d'écriture aux pouvoirs exorbitants ? *Réseaux*. 2006, vol. 137, n° 3, p. 97-131.

44 Kirschenbaum, Matthew G. *Track changes: a literary history of word processing.* Cambridge : Belknap Press, 2016.

45 Davallon, Jean. *L'économie des écritures sur le Web. Volume 1 : Traces d'usages dans un corpus de sites de tourisme.* Cachan : Hermès Science – Lavoisier, 2012 ; Giffard, Alain. Des lectures industrielles. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *op. cit.*, p. 117-216.

46 Souchier, Emmanuël. Le carnaval typographique de Balzac. Premiers éléments pour une théorie de l'irréductibilité sémiotique. *Communication & langages.* 2015, n° 185, p. 3-22.

API, architexte, interface graphique, logiciel, Web

Voir glossaire tome II, p. 3-25

Cette thèse sur les { API } web a donc pour principal point de vue le phénomène de l'écriture comme composite. Sans négliger toutes les dimensions vues ci-dessus, notre problématique est résolument techno-sémiotique. Il s'agit de **comprendre le lien entre les outils d'écriture et les formes sémiotiques**. Comment les { API }, par leurs propriétés techniques, participent-elles aux formes de l'écriture sur le { Web } ? Au nom de quels imaginaires ? Notre problématique est donc la suivante : **quelle est la pensée du texte présente dans ces outils d'écriture particuliers que sont les { API } web ?** En d'autres termes, **quelle est la fonction éditoriale**[47] **des { API } ?**

47 La dimension éditoriale, comme nous le montrons plus loin dans cette introduction, permet de prendre en compte la dimension sémiotique du texte en l'articulant avec des considérations culturelles, politiques, techniques et économiques.

D'une problématisation à un positionnement dans le champ scientifique

Cette problématisation de l'objet par l'angle techno-sémiotique est redevable d'au moins deux courants théoriques, qu'il nous faut ici expliciter afin de situer notre démarche dans le champ scientifique et commencer à proposer des hypothèses de travail.

La sémiotique des écrits d'écran

Premièrement, notre façon de penser les { API } comme outils d'écriture et notre problématique techno-sémiotique est issue de la sémiotique des écrits d'écran, initiée par Emmanuël Souchier en 1996[48], consolidée par des travaux collectifs au début des années 2000[49] puis étendue à de multiples objets au cours des années suivantes par différents chercheurs[50]. Partant du constat que l'informatique est une technique d'écriture, cette sémiotique a comme particularité de considérer les productions numériques (pages web, { logiciels }, { interfaces graphiques }…) comme des textes. Ainsi, on peut analyser comment ils configurent une situation de communication et textualisent les pratiques sociales.

Dans cette perspective, le concept d'{ architexte } est central. Il naît d'une nécessité technique. L'informatique introduit en effet une rupture dans l'histoire de l'écriture : la séparation entre le signe et son support, entre ce qui est inscrit et mémorisé et ce qui est affiché, donné à lire. « Trace et support ne vieillissent plus ensemble[51] […] » ou plus exactement ils ne sont plus indissociablement liés sous la même forme. Ce qui est inscrit, ce sont des données sous forme éléctromagnétique dans un disque dur. Ce qui est affiché, ce sont des lettres, des images, des mots, des vidéos…. Tout un ensemble sémiotique qui nécessite un incessant passage, une conversion[52] entre l'inscrit (le technique) et l'affiché (le sémiotique).

48 Souchier, Emmanuël. L'écrit d'écran, pratiques d'écriture & informatique. *Communication & langages.* 1996, nº 107, p. 105-119.

49 Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Xoana*. 1999, nº 6, p. 97-107 ; Davallon, Jean, Després-Lonnet, Marie, Jeanneret, Yves *et al. Lire, écrire, récrire : Objets, signes et pratiques des médias informatisés.* Paris : Éditions de la Bibliothèque Publique d'Information, 2003.

50 Gomez-Mejia, Gustavo. *Op. cit.*, 2011 ; Candel, Étienne. *Autoriser une pratique, légitimer une écriture, composer une culture : les conditions de possibilité d'une critique littéraire participative sur Internet : étude éditoriale de six sites amateurs.* Thèse de doctorat. Paris : Université Paris-Sorbonne, 2007 ; Angé, Caroline. *La question du sens : écrire et lire le fragment.* Thèse de doctorat. Paris : Université Paris 13, 2005 ; Jeanne-Perrier, Valérie. Des outils d'écriture aux pouvoirs exorbitants ? *Op. cit.*, p. 97-131.

51 Souchier, Emmanuël. L'écrit d'écran, pratiques d'écriture & informatique. *Op. cit.*, p. 108.

52 Doueihi, Milad. *Op. cit.*, 2008.

Cette scission pose le problème du passage d'un niveau à l'autre : on ne peut pas écrire sous une forme directement inscriptible. Il faut donc des outils qui assurent le passage entre ces différents niveaux. Ces outils, ce sont les { architextes }, « [...] les outils qui permettent l'existence de l'écrit à l'écran [...]. Le texte naît de l'architexte qui en balise l'écriture[53] ». Ils désignent aussi bien un traitement de texte, qu'un navigateur web ou qu'un champ de recherche Google... Ce sont tous les outils qui permettent de manipuler et de produire des textes numériques. Ils ont comme particularité d'être eux-mêmes écrits, ce qui signifie qu'ils encapsulent, par leur conception même, une vision de l'activité qu'ils permettent. Par exemple, un traitement de texte, par la manière dont l'{ interface } est organisée, par les actions qu'il permet ou autorise, donne à lire une certaine conception du texte que cet outil permet d'écrire. L'{ architexte } est le cadre instituant des écrits d'écran : il permet l'écriture tout en la modelant, en lui donnant forme.

API, architexte, interface, petites formes, programmation, Web

Voir glossaire tome II, p. 3-25

Si on veut donc analyser la conception du texte présente dans les { API } web, il faut considérer ces dernières comme des { architextes } : des outils d'écriture du { Web } qui permettent cette écriture en la cadrant, ce au nom de certaines conceptions analysables sémiotiquement. Et si les { API } web sont des outils historiquement liés à cette pratique d'écriture particulière qu'est la { programmation }, on peut faire l'hypothèse que ces outils convoquent cette culture particulière du texte dans l'écriture des { petites formes } et que cette culture spécialisée informe les textes de réseau.

Cette focale sur l'{ architexte } entraîne un déplacement par rapport à au moins deux traditions en sciences humaines et sociales, toutes deux liées à l'étude des objets techniques et notamment des médias. La première, c'est celle de la sociologie de l'innovation, la seconde, les études d'usage[54]. La sociologie de l'innovation, notamment autour du paradigme de la traduction[55], analyse un objet technique comme le lieu d'ajustements entre différents cadres socio-techniques, entre différentes conceptions de cet objet. Ces conceptions font l'objet de « traductions » : de reformulations par différents acteurs qui participent ainsi à cet ajustement des cadres socio-techniques. L'objet technique cristallise ainsi des jeux d'acteurs,

53 Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Op. cit.*, p. 103.

54 Jouët, Josiane. Retour critique sur la sociologie des usages. *Réseaux*. 2000, vol. 18, nº 100, p. 487-521.

55 Akrich, Madeleine, Callon, Michel et Latour, Bruno. *Sociologie de la traduction : textes fondateurs.* Paris : Presses de l'École des Mines, 2006.

des « scénarios[56] » de son usage, écrits selon la façon dont l'acteur impliqué traduit le cadre socio-technique. La sociologie de l'innovation met donc l'accent sur l'objet technique en tant qu'il est – dans sa conception même – le fruit de négociations entre les différentes parties impliquées. Ce n'est ni notre objet ni notre approche. Nous n'avons pas un regard sociologique mais sémiotique : c'est la question de l'écriture et du texte qui nous importe, plus que l'analyse des jeux d'acteurs[57].

Deuxièmement, nous ne faisons pas une étude d'usage des { API } web, ni une étude de la façon dont ces objets sont réappropriés dans certaines communautés[58] ou par rapport à certaines pratiques (journalisme, photographie, cartographie, etc.). Nous étudions la façon dont cet outil configure ces usages, par son histoire, par les imaginaires qu'il convoque, par son fonctionnement technique… C'est l'{ API } en tant que condition de possibilité qui nous intéresse et comment cette condition de possibilité est porteuse d'une certaine conception des pratiques qu'elle permet[59].

Une « théorie du support[60] »

Ce qui suppose – et c'est le second courant théorique essentiel à cette thèse – que la médiation technique n'est pas neutre. Elle est, certes, le résultat de négociations et de visions du monde qu'elle reconduit, mais notre argument est plus radical que ça. Il se situe au plan anthropologique : la technique est anthropologiquement constituée et constituante[61]. Inspirée notamment des travaux de Leroi-Gouhran[62] et de Simondon, cette théorie postule que l'humain n'est pas extérieur à ses objets techniques, mais qu'il devient humain par la technique. La technique est un facteur d'hominisation : « […] l'homme s'invente dans la technique et la technique s'invente dans l'homme[63] ».

56 Akrich, Madeleine. Comment décrire les objets techniques ? *Techniques & Culture.* 1987, nº 9, p. 49-64.

57 À cet égard, la notion de scénario développée par Madeleine Akrich est intéressante. Si elle est de prime abord proche de celle de la textualisation des pratiques sociales par le biais d'un architexte, les concepts employés sont clairement sociologiques (acteurs) plus que sémiotiques (signe, écriture, formes…).

58 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

59 Nous verrons quelques pages plus bas que cette remarque a une influence décisive sur la constitution de notre corpus.

60 Bachimont, Bruno. *Le sens de la technique : le numérique et le calcul.* Paris : Les Belles Lettres, 2010, p. 117.

61 C'est ce que l'école de Compiègne nomme la « thèse TAC ». Voir Steiner, Pierre. Philosophie, technologie et cognition. État des lieux et perspectives. *Intellectica*. 2010, vol. 1-2, nº 53-54, p. 7-40.

62 Leroi-Gourhan, André. *Le geste et la parole. 1 : Technique et langage.* Paris : Albin Michel, [2008] 1964 ; Leroi-Gourhan, André. *Le geste et la parole. 2 : La mémoire et les rythmes.* Paris : Albin Michel, [1991] 1964.

63 Stiegler, Bernard. Leroi-Gourhan : l'inorganique organisé. *Les Cahiers de médiologie.* 1998, nº 6, p. 187-194.

Quelles sont les conséquences d'une telle affirmation, particulièrement par rapport à notre problématique ? Nous parlons de l'écriture et plus particulièrement de l'écriture informatique. Or, l'écriture est une technologie intellectuelle[64], ce qui signifie qu'une modification du système d'écriture est aussi une modification de nos structures de pensée. Quelle est donc la modification induite par l'informatique en tant que technique d'écriture ? Bruno Bachimont a proposé la notion de « raison computationnelle[65] », marquée par la { discrétisation } et la formalisation du signe autour de l'importance du { calcul }. De façon plus générale, l'informatique introduit dans l'histoire de l'écriture un mouvement d'abstraction, dû à la formalisation du { calcul } et à la rupture sémiotique entre inscription et affichage[66].

API, algorithme, calcul, code, discrétisation, formalisme, logiciel, matériel, programme,

Voir glossaire tome II, p. 3-25

On peut donc supposer que les { API } web, en tant qu'outils d'écriture informatique, participent à ce mouvement d'abstraction { formelle }. Ce serait ce mouvement qui travaille et transforme le texte produit par les { API }, transformation dont les modalités sémiotiques précises restent à étudier.

64 Goody, Jack. *Op. cit.*, [2007] 1979.

65 Bachimont, Bruno. Intelligence artificielle et écriture dynamique : de la raison graphique à la raison computationnelle. Dans : Fabbri, Paolo et Petitot, Jean (dir.), *Au nom du sens. Autour de l'œuvre d'Umberto Eco.* Paris : Grasset, 2000.

66 Souchier, Emmanuël. L'écrit d'écran, pratiques d'écriture & informatique. *Op. cit.*, p. 108.

Un « matérialisme numérique »

Notre approche techno-sémiotique fait partie d'un mouvement contemporain plus large autour d'un « matérialisme numérique[67] ». Nous entendons par là le fait que nous accordons à la matérialité du support d'écriture, à la physicalité du { calcul }, à la manifestation physique du texte une grande importance. Il s'agit de considérer le numérique et ses objets propres ({ code }, { programme }, { algorithmes }, données, { API }…) non pas d'un point de vue instrumental (comment utiliser tel outil, tel { algorithme }, pour produire de nouveaux savoirs[68] ?), ni comme des entités immatérielles, mais du point de vue de leur élaboration, de leurs manipulations, de leurs transformations à travers ces manipulations. Il s'agit donc de les considérer comme des objets matériels : ils dépendent d'un agencement matériel (l'ordinateur, l'écran, le { calcul }), ils sont eux-mêmes amenés à être manipulés et transformés, ils sont le produit d'une fabrication hétérogène et ambivalente[69]. Lorsque nous parlons « d'abstraction de l'écriture », d'imaginaires ou de conceptions du texte, ce n'est donc pas pour entretenir un discours sur l'immatérialité du texte de réseau. C'est au contraire pour essayer de comprendre le rôle structurant du complexe { matériel } et { logiciel } qui permet l'affichage des textes de réseau et les tensions qui peuvent se créer avec l'« utopie de l'ordre[70] » du numérique. Une hypothèse de travail est que cette tension est centrale dans nos objets, entre un idéal abstrait du texte et un complexe { matériel } qui ne s'oppose pas à cette abstraction, mais au contraire permet de la construire.

67 Milad Doueihi, cité dans Jahjah, Marc. De la bibliographie matérielle aux « Digital Studies » ? *Revue française des sciences de l'information et de la communication* [en ligne]. 2016, nº 8. [Mis en ligne le 30 mars 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://rfsic.revues.org/1968. Voir aussi Casemajor, Nathalie. Matérialisme numérique et trajectoires d'objets : les artefacts numériques en circulation. *French Journal for Media Research* [en ligne]. 2014, nº 1. [Mis en ligne le 8 janvier 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://frenchjournalformediaresearch.com/index.php?id=263.

68 Il s'agit là d'une différence fondamentale entre notre démarche et celle des *Digital Humanities*. Même si nous partageons certains objets (le texte), nous ne prenons pas la médiation technique comme le moyen d'automatiser certaines manipulations du texte ou de constituer des corpus. C'est la médiation technique même qui est notre objet.

69 Voir par exemple pour le cas des « données publiques » : Goëta, Samuel. *Instaurer des données, instaurer des publics : une enquête sociologique dans les coulisses de l'*open data. Thèse de doctorat. Paris : École Nationale Supérieure des Télécoms, 2016.

70 Doueihi, Milad. *Op. cit.*, 2011, p. 30.

Ce matérialisme numérique n'est pas propre aux SIC, mais fait partie d'un « tournant matériel[71] » plus général en sciences humaines, qu'on voit poindre en philosophie[72], en anthropologie[73], en théorie des médias[74] et qui, dans le cas du numérique, est parfois articulé non pas à une discipline mais à une façon d'interroger des objets du point de vue de leur « infrastructure[75] ». Sans vouloir discuter du bienfondé de structurer la recherche en « tournants » successifs, il nous semble plus important de souligner que ce matérialisme numérique a deux conséquences au moins sur notre travail. La première porte sur notre définition du texte, la seconde sur notre conception de l'éditorialisation.

Texte et matérialisme

En ce qui concerne le texte, c'est un concept central dans notre thèse, mais dont les définitions varient selon les champs par lesquels on l'aborde. Ordinairement, on entend le texte comme une suite de mots, ce qu'on pourrait appeler la définition linguistique. Mais alors images, sons, mise en page, modalités de lecture sont évacués. Le second écueil, c'est de restreindre le texte au livre, ou encore à la littérature[76]. Or, le livre est un cas particulier du texte, même s'il est un référent culturel encore très fort. Il y'a des « [...] textes qui ne sont pas des livres[77] ». Ce sont ces textes qui nous intéressent. Nous entendons par texte, de façon générale, « [...] une configuration de signes de toute nature[78] », ces configurations constituant des « systèmes symboliques proposés à l'interprétation[79] ».

71 Pour un état de l'art sur cette notion, voir Casemajor, Nathalie. Matérialisme numérique et trajectoires d'objets : les artefacts numériques en circulation. *Op. cit.*

72 Dolphijn, Rick et Tuin, Iris van der (dir.). *New Materialism: Interviews & Cartographies.* Ann Arbor : Open Humanites Press, 2012.

73 Jacob, Christian. *Op. cit.*, 2010.

74 Kittler, Friedrich. *Mode protégé.* Dijon : Les Presses du Réel, 2015 ; Parikka, Jussi. *Digital contagions: a media archaeology of computer viruses.* New York : Peter Lang, 2007.

75 Yoo, Christopher S. et Blanchette, Jean-François (dir.). *Regulating the cloud: policy for computing infrastructure.* Cambridge : The MIT Press, 2015.

76 Notre rapport aux études littéraires est assez complexe. Institutionnellement tout d'abord, rappelons que cette thèse fut entamée en cotutelle entre le GRIPIC Paris-Sorbonne et la Chaire de Recherche sur les Cultures Numériques à l'Université Laval, qui dépendait du département des Études Littéraires. Notre méthode (la sémiotique), notre focale sur le texte et ses acteurs... Toutes deux sont issues des études littéraires, tout comme – en partie – les Sciences de l'Information et de la Communication. Nous insistons toutefois sur le fait que ce n'est pas « le littéraire » en tant que phénomène esthétique qui nous intéresse, ni le « texte » au sens d'une œuvre originale de l'esprit. Ce qui nous intéresse, c'est l'écriture, la médiation technique nécessaire à celle-ci, la façon dont cette médiation informe ce que nous lisons et comment cette médiation est donnée à voir dans le texte. Dans cette optique, la question de la littérature nous semble un cas d'écriture très particulier, symboliquement très fort, mais qui n'est pas notre objet.

77 McKenzie, Donald F. *La bibliographie et la sociologie des textes.* Paris : Cercle de la librairie, 1991, p. 30.

78 Pédauque, Roger T. *Le document à la lumière du numérique.* Caen : C & F éditions, 2006, p. 104.

79 Chartier, Roger, dans McKenzie, Donald F. *Op. cit.*, p. 6.

Ce que nous appelons « texte » dans cette thèse à donc quatre caractéristiques :

– Il est **un objet matériellement attesté** que l'on peut voir et manipuler (cet objet n'étant pas nécessairement un livre) :

– Il est un agencement de signes lisible et visible, ou plus précisément : le texte est lisible parce qu'organisé visuellement. Il est donc une **forme** et cette forme construit la lisibilité du texte selon des critères culturels.

– Cette forme, pour être lisible, utilise des « formes repères[80] », des stéréotypes… Des formes suffisamment répétées pour se sédimenter culturellement. Le texte fait donc l'objet d'une certaine { **standardisation** } **visuelle** qui le rend lisible et intelligible.

API, standardisation,

Voir glossaire tome II, p. 3-25

– Cet agencement de formes est toujours **ouvert à l'interprétation**. Le texte est lu et réinterprété. Il a une vie sociale qui fait que son sens ne peut être déterminé par la seule analyse de son organisation visuelle. En revanche, on peut étudier les « effets de sens des formes[81] » et la façon dont la forme dit quelque chose du texte.

Une telle définition du texte appelle deux remarques. Premièrement, avec cette définition, nous prenons nos distances par rapport aux études littéraires. Cette distance se mesure particulièrement bien dans le rapport que nous avons au fragment. Objet esthétique tout autant que médiatique, le fragment a une histoire en tant que forme littéraire mais aussi en tant que forme communicationnelle et de nombreux travaux ont souligné à quel point le numérique puisait dans cette riche histoire[82]. Si, dans notre titre, nous parlons de « briques et de blocs » et non de fragments, c'est parce que nous approchons les { API } et l'écriture informatique par sa face combinatoire. Or, alors que le fragment est « […] le regret d'une totalité perdue[83] », on peut faire l'hypothèse que penser le texte comme agencement combinatoire de briques ou de modules ouvre aux combinaisons possibles, aux agencements futurs. Alors que le fragment nous fait regarder vers le passé, la combinatoire porte notre attention vers le futur et la génération des possibles. Passer des fragments aux modules n'est donc pas innocent. C'est un choix de notre part qui nous amène à

80 Pédauque, Roger T. *Op. cit.*, p. 106.

81 Chartier, Roger, dans McKenzie, Donald F. *Op. cit.*, p. 6.

82 Angé, Caroline. Le fragment comme forme texte : à propos de Fragments d'un discours amoureux. *Communication & langages*. 2007, n° 152, p. 23-34 ; Fournout, Olivier. *Théorie de la communication et éthique relationnelle : du texte au dialogue*. Cachan : Hermès Science – Lavoisier, 2012.

83 Fabbri, Paolo. *Le tournant sémiotique*. Cachan : Hermès Science – Lavoisier, 2008, p. 36.

considérer nos objets du point de vue d'une mécanique générative du texte et donc d'une totalisation à conquérir, plutôt que celui d'une esthétique par laquelle on regrette l'unité perdue.

Deuxièmement, nous opérons un tournant vers le matérialisme numérique et ce à deux égards. En premier lieu parce que, comme l'a montré Marc Jahjah[84], la prise en compte du texte comme objet vient historiquement de la bibliographie matérielle[85]. Deuxièmement, notre approche techno-sémiotique est matérialiste en ceci qu'elle se concentre sur les deux premiers niveaux du texte : c'est un objet (face technique) ; il est un agencement de signes (face sémiotique) et la dynamique croisée de ces deux niveaux ouvre à la question de la { standardisation } et de l'{ industrialisation } du texte. Nous pouvons donc faire l'hypothèse que les propriétés techniques des { API } non seulement participent aux formes de l'écriture, mais également aux modalités de son { industrialisation }.

API, calcul, standardisation, industrialisation

Voir glossaire tome II, p. 3-25

Éditorialisation et matérialisme

Une fois le texte défini, comment définissons-nous la notion d'éditorialisation, quels rapports avec notre définition du texte et le matérialisme numérique ? La notion d'éditorialisation connaît aujourd'hui un certain succès pour expliquer certaines dynamiques du numérique[86] et notamment l'effet du { calcul } informatique sur les modalités de publication et de transformation des textes de réseau. Les travaux contemporains actent donc de la nature scripturaire de ces objets, tout en rappellant le rôle constitutif du support[87] dans la manipulation du texte. Le problème étant que ces travaux étendent la notion d'éditorialisation à une philosophie globale du numérique, en s'affranchissant progressivement de la catégorie de texte. Or, l'éditorialisation est historiquement liée au concept de texte, que ce soit lorsqu'on parle de « pratiques lettrées[88] » ou d'« énonciation éditoriale[89] ». L'éditorialisation, c'est le processus par

84 Jahjah, Marc. De la bibliographie matérielle aux « Digital Studies » ? *Op. cit.*

85 Laufer, Roger (dir.). *Le Texte et son inscription*. Paris : Editions du Conseil National de la Recherche Scientifique, 1989 ; McKenzie, Donald F. *Op. cit.*

86 Vitali Rosati, Marcello. Qu'est-ce que l'éditorialisation ? *Sens Public* [en ligne]. 2016. [Mis en ligne le 18 mars 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.sens-public.org/article1184.html. Voir également l'état de l'art réalisé par Valluy, Jérôme. Editorialisation. Dans : *Terra HN* [en ligne]. Publication du 18 mai 2015. [Mis en ligne le 18 mai 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.reseau-terra.eu/article1333.html.

87 Bachimont, Bruno. Nouvelles tendances applicatives : de l'indexation à l'éditorialisation. Dans : Gros, Patrick (dir.), *L'indexation multimédia. Description et recherche automatiques*. Cachan : Hermès Science – Lavoisier, 2007, p. 313-326.

88 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *Des Alexandries. I, Du livre au texte*. Paris : Bibliothèque Nationale de France, 2001, p. 11-40.

89 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Les Cahiers de médiologie*. 1998, vol. 2, nº 6, p. 137-145.

lequel un texte est transformé et rendu visible et lisible selon des attendus culturels et des horizons de lecture. L'éditorialisation est une partie de la chaîne du texte, où différents acteurs spécialisés – et souvent différents de l'auteur du texte : typographes, correcteurs, éditeurs, imprimeurs… – interviennent sur le texte en le transformant. En cela, l'éditorialisation est intimement liée au texte et à sa matérialité. Lorsque nous parlons donc de « fonction éditoriale » des { API }, nous faisons le constat que les { API } font partie de cette chaîne éditoriale. Selon quelle culture du texte transforment-elles les écrits d'écran ? Voilà ce qui nous reste à découvrir.

Rassembler un faisceau d'hypothèses

API, architexte, industrialisation, petites formes, programmation

Voir glossaire tome II, p. 3-25

Clarifier notre position nous a permis de formuler quelques hypothèses quant à la fonction éditoriale des { API }. Il nous faut désormais rassembler ces hypothèses éparses et structurer ainsi notre propos. Rappelons que notre problématique générale est la suivante : quelle est la fonction éditoriale des { API } ? Au nom de quelle conception du texte les { API } nous permettent-elles d'écrire ?

Nous partons du constat empirique que les { API } produisent des formes combinables[90], ce qui nous permet de formuler notre hypothèse principale : les { API } sont des outils d'écriture qui promeuvent une vision combinatoire du texte de réseau. Notre tâche va donc être de montrer en quoi les { API } sont des outils combinatoires et surtout en quoi cette combinatoire transforme les textes de réseau contemporain.

Cette hypothèse se ramifie en cinq sous-hypothèses :

1) Parce que ce sont des outils d'écriture informatique, les { API } web font partie d'une histoire de l'informatique et de la { programmation }. On peut donc en faire la **généalogie**, en faisant l'hypothèse que leur caractère combinatoire est l'héritage de manières d'écrire qui traversent l'histoire de la { programmation } et de l'informatique.

2) Parce que ce sont des outils d'écriture, les { API } web sont porteuses d'**imaginaires** et l'on peut supposer que la combinatoire est un lieu puissant d'élaboration de ces imaginaires.

3) Parce que ce sont des { architextes }, les { API } web **informent les formes à l'écran**. On peut donc faire **l'analyse sémiotique de leur fonction éditoriale**, en faisant l'hypothèse que le modèle combinatoire dont héritent les { API } renforce la modularité et la manipulabilité des textes numériques.

90 Il s'agit là d'un constat personnel, fait au fil de nos navigations, mais partagé par les chercheurs travaillant sur les mêmes formes. Voir Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 166-167.

4) Parce que ce sont des outils d'écriture permettant de produire des « { petites formes } », les { API } web font partie d'un mouvement d'{ **industrialisation** } **de l'écriture**, dans lequel on peut supposer que leurs propriétés techniques jouent un rôle important.

5) Parce que ce sont des outils d'écriture informatiques, les { API } web nous **mettent en présence d'entités machiniques** intervenant dans la chaîne éditoriale des écrits d'écran. Devant ce constat, on peut faire l'hypothèse qu'à travers les textes de réseau se jouent des **enjeux politiques de monstration ou d'invisibilisation** des différents acteurs du texte.

À travers cette hypothèse et ces cinq sous-hypothèses, nous cherchons à saisir au mieux la complexité de notre objet, complexité due au fait que nous avons affaire à des outils d'écriture. Il s'agit de considérer ensemble, ou tout du moins en relation, leur aspect technique et informatique, leur aspect historique, le contexte économique, leurs effets sémiotiques, leurs enjeux politiques… Toutes ces dimensions sont présentes dans notre objet et l'approche techno-sémiotique permet de les articuler.

Constituer un corpus

Quel corps avons-nous constitué pour vérifier ces hypothèses ?

Notre corpus principal est tripartite. Notre question de base interroge la présence des { API } dans les écrits d'écran. Une première partie de notre corpus est donc composée de captures d'écran des pages web où apparaissent les « { petites formes } » générées par les { API }. Mais notre approche techno-sémiotique exige de comprendre comment sont produites ces formes. Il a fallu donc, c'est la deuxième partie de notre corpus, prendre des captures d'écran du { code } { HTML } qui correspond à ces formes. Enfin, pour comprendre comment fonctionnent ces { API } et quel discours est porté sur les { petites formes }, nous avons pris des captures d'écran de la { documentation } des { API }.

API,
code,
code source,
formes-textes,
HTML,
industrialisation,
documentation,
petites formes,
programmation,
standardisation,
tweet,
widget

Voir glossaire tome II, p. 3-25

Choix des API étudiées

En revanche, quelles { API } choisir ? À quelles formes s'intéresser en particulier ? Nous ne pouvions pas faire ce travail sur toutes les plateformes web qui répandent des { petites formes } grâce à leur { API }. Nous avons donc choisi Facebook et Twitter, deux réseaux socionumériques. Pourquoi ce choix ? Premièrement, nous faisons l'analyse d'une présence ordinaire. De façon empirique, nous avons constaté que les formes les plus présentes étaient les boutons Facebook, Twitter, les { tweets } ancrés… Nous avons donc choisi les plateformes qui semblaient omniprésentes, celles auxquelles nous pouvions difficilement échapper. Deuxièmement, nous étudions un phénomène d'{ industrialisation } de l'écriture. Il est donc cohérent de choisir des acteurs majeurs du marché, ceux qui ont historiquement développé leur domination économique autour des { API[91] }. De plus, nous étions déjà familier de l'organisation sémiotique de Facebook suite à notre mémoire de master recherche[92], ce qui facilitait – au moins dans un premier temps – le travail de thèse. Nous aurions pu choisir l'{ API } de Google Maps, qui obéissait au moins aux deux premiers critères énoncés ci-dessus. Mais notre problématique sur les { formes-textes } rendait l'objet cartographique plus compliqué à interroger, car il fait appel à une culture (celle de la cartographie et des systèmes d'information géographique) fort différente.

91 Bodle, Robert. Regimes of Sharing. *Op. cit.*, p. 320-337 ; Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

92 Goyet, Samuel. *Facebook à l'épreuve de la mort. L'écriture du deuil à travers la fonction « groupes » : le cas « hommage à Bixente Lopez »*. Mémoire de Master. Paris : Paris-Sorbonne, 2011.

Nous avons donc retenu les { API } de Facebook et Twitter et plus précisément toutes les parties de l'{ API } qui permettent de produire ce que ces plateformes nomment { *widgets* }, ce que nous nommons les « { petites formes } » des textes de réseau. Il nous restait à ce moment-là à composer notre corpus et ses trois volets : la { documentation } de l'{ API } ; les { petites formes } en contexte ; le { code source } des { petites formes }.

Nous avons dans un premier temps effectué des captures d'écran de toute la { documentation } des { API } qui concerne leur fonctionnement global et la production de { *widgets* }. Nous avons ensuite fait le relevé de toutes ces { petites formes }, soit trente-neuf en tout :

Facebook	**Twitter**
Fait avec […]	Bouton Suivre
Se connecter *via* Facebook	Bouton *Hashtag*
Bouton *Follow*	Bouton Mention
Bouton *Like*	Bouton Tweet / partager
Bouton *Send*	Carte app
Bouton *Share*	Carte *gallery*
Publication ancrées	Carte *summary*
Vidéos ancrées	Recherche / Liste ancrée
Feed dialog	*Timeline* ancrée
Plugin commentaire	Tweet ancré
Plugin page	Vidéo ancrée
	Fav intent
	Follow intent
	Reply intent
	Retweet intent
	Tweet / Share intent
	User intent

Pour comprendre l'action éditoriale des { API }, il fallait saisir ces formes en contexte. Le second volet de notre corpus est donc composé de captures d'écran de ces formes dans des pages web. Nous avons capturé à la fois l'ensemble de la page web, pour voir où se répartissaient ces formes dans la page et pouvoir ainsi observer la { standardisation } visuelle du texte, mais aussi les détails de ces formes pour analyser précisément leur aspect. En ce qui concerne la taille de ce corpus, nous avons jugé qu'un seul exemplaire de ces formes n'était pas suffisant. Nous avons donc décidé de capturer chaque forme sur trois pages web différentes. Cette se-

conde partie de corpus est donc – idéalement – constituée de cent-dix sept exemples (39 x 3 = 117) de { petites formes } intégrées dans des sites web.

code,
code source,
petites formes,
tweet

Voir glossaire tome II, p. 3-25

Notre méthode pour récolter cette seconde partie de corpus fut celle d'un « hasard guidé ». Nous avons commencé par aller sur les sites que nous consultons habituellement, là où nous avions déjà remarqué quelques éléments intéressants. Le résultat fut insuffisant. Nous avons donc été sur Twitter et cliqué sur les dix premiers liens qui se présentaient sur notre profil, puis nous avons vérifié si des formes qui nous manquaient étaient présentes sur ces sites. Si ce n'était pas le cas, nous reprenions les dix premiers liens, jusqu'à constater une forme de saturation[93] où nous n'arrivions pas à trouver de nouvelles { petites formes } à capturer. Nous n'avons fait qu'une seule entorse à ce hasard guidé : le choix du site pornographique letagparfait.com. C'est le seul site où nous avons d'emblée fait une hypothèse sur l'usage de ces formes. Il nous semblait en effet particulièrement étonnant de trouver, sur un site pornographique, des boutons permettant d'en partager le contenu sur Facebook et Twitter, tant les sites pornographiques impliquent des pratiques qui – en général – ne se partagent pas et restent dans la sphère privée. Trouver des boutons jusqu'ici, cela serait en quelque sorte déjà confirmer l'hypothèse que ces formes sont circulantes à tel point que leur contexte de publication devient secondaire.

Avec cette méthode, il y a cinq formes que nous n'avons finalement pas rencontrées : le bouton *Hashtag*, le bouton *Mention*, la fonction « Fait avec […] », le bouton *Follow* et les vidéos ancrées Facebook. Il y a certaines formes pour lesquelles nous n'avons pas pu trouver trois exemplaires. Le corpus est donc finalement composé de **vingt-quatre** { **petites formes** }, étudiées dans **trente-sept sites** différents[94]. Les captures ont été réalisées entre **le 14 juillet et le 16 août 2015**, exceptées les captures des cartes *summary* de Twitter, qui ont été faites en avril et octobre 2014 afin de préparer une présentation orale de notre travail de thèse.

Une fois assemblé ce second volet du corpus principal, nous sommes passé au troisième volet et avons fait des captures d'écran des lignes de { code source } qui correspondent aux { petites formes } déjà capturées. Pour cela, nous avons utilisé l'extension Firebug pour le navigateur Fire-

93 Barthes, Roland. Éléments de sémiologie. *Communications*. 1964, nº 4, p. 133.

94 Dans des cas où certains sites condensaient plusieurs des petites formes recherchées, nous avons utilisé le même site pour différentes formes. Par exemple, on trouve sur slate.fr un bouton « Partager » de Twitter, mais aussi un bouton « *Like* » de Facebook. Dans ce cas, un site fournit deux formes de notre corpus.

fox, qui permet d'afficher le { code source } non pas de la page entière, mais le { code } qui correspond à un élément visuel de la page. Il nous suffisait donc de cliquer sur la { petite forme } pour voir s'afficher dans la fenêtre de notre navigateur, le { code } en regard de la forme. Cet outil a été essentiel en ceci qu'il permet de comparer dans le même espace graphique le { code } et le *widget*. Il permet donc une lecture savante, car comparative, de ces deux textes.

N'ayant pas demandé de consentement préalable aux personnes présentes sur les captures d'écran, nous avons choisi d'anonymiser notre corpus en considérant ce qui relevait d'informations publiques et d'informations privées. Tout le problème étant que les réseaux socionumériques troublent cette frontière. Si la plupart des { tweets } de notre profil sont publics, il n'en reste pas moins que nous y avons accès parce que précisément nous suivons la personne et que la plateforme met en visibilité ce { tweet } dans notre profil. En somme, notre seule identification sur Twitter ou Facebook nous permet d'accéder à ces { tweets }, dont on peut penser qu'ils n'apparaîtraient pas de la même façon sans identification préalable (puisque Twitter et Facebook éditorialisent l'affichage des publications en fonction des profils personnels). Nous avons donc choisi d'anonymiser nos captures d'écran **seulement dans le cas de particuliers** (amis, *followers* sur Twitter…) dont le visage et/ou le nom apparaissent du fait de notre identification préalable sur notre profil personnel Twitter ou Facebook. Les noms d'institutions publiques, de marques, de sociétés, d'organes de presse, les visages de célébrités n'ont pas été floutés. Nous avons estimé qu'il s'agissait là d'informations publiques. Les { tweets } de particuliers intégrés dans une page publique (article de presse par exemple) n'ont pas été anonymisés. Là encore, nous avons jugé qu'il s'agissait d'une rééditorialisation qui rend publics ces { tweets }, publicité dont la responsabilité incombe à l'auteur de l'article. Bien entendu, nous n'avons pas anonymisé notre propre nom et photo de profil, estimant que notre consentement était implicite.

Voici donc quel sera notre corpus principal. Le problème de ce corpus est qu'une grande partie (la documentation, ainsi que les extraits de code source) est illisible sous forme imprimée : les captures d'écran sont bien trop grandes pour pouvoir être mises en page de façon lisible. Pour permettre malgré tout de consulter notre corpus, nous avons choisi la solution suivante :

– les petites formes de notre corpus sont reproduites intégralement en annexe, dans le second tome de cette thèse ;

– les captures d'écran de la documentation, des extraits de code source et de l'intégralité des pages hôtes des petites formes sont gravées sur un CD joint à la thèse, accompagné d'une notice explicative. Nous espérons ainsi que nos lecteurs pourront consulter à loisir notre corpus.

Au fil des parties et selon les besoins, nous avons puisé dans d'autres sources, notamment des articles d'informatique théorique ou des guides de { programmation }. Dans ces cas-là, nous détaillons la composition de ces corpus secondaires au fil de l'argumentation. Avant d'en détailler le plan, nous souhaitons revenir sur deux points importants en ce qui concerne la constitution du corpus.

API, développeur, programmation,

Voir glossaire tome II, p. 3-25

La méthode des captures d'écran

Premier point, nous avons choisi la méthode des captures d'écran, aujourd'hui relativement courante dans la sémiotique des écrits d'écran. Cette méthode a l'énorme avantage de délimiter clairement un corpus, dans un environnement technique particulièrement labile. Cette même labilité pose en revanche certains problèmes. Premièrement, la capture d'écran annule les propriétés techniques des écrits de réseau, par exemple la possibilité de cliquer sur un lien hypertexte. Lorsque l'action que nous voulions analyser nécessitait une suite de clics, nous avons donc fait une capture de chaque étape. Deuxièmement, ce sont des écrits toujours susceptibles d'être modifiés, mis à jour. Nous en voulons pour exemple qu'entre le moment de la constitution de notre corpus et la date de rédaction de cette thèse (août 2016 – septembre 2017), Twitter a complètement refondu la présentation de son { API }. Même si après vérification, la plupart des textes n'ont pas changé, nous ne pouvions pas prévoir cette modification. Il faut donc reconnaître que nous avons pris un certain état documentaire des { API } web, un instantané à un moment précis (été 2015).

Un fil rouge techno-sémiotique

Notre second point tient à l'unité de ce corpus. Nous n'étudions pas l'utilisation des { API } web dans des communautés de pratiques (journalisme, { développeurs } indépendants, blogs…). Ce qui rassemble les sites de notre corpus, c'est **un fil rouge sémio-technique** : la récurrence de formes générées par des interfaces de programmation. D'où un aspect hétéroclite au premier abord : il n'y a aucun rapport entre un site de vidéos de football, une vidéo d'un chien jouant avec un enfant et un article du *Monde*, sauf la présence des { API } dans la chaîne de production de ces textes. C'est en cela que notre méthode du hasard guidé nous permet de constituer un corpus au plus près de notre problématique : il s'agit de se concentrer sur la dissémination de certaines formes, les procédures techniques qui permettent l'existence de ces formes et ainsi interroger le rôle éditorial de ces procédures.

Construire une argumentation

Notre thèse est composée de deux parties, regroupant cinq chapitres. Chaque chapitre est consacré à l'une des sous-hypothèses formulées à partir de la problématique suivante : **quelle est la pensée du texte présente dans ces outils d'écriture particuliers que sont les { API } web ?**

API,
calcul,
code,
formalisme,
logiciel,
matériel,
programmation,
programme,
subroutine

Voir glossaire tome II, p. 3-25

Deux parties…

Le mouvement global de la thèse peut être résumé en une phrase : **les { API } web sont le dernier avatar d'une science combinatoire de l'écriture devenue une industrie du texte**. Nous reprenons ici la remarque faite par Milad Doueihi que le numérique a ceci de particulier qu'il est le produit d'une science, l'informatique, devenue industrie puis culture[95] (le numérique en tant qu'il transforme nos objets de savoirs et nos pratiques liées à ces objets). Dans la perspective de comprendre la façon dont les { API } transforment le texte (objet culturel), nous remontons au niveau de l'informatique comme science, en mettant en évidence l'existence d'un modèle d'écriture combinatoire : le { calcul } { formel }. C'est la première partie de cette thèse, composée des chapitres I et II. Nous étudions ensuite comment ce modèle s'hybride avec des logiques industrielles et quels en sont les effets sémiotiques, économiques et politiques. C'est la seconde partie de cette thèse, composée des chapitres III, IV et V.

Mais qu'on ne s'y méprenne pas. Lorsque nous parlons d'un passage d'une science de l'écriture à une industrie du texte, il ne s'agit pas d'un passage au sens où l'on passerait d'un état à un autre. Ce n'est pas tant un passage qu'une articulation : les deux aspects cohabitent dans les { API } contemporaines, ce sont des objets qui permettent d'observer cette hybridation entre science et industrie, entre écriture et texte, entre { calcul } { formel } et culture humaniste. Nous invitons donc notre lecteur à ne pas se laisser abuser par l'historicité apparente de ce découpage. Ses vertus sont bien plus didactiques – il permet de clarifier notre propos – qu'historiques – nous ne prétendons pas décrire des effets de rupture.

95 Doueihi, Milad. *Qu'est-ce que le numérique ?* Paris : Presses Universitaires de France, 2013.

… et cinq chapitres

Le **chapitre I** est une **approche historique et généalogique** des { API } web. Nous partons de notre corpus et du constat d'une forte dimension **combinatoire** de ces outils et des formes qu'ils permettent de produire. Nous interrogeons cette dimension combinatoire en situant les { API } dans une histoire de la { programmation } et de l'informatique. D'où vient cet aspect combinatoire ? En quoi est t-il inhérent à la { programmation } comme pratique d'écriture spécialisée ? À quoi sert alors la combinatoire dans le contexte de la { programmation } ? À quels besoins techniques, économiques, pratiques répond-elle ? Nous montrons dans ce premier chapitre que la combinatoire, dans le cas des { API } web, permet une économie de l'écriture qui consiste à assembler des blocs de { code informatique } préécrits (I. 1). Cette façon particulière d'écrire n'est pas propre aux { API } web mais remonte à la structuration du marché informatique autour de la différence entre { logiciel } et { matériel } et aux modèles de { programmation } dominants dès les années 1960. Nous remontons ensuite d'un cran dans l'histoire de la { programmation } et montrons que l'idée d'abstraire, au sein d'un { programme }, des blocs de { code } remobilisables selon les contextes est inhérent à la naissance de la { programmation } dans les années 1940. Nous prenons comme exemple le concept des { *subroutines* } (I. 2), à travers lequel nous analysons cette conception abstraite de l'écriture informatique. Le dernier mouvement de cette première partie est une remontée au niveau de l'informatique théorique. Si les { API } web ont une part combinatoire, que cette part combinatoire est héritée de certains modèles de { programmation }, c'est-à-dire de la façon d'écrire des instructions à un ordinateur, est-ce que la combinatoire ne serait pas un trait plus fondamental de l'informatique en tant que science ? L'étude d'un article de Turing[96] en 1937 montrera que l'informatique est une science fondée sur une conception combinatoire – car calculatoire – de l'écriture, en tout cas d'un certain type d'écriture (I. 3). La combinatoire, si présente dans les { API } web, est donc une ligne de force de l'écriture informatique, que ce soit au niveau des outils d'écriture ({ API }, { programmation }) que de la façon même de concevoir l'activité d'un ordinateur.

96 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*. 1937, vol. s2-42, n° 1, p. 230-265.

API, architexte, calcul, discrétisation, documentation, formes-textes, industrialisation, interopérabilité

Voir glossaire tome II, p. 3-25

Le **chapitre II** repart des acquis du chapitre I en opérant un renversement. Arrivé au bout de cette généalogie des { API }, où nous avons montré que l'informatique est construite sur un modèle d'écriture codée et combinatoire, nous prenons notre point d'arrivée comme point de départ d'une nouvelle hypothèse : en quoi l'écriture informatique, construite autour du chiffre et de la combinatoire, recèle de puissants imaginaires de l'écriture ? Il s'agit de repartir des acquis de notre travail sur Turing pour en explorer les **imaginaires** du point de l'**anthropologie de l'écriture**. L'écriture décrite par Turing est une écriture chiffrée (II. 1), combinatoire (II. 2) et aux prétentions universelles (II. 3). Nous commençons par étudier les différents imaginaires du chiffre, entre puissance d'abstraction, mise en contact avec des entités divines et substitut de liens de confiance. Nous traitons ensuite de la combinatoire, considérée cette fois comme un trait fondamental de tout système d'écriture et comme une « technologie intellectuelle[97] » qui produit des formes spécifiques de pensée. Parmi ces formes de pensée, nous montrons que l'écriture combinatoire a joué un grand rôle en Occident dans la quête d'une « méthode universelle », dont la machine de Turing est le dernier avatar scientifique.

Les deux premiers chapitres plongent nos objets dans une histoire longue de l'informatique et plus largement de l'écriture et permet de donner un premier élément de réponse à notre problématique. En tant qu'outils d'écriture informatique, les { API } web reposent sur une conception combinatoire et universelle de l'écriture autour du { calcul }, pierre de touche de tout outil informatique.

Mais si les { API } sont des { architextes } qui informent les formes à l'écran, alors comment cet universalisme combinatoire transforme-t-il les écrits de réseau ? Comment ce modèle d'écriture s'industrialise-t-il et modèle-t-il les pages web que nous lisons tous les jours ? Le **chapitre III** est entièrement consacré à ces questions. Nous y faisons **l'analyse sémiotique de la façon dont l'universalisme combinatoire des { API } web travaille les écrits d'écran**. En partant de l'hypothèse que le numérique réactive une « culture anthologique[98] » du texte, nous montrons trois effets des { API } sur les pages web. Le premier est une modularisation (III. 1) : la page est conçue comme un agencement de modules abstraits qui viennent s'y ancrer de temps à autre. Cette dynamique n'est pas nouvelle mais trouve avec le { calcul } une dimension inédite. Notamment parce que le

97 Goody, Jack. *Op. cit.*, 2007 [1979].

98 Doueihi, Milad. *Op. cit.*, 2011, p. 105-138.

{ calcul } permet une grande manipulabilité des formes du texte (III. 2) et une grande { discrétisation } de ces formes (III. 3). Ces trois dynamiques – modularisation, manipulabilité, { discrétisation } – composent l'essentiel des effets des { API } sur les pages web contemporaines.

Comment modularité, manipulabilité et { discrétisation } des { formes-textes } servent-elles à travers les { API } une { industrialisation } de l'écriture[99] ? C'est la question qui oriente notre quatrième chapitre. Nous y faisons l'hypothèse que les { API } web sont une cheville ouvrière essentielle d'une « industrie des passages[100] » contemporaine. Il s'agit, à travers ces outils, d'équiper, de faire désirer, de valoriser et de standardiser la circulation des textes de réseau. Notre premier argument est qu'une { API }, parce qu'elle repose sur un principe de délégation de l'écriture, équipe la circulation des { formes-textes } (IV. 1). Mais cet équipement serait vain si l'écriture n'était pas motivée, rendue désirable. Il y a donc un « régime de désir[101] » institué par les { API } web (IV. 2). Dans ce qui rend désirable une { API }, il y notamment le fait que ce qui est promis à travers le clic sur des boutons « J'aime » ou « Partager » est une forme de sociabilité. Il nous faut donc comprendre par quelles médiations les { API } web en sont venues à incarner cette sociabilité. C'est l'étape d'instrumentalisation des { API } (IV. 3), où nous montrons comment la fonction d'{ interopérabilité } des { API } web est au cœur de multiples réappropriations qui participent à en redéfinir la fonction sociale et technique. Mais en augmentant la circulation des textes et leur rééditorialisation, les { API } web en augmentent également l'altération, la transformation. Ce qui pose, pour des industries comme Facebook ou Twitter, un problème d'image de marque. Comment alors contrôler ces transformations inhérentes à la circulation des textes ? Par la { documentation } des { API } web, qui fait de ces objets techniques un lieu d'{ industrialisation } de pratiques lettrées[102], c'est-à-dire de pratiques visant à contrôler l'image et le contenu du texte, malgré ses transformations et rééditorialiations (IV. 4).

99 Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 165-201.

100 Jeanneret, Yves. *Critique de la trivialité : les médiations de la communication, enjeu de pouvoir.* Paris : Éditions Non Standard, 2014.

101 Lordon, Frédéric. *Capitalisme, désir et servitude : Marx et Spinoza.* Paris : La Fabrique, 2013 [2010].

102 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 11-40.

API,
code,
code source,
développeur,
énonciation
computationnelle,
formes-textes

Voir glossaire tome II, p. 3-25

Nous arrivons ainsi au dernier niveau de la fonction éditoriale des { API } web, dans lequel nous mettons notamment en avant le fait que les { API } contrôlent la polyphonie inhérente à tout texte et que ce contrôle a tendance à invisibiliser l'action des machines qui produisent ce texte. Ce qui pose en dernière analyse **un problème d'ordre politique ou écologique**[103].

Les { API } web nous mettent en présence de formes particulières, générées par des machines au fonctionnement calculatoire. Mais le rôle de ces machines est, par un lourd travail éditorial, mis sous silence sémiotique au profit d'une circulation intense des { formes-textes }. D'où l'hypothèse générale suivante : dans les écrits d'écran se joue une invisibilisation des entités machiniques – parmi lesquelles les { API } web – permettant l'existence de ces textes et cette invisibilisation repose en grande partie sur les outils théoriques utilisés pour aborder ces textes. Il faut donc proposer une sémiotique non-anthropocentrée, une sémiotique qui permette de faire apparaître le rôle que jouent des entités non-humaines dans la production des écrits d'écran.

C'est l'enjeu de notre **chapitre V**, un enjeu qui marque un retrait épistémologique par rapport aux quatre autres. Nous y interrogeons les limites de nos concepts utilisés jusqu'ici (texte, écriture, { code }…), afin de comprendre quels en sont les effets sur la relation que nous tissons avec la présence machinique interrogée dans cette thèse. Pour cela, il faut commencer par reconstruire une altérité (V. 1), montrer en quoi un ordinateur est différent d'un humain tant en degrés qu'en nature. Ces deux entités se rencontrent malgré tout en un lieu : le { code source }, lisible par les humains et les machines. Mais quels rapports politiques se jouent dans ce texte hybride (V. 2) ? Quelle est l'« économie scripturaire[104] » du { code informatique } ? Elle se construit autour de deux imaginaires fondamentaux : l'imaginaire de la source (V. 3) et l'imaginaire de la commande (V. 4). Tous deux participent de l'idée que tout ce que nous voyons à l'écran est réductible à un seul texte (le { code source }), que ce texte est la consignation de la volonté d'êtres humains (les { développeurs }) et que l'ordinateur ne fait qu'exécuter cette volonté. Nous aurons alors identifié un imaginaire structurant de l'écriture informatique qui contribue à invisibiliser la part machinique des écrits d'écran.

103 Despret, Vinciane. *Au bonheur des morts : Récits de ceux qui restent.* Paris : Les Empêcheurs de penser en rond – La Découverte, 2015, p. 20.

104 De Certeau, Michel. *L'invention du quotidien. 1, Arts de faire.* Paris : Gallimard, 2010 [1990], p. 195-224.

Contre cet imaginaire, nous proposons enfin une méthode pour reconnaître et qualifier « l'{ énonciation computationnelle } » (V. 5), soit la part du texte lisible qui revient aux différentes entités calculatoires impliquées dans la production des écrits d'écran.

d'une sci
combina
de l'écrit
à une ind
du texte.

ence
toire
ure...
ustrie

API, code informatique, petites formes

Voir glossaire tome II, p. 3-25

Cette première partie s'appuie sur le constat, que les { API } – tant dans les formes qu'elles permettent de produire que dans leur organisation interne – ont une forte part combinatoire : elles structurent l'accès à des blocs de { code informatique }, elles produisent des { petites formes } combinables. Ce sont des outils d'écriture informatique reposant en partie sur des principes combinatoires. Ce constat est dicté par une observation globale de notre corpus.

À partir de là, nous formulons l'hypothèse – que nous prouverons dans le premier chapitre – que l'informatique en tant que science s'est construite sur un certain modèle d'écriture calculatoire et combinatoire. Cela nous permet de construire une autre hypothèse, d'ordre généalogique : les { API } seraient l'aboutissement industriel d'une écriture { formelle } et combinatoire qui traverse toute l'histoire de l'informatique (chapitre I) et qui mobilise des imaginaires de la science et de l'écriture (chapitre II).

Il s'agit donc d'interroger « […] l'historicité des { API }, pour montrer comment elles font partie du contexte technologique, économique et socio-politique actuel[105] ». Nous interrogeons cette historicité par la face combinatoire des { API }. Dans quelle mesure peut-on dire qu'une { API } est combinatoire ? À quel niveau ? Comment est utilisée cette dimension combinatoire ? À quoi sert-elle ? Quels imaginaires de l'écriture ou de l'informatique mobilise-t-elle ?

Nous ne prétendons pas pour autant retracer une filiation directe entre les { API } web contemporaines et – par exemple – la machine de Turing inventée en 1937 ou les usages divinatoires du *Yi King* en Chine. Nous cherchons plutôt à montrer que les { API }, par leur aspect combinatoire, reconduisent tout autant que redéfinissent un certain nombre de façons d'écrire. Plutôt qu'une ligne droite, on peut résumer notre démarche par un élargissement de cercles

105 « […] *to historicize APIs as part of the specific technological, economic, and socio-political constraints of the day* ». Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

API, formalisme, logiciel, programme, *subroutine*

Voir glossaire tome II, p. 3-25

concentriques : l'histoire des { API }, comprise dans l'histoire du { logiciel }, elle-même comprise dans l'histoire de l'informatique, elle-même comprise dans l'histoire de l'écriture. La combinatoire traverse chacun de ces cercles, non sans subir quelques substantielles modifications lors de ce parcours.

Le premier chapitre est consacré aux trois premiers cercles : l'histoire des { API }, l'histoire du { logiciel } et l'histoire de l'informatique. Nous y montrons comment la combinatoire, dans les { API }, est étroitement liée à l'idée de faciliter l'écriture des pages web en créant des modules de fonctionnalités. Cette modularité est présente dès les débuts de l'industrie du { logiciel }, comme solution à la structuration du marché de l'informatique. Modularité qui est l'héritière d'une certaine pratique d'écriture combinatoire – la { *subroutine* } – créée au milieu des années 1940 pour faciliter l'écriture des { programmes }. Remontant d'un cran, nous montrons qu'un des modèles théoriques de l'ordinateur, la machine décrite par Turing en 1937, possède déjà un fonctionnement combinatoire, par sa dimension logique et calculatoire. Cette combinatoire ne porte plus sur l'écriture des { programmes }, mais sur les opérations faites par la machine, qui sont des opérations de calcul, et donc d'un certain type d'écriture : une écriture { formelle }, chiffrée et combinatoire. C'est par le modèle proposé par Turing, aujourd'hui dominant, que l'histoire de l'informatique est liée à l'histoire de l'écriture.

Le second chapitre est consacré à ce dernier cercle, soit l'histoire de l'écriture. Nous repartons des caractéristiques de l'écriture de la machine de Turing démontrées à la fin du chapitre I : une écriture combinatoire, chiffrée et { formelle }. Nous montrons que la combinatoire est l'une des caractéristiques de tout système d'écriture, et en interrogeons les imaginaires propres. Le chiffre, en tant que signe particulier d'écriture, recèle lui aussi des imaginaires puissants. Enfin, nous montrons que le { formalisme } de la machine de Turing et sa qualification de machine « universelle » s'inscrit dans une histoire longue des sciences et de la quête d'une méthode universelle par des moyens combinatoires.

LES API DANS L'HISTOIRE DU LOGICIEL ET DE L'INFORMATIQUE

Dans ce premier chapitre, nous partons de la dimension combinatoire des { API } pour montrer comment elle s'inscrit dans une histoire de la { programmation }, de l'industrie du { logiciel } et de l'informatique. En quoi une { API } peut être dite combinatoire ? D'où vient cette part combinatoire et à quoi sert-elle selon les différents contextes où elle est mobilisée?

API ,
codage,
code,
logiciel,
programmation,
programme,
standardisation,
subroutine

Voir glossaire tome II, p. 3-25

L'hypothèse principale de ce chapitre est que les { API } sont des outils d'écriture qui valorisent des principes combinatoires pour des raisons économiques, pour faciliter la { programmation } et pour des raisons techniques.

Chacune de ces étapes correspond à autant d'étapes dans l'histoire de notre objet. Histoire dont nous préciserons dans un premier temps le cadre général (I. Introduction) avant de montrer que la part combinatoire des { API }web contemporaines permet de répondre aux besoins d'une industrie de l'écriture fondée sur des enjeux de plasticité et de { standardisation } (I. 1). Cette combinatoire fut initialement exploitée pour faciliter l'écriture des premiers { programmes } (I. 2). C'est ce que nous montrons en nous plaçant à la naissance de la { programmation } à la fin des années 1940, à travers l'exemple de la { *subroutine* }, soit le { codage } de certaines opérations routinières sous forme de blocs de { code } préécrits et mobilisables dans différents contextes. De ce point de

API, application, calcul, formalisme, programmation, *subroutine*

Voir glossaire tome II, p. 3-25

vue, la { *subroutine* } est l'élement de base d'une forme d'écriture informatique combinatoire. Mais le fonctionnement combinatoire en tant que tel tient plus fondamentalement à l'informatique elle-même. Nous remontons alors encore d'un cran, en sortant de la question de la { programmation } (l'écriture pour la machine) et en abordant les débuts de l'informatique (le fonctionnement de la machine). Grâce au commentaire d'un des textes fondateurs de l'informatique moderne : *On Computable Numbers, with an application to the Entscheidungsproblem*, publié en 1937 par Alan Turing, nous montrons que l'informatique en tant que science est fondée sur une certaine conception automatisée et machinique de l'écriture (I. 3).

INTRODUCTION
UNE INFORMATIQUE ENTRE LOGIQUE FORMELLE ET SCIENCES DE L'INGÉNIEUR

Avant de commencer ce chapitre, il convient de faire quelques mises au point afin d'éviter toute ambiguïté dans les pages qui vont suivre. Nous allons porter un regard historique sur nos objets : d'où viennent les { API }, en quoi sont-elles combinatoires et en quoi cet aspect combinatoire hérite de pratiques plus anciennes dans l'histoire de la { programmation } ?

L'ambiguïté est que l'informatique est en quelque sorte entre deux eaux : elle est, dans ses fondations théoriques, issue de la logique { formelle } ; elle est en tant qu'industrie et dans ses réalisations effectives – la mise au point de calculateurs automatiques – une branche de l'ingénierie. D'un côté, l'informatique est une affaire de logiciens et de mathématiciens qui ont pour préoccupation première de formaliser au mieux un { calcul } ; de l'autre et dans le même temps, elle est une affaire d'ingénieurs qui ont pour tâche de réaliser une machine capable d'effectuer des { calculs } à grande vitesse[105].

Cet équilibre entre logique { formelle }et sciences de l'ingénieur, entre logique et logistique, est un axe central de ce chapitre. Il a une incidence sur la séquence temporelle retenue. Si l'on insiste sur la dimension ingénieuriale, un ordinateur est une machine à calculer qui tient de l'histoire longue de ces techniques[106], voire d'une anthropologie du calcul et de l'écriture[107] (ce qui sera l'objet de notre second chapitre). Si on insiste sur la dimension logique, l'informatique naît avec les travaux en logique { formelle } des années 1930 et rejoint ensuite le courant de l'histoire des techniques de calcul pendant la seconde guerre mondiale[108]. L'histoire de l'informatique s'étend alors de 1930 à aujourd'hui.

105 Tedre, Matti. *The science of computing: shaping a discipline*. Boca Raton : CRC Press, 2015.

106 Williams, Michael R. *A History of Computing Technology*. Englewood Cliffs : Prentice-Hall, 1985 ; Knuth, Donald E. *Éléments pour une histoire de l'informatique*. Paris : Société mathématique de France, 2011. Michael Williams donne l'exemple le plus frappant de ce genre d'historiographie : l'histoire des ordinateurs remonte à l'ingénieur anglais Charles Babbage et sa « machine analytique », à la Pascaline, aux logarithmes de Neper et plus anciennement aux abaques grecques. Il s'agit d'une histoire des outils de calcul au sens très large qui inclut toutes les inventions ayant permis aux hommes de faciliter le maniement des nombres.

107 Herrenschmidt, Clarisse. *Les trois écritures : langue, nombre, code*. Paris : Gallimard, 2007 ; Ramunni, Girolamo. *La physique du calcul : histoire de l'ordinateur*. Paris : Hachette, 1989.

108 Campbell-Kelly, Martin et Aspray, William. *Computer: a history of the information machine*. New York : Basic Books, 1996 ; Ceruzzi, Paul E. *A History of Modern Computing*. Cambridge : MIT Press, 2003 [1998].

API formalisme

Voir glossaire tome II, p. 3-25

C'est cette dernière séquence chronologique que nous choisissons pour ce chapitre : des premiers travaux d'Alan Turing et d'Emil Post à la fin des années 1930 jusqu'aux { API } contemporaines. Cette période, particulièrement les années 1930 – 1960, a été principalement analysée sous l'angle d'une histoire sociale des techniques[109], que ce soit une histoire globale[110], ou centrée sur des aspects spécifiques : l'invisibilisation progressive des femmes dans l'informatique[111], l'apparition de nouvelles expertises professionnelles[112]…

Notre focale sur l'écriture nous pousse à écrire une histoire un peu différente, qui se concentre sur la façon dont l'informatique hérite et généralise un certain type d'écriture calculatoire venu de la logique { formelle }. Cela met l'accent sur les auteurs qui ont mis en avant la question de l'écriture, en premier lieu Alan Turing puis John Von Neumann. Ce faisant, nous reconduisons une façon d'écrire l'histoire de l'informatique qui se concentre sur certaines figures tutélaires, au dépens d'autres personnalités et en gommant certains modèles alternatifs. En résumé, l'histoire « officielle » de l'informatique fait de l'axe Alan Turing – John Von Neumann – Norbert Wiener les grands inventeurs de cette discipline[113], négligeant par là d'autres acteurs de cette histoire. Les spécialistes actuels montrent que cette façon d'écrire l'histoire de l'informatique doit être nuancée[114] et nous tiendrons compte de leurs travaux. En revanche, notre but n'est pas d'écrire une contre-histoire de l'informatique, ou d'en défendre une certaine vision. Notre but est de comprendre en quoi les { API } sont des outils d'écriture combinatoire. Notre hypothèse est que l'informatique, en tant que science, est construite sur un modèle d'écriture fortement combinatoire dont les { API } héritent. Nous nous intéressons donc au modèle dominant et d'une manière qui met l'accent sur la question de

109 Ce qui n'est pas sans causer certaines polémiques sur la façon d'écrire cette histoire, notamment quant au savoir informatique nécessaire pour comprendre et prendre en considération les innovations strictement techniques – et non seulement les logiques sociales autour de ces innovations. Voir Knuth, Donald E. *2014 Kailath Lecture: Let's Not Dumb Down the History of Computing Science* [captation vidéo en ligne]. Conférence donnée à l'université de Stanford le 7 mai 2014. [Mis en ligne le 30 mai 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://kailathlecture.stanford.edu/2014KailathLecture.html.

110 Ceruzzi, Paul E. *Op. cit.*

111 Hicks, Marie. *Programmed inequality: how Britain discarded women technologists and lost its edge in computing*. Cambridge : MIT Press, 2017.

112 Ensmenger, Nathan. *The computer boys take over: computers, programmers, and the politics of technical expertise*. Cambridge : MIT Press, 2010.

113 Randell, Brian. *On Alan Turing and the origins of digital computers*. Newcastle : University of Newcastle, 1972 ; Aspray, William. *John Von Neumann and the Origins of Modern Computing*. Cambridge : MIT Press, 1990.

114 Sieg, Wilfried, Szabó, Máté et Mclaughlin, Dawn. Why Post Did [Not] Have Turing's Thesis. Dans : Omodeo, Eugenio G. et Policriti, Alberto (dir.), *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*. Berlin : Springer International Publishing, 2016, p. 175-208 ; De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Journal of Logic and Computation*. 2015, vol. 25, n° 4, p. 1011-1046 ; Bullynck, Maarten, Daylight, Edgar G. et De Mol, Liesbeth. Why did computer science make a hero out of Turing? *Communications of the ACM*. 2015, vol. 58, n° 3, p. 37-39.

l'écriture. Cette problématisation nous amène à des auteurs (Von Neumann, Turing) qui non seulement réservent une grande place à la notion d'écriture, mais dont les modèles se sont aussi progressivement imposés comme les modèles dominants en informatique.

Ces clarifications sont nécessaires car il est essentiel pour comprendre cette thèse de bien comprendre le modèle que nous décrivons dans ce premier chapitre. Nous y montrerons en effet que la combinatoire tient une place prépondérante dans les { API } et dans l'écriture informatique, ce qui nous servira ensuite à en analyser les imaginaires structurants (chapitre II). Dans la seconde partie, nous nous demanderons comment ce modèle d'écriture s'industrialise et transforme les textes de réseau (chapitre III), nous interrogerons l'idéologie de la circulation qu'il sert (chapitre IV) et les relations qu'il nous permet de construire avec les ordinateurs (chapitre V). À ce titre, le chapitre qui suit est autant une façon d'historiciser notre objet – les { API } – que de poser les fondements théoriques des chapitres à suivre.

1 LES API : UNE ÉCRITURE COMBINATOIRE AU SERVICE D'UNE ÉCONOMIE DE L'ÉCRITURE LOGICIELLE

Nous avons jusqu'ici pris comme point de départ que les { API } sont des outils d'écriture combinatoire. Il s'agit maintenant de préciser cette idée, en commençant par la définition de ce qu'est la combinatoire (I. 1. A). Une { API } est un outil d'écriture combinatoire, au sens où elle met à disposition des internautes des modules élémentaires qui servent à écrire une page web ou une { application } (I. 1. B). La combinatoire est alors envisagée comme combinaison de modules standards, plus ou moins adaptables selon les contextes et la { littératie } de l'écrivant, ce que nous montrerons avec l'exemple des { *widgets* } (I. 1. C). Cette plasticité est plus largement une solution à un marché de l'informatique structuré auour de la différence entre le { matériel } (*hardware*) et le { logiciel } (*software*) (I. 1. D) et intègre à partir des années 1970 certains modèles de { programmation } (I. 1. E).

API, application, littératie, logiciel, matériel, programmation, *widget*

Voir glossaire tome II, p. 3-25

1 A La combinatoire, une première définition

La question centrale qui nous occupe est de montrer en quoi les interfaces de programmation web sont des outils d'écriture à dimension combinatoire. Il faut pour cela définir ce que nous appelons combinatoire, autour de la notion de finitude (I. 1. A. a), définition qui se concentre sur la mécanique d'un système combinatoire plutôt que sur ses mises en pratique dans un contexte culturel et technique précis (I. 1. A. b). Cette définition large permettra dans la partie suivante de bien comprendre en quoi les { API } sont une certaine mise en application de principes combinatoires.

API

Voir glossaire tome II, p. 3-25

a Combinatoire et finitude

Par combinatoire, nous entendons un système réglé de permutations entre un nombre fini d'éléments. Par exemple, une boîte de Lego® est un système combinatoire. Il y a un certain nombre de briques élémentaires – qui ne peuvent être divisées en plus petites briques – et qui ne peuvent s'assembler qu'à condition qu'elles aient chacune au moins une zone de libre. C'est leur règle de permutation. Le nombre de briques, ainsi que leurs zones exploitables, définissent le nombre de combinaisons possibles. De la même façon, on peut dire du langage qu'il est une structure combinatoire, au moins partiellement. Notre alphabet est la liste des éléments combinables, tandis que la grammaire et l'orthographe donne la liste des permutations autorisées[115].

Pourquoi le langage est-il un système partiellement combinatoire ? Du point de vue discursif tout d'abord. Certains mots (entendre certaines combinaisons) n'ont pas de sens et ne sont donc pas autorisés. Mais cela tient surtout au nombre fini de permutations d'un système combinatoire. Le langage autorise à répéter une lettre *ad indefinitum*. Ce faisant, il y a un nombre de mots tendant vers l'infini, ne serait-ce qu'avec la lettre A – si par exemple elle est répétée x fois. Il est donc essentiel, pour définir un strict système combinatoire, d'avoir un ensemble de règles qui organise les permutations et permet d'en déterminer le nombre total. Si par exemple on postule que dans un système combinatoire à quatre éléments de base (A, B, C, D), il ne peut y avoir de redoublement d'éléments (la combinaison

115 Ce qui laisse de côté un nombre considérable de combinaisons.

AAAA, BB ou CCC est impossible) et que ce système est commutatif[116], alors cela restreint fortement le nombre de combinaisons. La règle que n'applique pas le langage et qui fait que le nombre de combinaisons tend vers l'infini, c'est que le nombre maximal d'éléments utilisables dans une combinaison n'est pas défini. Par exemple, si nous voulons trouver toutes les combinaisons possibles de A,B, C et D, il faut déterminer en amont si on autorise des combinaisons de plus de quatre lettres et jusqu'à quelle limite – six, sept, huit lettres, etc. Ce n'est qu'une fois défini ce nombre fini qu'on peut calculer le nombre total de combinaisons[117].

b Une conception mécaniste de la combinatoire

Pourquoi est-ce important d'avoir un nombre fini de combinaisons, un nombre fini d'éléments de base et un nombre fini déterminant la plus longue combinaison possible ? C'est à ces conditions qu'on peut savoir – par le calcul – qu'il existe un nombre donné de combinaisons possibles, même si ce nombre est colossal et difficilement appréhendable. Il n'y a pas besoin d'effectivement noter ou calculer toutes les combinaisons possibles. On peut déterminer en amont le nombre de combinaisons possibles, au moins de façon virtuelle, c'est-à-dire sans actualiser ces possibles par le calcul. Toute la force de la combinatoire réside dans ce rapport entre une grande économie de moyens – quelques règles et éléments de base – et un grand nombre possible de résultats. Il suffit d'appliquer méthodiquement deux ou trois principes pour arriver à épuiser toutes les combinaisons possibles, ce qui revient à accéder à une totalité finie. Cette définition fait donc la part belle à une conception mécaniste de la combinatoire.

Cette conception mécaniste permet d'interroger au mieux les imaginaires de la combinatoire, les discours totalisants qu'elle sert et justifie. Ce qui nous intéresse, c'est sur quelles bases fonctionnent un système combinatoire et ensuite dans quel contexte historique, philosophique, technique, voire esthétique il est utilisé. Si nous reprenons nos deux exemples ci-dessus, on peut supposer que dans le cas des Lego®, le nombre de combinaisons possibles soutient un discours sur la créativité. On peut, avec une boite de Lego®, créer une foule de choses : il suffit de faire fonctionner son

116 Le principe de commutativité pose l'équivalence des combinaisons utilisant les mêmes éléments mais dans un ordre différent. Le langage, à cet égard, est non-commutatif : le mot « chien » n'est pas le même que le mot « niche », même s'il est composé des mêmes éléments de base. L'anagramme est un exemple de pratique du langage qui en exploite le principe de non-commutativité.

117 Le nombre de combinaisons possibles pour un ensemble fini n, commutatif et sans redoublements, est de 2^n-1. En l'occurrence, il y 15 combinaisons possibles : 2^4-1 = (2x2x2x2)-1 = 16-1=15. Ce calcul présuppose qu'il ne peut y avoir de combinaison plus longue que n.

imagination – et quelques principes architecturaux et combinatoires. De même, dans le cas du langage, il est certain que certaines combinaisons n'ont aucun sens et trouver toutes les combinaisons possibles est un travail de Sisyphe[118]. En revanche, s'imaginer que le langage fonctionne comme un système combinatoire et qu'on peut donc potentiellement « l'épuiser », est une idée opérante qui a inspiré de nombreux travaux[119].

118 Précisément parce que le langage n'est pas un système combinatoire à strictement parler, puisque le nombre de combinaisons n'est pas fini.

119 Un des exemples les plus fameux est le *Cent Mille Milliards de Poèmes* de Raymond Queneau ou les jeux mathématico-langagiers de l'OuLiPo. Ce sont des exemples où la combinatoire nourrit une poétique : c'est une machine à générer des textes. Quelques siècles auparavant, cette fois dans une perspective mathematico-théologique, Leibniz avait calculé le nombre total de livres que l'on peut écrire – ce qu'il assimile à tout ce qui peut être dit – avec un alphabet de 24 lettres. On peut alors calculer ce qu'il nomme la « Restitution Universelle » (l'*Apocatastasis*) : le moment où l'histoire ne sera plus que répétition du déjà-dit. Voir Leibniz, Gottfried W. *De l'horizon de la doctrine humaine 1693 ; La restitution universelle, 1715 = Apokatastasis pantōn*. Paris : Vrin, 1991.

1 B L'API web comme outil d'écriture combinatoire

API, code, développeurs, HTML, JavaScript,< objet, programme, tweet, *widgets*

Voir glossaire tome II, p. 3-25

Sur ces bases, en quoi une { API } web est-elle un outil d'écriture combinatoire ? Parce qu'elle permet de composer une page ou un { programme } à partir de blocs combinables afin de faciliter l'écriture de la page ou du { programme } (I. 1. B. a). Cela sera montré à travers le cas des *mashups* et de la curation, deux pratiques dans lesquelles les { API } et leur fonctionnement combinatoire ont une place prépondérante (I. 1. B. b)

a Une combinatoire au service d'une économie de l'écriture

Une { API } permet fondamentalement deux opérations : récupérer auprès d'un serveur certaines données d'une plateforme comme Facebook ou Twitter ; récupérer des extraits de { code } { HTML } ou { JavaScript } qui permettent d'intégrer dans une page web certaines formes précodées : { tweets }, statut, cartes… ce que ces plateformes nomment des { *widgets* }. Dans les deux cas, une { API } est la mise à disposition d'« […] un jeu de fonctions pré-écrites, souvent standards, que les développeurs peuvent utiliser[120] », pour écrire par exemple un { programme }, une page web ou n'importe quel autre { objet } informatique. Cela revient donc à insérer dans un projet d'écriture – un { programme } ou une page web[121] – des éléments exogènes qui y remplissent une fonction bien précise. L'idée principale, pour un { développeur }, est soit de disposer de données auxquelles il ne pourrait avoir accès lui-même – ou alors au prix d'un travail laborieux[122] ; soit de gagner du temps en récupérant auprès d'une { API } des blocs de { code } tout faits qu'il peut réutiliser en les modifiant légèrement pour les adapter à son { programme } ou sa page. C'est en cela qu'une { API } est un outil d'écriture et de surcroît un outil d'écriture

120 « […] *a set of prewritten, often standardized, functions that can be used by developers* ». Tunstall, Michael, Markantonakis, Konstantinos, Sauveron, Damien, *et al.* Smart Cards: Communication Protocols and Applications. Dans : Bidgoli, Hossein (dir.), *Handbook of Computer Networks. Volume 3 : Distributed networks, Network planning, Control, Management, and New Trends and Applications*. Hoboken : John Wiley & Sons, 2008, p. 266.

121 Un programme n'est pas une page web , même si cette dernière est représentée sous forme de langage HTML. Une page web est le résultat de l'interprétation d'un code HTML, lequel est un langage déclaratif, c'est-à-dire qu'y est décrit la structure documentaire d'une page. Un programme en revanche n'est pas uniquement déclaratif. Il y a un certain nombres d'opérations logiques automatisées (si ceci, alors fait cela, crée telle entité, etc.), qui sont présentes dans le Web via des scripts, mais qui ne sont pas prises en charge directement par le code HTML. De plus, l'ordinateur ne calcule pas un programme comme il calcule une page web . Le code d'un programme doit d'abord être compilé, puis interprété, alors qu'une page web – scripts compris – est interprétée à la volée, sans en passer par une compilation préalable.

122 Typiquement, la liste de contacts Facebook d'une personne par exemple. Il est plus simple de récupérer ces informations en faisant une requête API, plutôt que de concevoir un programme qui fasse le même travail, voire de copier manuellement la liste de contacts.

combinatoire : elle permet d'accéder à des « briques de construction[123] » préécrites de données ou de fonctionnalités qui ont vocation à être réutilisées dans différents contextes d'écritures. Il s'agit donc bien d'une forme combinatoire – sous la forme d'un agencement de différents modules – qui permet une économie de l'écriture informatique. Nous entendons pour l'instant par économie un gain de temps et une commodité d'écriture : isoler des blocs de fonctionnalités réutilisables permet d'aller plus vite pour écrire un { programme } ou une page web. Les { API } sont donc des outils d'écriture qui facilitent la { programmation } informatique.

API, application, architexte, logiciel, programmation, programme, Web

Voir glossaire tome II, p. 3-25

b *Mashup* et curation : deux exemples des liens entre API et écriture combinatoire

Cette dimension combinatoire des { API } web se remarque sûrement le mieux dans la façon dont ces outils ont configuré et accompagné certaines pratiques éditoriales à partir des années 2000. Dans le cadre de ce que Laurence Allard a nommé « l'expressivisme digital[124] », on a en effet vu fleurir de nombreuses formes textuelles comme les *mashups* et les outils web de curation. Dans les deux cas, les { API } jouent un rôle central.

Le *mashup* — de l'anglais purée ou compote – est une façon de composer par assemblage, mélange. C'est une forme ancienne de composition mais qui a pris dans les années 1960 et 1970 une tournure politique par les détournements situationnistes, puis est devenue largement répandue avec le développement du { Web }. Qu'on pense aux remix, aux images détournées, modifiées, aux textes composés de plusieurs sources rassemblées… Loin de ne concerner que les écrits ordinaires de réseau, ce mode par composition concerne aussi les pratiques de { programmation }. On parle de *mashup* dans le cas d'une { application } qui mélangerait plusieurs fonctionnalités tirées d'autres { applications }.

La curation est un terme issu du monde muséal et juridique, désignant sur le { Web } le fait de rassembler des documents épars jugés pertinents quant à un sujet ou un thème donné. Un site comme Storify par exemple permet d'écrire un « récit » composé d'extraits de sites web agencés selon un ordre voulu. À la différence du *mashup*, le texte ainsi produit n'est pas une synthèse nouvelle et inédite, mais une collection de déjà-dit et de déjà-publié.

123 « *building blocks* ». Article « Application Programming Interface (API) ». Dans : Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New York : Facts On File, 2009 [2003], p. 20.

124 Allard, Laurence. L'impossible politique des communautés à l'âge de l'expressivisme digital. *Op. cit.*, p. 105-126.

Ces deux exemples montrent la façon dont les { API } web utilisent des principes combinatoires et cadrent dans ce sens les formes d'écriture qu'elles permettent. Les *mashups* autant que les { logiciels } de curation reposent en effet techniquement sur l'utilisation d'{ API }. Dans le cas des *mashups*, il serait impossible de composer une telle forme sans accès structuré aux unités mélangées. Si je veux par exemple créer une { application } qui mélange les horaires de bus de la RATP et un fond de carte, je dois avoir accès à certains jeux de données de la RATP et de Google Maps ou autre système cartographique en ligne. Je dois donc faire appel à l'{ API } pour cibler quelles données vont être les plus utiles par rapport au { programme } que je veux écrire. Ce faisant, je mets en œuvre un type d'écriture combinatoire, en ceci que j'actualise une combinaison possible entre deux jeux de données, combinaison rendue possible par la mise en place d'une { API }, c'est-à-dire d'un accès structuré à ces données. Les { API } ont été dans ce cadre un facteur essentiel du développement technique de cette forme culturelle qu'est le *mashup*[125]. Parce qu'elles permettent l'accès à des briques élémentaires, elles outillent ces pratiques créatives et, en tant qu'{ architextes[126] }, les cadrent vers un modèle combinatoire d'écriture.

Quant à la curation, là encore les { API } jouent un rôle crucial. Prenons l'exemple de Storify, site web emblématique de cette forme d'écriture :

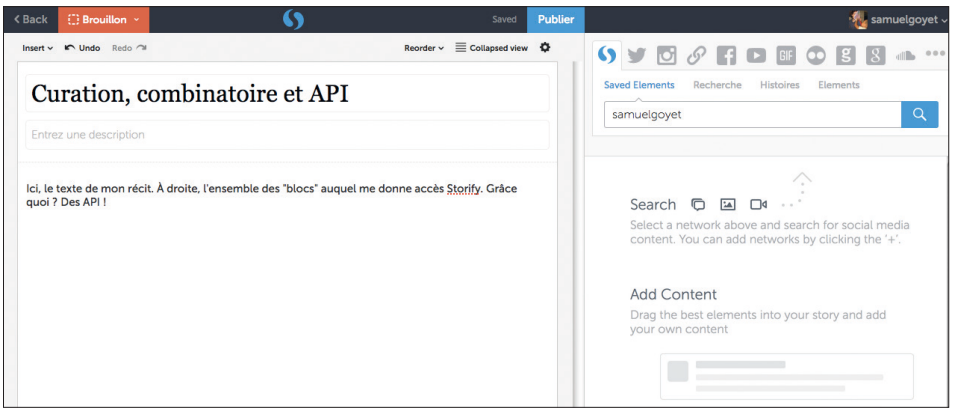


Fig.1. Le site de curation Storify : l'exemple d'une écriture combinatoire permise par des API. Capture d'écran du site https://editor.storify.com/ (profil personnel, détail). Capture réalisée le 9 février 2017.

125 Benslimane, Djamal, Dustdar, Schahram et Sheth, Amit P. Services Mashups: The New Generation of Web Applications. *Op. cit.*, p. 13-15.

126 Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Xoana*. 1999, nº 6, p. 97-107.

Dans la colonne de gauche, le texte composé par l'utilisateur. Dans la colonne de droite, une barre de recherche (cadre avec les mots « samuelgoyet » inscrits par défaut assorti d'un bouton bleu avec une icône de loupe) permet de faire une recherche parmi de multiples sites (suite d'icônes dans l'onglet supérieur) : Storify bien sûr, mais aussi Twitter, Instagram, Facebook, Google, YouTube, Soundcloud… Les résultats de ces recherches peuvent être glissés/déposés dans la colonne de gauche, afin de composer le « récit » publié sur Storify. Ces résultats sont en quelque sorte les briques élémentaires du texte à venir, les unités qui font l'objet d'une curation. Mais comment, depuis le site de Storify, peut-on rechercher et manipuler des textes provenant d'autres sites ? Grâce précisément à des { API }, qui permettent l'accès aux données de Google, Twitter, Soundcloud, etc. Dans ce cas précis, les { API } déterminent en quelque sorte l'étendue des briques mobilisables dans la curation, le nombre de blocs différents auquel l'utilisateur aura accès.

API, standardisation, *widget*

Voir glossaire tome II, p. 3-25

Dans les deux cas – les *mashups* et la curation – les { API } sont essentielles tant techniquement que sémiotiquement et chaque pratique explore un aspect différent de la combinatoire promue par ces outils. Dans le cas des *mashups*, les { API } permettent d'actualiser des combinaisons possibles, dans le cas de la curation, les { API } fournissent un réservoir de blocs qui permet ensuite le jeu combinatoire.

Par cette dimension combinatoire et la manière dont elle intègre des pratiques, les { API } rentrent donc dans une dynamique plus générale des médias informatisés : celle du plastigramme. Le plastigramme est cette « forme à la fois pérenne et plastique, qui comporte une mémoire mais procède d'une transformation réglée, produit ses effets à la jointure entre l'espace des auctorialités et celui des formes de l'expression[127] ». C'est une forme propre aux médias informatisés en ceci que « les formes écrites engendrées par l'informatique sont fondamentalement (et non occasionnellement) destinées à se propager dans leur propre réécriture[128] ». Les { API } sont des plastigrammes, car ce sont des outils d'écriture toujours pris dans une tension entre { standardisation } et plasticité. Toute la difficulté est en effet de proposer des blocs qui soient à la fois suffisamment standards, pour qu'il puissent être utilisés dans de nombreux contextes, mais aussi suffisamment « plastiques » pour qu'ils puissent s'adapter à ces contextes.

Comment cette tension entre { standardisation } et plasticité se joue dans notre corpus ? Nous avons entrevu le rôle de la combinatoire dans les *mashups* et dans la curation. Mais quelle forme prend cette combinatoire d'éléments dans le cas des { *widgets* } de Facebook et Twitter ? Et dans quel sens les { API } de ces services reconfigurent cette combinatoire ?

127 Tardy, Cécile et Jeanneret, Yves (dir.). *L'écriture des médias informatisés : espaces de pratiques*. Paris : Hermès Science – Lavoisier, 2007, p. 212.

128 Tardy, Cécile et Jeanneret, Yves (dir.). *Idem*, p. 210.

1 C Les *widgets* : une combinatoire qui ouvre à la question de la personnalisation et de la littératie

API, architexte, code, documentation, HTML, littératie, petites formes, tweet, *widget*

Voir glossaire tome II, p. 3-25

Le second exemple des liens entre API web et écriture combinatoire sont les { *widgets* }. Ce sont des { petites formes } très présentes sur les écrits de réseau contemporains, dont la production est réglée par les { API }. Facebook et Twitter, grâce à leurs { API }, donnent accès à des blocs de { code } préécrits qui permettent de produire ces { petites formes }. En étudiant la façon dont Facebook et Twitter présentent ces { *widgets* }, on peut mettre en avant une logique liée à la combinatoire : la personnalisation de blocs standardisés (I. 1. C. a). Le degré de personnalisation possible ouvre alors le champ à des utilisations différenciées selon la { littératie } propre au scripteur (I. 1. C. b). Notre analyse montre que les { API } prévoient, dans la { documentation }, une multitude de niveaux de { littératie }, en fonction des manipulations plus ou moins fines du { *widget* }.

a Standardisation et personnalisation des blocs combinables

Facebook tout comme Twitter mettent en avant, dans la { documentation }, les possibilités de personnalisation de ces formes[129]. Ces possibilités sont limitées. Elles portent principalement sur deux aspects : l'aspect du { *widget* } et le texte qu'il affiche. Facebook tout comme Twitter proposent par exemple d'en adapter la taille, la couleur des liens qui y apparaissent, le « thème graphique » – bien souvent limité à « sombre » (*dark*) ou « clair » (*light*). Les boutons « *Like* » ou « Partager » peuvent également être légèrement adaptés : on peut choisir de n'afficher que l'icône du bouton, le compte des clics enregistrés sur ce bouton, ou bien encore le texte accompagnant le bouton[130]. Le deuxième type de manipulations possibles porte sur le contenu du texte affiché. Il est surtout visible dans l'{ API } de Twitter. Au moment où l'on insère par exemple un { tweet } dans une page web, on peut choisir de faire apparaître ou non l'image qui peut accompagner le { tweet } ou encore les réponses à ce { tweet }. Il en va de même pour les modules qui permettent d'afficher tous les { tweets } d'un utilisateur : en plus des modifications de l'aspect

129 Voir annexe n° 1.

130 Voir annexe n° 2.

du texte, on peut ne pas faire apparaître les réponses à d'autres { tweets } ou décider ne pas faire apparaître automatiquement les photos ou vidéos postées dans certains { tweets }. Ces personnalisations sont assez limitées mais montrent d'emblée qu'elles sont de l'ordre de pratiques lettrées, c'est-à-dire d'un « double contrôle » : contrôle « sur la forme et la lettre, visant à construire la lisibilité du texte […] » ; « contrôle sur les contenus, en référence à des normes culturelles […][131] ». Contrôle sur l'image du texte - taille du bouton, couleur des liens – et sur son contenu – afficher les images, les réponses… Les { *widgets* }, par leur dimension plastique, intègrent donc des rapports savants au texte de réseau. En étant intégrées au cœur des outils, ces pratiques sont déléguées aux utilisateurs de ces { architextes }, ce qui pose la question de leur culture du texte et de l'écriture - ou { littératie }.

b Personnalisation et littératie

Là encore, les { architextes } guident les pratiques. Facebook tout comme Twitter proposent deux manières d'exercer ces pratiques lettrées. Soit par un système de cases que l'on peut cocher ou décocher selon ce que l'on veut voir apparaître, soit par une manipulation du { code } { HTML } (voir captures ci-dessous)

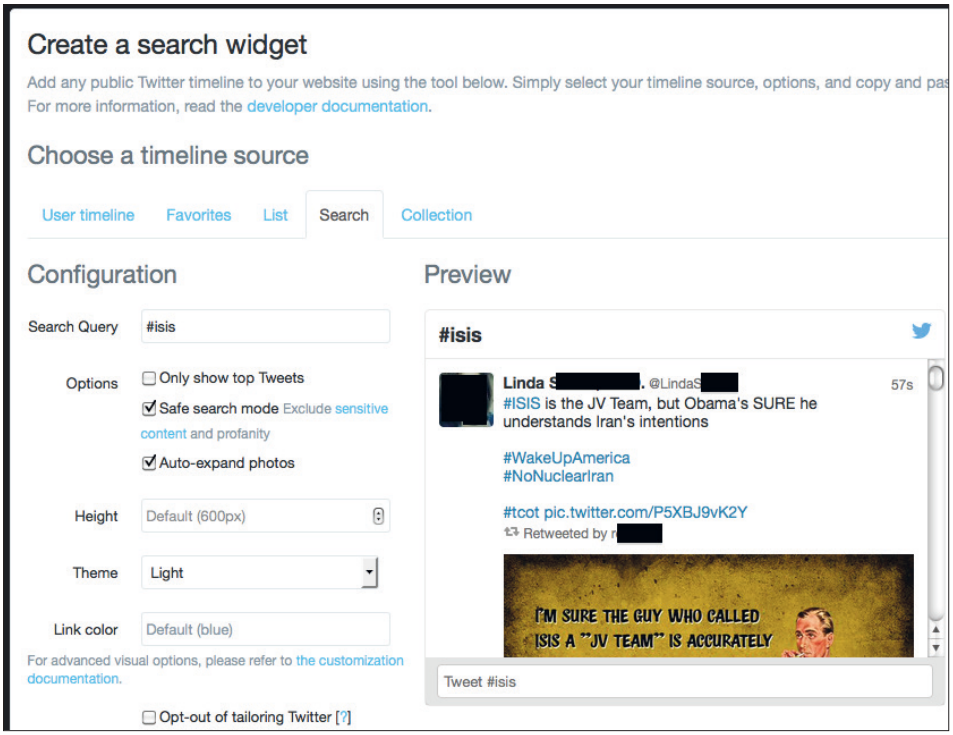
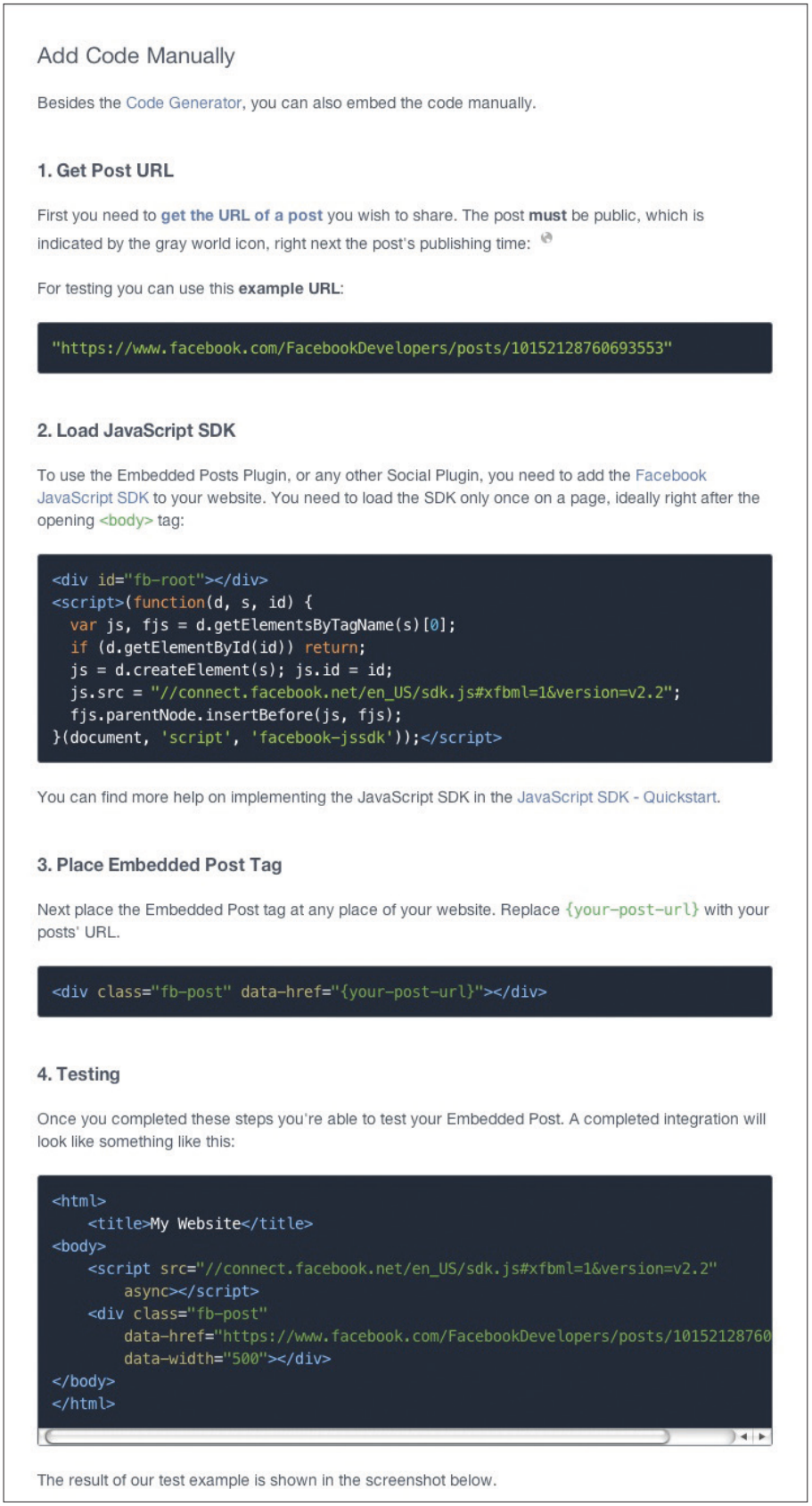


Fig.2. Exemples de personnalisation d'un *widget* par des cases cochables. Captures d'écran du site dev.twitter.com et developers.facebook.com (détails). Captures réalisées les 15 et 29 juillet 2015.

131 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 12.



Add Code Manually

Besides the Code Generator, you can also embed the code manually.

1. Get Post URL

First you need to **get the URL of a post** you wish to share. The post **must** be public, which is indicated by the gray world icon, right next the post's publishing time:

For testing you can use this **example URL**:

```
"https://www.facebook.com/FacebookDevelopers/posts/10152128760693553"
```

2. Load JavaScript SDK

To use the Embedded Posts Plugin, or any other Social Plugin, you need to add the Facebook JavaScript SDK to your website. You need to load the SDK only once on a page, ideally right after the opening <body> tag:

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.2";
  fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));</script>
```

You can find more help on implementing the JavaScript SDK in the JavaScript SDK - Quickstart.

3. Place Embedded Post Tag

Next place the Embedded Post tag at any place of your website. Replace {your-post-url} with your posts' URL.

```
<div class="fb-post" data-href="{your-post-url}"></div>
```

4. Testing

Once you completed these steps you're able to test your Embedded Post. A completed integration will look like something like this:

```
<html>
    <title>My Website</title>
<body>
    <script src="//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.2"
        async></script>
    <div class="fb-post"
        data-href="https://www.facebook.com/FacebookDevelopers/posts/10152128760
        data-width="500"></div>
</body>
</html>
```

The result of our test example is shown in the screenshot below.

Fig.3. Exemples de personnalisation d'un *widget* via le code HTML. Capture d'écran du site developers.facebook.com/docs/plugins/embedded-posts (détails). Capture réalisée le 29 juillet 2015.

Les deux méthodes produisent globalement les mêmes effets visuels, mais les deux ne supposent pas la même { littératie }. La première, celle des cases à cocher, ne nécessite pas de connaissance précise du { code } { HTML }. Les pratiques lettrées permises par l'{ architexte } sont résumées à des solutions binaires oui / non sans qu'il ne soit jamais requis d'écrire une seule ligne de { code }. Le { code } est généré automatiquement par Twitter pour correspondre aux choix faits par le créateur du { *widget* }. Il s'agit bien d'une forme d'écriture, mais d'une écriture qui passe – encore – par une couche architextuelle qui traduit les cases cochées en ligne de { code }. La seconde manière d'opérer, c'est d'écrire le { *widget* } directement sous sa forme { HTML }. Facebook qualifie cette façon de faire de « manuelle » : *add code manually* (« ajouter du code manuellement[132] »). Cette façon est un peu plus laborieuse et la procédure est dûment détaillée par Facebook, ainsi que la syntaxe { HTML } de tous les paramètres sur lesquels le { développeur } peut intervenir. La principale différence avec le premier cas est que les paramètres peuvent être réglés plus finement. Le { *widget* } peut donc être adapté plus spécifiquement à la page qui l'accueille. Cela nécessite en revanche une plus grande connaissance de la { programmation }. L'adaptabilité des formes générées par une { API } se paye donc en retour par la nécessité d'une plus grande { littératie } informatique.

API, architexte, code, développeur, HTML, littératie, programmation, *widget*

Voir glossaire tome II, p. 3-25

132 C'est l'occasion de souligner la valeur donnée à l'écriture du code – ici l'HTML – comme façon plus « directe » d'écrire une page web , sans en passer par une médiation architextuelle. Ce qui est techniquement faux mais symboliquement d'importance, comme le montrera le dernier chapitre de cette thèse.

1 D La combinatoire comme réponse à un marché structuré entre *software* et *hardware*

API, codage, littératie, logiciel, matériel, objet, programme, standardisation, *widget*

Voir glossaire tome II, p. 3-25

Si les API web sont bien des outils d'écriture combinatoire, le cas des { *widgets* } nous a déplacé vers la question de l'adaptabilité des { objets } auxquels donne accès l'{ API }. Cet accès n'est pas uniforme mais permet des manipulations différentes selon la { littératie } de l'écrivant.

La combinatoire amène donc à cette tension entre plasticité et { standardisation } des éléments combinés, qui prend dans le cas des { API } web la forme de la personnalisation de { *widgets* }. S'agit-il là d'une spécificité propre aux { API } contemporaines, ou est-ce que cette tension n'est pas fondamentalement liée à l'industrie du { logiciel } ? Notre approche communicationnelle et non seulement technique ou sémiotique, ainsi que notre objet (l'écriture) nous poussent à répondre à cette question en allant voir du côté de l'histoire du { logiciel }. Parce que nous avons affaire à des outils d'écriture (les { API } web) qui ont été industrialisés, équipant des pratiques et produisant certaines formes caractéristiques, on ne peut comprendre leur aspect combinatoire sans remonter d'un cran et les placer dans une histoire du { logiciel } et plus spécifiquement dans une histoire de l'exploitation industrielle de l'écriture de { programmes }. Comme le soutient en effet l'historien de l'informatique Edward Steinmueller, on peut comprendre le développement du marché du { logiciel } comme « [...] une tension établie de longue date entre la standardisation et la personnalisation des applications[133] ». D'où vient cette tension (I. 1. D. a) ? En quoi est-elle liée à des impératifs techniques, en quoi structure-t-elle des modèles économiques (I. 1. D. b) où l'écriture combinatoire joue un rôle essentiel (I. 1. D. c) ?

133 « [...] *the long-established tension between standardisation and customization in application software* ». Steinmueller, Edward W. The International Software Industry : An Analysis and Interpretive History. Dans : Mowery, David C (dir.), *The international computer software industry: a comparative study of industry evolution and structure*. Oxford : Oxford University Press, 1996, p. 44.

a La différence entre matériel et logiciel…

La tension entre { standardisation } et plasticité remonte à un principe de base de l'informatique : la différence entre le { matériel } (*hardware*) et le { logiciel } (*software*), où le { matériel } est une machine « générale » quand le { logiciel } est « […] le jeu d'instructions qui ordonne à l'ordinateur de faire une tâche spécifique[134] ». La machine est générale au sens où elle peut imiter le comportement d'une autre machine si tant est que ce comportement puisse être représenté sous la forme d'une suite d'instructions, c'est-à-dire d'un { programme }. Ce qui pose toute la question de la portabilité d'un { logiciel }. Si – en théorie – tout ordinateur peut imiter n'importe quelle autre machine par le biais d'un { logiciel }, dans quelle mesure ce { logiciel } fonctionne t-il sur d'autres machines ? Est-il interopérable ou au contraire lié à une machine en particulier (*machine-specific*) ?

Aujourd'hui, cette question nous semble incongrue car nous vivons dans un environnement numérique où le { logiciel } tient une place prépondérante et n'est quasiment plus dépendant d'une machine donnée. Mais cette situation est le résultat de décennie de développement d'une industrie. Les premiers { logiciels } sont éminemments *machine-specific :* une suite d'instructions dépend fortement de la machine pour laquelle ces instructions sont rédigées. Cela tient à l'{ encodage } de ces machines. Pour la même opération, le { codage } pour la machine de l'*Institute for Advanced Studies* (IAS) de Princeton n'est pas le même que la machine du *National Physical Laboratory* des environs de Londres. Ce n'est qu'au fil du développement technico-industriel de l'informatique à partir des années 1950 que la distinction entre *software* et *hardware* va se faire, dans le sens d'un affranchissement progressif – mais jamais total – du *software* vis-à-vis du *hardware*. Il s'agit en quelque sorte de retrouver l'idéal initial d'un { matériel } « neutre » vis-à-vis du { logiciel }, un { matériel } capable d'imiter n'importe quelle autre machine.

Cette scission n'est jamais nette. Il existe des { logiciels } commerciaux implantés directement dans du { matériel } et qui ne peuvent pas être exécutés sur une autre machine (le *firmware*) ou du { matériel } qui est lui même programmable et « plastique » comme un { logiciel } (*programmable hardware*). Ou plutôt sa netteté est illusoire : aucun { logiciel } ne peut

134 « […] *the set of instruction that direct a computer to do a specific task* ». Ceruzzi, Paul E. *Op. cit.*, p. 80.

API, application, développeur, documentation, langage de haut niveau, langage de programmation, langage orienté objet, logiciel, matériel, objet, programmation, programmation modulaire, programme, système d'exploitation,

Voir glossaire tome II, p. 3-25

se départir du { matériel }. Mais le fait qu'elle soit floue ne remet pas en cause son efficacité industrielle et plus largement communicationnelle. Certes, il ne peut y a voir de { logiciel } sans { matériel }, mais le marché du { logiciel } repose encore aujourd'hui largement sur cette distinction et sur l'idée que le { logiciel } doit être indépendant du { matériel } sur lequel il s'exécute.

b …qui structure l'émergence du marché du logiciel

D'un point de vue industriel et économique, cette question est d'une grande importance. Si, par différents expédients techniques, il est possible de dissocier le *software* du *hardware*, il est alors possible de baisser le coût d'adoption des nouvelles machines[135], mais aussi d'externaliser le développement { logiciel }. Les premiers { langages de programmation } dits de « { haut niveau[136] } » vont dans ce sens : le Fortran (1957), puis le Cobol (crée en 1960 par IBM) permettent d'écrire des { logiciels } sans nécessairement devoir connaître la totalité du fonctionnement de la machine sur laquelle le { logiciel } peut être exécuté. De même, IBM – l'une des premières entreprises à élargir le marché de l'informatique aux entreprises et non plus aux seules administrations et universités – vend avec ses machines des outils pour programmer (la { documentation } du { langage de programmation[137] }) et fait dès les années 1950 la différence entre le { système d'exploitation } et les { applications }. Cette scission ne fera que s'accroître et contribuera à structurer le marché de l'informatique dans les années 1960 et 1970 entre des vendeurs indépendants de { logiciels } et des fabricants de { matériel } qui s'appuient sur ce tissu économique de { développeurs } et vendeurs de { logiciels }.

L'arrivée de la micro-informatique dans les années 1980 transforme encore un peu plus le marché informatique. Jusqu'ici, les éditeurs de { logiciels } travaillaient majoritairement en suivant les pratiques d'appels d'offres héritées des contrats d'ingénieurs[138]. Une organisation cherche un { logiciel } pour résoudre un problème interne, elle publie un appel d'offre

135 Si un logiciel n'est plus intrinsèquement dépendant d'une machine, on peut changer de machine sans devoir racheter tous les logiciels.

136 Dont la syntaxe et le niveau d'abstraction est éloigné du langage machine et plus proche d'un langage naturel comme l'anglais.

137 Steinmueller, Edward W. The International Software Industry : An Analysis and Interpretive History. Dans : Mowery, David C (dir.), *op. cit.*, p. 25.

138 Campbell-Kelly, Martin. Software as an economic activity. Dans : Hashagen, Ulf, Keil-Slawik, Reinhard et Norberg, Arthur L (dir.), *History of computing: software issues. International Conference on the History of Computing, ICHC 2000, April 5-7, 2000, Heinz Nixdorf MuseumsForum, Paderborn, Germany*. Berlin : Springer -Verlag Berlin Heidelberg, 2002, p. 193.

auquel des éditeurs répondent avec budget et livrable. Lorsque, dans les années 1980, l'informatique sort des bureaux et intègre les foyers, les nouveaux acteurs que sont Lotus, Microsoft (éditeur de MS-DOS) ou encore Corel (éditeur de WordPerfect, { logiciel } de traitement de texte) misent sur la massification du marché. Le modèle économique dominant n'est plus le développement au coup-par-coup de { logiciels } personnalisés, mais un fort investissement de recherche pour mettre au point un { logiciel } de masse. Le coût de développement est compensé par un faible coût de réplicabilité du { logiciel }, qui peut donc être distribué massivement ce qui permet de compter sur un fort volume de vente[139].

c Des *widgets* aux modules

Notre hypothèse est que ce progressif affranchissement du { logiciel } vis-à-vis du { matériel } n'a pas que des répercussions sur les modèles d'affaires de l'industrie, mais également sur les manières d'écrire les { programmes }. L'une de ces conséquences, c'est la mise au point de modules que l'on peut combiner pour faciliter l'écriture des { logiciels }. Ils sont standards et documentés, afin qu'un { développeur } n'ait pas besoin de savoir intégralement comment fonctionne la machine ou le { logiciel } auquel il emprunte un module. Ces modules sont, dans une certaine mesure, personnalisables et adaptables si bien qu'un éditeur peut répondre à une demande très spécifique. Les { API } web héritent de cette histoire plus longue de la modularité dans la { programmation }. Pour le montrer, nous nous intéresserons aux méthodes de génie logiciel, c'est-à-dire aux modèles d'écriture d'un { programme }. Nous ferons l'analyse de deux de ces modèles : la { programmation modulaire } puis les { langages orientés objet }. Ces deux modèles, par leurs concepts clés (module ; { objet }) sont de bons exemples de l'influence d'une pensée combinatoire dans la façon d'écrire des { logiciels } informatiques.

139 À titre d'exemple, le logiciel Mark IV, un logiciel de gestion de dossiers à destination des entreprises, crée par Informatics en 1967, se vend à 3 000 copies en 1984. C'est l'un des plus vendus de l'époque sur le marché interentreprises. WordStar, traitement de texte édité par MicroPro et distribué pour les ordinateurs personnels, se vend à 700 000 exemplaires. Campbell-Kelly, Martin. *Idem*, p. 196.

1 E La modularité en programmation : écriture combinatoire et besoins d'un marché

API, *bugs*, code, développeur, fonction, logiciel, matériel, ordinogramme, programmation, programmation modulaire, programmation orientée objets, programme, technologies de l'intellect, *widget*

Voir glossaire tome II, p. 3-25

Les { API } sont des outils d'écriture combinatoire en ceci qu'ils mettent à disposition des modules élémentaires d'écriture : blocs de { code }, de données, etc. Cette modularité prend différentes formes et différents enjeux selon les contextes ({ *widgets* }, curation…) mais s'ancre dans la différence entre le { matériel } et le { logiciel }, différence qui structure le marché informatique à partir des années 1960. Ce qui nous amène à un déplacement : la combinatoire ne serait pas le fait des seuls { API } web contemporaines, mais serait une tendance plus lourde de la { programmation } et de son optimisation pour le marché du { logiciel }. Nous vérifions cette hypothèse en étudiant deux modèles de { programmation } : la { programmation modulaire } (I. 1. E. a, b, c) et la { programmation orientée objet } (I. 1. E. d). Comme dans les parties précédentes, l'enjeu est de comprendre d'où vient l'aspect combinatoire des { API }, son histoire et son économie.

a La programmation modulaire : une optimisation de l'écriture des programmes

Les termes de « module » ou de « modularité » que nous avons utilisés jusqu'ici sont des termes issus de l'informatique, qui ont fait florès à partir des années 1970. La modularité, dans ce contexte, est une façon d'écrire un { programme } informatique. C'est un principe de design, un modèle d'écriture qui s'ancre dans une tradition plus ancienne : celle de la programmation structurée[140]. Cela consiste, comme le résume Tania Bucher, à « […] diviser un produit en plusieurs sous-systèmes ou modules, ces derniers pouvant être combinés entre eux[141] ». Ce modèle est progressivement devenu le paradigme dominant de { programmation }, au fur et à mesure de la complexification des { logiciels[142] }. On en trouve un exposé dans les manuels et articles de l'époque : le *Designing Computer*

140 La programmation structurée (*structured programming*) est une façon de programmer née à la fin des années 1960. Elle repose sur l'idée que tout programme informatique peut être découpé en blocs de fonctions ensuite ordonnés et répétés au besoin. Cette manière de programmer a pour objectif de clarifier l'écriture d'un programme et de limiter au maximum les « sauts » d'une ligne de code à une autre. La structuration du code en bloc permet de suivre pas à pas – de bloc en bloc –le déroulé du programme. Voir Dijkstra, Edsger W. Letters to the Editor: Go to Statement Considered Harmful. *Communications of the ACM*. 1968, vol. 11, n° 3, p. 147–148 ; Dahl, Ole-Johan, Dijkstra, Edsger W. et Hoare, Charles Anthony R. (dir.). *Structured programming*. London : Academic Press, 1972.

141 « *In general, modularity refers to the principle of breaking up a product into subsystems or modules, which can be recombined.* » Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op.cit*.

142 *Ibidem*.

Programs (Concevoir des programmes informatiques) de Gauthier et Ponto en 1970, puis l'article « *On the criteria to be used in decomposing systems into modules* » (Des critères à utiliser pour diviser un système en modules) de David Parnas deux ans plus tard. Le recours à cette écriture modulaire est dicté par des considérations managériales : il s'agit d'optimiser l'écriture des { programmes }, en trouvant une manière de travailler qui permette d'aller plus vite, de mieux repérer les { *bugs*[143] } et de faire facilement travailler les { développeurs } en équipes séparées.

Comme le souligne Bucher, la modularité comme principe d'écriture de { logiciel } cherche à mieux gérer la complexité d'un { programme } informatique. Il faut là aussi bien comprendre le contexte de l'époque. L'évolution rapide de l'informatique, tant du point de vue technique qu'économique, entraîne une complexification des { programmes }, qui contiennent de plus en plus d'instructions et de ramifications. Comment gérer cette complexité et d'abord comment se la représenter ? On utilisait alors le *flowchart*[144] ou { ordinogramme } : une ensemble de boîtes, reliées par des flèches. Chaque boîte est une ligne d'instructions ou une { fonction } du { programme }, les flèches indiquent l'ordre du { programme }. L'{ ordinogramme } permet de représenter synthétiquement le { programme }, dans la vieille tradition des { technologies de l'intellect } comme la carte[145] ou l'arbre[146]. Le problème étant que cette technologie a des limites, notamment quand les { programmes } dépassent les 10 000 instructions[147].

143 Un *bug* est une erreur dans un programme informatique. Ce *bug* peut être mineur et ne pas affecter le fonctionnement global du logiciel, mais il peut aussi bloquer complètement le calcul. Le plus souvent, le *bug* est une erreur humaine, une coquille dans la rédaction du code qui le rend inopérable pour une machine.

144 Pour une histoire des ordinogrammes, de leur rôle communicationnel et sociopolitique, voire Ensmenger, Nathan. The Multiple Meanings of a Flowchart. *Information & Culture*. 2016, vol. 51, n° 3, p. 321-351. L'auteur insiste notamment sur la fonction communicationnelle de l'ordinogramme : il permet moins de planifier en avance un programme que d'expliciter et de rendre clair à d'autres programmeurs ou à ses supérieurs un programme fraîchement écrit.

145 Robert, Pascal et Souchier, Emmanuël. La carte, un média entre sémiotique et politique. La carte au rivage des SIC. *Communication & langages*. 2008, n° 158, p. 25-29.

146 Eco, Umberto. *De l'arbre au labyrinthe. Études historiques sur le signe et l'interprétation*. Paris : Grasset, 2011 [2010].

147 « L'ordinogramme était une abstraction utile pour des systèmes informatiques entre 5 000 et 10 000 instructions, mais au-delà de cet ordre de grandeur elle s'avère insuffisante […] » (« *The flowchart was a useful abstraction for systems with on* [sic] *the order of 5,000-10,000 instructions, but as we move beyond that it does not appear to sufficient* […] »). Parnas, David L. On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM*. 1972, vol. 15, n° 12, p. 1056.

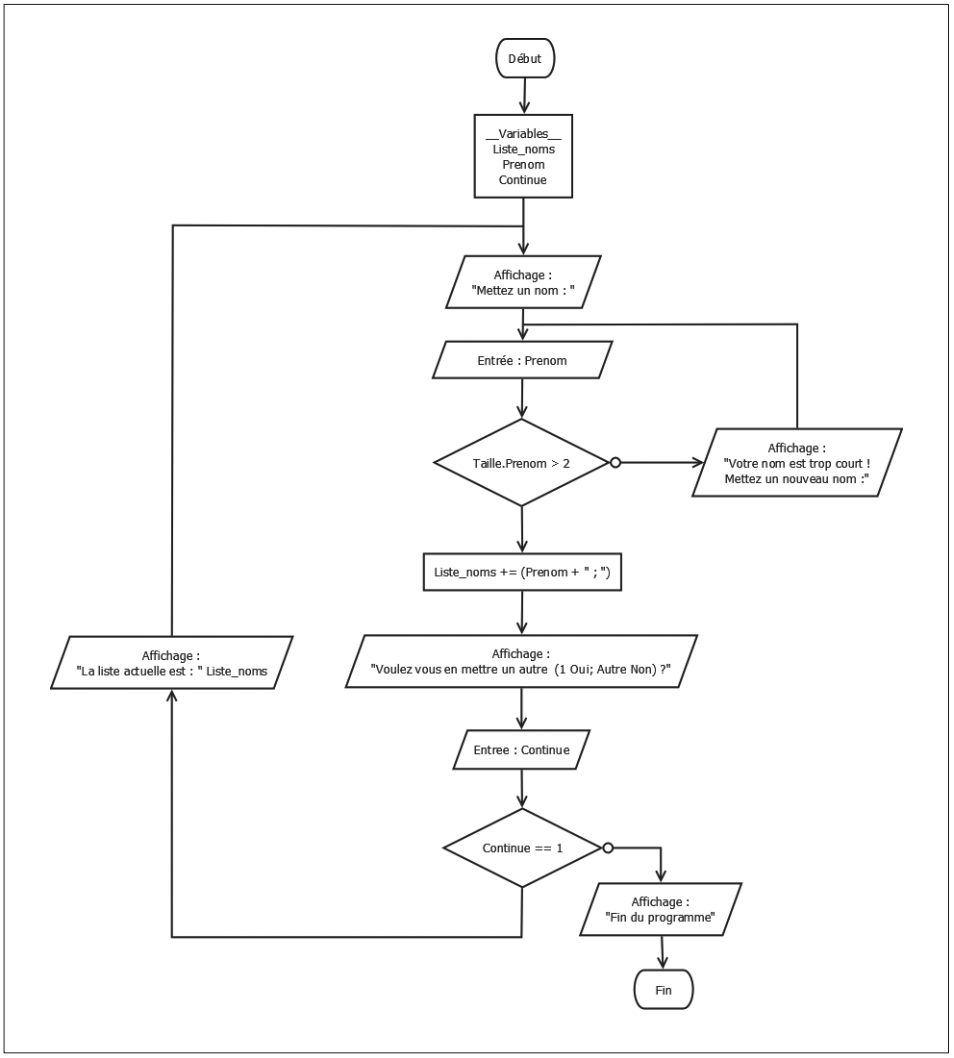


Fig.4. Exemple d'ordinogramme. Source : Open Classrooms. Introduction aux algorigrammes [en ligne]. 2013. [Mis à jour le 10 janvier 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://openclassrooms.com/courses/introduction-aux-algorigrammes.

b La modularité en réponse à la complexité : masquer pour rendre lisible et intelligible

développeur, interface, logiciel, masquage d'information, programmation, programmation modulaire, programme

Voir glossaire tome II, p. 3-25

Pour s'adapter à ce changement de paradigme dans l'écriture des { programmes }, l'informaticien canadien David Parnas s'appuie sur Gauthier et Ponto qui conçoivent la { programmation modulaire } de la manière suivante : « Une segmentation claire d'un projet assure la modularité du système. Chaque tâche forme un module du programme séparé et distinct des autres[148]. » Une équipe de { développeurs } se charge d'écrire un module, une autre un deuxième, etc. Ainsi, le { programme } n'est plus compris comme une suite plus ou moins complexe d'instructions mais

148 « *A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module.* » Gauthier, Richard L et Ponto, Stephen D. *Designing Systems Programs*. Englewood Cliffs : Prentice-Hall, 1970. Cité par Parnas, David L. On the Criteria to Be Used in Decomposing Systems into Modules. *Op. cit.*, p. 1056.

comme un assemblage de différentes tâches, chaque tâche pouvant être isolée et recombinée à d'autres. Sur ces bases, Parnas introduit le principe du « { masquage d'information[149] } » (*information hiding*) : il s'agit de masquer le plus possible la façon dont fonctionne un module, afin de ne pas surcharger le travail des autres { développeurs }. Bien sûr, ceux qui écrivent un module connaissent son fonctionnement, mais les autres groupes n'ont pas à connaître le fonctionnement intégral de tous les modules d'un { programme }. Ainsi, ils peuvent travailler en relative indépendance et le { programme } devient plus compréhensible car il peut être résumé en quelques fonctionnalités clés – celle de chaque module – sans qu'il y ait besoin de rentrer dans le détail. L'{ interface }, c'est-à-dire les façons d'accéder au module et à son action, en « […] révèle le moins possible sur son fonctionnement interne[150] »

Selon Parnas, les bénéfices de la { programmation modulaire } sont de trois ordres[151]. Un bénéfice managérial tout d'abord : chaque groupe de { développeurs } travaillant sur un module séparé sans qu'ils aient besoin de communiquer outre-mesure, le développement du { logiciel } ira plus vite. Un bénéfice commercial ensuite, puisqu'on peut modifier une partie du { programme } (un module) sans devoir modifier le tout[152]. Il suffit d'adapter un des modules aux demandes spécifiques d'un client. Un bénéfice cognitif enfin, puisque qu'on divise un système en unités plus faciles à comprendre, relire, corriger plutôt que le système entier en une seule partie. Il est ainsi plus facile de repérer d'éventuelles erreurs, de comprendre le fonctionnement général du { programme }, etc.

c Du managérial au scriptural

Jusqu'ici, la { programmation modulaire } est plus une organisation des tâches – un problème managérial – qu'un véritable modèle d'écriture des { programme } – un problème scripturaire. Comment donc la notion de « modules » passe du managérial au scriptural ? Il y a, dans le texte de Parnas, un flou qui autorise ce passage. Au début de son article, il définit un « module » comme « […] la respons-

149 *Ibidem*.

150 « […] *to reveal as little as possible about its inner workings* ». *Ibidem*.

151 Parnas, David L. *Idem*, p. 1054, paragraphe « Expected Benefits of Modular Programming ».

152 Il faut se rappeler le fonctionnement du marché du logiciel à l'époque. Des éditeurs indépendants répondaient à des appels d'offres. Il était plus simple d'adapter un logiciel existant aux besoins du client, plutôt que de devoir systématiquement réécrire un logiciel. La programmation modulaire permet une adaptabilité plus rapide et plus économique des logiciels.

abilité d'une tâche plutôt que la sous-partie d'un programme[153] ». De même, il insiste dans sa conclusion sur le fait qu'un module n'est pas une { *subroutine*[154] }. Un module ne peut donc être assimilé à un extrait de { programme } informatique. Et pourtant, lorsqu'il donne des exemples de modularisation, à chaque module semble correspondre une fonction dans le { programme }. Si l'on compare cela à un texte imprimé, cela revient en quelque sorte à dire qu'il faut considérer un livre non pas comme un enchaînement d'actions (dont la mise à plat serait l'{ ordinogramme }) mais comme un agencement de paragraphes ou de chapitres (qui seraient autant de modules) tout en disant que chaque paragraphe ou module participe à la progression dramatique du livre.

I 2
subroutine
Voir p. 111

**API,
appel,
code,
développeur,
logiciel,
masquage
d'information,
module,
ordinogramme,
programmation,
programmation
modulaire,
programme,
subroutine,
tweet,**

*Voir glossaire
tome II, p. 3-25*

Comme souvent, le paradoxe tient à ce que nous ne parlons pas exactement de la même chose, mais que ces choses différentes se rejoignent en un certain point. Quand Parnas donne des critères de { programmation modulaire }, il propose une méthode qui vise à remplacer l'{ ordinogramme }. Il se situe donc au niveau de **la représentation graphique d'un { programme } informatique**, l'équivalent du plan d'un livre imprimé. À ce niveau, à chaque module ne correspond pas un moment logique du { programme } (par exemple une { *subroutine* }), mais une fonction dans l'accomplissement global du { programme }. Pour continuer l'exemple du récit du livre imprimé, cela revient à différencier un moment dans le récit – un meurtre – et sa place matérielle dans le livre. Le meurtre joue une fonction essentielle dans le récit, mais son traitement peut être plus ou moins long : deux mots, deux lignes, deux chapitres… De la même façon, un module au sens de la { programmation modulaire }, c'est une fonction du { programme }. Il ne peut contenir que quelques dizaines de lignes de { code } ou des milliers, ce qui importe est sa fonction. La { programmation modulaire }, c'est passer de l'{ ordinogramme } aux modules, c'est-à-dire de la décomposition d'un { programme } en étapes logiques à une décomposition fonctionnelle : tel ensemble d'étapes logiques sert à obtenir tel résultat, tel autre tel résultat et l'ensemble forme le { programme }. Il s'agit donc bien d'un problème de design au sens à la fois d'une représentation et d'une manière de faire.

153 « […] *a responsability assignment rather than a sub-program* ». *Ibidem*.

154 Une *subroutine* est une fonction précise dans un programme (par exemple trier un jeu de données dans un certain ordre) que l'on a précédemment isolée et que l'on peut mobiliser à loisir dans différents programmes.

Mais c'est sur ce dernier point que le paradoxe arrive. Dans ce contexte très précis (l'industrie du { logiciel }), ce type de représentation a des incidences managériales, puisqu'elle a pour objectif d'organiser le travail de personnes qui écrivent des { programmes }. Elle a également des incidences scripturaires, puisqu'il s'agit d'un type d'écriture, que ce soit l'écriture d'un { programme } ou celle d'un module.

La { programmation modulaire } est donc tout à la fois un mode d'organisation des tâches, un type de représentation d'un { programme } et une façon d'écrire ce { programme }. Elle a pour enjeu principal d'aider à écrire des { programmes } complexes, tout en allant au plus rapide et au plus efficace. Pour cela elle recourt à des procédés combinatoires : division de la complexité en unités simples (modules), autosuffisance des unités simples (un module peut être compris sans avoir besoin d'une vue d'ensemble du { programme }) et réassemblage des modules pour créer le { programme } entier.

Les { API } héritent de la { programmation modulaire } à deux titres. D'abord parce que ce modèle de { programmation } met l'accent sur la notion de modules et donc sur l'accès à ces modules. Les interfaces de programmation jouent un rôle essentiel pour structurer cet accès. Deuxièmement, Cleidson Souza et David Redmiles ont montré comment les { API } fonctionnent comme une mise en application du principe de { masquage d'information[155] }. Elles isolent un bloc de fonctionnalités – un module – séparée en une partie publique (documentée) et une partie privée (masquée). Cette scission permet à des { développeurs } de s'approprier plus facilement des fonctionnalités complexes, puisqu'ils n'ont pas à en comprendre tous les tenants et aboutissants, tout en protégeant le fonctionnement du { logiciel } ou de l'entreprise qui fournit ce bloc de fonctionnalités. Bucher applique à juste titre ce principe aux { API } web : le système d' « { appel } » repose sur ce principe d'information cachée. Lorsque qu'une personne récupère des données grâce à l'{ API } de Facebook, ou qu'elle intègre un { tweet } dans un article, elle n'a pas besoin de savoir précisément comment fonctionnent ces plateformes et cela ne met pas en cause l'intégrité des données qui transitent[156].

155 Souza, Cleidson R. et Redmiles, David F. On The Roles of APIs in the Coordination of Collaborative Software Development. *Journal of Computer Supported Cooperative Work*. 2009, vol. 18, n° 5-6, p. 445–475.

156 Par cette dimension cachée, une API possède un caractère contractuel et est un outil de contrôle sur ce que peuvent écrire les utilisateurs. Voir IV. 2.

d Des modules aux objets : la programmation orientée objet (POO)

API, langage de programmation, langage orienté objet, logiciel, objet, programmation, programmation orientée objet, programme

Voir glossaire tome II, p. 3-25

Les { API } et la modularité jouent un rôle important dans un second modèle de { programmation } : la { Programmation Orientée Objet } (POO). Comme son nom l'indique, la POO repose sur le concept fondamental d'{ objet }, un peu différent de celui de module étudié précédemment. Un { objet }, en informatique, est une entité formée **d'attributs** (ses caractéristiques en quelque sorte) et de **méthodes** (les façons d'accéder à cet { objet } et à ses caractéristiques). On définit ensuite les **relations** de cet { objet } avec d'autres { objets }, ce qui permet de programmer un certain nombre de **messages** entre { objets }. La POO est un modèle de { programmation } qui consiste à répondre à un problème en faisant un modèle abstrait de ce problème sous forme d'{ objets } et des relations entre eux. Par { objet }, il faut donc bien comprendre la représentation d'un objet physique sous forme informatique. Imaginons par exemple que l'on veuille programmer le { logiciel } de navigation d'une voiture. Pour faire cela en utilisant le paradigme de POO, il faut commencer par définir ce qu'est informatiquement une voiture : un volume d'une certaine taille, d'un certain poids, avec quatre pneus, pouvant démarrer et s'arrêter[157]. Ce sont là ses **attributs**. Cet objet se meut sur une route à une vitesse devant être contrôlée. Il y a donc des *relations* de cet { objet } avec d'autres { objets[158] }. L'ensemble des relations possibles entre les { objets } et le déroulé de ces relations va constituer le { programme } en tant que tel. Lorsque nous parlons d'{ objets } dans ce contexte, c'est un { objet } au sens informatique du terme : une abstraction qui va servir d'élément de base à une écriture combinatoire et ce par ses propriétés. La voiture peut se combiner à la route parce qu'un de ses attributs (les pneus) permet cette combinaison.

157 On remarque qu'il n'est pas ici question de la couleur de la voiture, de sa marque, d'éventuels passagers… La représentation d'un objet informatique est une abstraction d'un objet physique, certaines propriétés en sont par là négligées.

158 D'où l'on voit qu'il faut définir tous les objets pertinents pour la situation à programmer. Ici, ce qu'est une route, un panneau de limitation de vitesse, un feu tricolore, etc.

La POO repose sur des { langages orientés objet }, c'est-à-dire des { langages } qui permettent de définir et de manipuler ces { objets }. Les premiers { langages orientés objet } (Simula et SmallTalk) sont créés à la fin des années 1960. Les années 1980 et surtout 1990 voient une explosion du développement de ces { langages } : C++ (1985), Python (1990), Java (1995), Ruby (1995)… Pourquoi ce moment là ? Steinmueller soutient que le succès de la POO est le résultat d'un double mouvement. Le premier est la continuation de la recherche d'un modèle de { programmation } qui permette la production de { logiciels } sur-mesure, notamment pour les entreprises, malgré l'arrivée de la micro-informatique et des { logiciels } grand public. Le second, c'est la montée en puissance des organisations en réseau – Internet en tête – qui nécessite la mise au point de standards pour assurer la compatibilité entre des machines et { logiciels } différents.

Or, sur quel principe la POO et plus spécifiquement un { langage orienté objet } ? Il permet de « fournir des modules standards pour la création de logiciels plus complexes[159] ». Plus fondamentalement :

> *« Un étalon de flexibilité est intégré dans le design de chaque objet. Cette approche peut engendrer le même type d'externalités[160] qui furent crées par le marché de masse de la micro-informatique, au fur et à mesure que des objets de plus en plus utilisés sont optimisés pour des environnements spécifiques, ses utilisateurs améliorant par là leur savoir-faire dans l'utilisation de ces objets[161]. »*

En d'autres termes, par leur architecture, les { langages orientés objet } anticipent la plasticité des { objets } informatiques qu'ils mettent en circulation. Cette plasticité est en lien étroit avec la structure tant économique que technique du marché où s'insère cette façon d'écrire des { programmes }. Il faut un modèle de { programmation } qui permette d'écrire des { programmes } facilement adaptables à des contextes d'utilisation différents, mais aussi à des environnements techniques de plus en plus complexes.

159 « […] *provide standard modules for the creation of more complex software systems* ». Steinmueller, Edward W. The International Software Industry: An Analysis and Interpretive History. Dans : Mowery, David C (dir.), *op. cit.*, p. 43.

160 En économie, on appelle externalité le fait que l'activité d'un agent crée un effet sur un autre agent, qu'il soit bénéfique ou maléfique, sans que ces agents soient conscients des effets.

161 « *A measure of flexibility is embedded within the design of each object. This approach can generate the same sort of externalities that were created by the mass market for personal computer software, as frequently reused objects are optimized for specific environments and as users improve their skills in the use of such objects* ». Steinmueller, Edward W., *idem*, p. 43-44.

API, application, discret, documentation, industrialisation, interface, langage de programmation logiciel, objet, programmation, programmation modulaire, programmation orientée objet, programme, standardisation, *widget*

Voir glossaire tome II, p. 3-25

L'étude de la POO montre d'abord à quel point la logique combinatoire des { API } web relève d'une logique plus ancienne en { programmation }. Toute la force de ces { langages } repose en effet sur l'abstraction et la combinaison d'unités { discrètes } : les { objets }. Cependant, pour mettre en circulation ces { objets }, il faut en documenter les méthodes d'accès. C'est ce en quoi consiste une { API } : la { documentation } explicite comment accéder et manipuler les { objets } d'un { programme }. C'est à cette condition que les externalités dont parle Steinmueller pourront être efficaces : sans { documentation }, il est très difficile de savoir à quoi correspond un { objet }, quels en sont les attributs et les méthodes d'accès. Difficile alors de mettre en circulation dans d'autres { programmes } ces { objets }, amplifiant par cette circulation l'adoption et le développement rapide d'un { langage[162] } ou d'un { logiciel }. Il y a donc un lien très fort entre POO et { API }, qui se joue à l'intersection entre un modèle d'écriture combinatoire et l'{ industrialisation } d'un modèle économique, fondé sur les externalités positives créées par la circulation d'{ objets } informatiques, à condition de fournir une { interface } à même de mettre en circulation ces { objets }.

162 Le cas du langage Java, lancé en 1995, est à ce titre exemplaire. Son adoption massive, sa stabilité et son développement rapide est dû en grande partie à une API bien documentée qui a permis à de multiples développeurs d'utiliser ce langage devenu rapidement incontournable. Voir Thimpleby, Harold W. Article « Java ». Dans : Ralston, Anthony, Reilly, Edwin D. & Hemmendinger, David (dir.), *Encyclopedia of computer science*. Londres : Nature Publishing Group, 2000, p. 937-941.

Cette première plongée dans l'histoire de la { programmation } a montré que l'écriture combinatoire est utilisée comme réponse à un marché du { logiciel } structuré autour d'une tension entre {standardisation} et personnalisation. Il s'agit de faciliter l'écriture en disposant de blocs standards, préfabriqués, tout en pouvant rapidement combiner et arranger ces blocs pour produire des { logiciels } adaptés aux demandes du marché. Les paradigmes de { programmation } que sont la { programmation modulaire } ou la { programmation orientée objet } reposent tout deux sur ces attendus combinatoires, dont les { API }web héritent tout en les diffusant à une échelle inédite. Les { API }web, en tant que plastigramme, fournissent en effet les blocs élémentaires d'une page web ou d'une { application }. Ces blocs sont combinables mais sont en même temps adaptables à leur contexte d'utilisation. Or, c'est notamment le cas avec les { *widgets* }, ce type d'écriture est, par le biais des { API } web, mis à disposition du plus grand nombre. Il s'agit donc en quelque sorte d'une extension du principe de modularité initialement lié à la structuration propre du marché informatique.

API, calcul, codage, code, logiciel, matériel, programmation, programme, standardisation, *subroutine*

Voir glossaire tome II, p. 3-25

De cette étude de la combinatoire du point de vue de l'industrie du { logiciel }, nous remontons au début de la { programmation } avec l'invention de la { subroutine }, soit le { codage } de certaines opérations routinières sous forme de bloc de { code } préécrit et mobilisables dans différents contextes. De ce point de vue, la { subroutine } est l'élement de base d'une forme d'écriture informatique combinatoire. Nous montrons qu'une telle conception de l'écriture s'appuie sur un paradigme logique de la { programmation } né au début des années 1950 en réponse à la complexité de la { programmation } des ordinateurs précédents (I. 2. a). C'est dans ce paradigme logique que va naître une forme d'écriture combinatoire : les { *subroutines* }, visant à faciliter l'écriture des { logiciels }, ce que montrera l'analyse d'un texte de John Von Neumann et Herman Goldstine (I. 2. b). Cette idée est reprise par David Wheeler, qui insistera sur les bénéfices de la { subroutine } en termes de flexibilité et de lisibilité du { programme } (I. 2. c).

2 LA *SUBROUTINE* COMME ANCÊTRE DES API

Nous venons de montrer que les { API } sont des outils d'écriture informatique porteurs d'une conception combinatoire et modulaire de l'écriture. Cette conception prend place dans une économie de la { programmation } fondée sur la différence entre { logiciel } et { matériel }. Les modules de fonctionnalité mis à disposition par une { API } permet un jeu entre la plasticité d'un { logiciel } ou d'une page et sa { standardisation }. À condition toutefois de documenter les { API }, c'est-à-dire d'expliciter la composition des modules ainsi que les conditions de leur mise à disposition.

Sur ces bases, nous pouvons continuer notre généalogie de ces objets en faisant l'hypothèse que ces deux tendances (combinatoire de modules ; nécessité de l'explicitation des règles de combinaison) n'est pas le propre des { API } web, ni des modèles de { programmation } proposés à partir des années 1970, mais qu'elles sont présentes dès la naissance de la { programmation } moderne à la fin des années 1940. Nous nous plaçons donc dans une histoire longue de la { programmation }, en montrant l'existence d'une pratique particulière, ce qu'on appelle la { *subroutine* }. Le principe est extrêmement simple. Un { programme } est habituellement séparé en plusieurs séquences, qui sont autant d'étapes du { calcul }. Certaines séquences sont génériques et peuvent servir dans d'autres { programmes }. Créer une { *subroutine* }, c'est repérer, dans un { programme }, de telles séquences pour les isoler puis les constituer en un bloc autonome remobilisable à souhait. Le terme apparaît pour la première fois dans un rapport de 1945 décrivant l'ENIAC[163] et son fonctionnement[164]. Le concept de { *subroutine* } est ensuite largement repris dans d'autres rapports techniques[165],

163 L'ENIAC (*Electronic Numerical Integrator and Computer*) est un calculateur électronique construit en 1944 à l'Université de Pennsylvanie.

164 Eckert, John A., Mauchly, John W., Goldstine, Herman. H., *et al. Description of the ENIAC and Comments on Electronic Digital Computing Machines*. Rapport nº AD498867 du 30 novembre 1945. Philadelphie : Moore School of Electrical Engineering, p. 31, section 3-7.

165 Curry, Haskell B. et Wyatt, Willa A. *A Study of inverse interpolation of the ENIAC*. Rapport nº AD0640621 du 19 août 1946. Aberdeen : Army Ballistic Research Lab, , p. 30 ; Goldstine, Herman H. et Von Neumann, John. *Planning and Coding of Problems for an Electronic Computing Instrument: report on the mathematical and logical aspects of an electronic computing instrument. Part II*. Princeton : Institute for Advanced Study, 1947. Curry et Wyatt n'utilisent pas le terme de subroutines mais de « *stages* » (étapes) dans la construction d'un programme. Le concept de programmation reste globalement identique à celui développé par Eckert et Mauchly en 1945.

API, documentation, programmation, *subroutine*

Voir glossaire tome II, p. 3-25

dans des manuels de { programmation[166] } ou encore dans des actes de colloques[167]. Le fait que les { *subroutines* } soient mentionnées par plusieurs auteurs, utilisées sur plusieurs machines et dans différents contextes tend à prouver qu'elles sont rapidement devenues une méthode ordinaire de { programmation } pour la communauté scientifique de l'époque.

Nous faisons l'hypothèse que ces { *subroutines* }, par leur caractère combinatoire et par les impératifs de { programmation } auxquelles elles répondent, sont en quelque sorte les ancêtres des { API }, ou en tout cas montrent que les { API } sont une des dernières manifestations de principes plus anciens dans l'histoire de la { programmation }. Quels sont ces principes et en quoi font-ils appel à une pensée combinatoire ?

Pour vérifier cette hypothèse et répondre à ces questions, nous étudierons deux textes : un rapport de Herman H. Goldstine (1913-2004) et de John Von Neumann (1903-1957), *Planning and Coding for an Electronic Computing Instrument* (Préparer et coder un outil électronique de calcul) ainsi qu'un article de 1952 de David J. Wheeler (1927-2004), *The use of sub-routines in programmes* (De l'utilisation des *subroutines* dans des programmes informatiques). Non pas que ces deux textes soient les seuls sur le sujet, comme nous venons de le dire. Goldstine, Von Neumann et Wheeler ne sont peut être même pas d'un point de vue informatique les solutions les plus efficaces pour programmer. En revanche, ils ont tous les deux vocation à diffuser une certaine version de ce qu'est programmer : ils écrivent leur rapport afin de diffuser leurs concepts de { programmation } ; Wheeler tire son article du livre écrit avec Maurice Wilkes et Stanley Gill : *The Preparation of Programs for an Electronic Digital Computer (La préparation de* programmes *pour un calculateur numérique électronique)*, l'un des premiers manuels de { programmation }. Ces deux textes nous intéressent donc en tant qu'ils témoignent d'un modèle de { programmation } qui va devenir dominant et où des pratiques d'écriture combinatoires comme les { *subroutines* } tiennent une place prépondérante.

166 Hopper, Grace. A-2 Compiler and Associated Routines for Use with UNIVAC. Dans : Adams, Charles W., Gill, Stanley et Combelic, Donn (dir.), *Digital Computers. Advanced Coding Techniques*. Cambridge : Massachussets Institute of Technology, 1954, p. 30-40 ; Wilkes, Maurice V., Gill, Stanley et Wheeler, David J. *The Preparation of Programs for an Electronic Digital Computer.* Cambridge : Addison-Wesley Press, 1951.

167 Wheeler, David J. The use of sub-routines in programmes. Dans : *Proceedings of the 1952 ACM national meeting (Pittsburgh).* New York : ACM Press, 1952, p. 235-236.

Reste à montrer comment ces pratiques, apparues dès les débuts de la { programmation } pour simplifier l'écriture pour la machine, reposent sur une pensée combinatoire ? Dans un premier temps, nous reviendrons sur le contexte historique et technologique de l'époque. En effet, pour qu'on puisse s'interroger sur la { programmation } et mettre au point des moyens scripturaires pour l'optimiser, encore faut-il que programmer, ce soit écrire, ce qui n'est pas le cas jusqu'au début des années 1950. Il faut que la { programmation } naisse en tant que pratique d'écriture pour qu'on puisse penser à mettre au point des blocs de fonctionnalités précodées. Nous soutenons que cette naissance est un passage de l'architecture à l'écriture, autour de l'adoption de l'architecture dite de Von Neumann (I. 2. A). Ce tournant acté, nous passons au texte de Goldstine et Von Neumann sur les { *subroutines* }, en montrant comment elles reconduisent une pensée abstraite et combinatoire de l'écriture (I. 2. B). Enfin, la lecture du texte de David Wilkes servira à montrer comment la réflexion sur des procédures comme les { *subroutines* } s'accompagne très vite d'une réflexion sur la façon de les rendre lisibles et intelligibles, par une { documentation } adaptée (I. 2. C)

2 A 1946–1951 : *Moore School*, « invention de la programmation » et paradigme logique. De l'architecture à l'écriture

Avant de nous confronter à ces textes, il est indispensable de resituer le contexte historique et technique de l'époque (1946 et 1951)[168], d'autant plus que la chronologie de l'histoire de l'informatique à cette période-là est particulièrement dense[169]. Faute de quoi, on aura peine à comprendre pourquoi la question de la { programmation } se pose de cette façon à ce moment-là. Dans la tension entre ingénierie et logique { formelle } qui constitue une des tensions de l'informatique, la fin des années 1940 marque un déplacement, une recomposition des équilibres. Alors qu'à la sortie de la guerre, le { calcul } était fortement ancré dans des manipulation matérielles (I. 2. A. a ; I. 2. A. b), la mise au point du concept d'ordinateur à { programme } en mémoire interne va remettre l'accent sur l'abstraction logique des { programmes } (I. 2. A. c). Nous soutenons que cette reconfiguration des équilibres entraîne une scripturisation de la { programmation }, qui fait passer celle-ci de la logistique à la logique (I. 2. A. d).

I Introduction
une des tensions de l'informatique
Voir p. xxx

calcul, formalisme, programmation, programme

Voir glossaire tome II, p. 3-25

a L'informatique à la sortie de la guerre

La sortie de la guerre est un moment où, dans l'histoire de l'informatique, commence à se poser la question de la { programmation } à proprement parler, c'est-à-dire de la manière d'écrire les instructions pour les machines développées au cours des dix années précédentes. Ce tournant de l'architecture – comment fabriquer un calculateur électronique automatique ? – vers l'écriture – comment lui donner des instructions, quelles sont les bonnes façons de coder ? – a lieu au cours de des années 1945 - 1946. Il est le résultat de trois facteurs : la difficulté rencontrée lors de la { programmation } du premier ordinateur entièrement électronique, l'ENIAC (*Electronic Numerical Integrator And Computer*) ; la proposition de l'architecture dite « de Von Neumann » suite au *First Draft of a report on the EDVAC* (Première version d'un rapport sur l'EDVAC) de

168 Nous avons choisi de reprendre le découpage de Campbell-Kelly, en rajoutant simplement l'année 1946 pour des raisons qui sont expliquées plus bas.

169 Pour resituer les évènements, nous nous sommes fondé sur deux monographies : Williams, Michael R. *Op. cit.* ; Ceruzzi, Paul E. *Op. cit*. Les dates qui suivent sont issues de recoupements entre ces deux ouvrages et nous avons jugé bon, pour le confort de lecture, de ne pas systématiquement renvoyer aux ouvrages concernés pour chaque date mentionnée. En revanche, lorsqu'il s'agit d'analyse et non plus de données strictement factuelles, nous donnerons bien évidemment la source. Voir également annexe n° 3 pour une chronologie synthétique.

John Von Neumann ; l'adoption massive de cette architecture, entre autres choix techniques, suite au séminaire « *Theory and Techniques for the design of Electronic Digital Computers* » (Théories et techniques pour la fabrication de calculateurs électroniques et numériques) qui s'est tenu du 8 juillet au 31 août 1946 à la *Moore School of Electrical Engineering* de l'Université de Pennsylvanie[170].

b Les difficultés de programmation de l'ENIAC, ou le primat du matériel

Au début des années 1940, John Eckert et John Mauchly présentent à l'Université de Pennsylvanie un projet de construction d'un ordinateur : L'ENIAC. Les premières unités sont fonctionnelles en 1944. L'ENIAC est pleinement opérationnel en 1945. Il est arrêté en 1955, après dix ans au service de la recherche et de l'armée américaine. L'ENIAC a la particularité d'être un des premiers calculateurs entièrement électroniques. Les machines précédentes, comme le Harvard Mark I, étaient encore électromécaniques. Elles comportaient des composants électroniques, comme des tubes à vide, mais aussi des composants mécaniques comme des roues dentées. L'ENIAC est une machine qui contient uniquement des composants électroniques et consacre, par sa vitesse et sa puissance de calcul largement accrues, la supériorité des calculateurs entièrement électroniques.

Se pose en revanche le problème de la { programmation } de ce calculateur : comment lui faire parvenir des instructions ? Comment planifier une opération ? Comme le rappellent Liesbeth de Mol, Martin Carlé et Maarten Bullynck, il est essentiel de prendre en compte la matérialité de la machine[171]. L'ENIAC est composé d'une quarantaine d'unités, dont 30 dédiées au { calcul }. L'ensemble occupe une salle entière. La transmission de données ou d'instructions se fait par le branchement de câbles qui émettent des pulsations électriques que chaque unité de { calcul } traduit en données computables (deux impulsions = le chiffre 2, par exemple) ou en instructions[172]. Programmer dans ce contexte, c'est relier les unités entre elles par ces câbles. C'est une opération essentiellement logistique, au sens où elle consiste en une série de gestes qui relient certains éléments

170 On y croise notamment John Von Neumann, Maurice Wilkes, John Mauchly, Herman Goldstine, ou encore Howard Aiken, qui a mis au point avec IBM le Harvard Mark I, premier supercalculateur électromécanique. La plupart des projets qui naissent à l'époque sont issus des discussions tenues à la *Moore School* pendant l'été 1946. Voir Williams, Michael R. *Op. cit.*, p. 349.

171 De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Op. cit.*, p. 1011-1046.

172 Il y a deux circuits possibles : un circuit de données et un circuit de programmation. Pour plus de détail sur l'architecture matérielle de l'ENIAC, voir De Mol, Liesbeth. *Code source sans code: le cas de l'ENIAC.* Communication donnée dans le cadre du séminaire « Codes Sources », LIP6, UPMC-Paris 6, 22 juin 2016.

architecture de Von Neumann, code source, logiciel, programmation, programme

Voir glossaire tome II, p. 3-25

de la machine. Si on peut planifier ces gestes, il n'y a pas à proprement parler « d'écriture » du { programme } au sens où il n'y a pas d'abstraction graphique (le { code source } qui est lisible par l'humain et par la machine. Il n'y a pas d'écriture pour la machine. Ce qui rend très difficile la { programmation } et surtout la reprogrammation de l'ENIAC :

> « [...] *pour tout nouveau programme, des câbles doivent être branchés dans les unités qui correspondent, des adaptateurs doivent être utilisés sur les connexions qui le nécessitent, des cadrans et des interrupteurs doivent être réglés aux valeurs adéquates, etc. Ainsi, il n'est pas étonnant que la planification d'un calcul, pas uniquement la traduction des opérations mathématiques en un schéma plus général mais aussi la matérialisation de ce schéma en un branchement de câbles et d'interrupteurs, pouvait prendre des semaines et devait être réalisée avec le plus grand soin*[173]. »

Tout le problème, du point de vue de la commodité d'utilisation, tient donc à la place prépondérante de la matérialité de la machine, du *hardware*. Ce terme même de *hardware* est anachronique, surtout si on l'oppose au { logiciel }, au *software*, puisqu'il n'y a pas à proprement parler de « { logiciel } ». Il y a une certaine configuration matérielle de la machine, rendant sa manipulation très fastidieuse.

c Programmes en mémoire interne et scripturisation de la programmation

Devant ces difficultés va s'engager une intense réflexion sur ce qu'est programmer et surtout comment optimiser l'écriture des instructions pour la machine. John Eckert, Von Neumann, Haskell Curry, John Mauchly, Alan Turing, Maurice Wilkes ou un peu plus tard David Wheeler… Tous ces informaticiens et scientifiques connaissent le fonctionnement contraignant de l'ENIAC et vont participer par leurs travaux à redéfinir ce qu'est programmer.

173 « [...] *for each new program, cables had to be plugged in the right devices, adaptors used on the right connections, dials and switches set for the right values, etc. Thus, it need not surprise* [sic] *that the planning of a computation, not only the translation of mathematics into a general scheme but also the realization of the scheme as a combination of cables and switch-settings, could take weeks and had to be done with utmost care.* » De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Op. cit.*, p. 1013.

C'est dans ce contexte que Von Neumann écrit son *First Draft of a report on the EDVAC* (Première version d'un rapport sur l'EDVAC) soumis à la *Moore School of Engineering* le 30 juin 1945. Dans son rapport, Von Neumann propose pour la première fois un modèle d'organisation matérielle des ordinateurs appelé depuis « { architecture de Von Neumann[174] } ». C'est ce modèle qui sera entériné lors du séminaire de la *Moore School* à l'été 1946 et c'est ce modèle qui sera utilisé pour fabriquer la machine de l'IAS[175] dont *Planning and Coding* [...] accompagne la conception.

En quoi consiste l'{ architecture de Von Neumann } ? C'est un modèle théorico-pratique, au sens où les auteurs du *First Draft* mettent l'accent sur l'organisation logique d'un ordinateur et non pas sa réalisation matérielle. L'ordinateur décrit par Von Neumann est composé de 5 unités[176].

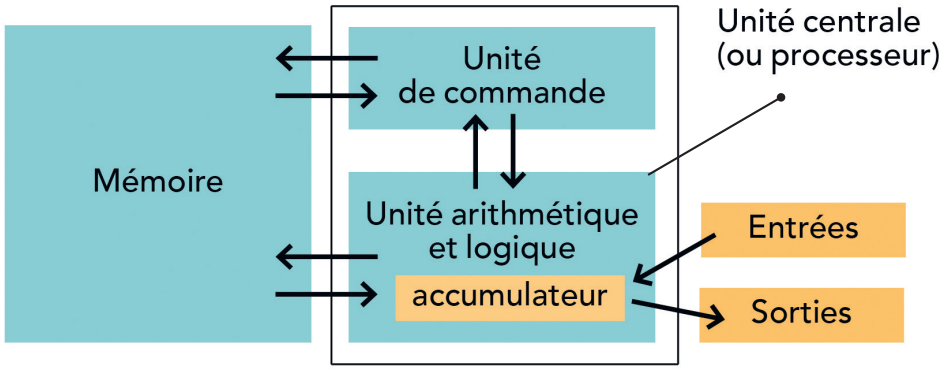


Fig.5. Schéma de l'architecture de Von Neumann, d'après Sacha Krakowiak. Disponible sur Krakowiak, Sacha. Le modèle d'architecture de Von Neumann. *Interstices* [en ligne]. 2011. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://interstices.info/modele-architecture-ordinateurs

174 Il existe une controverse historiographique sur la parenté de cette architecture et notamment sur le concept précis de « programme en mémoire interne ». Selon nombre d'historiens, John Von Neumann aurait pris à son compte des travaux menés notamment par John Eckert et John Mauchly. Ce qu'on appelle aujourd'hui architecture « de Von Neumann » est selon toute vraisemblance en réalité un travail réalisé par d'autres. Toutefois, la controverse historiographique nous intéresse moins que l'adoption de ce modèle de construction et les conséquences qu'il entraîne sur ce qu'est programmer. Voir Williams, Michael R. *Op. cit.,* p. 298-303 ; Ceruzzi, Paul E. *Op. cit.*, p. 21-23.

175 À en suivre De Mol, Carlé et Bullynck, la machine de l'IAS et EDVAC ne sont que deux noms d'une même machine. De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Op. cit.*, p. 1017.

176 Il ne s'agit plus ici d'unités au sens où nous employions ce mot pour l'ENIAC. Dans le cas de l'ENIAC, une unité s'entend au sens matériel du terme : c'est une machine qui effectue un certain type d'opérations (addition, multiplication, etc.). Dans le cas de l'EDVAC, il faut entendre « unité » au sens formel du terme : une unité est ce qui remplit une certaine fonction.

La première est l'unité de { calcul }, ou *Central Arithmetic* (CA, unité arithmétique sur le schéma). Elle est chargée d'opérer tous les { calculs }. La seconde est l'unité de contrôle, ou *Central Control* (CC, unité de commande sur le schéma). Elle vérifie que les { calculs } sont effectués selon l'ordre des instructions. La troisième unité est la mémoire (*Memory*, M) qui stocke les instructions ainsi que les données. La quatrième unité est une unité d'*input* (I) ou « entrée ». C'est par là que l'humain peut transmettre les données et instructions à la machine. La cinquième unité est une unité d'*output* (O), ou de « sortie », par laquelle la machine restitue les résultats définitifs de ses { calculs[177] }. Le point clé de cette architecture se situe entre CC, CA et M : les données (traitées par CA) et les instructions (traitées par CC) seront stockées par la même unité (M) et sous une forme similaire. C'est le principe des *stored-program computers :* ordinateurs à { programmes } en mémoire interne, qui est le mode de fonctionnement de tout les ordinateurs actuels. Désormais, { programme } et données seront { encodées } sous la même forme et stockées sur le même support. Il y a donc une sorte d'indifférenciation entre le { programme } en tant que jeu d'instructions et les données qu'il utilise.

calcul, carte perforée, codage, code source, langage de programmation programmation, programme

Voir glossaire tome II, p. 3-25

II | 3 | C | b
une plus grande plasticité de la machine
Voir p. 236 pour le concept de « machine universelle »

C'est cette indifférenciation qui permet une plus grande plasticité de la machine et une { programmation } grandement facilitée[178]. Il n'y a plus besoin de réarranger les câbles et autres interrupteurs de l'ordinateur. Il suffit de réécrire des instructions et/ou de lui fournir un autre jeu de données. La { programmation } devient à proprement parler une opération d'écriture pour la machine, étant donnée que ces instructions prennent une forme encodée sur un support (une { carte perforée } ou un fichier texte comme aujourd'hui) et ne consiste plus en une réorganisation matérielle de la machine. Cette « scripturisation » fait bénéficier la { programmation } d'une possibilité fondamentale de l'écriture : une plus grande abstraction et manipulabilité des instructions-machine. « Avec un ordinateur à programme en mémoire interne, une séquence d'instructions peut être stockée et réutilisée plus d'une fois[179]. »

177 Von Neumann, John. *First draft of a report on the EDVAC.* Rapport du 30 juin 1945. Philadelphie : Moore School of Electrical Engineering, University of Pennsylvania, p. 1-4. Paragraphe « *Main Subdivisions of the system* ». Von Neumann mentionne une sixième unité, nommée *Outside Recording* (R), qui fait le lien entre I et O et fait que les résultats sont « [...] perceptibles plus ou moins directement par les organes humains » (« [...] *sensed more or less directly by human organs* », p. 3). Von Neumann est toutefois moins clair sur le statut précis de cette sixième unité. Il est probable que cette sixième unité corresponde à ce que l'on appellerait en termes contemporains une interface : interface graphique, ligne de commande, etc.

178 Von Neumann le rappelle dès la page 2 : « *if the device is to be* elastic, *that is as possible* all purpose [...] » (Si la machine doit être *plastique*, c'est-à-dire la plus généraliste possible »). C'est l'auteur qui souligne.

179 « *With a stored-program computer, a sequence of instructions that might ve needed more than once could be stored.* » Ceruzzi, Paul E. *Op. cit.*, p. 84.

Ce changement n'est donc pas simplement technique. Il a forte une incidence sur ce qu'est la { programmation } et sur les savoirs professionnels qu'elle mobilise. Avec un ordinateur à { programmes } en mémoire interne, il peut y avoir un { code source }, c'est-à-dire un document écrit qui spécifie en un { langage } compréhensible pour les humains et la machine les instructions à opérer, ainsi que les données à traiter. Ce document est lisible par les deux entités en même temps, ou tout du moins sous la même forme[180]. Dans le cas de l'ENIAC, il peut y avoir bien sûr un diagramme schématique des câbles à brancher. Mais ce document sera lu uniquement par les opérateurs chargés de ces branchements. La machine ne lira que les données qu'elle doit calculer et certaines instructions, mais pas l'intégralité. Il ne peut pas y avoir de { code source }, puisqu'il n'y a pas à proprement parler de codification graphique[181] des instructions données à la machine.

d De la logistique à la logique : naissance de la programmation et primat du logiciel

La scripturisation de la { programmation } produit donc un déplacement entre les deux pôles (ingénierie / logique) fondamentaux de l'informatique. La { programmation } devient une opération à dominante logique plutôt que logistique. Programmer la machine, ce n'est plus connecter les bons câbles selon un schéma prédéfini, c'est écrire de façon la plus univoque possible des instructions pour une machine. Si la machine est correctement programmée en amont – entendre du point de vue logique et abstrait – alors il ne devrait pas y avoir d'erreurs[182]. Ou plutôt : l'erreur ne peut venir que d'humains ayant mal formulé les instructions, mais elle ne peut pas venir de la machine. La conception de l'EDVAC par Goldstine et Von Neumann est donc le résultat d'un « ethos » de la { programmation } :

180 Cela ne veut pas dire qu'une machine lit rigoureusement le *même document* que l'être humain. Dans bien des cas, le code source, pour être exécuté par la machine, doit être compilé, c'est-à-dire converti en un format directement opérationnel. Il n'en reste pas moins que le code source fait agir la machine, même si cette action est encore une conversion de format.

181 Par codification graphique dans ce contexte, nous entendons une écriture qui soit lisible à la fois par un humain et une machine.

182 Le fait qu'il y ait toujours, de manière quasi consubstantielle à tout programme, des *bugs* ne contredit pas notre argument. Nous parlons ici d'une certaine conception idéale de la programmation. Nous ne parlons pas encore de la programmation en tant qu'activité qui doit négocier avec une matérialité (la machine) et qui doit donc faire avec l'erreur non-humaine.

« Pour Von Neumann, une fois qu'un problème est traduit en termes mathématiques, la programmation en tant que telle ne présente pas de difficultés particulières et il n'y a pas de raisons que de graves problèmes apparaissent[183]. »

architecture de Von Neumann, code, logiciel, matériel, programmation, programme, *subroutine*

Voir glossaire tome II, p. 3-25

Le paradigme technologique de Von Neumann est un paradigme symbolique qui engage une vision de la { programmation } et du rapport à la machine. Si ce qui importe est la bonne rédaction des instructions, alors la logique prime sur la logistique et en dernière instance le *software* prime sur le *hardware*[184].

Ce qui se joue est donc une passation symbolique du { matériel } vers le { logiciel }. Données et instructions peuvent être comprises par la machine sous forme écrite, sans que le { matériel } doive être modifié. Programmer, ce n'est plus agencer correctement les différents composants d'un ordinateur, ce qui suppose un savoir faire en électronique et en mécanique. Programmer, c'est savoir penser et formuler un problèmes en termes univoques et en détaillant avec la plus grande précision l'ensemble des opérations à effectuer pour résoudre ce problème, de façon idéalement indépendante de la matérialité de la machine. Non pas que la dimension logique soit absente dans le cablâge d'une machine comme l'ENIAC. Mais lorsque les instructions sont stockées en mémoire interne, cette dimension logique de la { programmation } prend une place prépondérante et la dimension logistique est reléguée au second plan. Par conséquent, la balance penche plutôt du côté des mathématiciens que des ingénieurs. L'{ architecture de Von Neumann } et surtout le concept d'ordinateurs à { programme } interne opère donc « [...] la séparation entre programmation et architecture matérielle [...] [185] » séparation qui prend la forme d'une scripturisation de la { programmation }. Programmer, ce n'est plus câbler du { matériel }, c'est écrire du { logiciel }. Ce passage à l'écriture produit une première abstraction vis-à-vis de la machine pour laquelle on écrit. Cette architecture sera entérinée comme modèle de construction des ordinateurs lors du séminaire de la Moore School, *Theory and Techniques* [...] quelques mois plus tard.

I 1 D a
une première abstraction vis-à-vis de la machine...
Voir p. 97

183 « *According to Von Neumann, once the translation of a problem into mathematics is achieved, the actuel coding does not present any real challenge and no real problems are deemed to arise.* » De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Op. cit.*, p. 1017.

184 Encore une fois, le fait que Von Neumann ait probablement sous-estimé que des erreurs puissent subsister malgré la mathématisation des problèmes ne change rien à notre propos. Un tel modèle de la programmation est aussi une certaine conception de l'écriture qui s'appuie sur une forte abstraction, où des procédures combinatoires sont de première importance. C'est moins l'efficacité pratique du modèle de Von Neumann qui nous intéresse que les imaginaires de l'écriture qu'il mobilise.

185 « [...] *programming as separate from hardware design* [...] ». Kryza Joasia et Grzesiek, Sedek. Source Code. Dans : Fuller, Matthew (dir.). *Software studies: a lexicon*. Cambridge : MIT Press, 2008, p. 238.

Ce déplacement du logistique au logique explique que c'est à partir de 1947 que commencent à émerger des rapports, articles, manuels sur la { programmation } en tant que pratique d'écriture. Comment simplifier la rédaction des instructions pour la machine ? Quelles méthodes adopter pour gagner du temps, optimiser la lisibilité du { code } ou son efficacité technique ? C'est l'objet de deux documents, largement diffusés et qui ont constitué la base théorique de nombreux scientifiques de l'époque[186] : le *Planning and Coding* [...] de John Von Neumann et Herman Goldstine et *The Preparations of Programs for an Electronic Digital Computer*[187] (Préparer des programmes pour un calculateur numérique et électronique) de Maurice V. Wilkes, David J. Wheeler et Stanley Gill. Les historiens de l'informatique[188] traitent largement de l'importance de ces deux ouvrages dans la naissance de la { programmation }. La plupart le font du point de vue d'une histoire sociale de la technique. Notre point de vue est celui de l'histoire de l'écriture, étant entendu que l'informatique est une technique d'écriture. Plus particulièrement, nous nous intéressons à la place de la combinatoire dans les premiers écrits sur la { programmation }, autour du concept de { *subroutines* }. Pour ce faire, nous analyserons la section de *Planning and Coding* [...] consacrée à ce concept et un article de David Wheeler, « De l'usage des *subroutines* dans des programmes informatiques[189] ».

186 Campbell-Kelly, Martin. From Theory to Practice: The Invention of Programming, 1947-51. Dans : Jones, Cliff B. et Lloyd, John L. (dir.), *Dependable and Historic Computing: Essays Dedicated to Brian Randell on the Occasion of His 75th Birthday*. Berlin : Springer Berlin Heidelberg, 2011, p. 23-37.

187 Wilkes, Maurice V., Gill, Stanley et Wheeler, David J. *Op. cit*.

188 Campbell-Kelly, Martin. From Theory to Practice: The Invention of Programming, 1947-51. Dans : Jones, Cliff B. et Lloyd, John L. (dir.), *op. cit.*, p. 23-37 ; Aspray, William. *Op.cit.* ; Ceruzzi, Paul E. *Op. cit*. Chapitre 1 : « *The Advent of Commercial Computing. 1945–1956* », p. 20-24.

189 Wheeler, David J. The use of sub-routines in programmes. Dans : *Op. cit.*, p. 235-236.

2 B Les *subroutines* de Goldstine & Von Neumann : une pensée combinatoire au service d'une optimisation de l'écriture informatique

Planning and Coding [...] est une série de rapports dont la rédaction s'échelonne entre 1947 et 1948[190]. Ils accompagnent le développement de la machine de l'*Institute for Advanced Studies* (IAS) de Princeton, projet dirigé par Von Neumann et financé par l'*Army Ordnance Departement* (branche de l'armée états-unienne s'occupant du développement et de l'approvisionnement en armes) dans le cadre du contrat n° W_36_034_ORD_7481[191]. Le programme est commencé en 1946, en même temps que le séminaire *Theory and Techniques* [...] de la *Moore School*. La machine sera déclarée fonctionnelle le 10 juin 1952[192].

Ce rapport fait suite à un premier document, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument* (Remarques préalables à la conception logique d'un outil électronique de calcul), considéré par ses auteurs comme la première partie de *Planning and Coding* [...][193].Cette première partie traite, comme son nom le laisse penser, de l'{ architecture matérielle } de la machine. La seconde partie est constituée par les trois volumes de *Planning and Coding* [...], qui traitent eux de la { programmation } de cette machine. Le premier volume porte sur les principes généraux de la { programmation } ; le deuxième sur l'{ encodage } (*coding*) de problèmes simples et complexes ; le troisième volume est intitulé « *Combining Routines* » (Combiner des routines) et – comme son titre l'indique – il est entièrement dédié à la { programmation } de routines et à leur combinaisons. Ce dernier volume comporte trois mouvements : la justification du retour à des { *subroutines* } ainsi que la définition de ces dernières ; l'application de ces { *subroutines* } aux problèmes étudiés dans le volume II ; enfin une discussion sur l'espace mémoire requis pour stocker ces { *subroutines* } et les conséquences sur la vitesse de la machine. C'est le premier mouvement du troisième volume qui nous intéresse, puisque c'est dans celui-ci que les auteurs développent l'idée de précoder des fonctionnalités et de les combiner entre elles. C'est aussi dans le volume

application, architecture matérielle, codage, programmation, programme, *subroutine*

Voir glossaire tome II, p. 3-25

190 Le rapport qui nous intéresse, la seconde partie du troisième volume, est daté du 16 août 1948.

191 Goldstine, Herman H. et Von Neumann, John. *Op. cit*. La page liminaire de chaque volume de ces rapports porte la mention explicite du contrat liant l'IAS au Département de la Défense.

192 Williams, Michael R. *Op. cit.*, p. 352.

193 Goldstine, Herman H. et Von Neumann, John. *Op. cit.*, Préface, § 2.

III que l'on peut clairement lire à quel point l'informatique naissante est redevable d'une pensée combinatoire. Cela se remarque particulièrement sur trois points : la dimension analytique des problèmes calculables par la machine (I. 2. B. a) ; la recherche d'une économie et d'une facilitation de l'écriture des { programmes } (I. 2. B. b) ; la quête d'une plus grande flexibilité dans l'usage des { *subroutines* } (I. 2. B. c).

a Une combinatoire à l'image de la nature des problèmes computables

La prégnance de la pensée combinatoire tient d'abord à la nature des problèmes computables et aux possibilités de la machine. Ces problèmes ont l'avantage d'être « complets » au sens où ils sont formulés en des termes univoques qui permettent d'arriver à un résultat clair. En revanche, il est plus difficile de savoir si « [...] le problème en question se poserait, dans des cas concrets, comme une entité indépendante, ou plutôt comme l'une des composantes d'un problème plus grand et plus complexe[194] ». Comment articuler problème simple et problème complexe ? Les auteurs identifient deux manières de faire. La première est de se servir de l'expérience acquise lors de l'{ encodage } de problèmes simples pour encoder des problèmes complexes, mais en « [...] encodant de façon exhaustive toutes les parties du problème complexe, même si des problèmes simples similaires ont été encodés auparavant[195] ».

La seconde consiste à

> « [...] *commencer par encoder des problèmes simples (partiels), indépendamment du contexte (les problèmes plus complets) dans lesquels ils* [les problèmes simples] *peuvent apparaître, pour ensuite insérer ces séquences codées comme des unités indépendantes, quand se pose un problème complexe dont elles sont une partie*[196] ».

194 « [...] *the problem in question would in actual practice not occur by itself, as an isolated entity, but rather as one of the constituents of a larger and more complex problem* ». Goldstine, Herman H. et Von Neumann, John. *Idem*. Volume III, p. 1.

195 « [...] *to code all the parts of the complicated problem explicitly, even if equivalent simple problems have been coded before* ». *Ibidem*.

196 « [...] *to code simple (partial) problems first, irrespective of the context (more complete problems) in which they may occur subsequently, and then to insert these coded sequences as wholes, when complicated problem occurs of which they are parts.* ». *Ibidem*.

II | 3 | B
il existe des problèmes complexes
Voir p. 315

Goldstine et Von Neumann appliquent ici une stricte pensée analytique : il existe des problèmes complexes, divisibles en problèmes simples[197]. Or, ces problèmes simples qui se rencontrent lors de problèmes complexes se rencontrent souvent dans d'autres problèmes complexes, c'est-à-dire dans d'autres contextes d'utilisation de la machine.

Il n'y a donc pas que la façon de poser les problèmes qui entre en jeu (un problème doit pouvoir être réduit en parties simples), il y a aussi la nature de la machine. Rappelons que l'ordinateur conçu par Goldstine et Von Neumann doit être « généraliste », c'est-à-dire être capable de résoudre des problèmes de mathématiques, de physique, de ballistique, de météorologie, etc. Tant que ces problèmes sont mathématisables, il est possible de retrouver certaines séquences de { calcul } dans des problèmes appartenant pourtant à des disciplines différentes, à des « contextes » différents. Cet appel à une pensée analytique (isoler au sein de problèmes complexes des problèmes simples) et abstraite (extraire ces problèmes simples de leur contexte d'origine pour les remobiliser dans d'autres contextes) est donc le premier aspect fondamental d'une pensée combinatoire de l'écriture informatique. Il s'appuie tant sur la manière – mathématique – dont les problèmes sont envisagés et sur l'outil de { calcul } utilisé – une machine généraliste.

calcul, codage, programmation,

Voir glossaire tome II, p. 3-25

197 C'est toute la force de la logicisation et de la formalisation propre à l'informatique que d'envisager des problèmes sous l'angle du computable, c'est-à-dire de ce qui peut-être réduit en unités calculables par une machine. Voir Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Technologies, Idéologies, Pratiques*. 1999, nº 4, p. 198.

b Une combinatoire pour optimiser l'écriture des programmes informatiques

Cette pensée combinatoire sert également une optimisation et une économie de l'écriture. Goldstine et Von Neumann ne sauraient être plus clairs :

> « *Nous souhaitons* [dans ce chapitre] *développer des méthodes qui nous permettront, quand un problème se présente comme la partie d'un problème plus complexe, d'utiliser la séquence codée de ce problème simple comme une entité indépendante, comme un tout, pour ainsi éviter de devoir recoder ce problème à chaque fois qu'il apparaît comme une partie d'un nouveau contexte, c'est-à-dire d'un nouveau problème*[198]. »

Plutôt que de faire confiance à leur expérience et de recoder à chaque fois l'intégralité du problème, Goldstine et Von Neumann proposent d'identifier les opérations récurrentes entre différents problèmes et d'{ encoder } ces opérations comme des blocs indépendants. Et ce, encore une fois, principalement par souci d'économie : « [...] cela serait très pratique de ne pas avoir à recoder [ces parties de problèmes] quand ils doivent être utilisés comme partie d'un autre problème, mais d'être en mesure de les utiliser plus ou moins à l'identique, comme entité indépendante[199] ». La méthode de { programmation } qu'ils proposent est donc une méthode avant tout commode, qui évite de tout devoir systématiquement reécrire (réencoder) à chaque fois que la machine traite un problème différent.

Il faut ici se rappeler la difficulté de { programmation } de l'ENIAC : chaque nouveau {calcul} nécessitait des semaines pour recâbler la machine. Goldstine et Von Neumann cherchent à éviter ces désagréments. Mais plus qu'à accélérer la préparation de la machine, ils cherchent à en optimiser le fonctionnement et à en faciliter la { programmation } : « Savoir [isoler et encoder la partie d'un problème complexe] est d'une très grande importance. Il est fort probable que cela ait une influence décisive sur la facilité et l'efficacité d'utilisation d'un calculateur automatique comme celui que nous souhaitons fabriquer[200]. » Le mot important ici est

198 « *We wish to develop here methods that will permit us to use the coded sequence of a problem, when that problem occurs as part of a more complicated one, as a single entity, as a whole, and avoid the need for recoding it each time when it occurs as a part in a new context, i.e. in a new problem.* » Goldstine, Herman H. et Von Neumann, John. *Op. cit*. Volume III, p. 2.

199 « [...] *it would be very convenient not to have to recode any* [*parts of more complicated problems*] *when it is to be used as part of another problem, but to be able to use it more or less unchanged, as a single entity* ». Goldstine, Herman H. et Von Neumann, John. *Idem*. Volume III, p. 1.

200 « *The importance of being able to* [*use the coded sequence of a problem*] *is very great. It is likely to have a decisive influence on the ease and the efficiency with which a computing automat of the type that we contemplate will be operable.* » Goldstine, Herman H. et Von Neumann, John. *Idem*. Volume III, p. 2.

calcul, carte perforée, programmation, *subroutine*

Voir glossaire tome II, p. 3-25

« utilisation » (*operable*[201]) : un des enjeux de leur rapport est de rendre la { programmation } de leur calculateur plus simple. Bien sûr, la machine de l'IAS fonctionne encore avec un système de { cartes perforées } et demande des savoirs spécialisés en mathématique. Il n'est pas encore question de démocratiser la { programmation }. En revanche, on peut penser qu'à partir du moment où la { programmation } commence à s'émanciper de la matérialité de la machine, émerge la question de l'utilisation de cette machine par d'autres personnes que ses concepteurs.

c Une combinatoire pour valoriser la flexibilité dans l'écriture

Gagner du temps dans la préparation de la machine et en faciliter la { programmation }. Voilà l'optique dans laquelle Goldstine et Von Neumann théorisent l'utilisation de séquences codées indépendantes et combinables selon les contextes. Ces « séquences codées » sont ce qu'ils nomment une « { *subroutine* } » qu'ils distinguent de la « routine » ou « *main routine* » (routine principale). La routine est « [...] la séquence codée d'un problème[202] ». C'est l'ensemble des instructions et des données fournies à la machine afin qu'elle calcule la solution à un problème. Une { *subroutine* } est « [une routine] créée dans le but de pouvoir éventuellement remplacer [certaines parties] d'autres routines [...][203] ». Le mot clé est cette fois « remplacer » (*substitution*). Une { *subroutine* } se caractérise par sa permutabilité, son interchangeabilité. C'est ce caractère flexible qui fait sa facilité d'utilisation : chaque bloc d'instructions (chaque { *subroutine* }), s'il est correctement encodé, doit pouvoir être mobilisé dans n'importe quelle routine principale sans que l'on doive réécrire la totalité de ce bloc. C'est pourquoi Goldstine et Von Neumann posent ensuite la question des modifications à apporter à ces { *subroutines* } selon le contexte. Parmi ces modifications se pose l'épineuse question de l'espace mémoire alloué à la { *subroutine*[204] }. Goldstine et Von Neumann recommandent de ne pas systématiquement mettre une { *subroutine* } au même endroit dans la mémoire sans considérer la routine dans laquelle s'insère cette { *subroutine* }. Sans quoi il serait très difficile de combiner les { *subroutines* } entre elles, dans le cas par exemple où les deux emplacements des { *subroutines* }

201 « Opérable » existe en français mais pour des raisons de fluidité de la langue, nous avons transformé l'adjectif en nom, même si une « efficacité d'utilisation » est une formule peu heureuse.

202 « [...] *the coded sequence of a problem* ». *Ibidem*.

203 « [*a routine*] *which is formed with the purpose of possible substitution into other routines* [...] ». *Ibidem*.

204 Rappelons que dans un ordinateur, chaque opération occupe une certaine place mémoire. À l'époque, chaque partie d'un calcul, d'une routine, est mis à une certaine « position » dans la mémoire, position qui va déterminer l'ordre des instructions.

se chevauchent[205]. Tout le paradoxe étant qu'on ne peut non plus prévoir toutes les combinaisons possibles entre { *subroutines* } et qu'il est donc très difficile de garder le plus d'espace mémoire libre au cas où une nouvelle combinaison apparaisse. Devant ce dilemme, Goldstine et Von Neumann privilégient encore la permutabilité :

> « […] *Il sera très certainement très important de mettre au point une large "bibliothèque" de subroutines et de pouvoir l'utiliser avec une grande liberté. Toutes les solutions à ce dilemme* [de la mémoire] *basées sur une position fixe attribuée aux subroutines risquent fort d'être peu pratiques et d'une flexibilité insuffisante. En raison de quoi, nous devons postuler la variabilité, en fonction de ce qu'elle remplace, de la position initiale d'une subroutine*[206]. »

Les { *subroutines* }, comme méthode pour faciliter la { programmation }, ont comme enjeu central la flexibilité. Il faut entendre par ce terme leur capacité à être réinsérées dans de multiples types de { calcul }, mais aussi à pouvoir être combinées entre elles afin de gagner du temps dans la { programmation } d'un problème. On voit bien ici à quel point leur pensée de la { programmation } est redevable d'une pensée combinatoire : l'important est d'optimiser le nombre de permutations possibles (de « remplacements ») dans un système fixe (l'ensemble des problèmes calculables par la machine). Quand Goldstine et Von Neumann tranchent le problème des emplacements mémoire des { *subroutines* } en postulant la « variabilité » de ces emplacements, ils le font au nom d'une flexibilité plus grande des { *subroutines* }. Cette flexibilité est, selon nous, un avatar d'une combinatoire qui doit être la plus libre possible.

205 Imaginons une subroutine A qui utilise les espaces mémoire 1 à 4. Ces espaces mémoires ne peuvent être modifiés. Lors de l'encodage, on se rend compte qu'il est possible d'utiliser une subroutine B, qui utilise les espaces mémoires 3 à 10. La combinaison devient impossible, puisqu'une position dans la mémoire ne peut contenir qu'une seule valeur.

206 « […] *It will probably be very important to develop an extensive "library" of subroutines, and to be able to use it with great freedom. All solutions of this dilemma that are based on fixed positioning of subroutines are likely to be clumsy and of insufficient flexibility. Hence we should postulate the variability of the initial order position of a subroutine from one substitution to another.* » Goldstine, Herman H. et Von Neumann, John. *Idem*. Volume III, p. 3.

API, calcul, bibliothèque logicielle, code, développeur, documentation, programmation, programme, *subroutine*

Voir glossaire tome II, p. 3-25

Les { *subroutines* } de Goldstine et Von Neumann sont donc l'une des premières fois où est proposé l'usage, en { programmation }, de blocs de fonctionnalités précodées. Cette méthode d'écriture est rendue possible par la conjonction entre la nature des problèmes posés – ils sont tous posés en termes calculables et donc divisibles en unités simples - et la machine censée les résoudre – elle fonctionne sur du { calcul }. Cette calculabilité permet d'abstraire certains éléments récurrents pour les utiliser dans d'autres problèmes, dans d'autres contextes. C'est la notion de « remplacement » (*substitution*), essentielle dans ce texte. Enfin, l'organisation technique, notamment en ce qui concerne la mémoire, doit permettre de maximiser la flexibilité de ces blocs de fonctionnalités. On voit par ces trois aspects à quel point la { programmation }, dès ses débuts, est une écriture modulaire qui fait appel à une pensée combinatoire : diviser le complexe en éléments simples ; organiser les règles de permutation entre ces éléments ; maximiser, par la conception des éléments de base, le nombre de permutations possibles.

Cette question centrale de la flexibilité, liée à celle de la réappropriation des { *subroutines* }, ouvre à celle de leur lisibilité par d'autres scientifiques. En effet, à quoi sert une { bibliothèque } de fonctionnalités précodées si personne, à part leurs concepteurs, ne peut comprendre à quoi servent ces fonctionnalités ni comment elles fonctionnent ? En ouvrant la voie à une conception de la { programmation } comme écriture, permettant une abstraction et manipulation combinatoires de modules préconstruits, Von Neumann et Goldstine ouvrent également la voie au vieux problème de l'écriture orpheline de son scripteur[207]. Il ne suffit pas d'écrire le { code }, encore faut-il le commenter, le donner à lire et à comprendre. En l'occurrence, difficile de favoriser la réappropriation des { *subroutines* } si elles ne sont pas dûment expliquées. Elles ont beau être efficaces techniquement (pour la machine), elles ne le seront pas socialement (pour les humains). C'est ce qu'à compris David J. Wheeler dans son article « De l'utilisation des sub-routines dans des programmes informatiques » : l'un des enjeux des { *subroutines* } en { programmation } est leur intelligibilité pour les humains.

207 Platon. *Phèdre*. Paris : Flammarion, 2008. Traduction Luc Brisson.

2 C David Wheeler : flexibilité et lisibilité

David J. Wheeler (1927-2004) est un informaticien britannique. Son article de 1952, *The use of sub-routines in programmes* (« De l'utilisation des *subroutines* dans des programmes informatiques »), est une continuation du livre écrit un an plus tôt avec Maurice V. Wilkes et Stanley Gill : *The Preparations of Programs for an Electronic Digital Computer*. Ce livre est initialement rédigé comme { documentation } de la machine sur laquelle ils travaillent alors : l'EDSAC (*Electronic Delay Storage Automatic Calculator*). L'EDSAC est un ordinateur électronique à { programme } en mémoire interne développé au laboratoire de mathématiques de l'Université de Cambridge. Le projet est lancé en 1946, sous la supervision de Wilkes et la machine est opérationnelle trois ans plus tard, le 6 mai 1949. Wheeler est surtout connu pour avoir repris le concept de { *subroutines* } de Goldstine et Von Neumann en y apportant des modifications décisives connues sous le nom de *Wheeler Jump* (Saut de Wheeler). Le saut de Wheeler prolonge les attendus des { *subroutines* } (I. 2. C. a), mais surtout Wheeler met l'accent sur la nécessité de rendre lisible par d'autres { développeurs } le fonctionnement d'une { *subroutine* }, préfigurant les débats sur l'utilité d'une { documentation } pour les { API } (I. 2. C. b).

a Le « Saut de Wheeler » comme prolongement de l'abstraction des *subroutines*

Les { *subroutines* }, telles que conçues dans *Planning and Coding* [...] sont intéressantes en tant que concept de { programmation }. En revanche, leur mise en place est difficile du fait de la variabilité de la position, dans la mémoire, des { *subroutines* }. Étant donné que l'emplacement d'une { *subroutine* } dépend de son contexte d'utilisation, il est impossible d'écrire en amont cet emplacement. On opte donc pour une position mémoire « par défaut ». C'est pourquoi il faut ensuite procéder à une « relocalisation » : une fois la *main routine* écrite, on peut décider de la position d'une { *subroutine* }. Il faut qu'un opérateur humain écrive une « routine préparatoire » (*preparatory routine*) qui modifie la { *subroutine* } de telle sorte qu'aux emplacements mémoire « par défaut » soient substitués les emplacements mémoire effectifs qui serviront dans ce contexte précis.

Cette routine préparatoire de relocalisation était en général longue de 59 instructions et demandait un travail fastidieux à l'opérateur[208].

API, appel, bibliothèque logicielle, *bug*, calcul, code, programmation, programme, *subroutine*

Voir glossaire tome II, p. 3-25

Wheeler trouve une solution pour automatiser la relocalisation des { *subroutines* }. Elle sera désormais assurée par la machine elle-même et nécessitera beaucoup moins d'instructions, donc moins de travail humain. Cette automatisation est fondée sur une délégation du contrôle, c'est-à-dire de l'ordre des instructions, de la routine vers la { *subroutine* }. Dans un { programme } caractérisé par exemple par la suite d'instructions m[209], Wheeler intègre une instruction qui dit à la machine que sous certaines conditions (par exemple un { calcul } dont le résultat serait inférieur à zéro), il lui faut exécuter un sous-programme[210] précis – une { *subroutine* }, caractérisée par la séquence d'instructions n, n+1, etc. Le contrôle passe de m à n, de la routine à la { *subroutine* }. Le { programme } principal est comme « mis en attente ». La { *subroutine* } s'exécute et à la fin de cette { *subroutine* }, une instruction n renvoie à une instruction dans la séquence m : la { *subroutine* } rend le contrôle à la routine principale. C'est en cela que consiste le « saut » de Wheeler : l'appel à une { *subroutine* } à n'importe quel point du { programme } principal, la délégation du contrôle du { programme } à la { *subroutine* }, enfin le retour au point désiré dans le { programme } principal. Ces « sauts » sont l'ancêtre du système d'« { appel } » dans les { *subroutine* } et les { API } contemporaines[211]. Ils permettent une plus grande flexibilité dans l'usage des { *subroutines* } – il n'y a plus besoin de les relocaliser manuellement. Ils renforcent également l'abstraction des { *subroutines* }, qui deviennent véritablement des blocs de { code } indépendants du { programme } principal dans lequel ils s'insèrent.

b Rendre lisible et intelligible les *subroutines*

Mais l'apport de Wheeler aux { *subroutines* } ne s'arrête pas à une optimisation technique. C'est aussi l'un des premiers à souligner l'importance de les rendre lisibles et de coder d'une telle manière à mieux repérer les erreurs de { programmation }. Il détaille ces problèmes dans son article de 1952, « De l'utilisation de sub-routines dans des

208 Campbell-Kelly, Martin. From Theory to Practice: The Invention of Programming, 1947-51. Dans : Jones, Cliff B. et Lloyd, John L. (dir.), *op. cit.*, p. 31.

209 m étant l'instruction initiale, m+1 l'instruction suivante, etc.

210 Caractérisé par exemple par la séquence n, n+1, etc.

211 « *This so-called Wheeler Jump was the predecessor of the modern sub-routine call* ». (« Ce qui fut appelé " Saut de Wheeler " préfigurait les appels aux *subroutines* [dans la programmation] moderne »). Ceruzzi, Paul E. *Op. cit.*, p. 32.

programmes informatiques[212] ». Il repart des bases théoriques fixées par Goldstine et Von Neumann : une { *subroutines* } est définie comme « [...] une partie indépendante d'un programme, qui a la faculté de pouvoir être utilisée dans d'autres programmes »[213]. En stockant l'ensemble des { *subroutines* } dans une { bibliothèque }, le travail des programmeurs en est rendu plus facile : « Si des bibliothèques de *subroutines* peuvent être utilisées pour la plus grande partie d'un code, alors construire la partie restante du programme est un travail évidemment bien plus rapide que de réecrire le code en partant de zéro[214]. » Là encore, la mise au point des { *subroutines* } vise une économie de l'écriture : plutôt que de réinventer la roue à chaque { programme }, mieux vaut stocker certaines parties de { programmes } qui sont souvent réutilisées. Bien évidemment, ces parties de { programmes } devront être adaptées à leur contexte d'utilisation, mais il est plus rapide de modifier une { *subroutine* } existante plutôt que de la reprogrammer entièrement.

I | 2 | B | b
une économie de l'écriture
Voir p. 125

Wheeler se démarque toutefois de Goldstine et Von Neumann par son intérêt pour la résolution des erreurs de { programmation }. En isolant des blocs autonomes de { code }, il devient « [...] plus facile de les tester isolément[215] ». Ainsi quand le { programme } complet est écrit, si les { *subroutines* } ont été testées auparavant, on peut se concentrer uniquement sur le { programme } principal. Le risque de { *bugs* } dans un { programme } en est dès lors réduit[216]. Surtout, Wheeler souligne trois difficultés dans l'écriture la mise au point d'une { bibliothèque } de { *subroutines* } :

a) La { *subroutine* } ne doit pas être codée dans « [...] sa forme la plus simple. En règle générale, il sera nécessaire de coder [la *subroutine*] au format standard de la bibliothèque[217] ».

212 Wheeler, David J. The use of sub-routines in programmes. Dans : *Op.cit.*, p. 235-236.

213 « [...] *a self-contained part of a programme, which is capable of being used in different programmes* ». Wheeler, David J. *Idem*, p. 235.

214 « *If library sub-routines exist for the major part of a code then the task of constructing the remaining part of the programme is naturally very much less than if the code had to be written from the very beginning.* » *Ibidem*.

215 « [...] *tested in isolation from the rest of the programme* ». *Ibidem*.

216 Dans le même texte, Wheeler propose l'idée d'une « routine de contrôle » (*checking routine*) qui, parallèlement à la routine principale, vérifie que l'ordre des instructions est bien respecté, permettant de repérer les erreurs dans l'exécution d'un programme.

217 « [...] *in its simplest possible form. It will usually be necessary to code* [*the subroutine*] *in the library standard form* [...] ». *Ibidem*.

b) Il faut programmer « [...] de telle manière à ce que l'opération soit, dans une certaine mesure, généralisable[218] ».

c) Une fois la librairie proprement écrit, « [...] reste le travail considérable d'en écrire une description afin que ceux qui ne seraient pas familier de cette façon de programmer puissent malgré tout l'utiliser facilement[219] ».

Ce dernier point, que Wheeler estime la tâche « la plus difficile », est capital. L'aspect combinatoire de l'écriture des { *subroutines* } est acté. Cette combinatoire promeut une forme de flexibilité et de modularité des blocs de fonctionnalités. Or, c'est le point aveugle de Von Neumann, il est nécessaire d'expliquer la façon de programmer ces blocs, afin de permettre à d'autres scientifiques de les utiliser dans d'autres environnements ou sur d'autres machines. Sans quoi, une librairie de { *subroutines* } resterait non seulement liée intrinsèquement à une seule machine[220] – ici l'EDSAC – mais ne pourrait également être utilisée que par les programmeurs de cette machine.

API, architecture de Von Neumann, code, documentation, format, logiciel, matériel, programmation, programme, *subroutine*

Voir glossaire tome II, p. 3-25

Sans parler de démocratisation, il est clair que les documents que nous avons analysés cherchent à élargir la pratique de la { programmation }, à fournir des méthodes d'écriture qui ne se confinent pas à une seule machine et à un seul laboratoire. Cette extension de la { programmation } est rendue possible par l'{ architecture de Von Neumann } et la différenciation entre *software* et *hardware*. Puisque le { logiciel } (*software*) est abstrait du { matériel } (*hardware*), la { programmation } — l'art d'écrire des { logiciels } – peut être indépendante de la machine sur laquelle est exécuté le { programme }. C'est pourquoi il devient essentiel d'écrire, comme le suggère Wheeler, ce qu'on appelle aujourd'hui une { documentation } et sur laquelle repose en grande partie les { API } contemporaines : l'explicitation des méthodes, des { formats } de données et de la manière dont on peut utiliser les modules de { code informatique }. Ce qui est en jeu avec cette { documentation }, c'est la réappropriation des { *subroutines* }, en d'autres termes leur circulation sociale. Sans celle-ci, il serait très difficile de savoir comment sont constitués les modules combinables et les actions qu'ils permettent. Il serait donc très difficile de faire circuler les { *subroutines* }, de les utiliser dans d'autres contextes et par d'autres personnes que ses concepteurs.

218 « [...] *to code in such a manner that the operation is generalized to some extent* ». *Ibidem*.

219 « [...] *there still remains the considerable task of writing a description so that people not acquainted with the interior coding can nevertheless use it easily* ». *Ibidem*.

220 Ce qu'on appelle en anglais *machine-bound*. Littéralement : lié à une machine.

Les années 1946-1951 sont un moment de l'histoire des rapports entre écriture et informatique, autour de la naissance de la { programmation } au sens moderne du terme, c'est-à-dire de l'écriture de { programmes } pour un ordinateur. Avec l'adoption de l'{ architecture de Von Neumann } et surtout celle des ordinateurs à { programme } en mémoire interne naît l'abstraction du { logiciel } (*software*) vis-à-vis du { matériel } (*hardware*). Cette abstraction consacre non seulement la { programmation } comme écriture (et non plus comme réarrangement de la matérialité de la machine) mais consacre également une conception logique de la { programmation } : la logique interne du { programme } est plus importante que son exécution par une machine donnée et donc un { logiciel } doit pouvoir être exécuté sur d'autres machines et dans d'autres environnements techniques.

À partir de ce moment naît une préoccupation pour l'écriture des { programmes } informatiques et l'élaboration de méthodes pour la faciliter. Parmi ces méthodes, les { *subroutines* } constituent la préfiguration de ce que sont les { API } contemporaines : des blocs de fonctionnalités précodés faits pour pouvoir s'adapter à de multiples contextes d'utilisation. L'étude des manuels préconisant la mise au point de { *subroutines* } montre à quel point ces objets informatiques sont pétris d'une pensée combinatoire : divisions d'un { programme } en unités simples ; abstraction de ces unités en blocs permutables ; solutions techniques pour optimiser les possibilités de permutations (le « Saut de Wheeler » en étant un bon exemple). Ce qui pose finalement le problème de la circulation sociale de ces blocs de fonctionnalités, problème qui ouvre à toute la question des moyens pour faire comprendre à l'humain le mode de fonctionnement de la machine.

L'hypothèse de départ de cette sous-partie est donc confirmée : la combinatoire est présente dès les début de la { programmation }. De solution pratique pour faciliter l'écriture pour la machine, elle conduit à une plus grande abstraction et flexibilité des éléments de base de cette combinatoire. Peut-on encore remonter d'un cran ? Au-delà de l'industrie du { logiciel } (I. 1), au-delà de la { programmation } (I. 2.), est-ce que la combinatoire ne tient pas une place prépondérante en informatique en raison de la manière dont celle-ci se constitue en discipline scientifique à la fin des années 1930 ? Et quel rôle joue alors la combinatoire dans ce contexte ?

Dès « l'invention de la { programmation } », on trouve des pratiques d'écriture combinatoire qui ont pour but de faciliter l'écriture des { programmes }. Ces pratiques se déploient autour d'un paradigme logique de la { programmation } : programmer, c'est écrire une suite d'actions et non plus agencer la matérialité de la machine. Une combinatoire peut alors s'exercer, mais dans le stricte espace de l'écriture des { programmes }, c'est-à-dire un espace abstrait, idéalement indépendant de la machine sur laquelle s'exécute ce { programme }.

programmation, programme

Voir glossaire tome II, p. 3-25

API, calcul, programmation, programme

Voir glossaire tome II, p. 3-25

Nous remontons encore d'un cran, en sortant de la question de la { programmation } (l'écriture pour la machine) pour aborder les débuts de l'informatique (le fonctionnement de la machine). Notre hypothèse est que l'informatique elle-même, par le biais de l'écriture, est porteuse d'une pensée combinatoire dont hérite la { programmation } et les { API } contemporaines. Pour vérifier cette hypothèse, nous commentons un des textes fondateur de l'informatique moderne : *On Computable Numbers, with an application to the Entscheidungsproblem d'Alan Turing*, paru en 1937. Après avoir fait le point sur les travaux traitant du rapport entre écriture et informatique (I. 3. A), nous montrons dans quelle mesure Turing donne une définition de l'écriture telle qu'elle peut être automatisée par une machine (I. 3 .B). Enfin, nous montrons en quoi cette écriture est combinatoire (I. 3. C).

3 L'INFORMATIQUE AU CREUSET DE L'ÉCRITURE, DU CALCUL ET DE LA COMBINATOIRE

L'analyse des textes de Goldstine & Von Neumann, ainsi que celle de l'article de Wheeler, a montré l'importance, dès les débuts de la { programmation }, de la combinatoire autour d'un paradigme logique, ce afin de faciliter l'écriture. Isoler des blocs de fonctionnalités pour mieux les combiner entre eux permet d'économiser un temps précieux pour écrire un { programme }.

Nous allons désormais remonter d'un cran et montrer que dans l'informatique – et non plus seulement dans la { programmation } – il y'a l'exercice d'une pensée combinatoire. Notre hypothèse est que c'est par l'écriture et ses propriétés que la combinatoire s'introduit dans l'informatique. Pour la vérifier, il nous faudra d'abord préciser les liens ambigus entre écriture et informatique (I. 3. A.). Affirmer que les ordinateurs sont des machines d'écriture, est-ce que cela signife qu'elles servent à écrire, qu'elles écrivent elles-mêmes, qu'elles fonctionnent grâce à une certaine forme d'écriture ? Un état de l'art nous permettra de préciser à quel niveau d'écriture nous nous situons. Nous passerons ensuite à la naissance de l'informatique proprement dite, que nous incluons dans l'histoire de l'écriture, par le biais d'auteurs qui, au début du XX^e^ siècle, avaient lié écriture, { calcul } et combinatoire. Notre état de l'art montrera à quel point cette question de l'identité de l'écriture et du { calcul } – que pose Alan Turing – est centrale dans la compréhension contemporaine de l'informatique.

Nous étudierons donc un texte d'Alan Turing (1912-1954) de 1937, où il décrit une machine, la fameuse « machine de Turing[221] », modèle théorique des ordinateurs modernes. Ce texte étant un des textes fondateurs de la science informatique, il permet d'en mettre à jour certains imaginaires structurants. L'analyse montrera comment Turing lie le { calcul }, l'écriture et l'automatisme, en faisant du { calcul }, un type d'écriture automatique, c'est-à-dire réalisable par une machine (I. 3. B). Nous concluerons en précisant en quoi cette « écriture informatique » est combinatoire (I. 3. C).

221 À ne pas confondre avec l'expression « test de Turing » qui n'est pas l'apanage de Turing mais de l'écrivain Arthur C. Clarke. Le test de Turing est censé déterminer « l'intelligence » d'une machine par le biais d'un jeu où la machine doit se faire passer pour un humain. Turing décrit ce jeu dans un texte de 1950. Voir Turing, Alan M. Computing Machinery and Intelligence. *Mind*. 1950, vol. 59, n° 236, p. 433-460.

3 A Écriture et informatique : des liens avérés mais ambigus

On peut résumer l'ambiguïté de la façon suivante : les ordinateurs sont-ils des machines *à écrire* ou des machines *d'écriture ?* Autrement dit, est-ce que ce sont des machines qui *permettent* d'écrire – un certain type d'outils d'écriture donc – ou est-ce que ce sont des machines dont *le fonctionnement repose* sur de l'écriture ? La sémiotique des écrits d'écran s'est construite sur cette ambiguïté d'un outil d'écriture lui-même écrit (I. 3. A. a), mais aussi sur l'idée – légèrement différente – que la { programmation } est une pratique d'écriture (I. 3. A. b). Ce qui ne résout en rien la question de savoir si les machines écrivent en tant que telles, ou si leur activité doit être qualifiée autrement (I. 3. A. c).

architexte, code, formalisme, interface, programmation

Voir glossaire tome II, p. 3-25

a Des « outils d'écriture écrits »

Lorsqu'Yves Jeanneret et Emmanuël Souchier proposent, pour étudier l'écriture informatique, la notion d' « { architextes[222] } », qu'Emmanuël Souchier reformule en « outils d'écritures écrits[223] », ils lient en un seul concept deux niveaux d'écriture. Ce sont des outils qui permettent d'écrire, de produire des textes en vertu de certains modèles culturels et avec des fonctions communicationnelles diverses. Ce sont aussi des outils écrits, en ceci que ces outils sont le produit d'une écriture – typiquement le { code informatique } – qui cadre et oriente les pratiques qu'ils permettent. Dans les deux cas, l'écriture est à comprendre comme l'organisation, à travers l'utilisation de signes, d'une situation de communication. Cette organisation mobilise des modèles culturels, une histoire des formes médiatiques… Autant d'éléments qui rendent ces pratiques indissociables d'un contexte culturel précis. Le courant des *Software Studies*, dans le monde anglophone, est né d'une volonté similaire de comprendre et d'analyser l'épaisseur culturelle des { interfaces } numériques, avec une focale particulière sur l'histoire des médias[224].

222 Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Pour la Science*. 2002, n° 33, p. 101.

223 Souchier, Emmanuël. Le livre au risque de l'écrit d'écran et des écrits de réseaux. Dans : Zali, Anne (dir.), *La grande aventure du livre : de la tablette d'argile à la tablette numérique*. Paris : Bibliothèque Nationale de France, 2013, p. 176-183.

224 Manovich, Lev. *The language of New Media*. Cambridge : MIT Press, 2000 ; Fuller, Matthew (dir.). *Op. cit.*, 2008.

b Le code comme pratique d'écriture

Cette interrogation sur les { interfaces } dans leur dimension écrite amène à s'interroger sur le { code } comme pratique d'écriture. Historiquement, les écrits d'écran viennent de travaux sur la mise en page du { code informatique }, visant à montrer comment cette écriture s'inscrit dans une histoire plus longue du texte[225]. Cela rejoint par ailleurs la proposition de Donald Knuth de considérer la { programmation } comme une « pratique lettrée[226] », où le souci d'intelligibilité et de lisibilité du { code } pour l'être humain l'emporte sur son efficacité machinique. Dans la lignée des *Software Studies*, certains chercheurs ont pris comme objet central le { code informatique }, du point de vue de sa circulation sociale et son appropriation par certaines communautés[227]. Il s'agit de considérer le { code } comme discours[228], avec sa philosophie[229] et ses systèmes de valeurs[230]. Toujours dans le monde anglophone, les *Critical Code Studies* étudient certes le { code informatique }, mais cette écriture y est envisagée du point de vue poétique et de ses effets esthétiques[231].

c Des machines écrivantes

Tous ces travaux traitent de la { programmation }, envisagée comme pratique d'écriture faite par des humains *pour* une machine mais surtout pour d'autres humains. De plus, nous nous situons à des couches d'écriture assez éloignées de ce que fait la machine de Turing. Car pour l'instant, il ne s'agit pas de déterminer si la machine en tant que telle écrit. Jean Lassègue et Giuseppe Longo posent de manière frontale la question dans un article publié en 2006 : « Que vaut la comparaison de Turing entre écriture et mécanisme ?[232] » Lorsque Turing compare l'écriture à un mécanisme, de quelle écriture parle-t-il ? Leur thèse est que l'écriture est dans ce contexte une écriture { formelle } et combinatoire,

225 Souchier, Emmanuël et Pomian, Joanna. Informatique et pratiques écrivantes. *Traverses*. 1988, n° 43, p. 121-130.

226 Knuth, Donald E. Literate Programming. *The Computer Journal*. 1984, vol. 27, n° 2, p. 97-111.

227 Mackenzie, Adrian. The Performativity of Code : Software and Cultures of Circulation. *Theory, Culture & Society*. 2005, vol. 22, n° 1, p. 71-92.

228 Coleman, Gabriella. Code Is Speech : Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers. *Cultural Anthropology*. 2012, vol. 24, n° 3, p. 420-454.

229 Berry, David M. *The philosophy of software : code and mediation in the digital age*. Basingstoke : Palgrave Macmillan, 2011. Pour un état de l'art plus complet sur les travaux sur le code dans le monde anglophone, voir p. 4-5.

230 Lessig, Lawrence. *Code and other laws of cyberspace*. New York : Basic Books, 1999.

231 Montfort, Nick, Baudoin, Patsy, Bell, John, *et al. 10 PRINT CHR$(205.5+RND(1));: GOTO 10*. Cambridge : MIT Press, 2012.

232 Lassègue, Jean et Longo, Giuseppe. What is Turing's Comparison between Mechanism and Writing Worth ? Dans : Cooper, S. Barry, Dawar, Anuj et Löwe, Benedikt (dir.), *How the World Computes : Turing centenary conference and 8th conference on computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012 : proceedings*. Berlin : Springer Berlin Heidelberg, 2012, p. 450-461.

calcul, codage, formalisme, langage machine, programmation

Voir glossaire tome II, p. 3-25

héritée des travaux du logicien Hilbert sur les systèmes { formels[233] }. C'est cette écriture qui permet l'abstraction { formelle } et combinatoire que Turing mécanise, car elle est réduite à une pure manipulation, indépendante du sens des unités manipulées[234]. Ils rejoignent en grande partie les conclusions de Bruno Bachimont : la machine n'écrit pas, mais manipule des symboles. À cause du { formalisme } hilbertien inhérent à la mécanisation du { calcul }, les signes ne font pas l'objet d'une interprétation contextuelle, mais ne font que renvoyer à une signification précodée qui va déclencher un mouvement de la machine[235]. Pour que la machine fonctionne, il faut en quelque sorte qu'elle ne fasse que manipuler aveuglément des symboles vides de sens. Bruno Bachimont ne réfléchit toutefois pas en tant qu'historien de l'écriture, comme Jean Lassègue et Giuseppe Longo, mais en tant que philosophe des techniques. La question qu'il se pose est : comment redonner du sens à ces manipulations ? Selon lui, c'est à l'humain qu'incombe cette tâche. Un autre théoricien des médias, Friedrich Kittler, est dans une logique similaire. À le suivre, l'écriture a disparu au moment de son automatisation par des machines. Tout comme Bruno Bachimont, Kittler fait la part belle à la matérialité du { calcul }, qu'il considère comme un nouveau stade historique de l'écriture, à partir du moment où elle est auto-reproductible. Il y a une écriture *de* la machine, mais cette écriture est indépendante du contexte, automatique, séparée d'un acte humain d'interprétation. Friedrich Kittler garde donc la catégorie de « l'écriture » pour qualifier les actions de la machine, sans plus préciser quelles sont les caractéristiques de cette écriture. Katherine Hayles traite aussi cette question mais étudie l'influence de la matérialité du dispositif technique sur les formes esthétiques qu'il permet de produire, formes qu'elle appelle des « technotextes[236] ». Dans sa perspective, « écrire » est une inscription de signes réalisées dans un contexte et une visée par un être humain. L'ordinateur n'est pas écrivant mais c'est une *technologie de l'inscription*[237] : « [...] un appareil [qui] produit des altérations de matière qui peuvent être lues comme des traces[238] ». Il existe une

233 Lassègue, Jean et Longo, Giuseppe. What is Turing's Comparison between Mechanism and Writing Worth? Dans : Cooper, S. Barry, Dawar, Anuj et Löwe, Benedikt (dir.), *idem*, p. 456.

234 Précisons toutefois que Lassègue et Longo s'appuient sur un texte de 1950 de Turing, ce qui a quelque influence sur leur réflexions quant à ce qu'est un mécanisme. Turing montre beaucoup plus de précautions dans celui de 1937, comme nous allons le montrer. Pour le texte de 1950, voir Turing, Alan M. Computing Machinery and Intelligence. *Op. cit.*, p. 433-460.

235 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 195-225 ; Bachimont, Bruno. *Op. cit.*, 2010. «Le noème du numérique», p. 156-160.

236 Hayles, N. Katherine. *Writing machines*. Cambridge : MIT Press, 2002, p. 26.

237 « [...] *inscription technology* [...] ». Hayles, N. Katherine. *Idem*, p. 24.

238 « [...] *a device* [*that*] *initiate material changes that can be read as marks* ». *Ibidem*. Nous avons choisi de traduire *marks* par traces, car ce terme fait référence à un enjeu sémiotique, d'autant plus que Hayles utilise le même mot quelques lignes plus haut pour qualifier les mots du livre imprimé : des traces (*marks*) d'encre sur du papier.

corrélation – établie par { codage } – entre une différence de potentiel électrique et une inscription (0 et 1) qui va servir de base au { langage machine }. C'est en ce sens que les machines sont dites *écrivantes* : l'altération de la matière électrique peut être lue et interprétée par une machine selon un { codage } précis. C'est donc une définition limitée de l'écriture qu'elle applique aux machines, définition qu'on pourrait qualifier avec Yves Jeanneret de « logistique[239] ». Sur ce point, Katherine Hayles rejoint Bruno Bachimont : un ordinateur n'écrit pas mais ne fait que permettre une certaine manipulation des textes. Il ne peut y avoir d'écriture que s'il y a une interprétation et une lecture faite par un humain.

Si le débat semble être tranché, pourquoi alors revenir à Turing et à son texte de 1937 pour déterminer si une machine de Turing « écrit » et si par extension on peut dire qu'un ordinateur – comme réalisation effective des principes énoncés par Turing – « écrit » ? Il y a plusieurs raisons à cela. La première porte sur l'imaginaire de la machine de Turing, souvent décrite comme composée d'une tête de lecture et d'écriture, alors que Turing n'emploie jamais ces termes. Il y a là un paradoxe à éclaircir. La deuxième raison est d'ordre historique. Dans son texte, Turing lie { calcul }, écriture et mécanisation. Si nous voulons comprendre comment certains imaginaires combinatoires propres à l'écriture passent dans l'informatique, puis dans les pratiques d'écriture afférentes comme la { programmation }, il est nécessaire de retracer certaines étapes de cette histoire. Troisième et dernière raison, qui porte cette fois sur nos rapports aux médias informatisés quand nous leur dénions une activité comme l'écriture. Il nous semble qu'il y a là une forme de monopole humain de la production de sens, qui confine les ordinateurs à être de simples outils exclus du champ de la culture[240]. Replacer les médias informatisés dans le champ de l'écriture, non plus seulement comme outil mais comme entité ayant une part agissante propre, serait donc préfigurer un nouveau rapport à ces machines[241].

239 Jeanneret, Yves. *Op. cit.*, 2014, p. 11.

240 Ce que nous avons appelé avec Cléo Collomb, l'« anthropocentrisme ordinaire » de nos rapports aux machines, reprenant certaines thèses de Gilbert Simondon. Collomb, Cléo et Goyet, Samuel. Meeting the machine halfway : Vers une sémiopolitique de l'agir computationnel. Communication à l'occasion du colloque international *Reconfiguring Humans and Non-Humans : images, texts and beyond*. Université de Jyväskyllä, Finlande, 29-30 octobre 2015.

241 Ce sera l'objet de notre ultime chapitre, consacré à une sémiotique non-anthropocentrée des médias informatisés.

Dans cette partie, nous traiterons des deux premières raisons. La troisième fera l'objet de notre dernier chapitre. Pour l'instant, nous en restons à la généalogie de la combinatoire et nous montrons comment Turing formule un modèle théorique de l'informatique où un certain type d'écriture combinatoire joue un rôle prépondérant. Nous traitons de l'écriture du point de vue matériel (l'inscription de symboles) et lorsque nous parlons de machine, nous parlons du modèle de machine tel qu'imaginé par Turing en 1937.

3 B *« We may compare a man… »* Turing entre écriture et calcul

En 1937, le jeune Alan Turing publie dans les *Proceedings of the London Mathematical Society* un article publié un an plus tard et intitulé : « *On Computable Numbers, with an application to the entscheidungsproblem*[242] » : « Des nombres calculables, avec une application au problème de la décision ». C'est un article de logique, où Turing répond à un problème posé par David Hilbert et Wilhelm Ackermann en 1928, le problème de la décision[243]. Il n'est pas le seul, à l'époque, a avoir tenté de résoudre ce problème (I. 3. B. a). En revanche, il est le seul à faire en quelque sorte un détour et à imaginer une machine – la « machine de Turing » – capable de calculer tout ce qui est de l'ordre du calculable, ce qui lui permettra ensuite de répondre au problème de la décision (I. 3. B. b). Nous verrons alors comment Turing décrit sa machine, notamment en comparant son activité à de l'écriture (I. 3. B. c).

calcul, formalisme

Voir glossaire tome II, p. 3-25

a *On Computable Numbers* […] : un texte ancré dans la logique du début du siècle

Aujourd'hui, Turing est célébré comme l'inventeur de l'informatique, notamment grâce à cet article de 1937. C'est en tout cas ce que la vulgate historique a retenu. Mais les spécialistes conviennent que cet article prend place dans un contexte précis, celui du problème de la décision. C'est un problème ancré dans la logique du début du xx^{e} siècle. L'une des questions fondamentales de l'époque est d'établir, par la logique, le fondement universel des raisonnements mathématiques. C'est pourquoi toute une branche de cette logique va se concentrer sur la { formalisation } du { calcul }, sur la nature des symboles à utiliser et leurs manipulations possibles[244]. Les travaux d'Hilbert établiront l'arithmétisation possible de la logique mathématique, c'est-à-dire la possibilité de déterminer par le { calcul } la vérité des propositions et énoncés mathématiques. Cette arithmétisation n'est cependant possible qu'en formalisant les signes utilisés, c'est-à-dire en considérant que chaque inscription équivaut à un seul et même « type » { formel }. Par exemple, « A » vaut pour tous les « a » possibles, peu importe leur réalisation matérielle. De

242 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 230-265. Par commodité, nous abrévierons le titre de l'article en « *On Computable Numbers* […] ».

243 Le problème de la décision consiste à savoir si on peut déterminer par un algorithme – une suite d'opérations finies et mécanisables – la vérité d'une proposition mathématique. En d'autres termes, il s'agit de savoir si on peut fonder la vérité des propositions mathématiques par des moyens internes aux mathématiques.

244 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 196-199.

plus, on peut formuler des énoncés logiques sur les relations entre A et non-A sans que l'on ait besoin de définir à quoi renvoie le signe « A ». C'est le principe de { formalisation } du { calcul }. On peut alors faire une démonstration logique par le biais d'une écriture { formelle } et qui obéit aux règles de l'arithmétique. Le { calcul } devient dès lors une forme d'écriture[245].

calcul, formalisme

Voir glossaire tome II, p. 3-25

Dans ce cadre scientifique, d'autres auteurs ont formulé des thèses très proches de celle de Turing, comme réponse à ce problème de la décision. Alonso Church tout d'abord, dont le nom est fréquemment associé à celui de Turing en raison de l'équivalence logique de leurs thèses, mais ausi Emil Post qui va proposer en 1936[246] une théorie quasi identique à celle de Turing dans la formulation mais fort différente dans ses enjeux[247]. Turing n'est donc pas celui qui a formulé l'idée que le { calcul } peut être réduit à une écriture { formelle }. Il n'est pas non plus le seul à avoir pensé que le domaine du calculable puisse s'étendre au-delà des frontières de la logique. En revanche, il est celui qui a clairement formulé le problème de la calculabilité en terme de mécanisme : Turing imagine une machine dont le fonctionnement est comparable à un humain calculant. Si cette machine – dans le sens bien particulier que lui donne Turing – peut arriver à un résultat sans intervention extérieure pendant le { calcul }, alors le problème est considéré comme calculable. Le succès de la thèse de Turing s'explique donc autant par son inventivité scientifique que par sa force rhétorique[248] : il part du savoir intuitif de ce qu'est un nombre calculable et élabore un modèle facilement représentable et compréhensible. On pourrait donc dire que Turing imagine l'ordinateur « à la hussarde », comme solution éventuelle à un problème plus fondamental : la quête d'un fondement universel des mathématiques.

245 Il faut bien différencier calcul, arithmétique et logique mathématique. La logique mathématique est la discipline mathématique qui s'intéresse aux preuves et au fait de savoir si elles sont vraies ou non. Le calcul est une suite d'opérations qui arrive, depuis une situation de départ, à un résultat. L'arithmétique est plus largement tout ce qui concerne les nombres et leurs interactions possibles, notamment les règles de calcul. Quand nous disons qu'Hilbert arithmétise la logique, nous disons qu'il cherche à déterminer par un calcul la vérité d'une preuve.

246 Post, Emil L. Finite combinatory processes—formulation. *The Journal of Symbolic Logic*. 1936, vol. 1, n° 3, p. 103-105.

247 De Mol, Liesbeth. Generating, solving and the mathematics of Homo Sapiens. Emil Post's views on computation. Dans : Zenil, Hector (dir.), *A Computable Universe. Understanding and Exploring Nature as Computation*. River Edge : World Scientific, 2012, p. 45-62 ; Sieg, Wilfried, Szabó, Máté et Mclaughlin, Dawn. Why Post Did [Not] Have Turing's Thesis. Dans : Omodeo, Eugenio G. et Policriti, Alberto (dir.), *Op. cit.*, p. 175-208.

248 De Mol, Liesbeth. Generating, solving and the mathematics of Homo Sapiens. Emil Post's views on computation. Dans : Zenil, Hector (dir.), *Op. cit.*, p. 48.

Pourquoi alors retenir Turing, plutôt que Post ou encore Church ? Premièrement parce que nous ne sommes pas dans une thèse de logique ou d'histoire de la technique, mais en Sciences de l'Information et de la Communication. À ce titre, la validité scientifique d'une thèse ou sa cohabitation avec d'autres nous intéresse tout autant que le fait qu'elle se soit imposée comme le modèle dominant. En d'autres termes, c'est parce que le modèle de Turing est aujourd'hui dominant, qu'il est intéressant communicationnellement. Deuxièmement, Turing pose de manière centrale la notion de mécanisme et d'automatisme alors que Post va progressivement se déplacer vers la psychologie et les limitations propres à l'esprit humain[249]. Cette question de la machine étant importante pour nous, le modèle de Turing est plus intéressant. Troisièmement, le texte de Turing donne une plus grande place à l'écriture que celui de Post et pour cette raison Turing retient là encore notre intérêt.

b Le calculable et le scriptible

Turing s'intéresse aux « nombres calculables », définis en première approche comme « […] les nombres réels dont la forme décimale peut être calculée par des moyens finis[250] ». Quelques phrases plus loin, il précise cette définition : « […] un nombre est calculable si sa forme décimale peut être notée [*written down*] par une machine[251] ». Voilà déjà deux glissements significatifs : le { calcul } devient une forme d'écriture réalisable par une machine ; cette machine se caractérise par sa finitude[252]. Ce qui se calcule est ce qui peut s'écrire automatiquement. Ce glissement témoigne de l'influence sur Turing des travaux d'Hilbert en logique, qui avait déjà formulé cette idée que ce qui est calculable peut être compris comme de l'écriture, c'est-à-dire de la manipulation de signes sur une feuille de papier[253].

249 De Mol, Liesbeth. *Idem*, p. 53 ; Post, Emil L. Finite combinatory processes—formulation. *Op. cit.*, p. 105.

250 « […] *the real numbers whose expressions as a decimal are calculable by finite means* ». Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op.cit.*, p. 230.

251 « […] *a number is computable if its decimal can be written down by a machine* ». *Ibidem*.

252 Il en va avec cette finitude de la dimension algorithmique du problème de la décision : il faut déterminer par une suite d'opérations finies si un énoncé est un théorème du premier ordre. Turing revient à plusieurs reprises sur cette notion importante de finitude. Il la justifie dans un premier temps par le caractère limité de la mémoire humaine (p. 231) mais cette finitude porte autant sur les opérations de calcul de la machine que sur son mode opératoire propre, ce que Turing appelle ses « configurations ».

253 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 197.

calcul

Voir glossaire tome II, p. 3-25

Le caractère fini du { calcul } tient à la nature des nombres dont parle Turing. Par nombres « calculables », il précise qu'il va traiter des ensembles dénombrables[254], c'est-à-dire à tout ce qui peut être *compté*. Ce qui suppose une quantité finie d'opérations pour opérer ce dénombrement.

c Les descriptions de la machine de Turing et la comparaison avec l'écriture

À partir de cette définition du { calcul }, Turing va ensuite décrire plus finement sa machine et définir ce qu'il entend par « machine à calculer automatique » (*automatic calculating machine*). On trouve bien souvent des descriptions de cette machine comme composée d'une bande de papier divisée en cases, d'une tête de lecture et d'écriture. La tête peut se déplacer sur l'axe horizontal de la bande vers la gauche ou vers la droite, passant de case en case. Elle peut écrire sur cette bande des 0 ou des 1, ou les effacer. Cette description est partiellement vraie mais insuffisante pour nous car elle ne permet pas de comprendre les enjeux propres au texte de Turing et la façon dont il redéfinit ce qu'est écrire.

La description que fait Turing repose sur une comparaison entre un calculateur humain et un calculateur machinique, comparaison qui est faite à deux reprises dans l'article[255]. La première, plus technique, permet de bien comprendre quelques concepts clés de la machine. Un humain qui calcule un nombre réel est comparé à une machine « [...] seulement capable d'un nombre fini d'états q1, q2, ..., qR qui seront appelés "*m*-configurations"[256] ». Cette machine est équipée d'une bande (*tape*) qui est « [...] l'équivalent du papier[257] » (*the analogue of paper*) pour le calculateur humain. Cette bande est divisée en cases (*squares*). Chaque case peut contenir un « symbole[258] » S(r). La machine n'a accès qu'à un seul symbole à la fois. Elle passe de case en case et « scanne » le symbole de la case : « [l]e "symbole scanné" est le seul dont la machine est, façon de parler,

254 La notion d'ensemble dénombrable est construite par Georg Cantor, mathématicien de la fin du XIX^e^ siècle. Toute la difficulté de cette pensée est que le dénombrable ne s'oppose pas au fini. Il y a des ensembles dénombrables infinis. L'ensemble n des entiers naturels (1, 2, 3, etc.) est dénombrable : on peut compter chaque élément de l'ensemble sans qu'il y ait de doublons. En revanche, il est infini. Mais l'essentiel est que cet ensemble puisse être compté, quand bien même ce décompte est très long : il n'en reste pas moins qu'un ordinateur peut en théorie compter l'ensemble n.

255 Il faut rappeler qu'à l'époque, *computer* désigne l'employé de bureau chargé d'effectuer divers calculs. Les « machines à calculer automatiques » ont pour vocation, entre autre, de remplacer ces employés en mécanisant leurs opérations. La comparaison avec l'humain est donc très limitée. Il ne s'agit pas de démontrer que « la machine » peut « remplacer l'humain » ou être plus intelligente, mais il s'agit de voir si un travail spécifique – le calcul – peut être automatisé.

256 « [...] *a machine which is only capable of a finite number of conditions q1, q2, ..., qR which will be called "*m-*configurations"* ». Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 231.

257 « [...] *the analogue of paper* [...] ». *Ibidem*.

258 Les guillemets sont de Turing.

" directement consciente "[259]. » À chaque fois que la machine passe de case en case, elle change d'état, de « *m*-configuration ». Elle scanne alors la case sur laquelle elle se trouve. Pour chaque *m*-configuration, il y a donc un symbole S(r). Ce couple *m*-configuration et symbole (S(r)) est appelé par Turing une *configuration*. Ce que fait la machine est déterminé par sa configuration à un instant t, c'est-à-dire à son état et au symbole qu'elle lit : « […] la configuration détermine le comportement possible de la machine[260] ». Si la case est blanche, la machine peut noter (*writes down*) un symbole. Si un symbole est déjà noté, elle peut l'effacer. Elle peut se déplacer sur la bande, de gauche à droite. Selon Turing, « […] ces opérations incluent tout ce qui est nécessaire au calcul d'un nombre[261] ».

La seconde description est encore plus explicite. Turing part de l'idée que « Habituellement, on calcule en écrivant certains symboles sur du papier[262]. » Il continue la comparaison, en supposant que « […] ce papier est divisé en cases, comme dans les livres d'arithmétique des enfants[263] ». En revanche, alors que la bi-dimensionnalité du papier est « parfois utilisée […] en arithmétique élémentaire » (« *In elementary arithmetic the two-dimensional character of the paper is sometimes used* »), c'est pour lui une propriété accidentelle du { calcul } : « Un tel usage peut toujours être évité et je pense qu'on agréera du fait que la bi-dimensionnalité du papier n'est pas nécessaire au calcul[264]. »

Turing donne ensuite une seconde caractéristique du { calcul } : un nombre fini de symboles doit être utilisé[265]. Il ne donne pas un nombre précis, mais réflechit sur des bases décimales, comme le { calcul } humain qui utilise dix symboles (les chiffres de 0 à 9), donc une quantité finie de symboles. Ce qui ne veut pas dire qu'on ne peut pas les combiner pour créer une potentielle infinité de nombres. Dans ce cas, la seule difficulté est de reconnaître « d'un coup d'œil » (« *at one glance* ») la différence entre

259 « *The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". » Ibidem*.

260 « […] *the configuration determines the possible behaviour of the machine* ». *Ibidem*.

261 « […] *these operations include all of those which are used in the computation of a number* ». *Ibidem*.

262 « *Computing is normally done by writing certain symbols on paper.* » Turing, Alan M. *Idem*, p. 245.

263 « […] *we may suppose that this paper is divided into squares like a child's arithmetic book* ». *Ibidem*.

264 « *But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential to computation.* » *Ibidem*.

265 « *I shall also suppose that the number of symbols which may be printed is finite.* » *Ibidem*. D'ores et déjà, Turing utilise pour décrire le calcul humain des termes propres à sa machine : *print, symbols*… Il recadre en quelque sorte l'activité de calcul pour qu'elle puisse être mécanisable.

deux nombres, particulièrement s'ils sont très longs[266]. Chaque étape du { calcul } est déterminée par le symbole vu par celui qui calcule et son « état d'esprit à ce moment » (« *his "state of mind" at that moment* »). Le couplage symbole lu / état d'esprit est analogue à la « configuration » décrite page 231 de l'article. « Lire » un symbole, c'est le reconnaître au premier coup d'œil. Calculer, c'est donc inscrire certains symboles sur du papier, ces symboles devant être en nombre fini et idéalement interprétables au premier coup d'œil. On passe d'un symbole à un autre et chaque symbole lu détermine ce qui sera lu et calculé par la suite.

Mais de quelle nature sont ces opérations de { calcul } ?

> « *Imaginons que les opérations faites par le calculateur puissent être divisées en "opérations simples" qui soient si élémentaires qu'il est difficile de s'imaginer plus petites unités. Chacune de ces opérations consiste en un changement du système physique constitué par le calculateur et sa bande* [de papier][267]. »

calcul, discret, discrétisation formalisme

Voir glossaire tome II, p. 3-25

Le { calcul } selon Turing est donc éminemment { discret } : il est une suite d'opérations discontinues. Ces opérations sont élémentaires, au sens où chaque problème complexe doit être précédé d'un découpage qui permette d'en isoler les problèmes simples afin de rendre ce problème calculable. Dans un second temps, chaque opération simple est calculée puis réassemblée pour traiter le problème complexe. Le { calcul } est donc une méthode analytique (découpage en unités simples) et combinatoire (recombinaison des unités simples) fondée sur une { discrétisation } des opérations. Cette { discrétisation } est ancrée dans une physicalité du { calcul } comme écriture : chaque opération est une nouvelle configuration de l'ensemble constitué par un calculateur et son support d'écriture[268].

266 Turing glose ici sur la différence, dans le système de calcul occidental, entre chiffre et nombre. Nous utilisons dix chiffres, ces chiffres peuvent être combinés entre eux pour produire une infinité de nombres. Mais rappelons que chaque nombre est considéré comme un symbole unique par Turing, symbole qui occupe une case de papier. Pour la machine, 4872 est un symbole simple, au même titre que 1 : dans les deux cas, ils occupent une case. Dans ces conditions, certains nombres particulièrement longs sont effectivement difficiles à distinguer. Turing prend l'exemple de 9999999999999999 et 999999999999999, difficile à distinguer sans laborieusement cocher chaque 9 jusqu'à établir la différence entre les deux nombres.

267 « *Let us imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape.* » *Ibidem*.

268 Cette physicalité du calcul explique en grande partie pourquoi notre approche, parmi d'autres, peut être qualifiée de « matérialiste ». À partir du moment où Turing mécanise le calcul comme écriture, étudier les procédures de calcul, c'est étudier un certain état de matière.

Le { calcul } ainsi défini et la liste des opérations simples établies[269], Turing peut terminer sa démonstration :

> *« Nous pouvons désormais construire une machine pour faire le travail du calculateur[270]. À chaque état d'esprit du calculateur correspond une "*m-*configuration" de la machine. La machine scanne un nombre* B *de cases, ce qui correspond aux* B *cases observées par le calculateur. À chaque mouvement, la machine peut changer le symbole de la case scannée, ou peut changer de case dans un rayon limite de* L *cases. Le mouvement ainsi réalisé, ainsi que la configuration qui s'en suit, sont déterminés par le symbole scanné et la* m-*configuration[271]. »*

Turing fait une comparaison entre l'humain et la machine, en ce qui concerne le seul { calcul }. Cette comparaison repose sur des termes bien précis, qui laissent peu de place à la confusion entre la machine et l'humain. Quand l'humain est dans un certain « état d'esprit », la machine est dans une « configuration ». Elle « scanne » alors que l'humain « observe ». Enfin, comparaison qui n'est pas présente ici mais patente tout au long de l'article, la machine n'écrit (*writes*) pas mais elle note (*writes down*). Cette comparaison n'est possible qu'en vertu d'une { formalisation } du { calcul } afin qu'il puisse être mécanisable.

269 Elles sont au nombre de deux : l'écriture ou l'effacement – le cas échéant – du symbole inscrit dans la case observée ; le changement de case observée. Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Idem*, p. 246.

270 Ce qui confirme notre interprétation précédente : le calcul qu'il décrivait plus haut était le calcul effectué par un humain.

271 « *We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an "*m-*configuration" of the machine. The machine scans* B *squares corresponding to the* B *squares observed by the computer. In any move the machine can change a symbol on a scanned square or can change anyone of the scanned squares to another square distant no more than* L *squares from one of the other scanned squares. The move which is done, and the succeeding configuration, are determined by the scanned symbol and the* m-*configuration.* » *Ibidem*.

3 C L'écriture selon Turing

Une fois décrite cette machine et établie l'analogie entre { calcul } et écriture, il nous reste à comprendre comment se tissent ces liens entre { calcul }, écriture et machine. Qu'est-ce qui, dans la manière dont la machine fonctionne, lie l'écriture et le { calcul } ? Qu'est-ce qui fait du { calcul } une sorte d'écriture ? Comment qualifier cette sorte d'écriture et en quoi est-elle combinatoire ?

calcul, discret, discrétisation, écriture-calcul, formalisme, machine automatique

Voir glossaire tome II, p. 3-25

Ce qui caractérise la machine de Turing, c'est sa finitude (I. 3. C. a) et sa { discrétisation } (I. 3. C. b), caractéristiques qui se répercutent à tous les niveaux de son fonctionnement et qui font de cette machine une machine combinatoire (I. 3. C. c). Cette écriture, nous avions proposé de l'appeler « { écriture-calcul[272] } » (I. 3. C. d) et en analysons les principales qualités, en mettant l'accent sur l'automatisme et sa dimension combinatoire. (I. 3. C. e)

a Une machine finie…

La machine est dite « finie » au sens où son nombre de configurations est limité et donc déterminable par avance. En d'autres termes, on peut la programmer : préécrire l'ensemble des actions qu'elle peut réaliser. Ces actions sont également en nombre limité : changer de case, écrire sur la case scannée, etc. Le nombre d'opérations à mener lors d'un { calcul } est fini lui aussi. C'est le principe même de la résolution algorithmique : un problème est calculable si l'on trouve une solution grâce à un nombre donné d'opérations de { calcul }. Cela s'appuie sur le caractère dénombrable des nombres dont s'occupe la machine. Seul les ensembles dénombrables peuvent être calculés, c'est-à-dire recensés par une suite finie d'opérations { discrètes }. Le nombre de symboles que manipule la machine est également fini, puisqu'à en suivre Turing, pour qu'une machine soit qualifiée de machine « à calculer[273] », il faut qu'elle n'inscrive que deux types de symboles : « […] ceux du premier type […] sont composés exclusivement de 0 et de 1 (les autres étant appelés symboles du second type […][274] ». Le { calcul } de la machine de Turing est donc une forme d'écriture binaire, qui n'utilise que deux symboles élé-

272 Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Communication +1* [en ligne]. 2016, vol. 5, nº 2. [Mis en ligne le 13 septembre 2016] [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://scholarworks.umass.edu/cpo/vol5/iss1/2

273 « *computing machines* ». Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 232.

274 « […] *the first kind* […] *consists entirely of 0 and 1 (the other being called symbols of the second kind)* […] ». *Ibidem*.

mentaires[275]. Cette finitude est enfin applicable à la signification de ces symboles[276] (principe de { formalisation }). Un « symbole » au sens de Turing n'a qu'un seul sens possible : celui de déclencher chez la machine un mouvement. Cette non-ambiguïté est un héritage du { formalisme } hilbertien[277] : pour que le { calcul } soit mécanisable, il faut que chaque symbole ait une seule signification. Il faut un langage le plus univoque possible, afin qu'à chaque symbole scanné, qu'à chaque « configuration » de la machine corresponde un seul et unique comportement à adopter. Si un symbole est plurivoque, la machine ne peut décider par elle-même comment l'interpréter[278] et un humain doit intervenir. Aujourd'hui, on dirait qu'elle « buggue[279] ». La monosémie est donc la condition de l'effectivité informatique du signe.

La finitude de la machine permet l'automatisation de ses opérations : « Si, à chaque étape [du calcul], le mouvement de la machine […] est *complètement* déterminé par la configuration, nous pouvons appeler cette machine une "machine automatique"[280] ». Lorsque Turing emploie le terme de « machine », il parle exclusivement de { machines automatiques }, c'est-à-dire déterminées par leurs configurations. Or, elles ne peuvent être entièrement déterminées que si on peut décrire l'ensemble de leurs configurations, ainsi que chaque mouvement qui corresponde à ces configurations. Et chaque mouvement doit être déclenché par un symbole univoque. Si, par exemple une configuration x est décrite comme « si le symbole scanné est 1, déplace toi d'une case vers la droite puis passe aux instructions de la configuration y », il ne saurait y avoir d'ambivalence quant à la lecture du « 1 ». À chaque symbole doit correspondre un seul et même « comportement » à adopter de la part de la machine. Ainsi, il n'y a aucunement besoin qu'un humain intervienne pendant le déroulé du { calcul }. La machine est dès lors « automatique ».

275 Turing ne s'étend pas sur les symboles du second type, qui ne servent pas au calcul strictement dit mais à mémoriser certaines valeurs calculées. Ce sont des sortes de raccourcis pour la machine.

276 C'est ce que Bruno Bachimont appelle « l'autothéticité » du signe informatique : « les expressions informatiques ne posent […] que leur propre effectivité […] ». Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 201.

277 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Idem*, p. 196.

278 C'est pourquoi, du moins dans le texte de 1937, la machine ne peut faire preuve d' « intelligence », en ce sens très précis d'autonomie dans l'interprétation d'un signe plurivoque.

279 La notion d' « erreur » est complexe chez Turing, car fluctuante selon que l'on parle du texte 1937 ou de 1950 sur l'intelligence. Nous ne creuserons pas plus loin cette piste de réflexion.

280 « *If at each stage the motion of a machine* […] *is* completely *determined by the configuration, we shall call the machine an "automatic machine"* […] ». Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 232.

b … et discrète…

calcul, discret, discrétisation, écriture-calcul, formalisme

Voir glossaire tome II, p. 3-25

En ce qui concerne la { discrétisation } de la machine de Turing, elle porte sur cinq plans. Son support est découpé en cases observées une par une. Les symboles observés dans ces cases forment une unité { discrète }, quelque soit leur degré de complexité pour un œil humain[281]. Les phases d'observation sont elles aussi { discrètes } : la machine « saute » d'une case à une autre, sans continuité dans la perception. Les problèmes amenés à être résolus par la machine sont découpables en unités simples calculables. Enfin, les mouvements de la machine sont { discrets }. Tout comme ses observations se font de case en case, elle inscrit des symboles case par case et non pas d'un geste continu. Inscrire la chaîne 101 par exemple correspond à trois mouvements distincts, tout comme ses déplacements latéraux qui ne peuvent être faits que case par case.

c … condition d'une écriture combinatoire

I | 1 | A | a
caractère fini de tout ensemble combinatoire
Voir p. 84

En quoi { discrétisation } et finitude sont-elles liées à une pensée combinatoire ? En raison du caractère fini de tout ensemble combinatoire. Pour qu'il y ait combinatoire, il faut qu'il y ait un nombre fini d'éléments à permuter et une liste finie de règles de permutation. C'est à cette condition que peut s'organiser un jeu systématique de permutations. Sur quoi portent ces permutations dans la machine de Turing ? Sur des unités élémentaires et discrétisées (0 et 1), inscrites dans des cases et qui ne peuvent être scannés que l'une après l'autre. La nature même des problèmes calculables est d'être discrétisables. Ils peuvent être découpés en suite d'opérations élémentaires. C'est par ces deux aspects – finitude et { discrétisation } – que l'on peut dire que la machine de Turing relève de la combinatoire : tout ce qui est de l'ordre du calculable peut être ramené à un ensemble fini et ordonné de permutations d'unités { discrètes }. Ces permutations peuvent être formalisées par des règles logiques et donc réalisées automatiquement par une machine.

Reste l'épineuse question de l'écriture. La machine de Turing est la mécanisation du { calcul } comme écriture. Mais qu'entend-on précisément par « écriture » dans ce contexte ?

281 999 est pour la machine de Turing un seul symbole, au même titre que 1 ou 18, puisqu'il n'occupe qu'une seule case.

d « Écriture-calcul » et « écriture-texte »

Nous pouvons définir l'écriture de la machine de Turing comme une « { écriture-calcul[282] } » automatisée. Ce sont ces trois aspects – écriture, calcul, automatisation – qui font la spécificité du modèle de Turing.

Cette écriture possède cinq principes :
– Principe de { formalisation } : les symboles ne renvoient pas à des sons ou à des entités du monde réel mais à des abstractions logiques. Le fait que ces symboles résultent d'une mesure physique[283] est purement conventionnel[284]. Il n'y a pas d'interprétation entendue comme l'attribution à un signe d'un sens possible selon le contexte (voir principe suivant).
– Principe de monosémie : ces symboles sont immédiatement reconnaissables et renvoyent à une seule signification.
– Principe de { discrétisation } : la machine manipule des symboles élémentaires et { discrets } (1 et 0[285]). Elle inscrit sur un support également { discret }.
– Principe combinatoire : chaque symbole peut être combiné à un autre indéfiniment, dans les limites de la mémoire disponible (ici, la taille du ruban de papier).
– Principe d'unidimensionnalité : cette écriture se déploie sur une ligne et non en deux dimensions.

Les quatre premiers principes ayant été jusqu'ici traités, nous insisterons sur le principe d'unidimensionnalité, car c'est un point décisif quant à la différence entre l'écriture humaine et l'écriture machinique mais peu considéré par les théoriciens jusqu'alors[286]. La machine de Turing écrit dans un espace unidimensionnel : la ligne, formée par les cases découpées dans la bande (*tape*) sur laquelle écrit la machine. Quand il dresse son analogie entre le { calcul } tel qu'il est fait par un humain et tel qu'il peut être fait par une machine, Turing précise bien que, selon lui, la bidimensionnalité de

282 Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Op. cit.*

283 Par exemple, un courant de zéro ou cinq volts dans un transistor.

284 C'est le principe de « l'idéalité computationnelle » décrit par Bruno Bachimont. Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *E-dossiers de l'audiovisuel* [en ligne]. 2012. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.ina-expert.com/e-dossier-de-l-audiovisuel-l-education-aux-cultures-de-l-information/pour-une-critique-phenomenologique-de-la-raison-computationnelle.html.

285 Le binaire est conventionnel. Définitivement adopté dans les années 1950, il est choisi parce qu'il est plus rapide et simple pour la machine de fonctionner sur un langage binaire. Mais un ordinateur est plus fondamentalement une machine à états discrets qu'une machine binaire. La discrétisation est au principe de son fonctionnement, l'utilisation de symboles binaires est une commodité. Ce qui n'empêche pas qu'il y ait un important imaginaire autour du chiffre et du binaire, comme nous le montrerons dans le chapitre suivant.

286 Ni Bachimont, ni Kittler ni Lassègue et Longo ne font cas de cette remarque de Turing sur le caractère superfétatoire de la bidimensionnalité dans le calcul. Seul Ingold se penche sur la question, mais sans faire explicitement référence au texte de Turing. Ingold, Tim. *Lines: a brief history*. London : Routledge, 2007. Chapitre 6, « *How the line became straight* », p. 152-170.

l'écriture n'est pas essentielle au { calcul[287] }. Un { calcul } peut se résumer à une longue ligne d'écriture, qu'on lirait de gauche à droite en prenant chaque signe l'un après l'autre pour mener l'opération. C'est là une rupture fondamentale avec ce que Jack Goody a appelé la raison graphique, selon lui consubstantielle à l'écriture et dont une des formes privilégiées est le tableau[288]. Turing évacue cet aspect de la raison graphique. Le fait que l'on puisse écrire de gauche à droite, mais aussi de bas en haut, en cercle ou en boustrophédon n'est pas nécessaire au { calcul }. On peut calculer sans en passer, par exemple, par un empilement vertical de symboles ou par une représentation arborescente des différentes opérations effectuées[289].

e Une « écriture-calcul » automatique

Turing, dans *On Computable Numbers* [...], lie donc trois fils auparavant séparés : écriture, { calcul } et machine. En repartant des travaux d'Hilbert qui avait déjà avancé que le { calcul } pouvait être une forme d'écriture, Turing y adjoint la dimension machinique, c'est-à-dire la possibilité de sa reproduction automatique. Mais une écriture qui se trouve alors singulièrement redéfinie. Cette écriture est binaire, { discrète }, monosémique, { formelle } et unidimensionnelle. *A contrario*, un être humain écrira à l'aide d'un alphabet comportant de multiples signes, articulé en mots ayant plusieurs significations, ce qui en fait la richesse d'interprétation selon les contextes et cette écriture produira un texte qui occupe un espace bi, voire tridimensionnel[290]. Pour différencier ces deux types d'écriture, nous avons proposé avec Cléo Collomb la notion d' « { écriture-calcul } » (*computational writing*) opposée à celle d' « { écriture-texte[291] } » (*textual writing*). L'écriture de l'ordinateur, telle que Turing en a fixé le modèle, est essentiellement un { calcul } qui produit un résultat exploitable par une machine ou par un humain. Ce dernier au contraire, pratique une écriture intimement liée à la production d'un texte, c'est-à-dire à un objet matériellement attesté, organisation bidimensionnelle de signes interprétés selon son contexte de lecture. À cette notion d'{ écrit-

calcul, discret, discrétisation, écriture-calcul, écriture-texte, formalisme, machine computationnelle

Voir glossaire tome II, p. 3-25

287 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 232.

288 Goody, Jack. *Op. cit.*, 2007 [1979]. Chapitre 4, « Écriture et classification, ou l'art de jouer sur les tableaux », p. 108-139.

289 Ce qui ne veut pas dire qu'il n'y a pas de saisie visuelle des symboles, saisie qui passe nécessairement par une conscience du support comme espace. Turing traite ce problème dans son passage sur ce qu'est « observer » pour sa machine. Comme dans les autres passages du texte, il discrétise la vision. Il la définit comme une somme finie de lettres observées à la fois – ce qu'on appelle l'empan visuel – et dit que sa machine ne peut observer qu'un seul symbole à la fois. Voir Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 245.

290 Si on conçoit l'architecture comme une forme de textualisation, on peut envisager la déambulation comme une lecture dans un espace en trois dimensions.

291 Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Op. cit.*, p. 5-8.

ure-calcul }, nous serions tenté d'ajouter celle d'automatisme. Une { machine computationnelle } écrit, au sens d'une { écriture-calcul }-automatique : elle manipule des symboles inscrits sur un support magnétique selon des règles { formelles } sans qu'aucune autre entité n'ait besoin d'intervenir pendant le { calcul } pour que la machine arrive à un résultat. C'est cela l'apport décisif de Turing : le { calcul } devient une écriture mécanisable, c'est-à-dire automatisable au sens où les mouvements de la machine sont corrélés à une lecture et interprétation[292] de symboles.

Si nous gardons le terme d'écriture, c'est parce que l'étude du texte de Turing montre qu'il y a là une pensée de l'écriture et à trop insister sur les ruptures par rapport à d'autres écritures, on en oublierait qu'un certain nombre des caractéristiques de l'écriture présentée dans l'article ressortent de dynamiques plus anciennes dans l'histoire de l'écriture. La { discrétisation } des signes existe depuis – au moins – les premiers alphabets phéniciens du deuxième millénaire avant notre ère et la manipulation { formelle } de signes est liée de près à la vocalisation de cet alphabet par les Grecs, comme le soutiennent Jean Lassègue et Giuseppe Longo[293]. Leur caractère fini et combinatoire relève fort probablement de tout système d'écriture : c'est l'hypothèse que nous allons développer dans le chapitre à venir. Quant à la notation binaire, certains[294] la font remonter au *Livre des Transformations (Yi King)* au troisième siècle avant notre ère. Il en va de même pour certains modèles combinatoires qui sont de véritables « machines » à produire des énoncés censés êtres vrais, tout du moins du point de vue { formel }, mais aussi théologique – contexte oblige[295]. Il ne s'agit pas de { calcul }, encore moins d'{ écriture-calcul }-automatique. Il n'en reste pas moins que la combinatoire et le { formalisme } jouent dans les deux cas un rôle prépondérant. On peut donc renverser la perspective. Après avoir cerné ce qui fait la particularité de la machine de Turing, il s'agit de voir en quoi certaines de ses caractéristiques et surtout l'aspect { formel } et combinatoire, relèvent d'une histoire plus longue de l'écriture. Si bien qu'il est possible que l'informatique soit une science combinatoire précisément parce qu'elle utilise et maximise une propriété de tout système d'écriture.

292 Si minimale et univoque soit cette interprétation.

293 Lassègue, Jean et Longo, Giuseppe. What is Turing's Comparison between Mechanism and Writing Worth? Dans : Cooper, S. Barry, Dawar, Anuj et Löwe, Benedikt (dir.), *op. cit.*, p. 450-461.

294 Knuth, Donald E. Two thousand years of combinatorics. Dans : Wilson, Robin et Watkins, John J. (dir.), *Combinatorics: Ancient and Modern*. Oxford : Oxford University Press, 2013, p. 3-5.

295 Nous pensons ici à l'*Ars Magna* de Raymond Lulle, système combinatoire permettant de produire tout un ensemble d'énoncés à propos de Dieu, des anges, du monde physique, de l'être humain… et dont la visée finale était la conversion des non-chrétiens.

COMBINATOIRE, CHIFFRE, UNIVERSALISME. TROIS IMAGINAIRES DE L'ÉCRITURE INFORMATIQUE

II

Nous venons de montrer que les { API } sont le développement contemporain de pratiques d'écriture combinatoires en { programmation } : les { *subroutines* }. Nous avons également montré que l'abstraction { formelle } et la combinatoire sont le mode même de fonctionnement d'un ordinateur en tant que ces machines sont fondées sur un certain modèle d'écriture : l'{ écriture-calcul automatique }. La combinatoire est ainsi inhérente à l'écriture informatique, qu'elle soit envisagée du point de vue humain (les { *subroutines* }) ou machinique (l'{ écriture-calcul } de la machine de Turing).

API, écriture-calcul, formalisme, programmation, *subroutine*

Voir glossaire tome II, p. 3-25

Mais plutôt que de s'arrêter là et de considérer que l'informatique inaugure « [...] le dernier acte historique d'écriture[296] », nous allons montrer dans ce chapitre que l'informatique prolonge certains imaginaires liés à l'écriture. Plus précisément, l'informatique en tant que science occidentale construite sur une écriture { formelle } et combinatoire mobilise plusieurs imaginaires.

À rebours d'une approche rationnaliste qui ne verrait dans l'informatique que l'aboutissement de la logique débarassée de toute ambiguïté, nous faisons valoir que cette science et les objets

296 Kittler, Friedrich. *Op. cit.*, 2015, p. 30.

API, écriture-calcul, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

techniques qu'elle permet de produire (les ordinateurs et tout machine fonctionnant grâce à de l'{ écriture-calcul automatique }) sont pétris d'imaginaires. Ces imaginaires sont multiples, pluriels et parfois contradicatoires. Comme l'a montré Milad Doueihi, l'informatique croise autant une pensée scientifique moderne qu'une pensée religieuse voire mystique[297]. De ce point de vue aussi l'objet technique est « composite[298] » : c'est un nœud qui rassemble plusieurs façons de penser, de voir et d'imaginer le monde.

Le présent chapitre a pour objectif de mettre à jour certains de ces imaginaires informatiques, ce qui permettra de montrer dans le chapitre suivant en quoi ils travaillent les formes produites par les { API }. Nous explorerons ceux liés à l'écriture, en repartant des caractéristiques de l'{ écriture-calcul automatique } : c'est une écriture chiffrée, combinatoire, aux prétentions universelles[299]. Pour ce faire, nous remontons parfois loin dans le temps et prenons des exemples tirés d'aires culturelles variées. Il s'agit d'un chapitre d'inspiration plus anthropologique et nous pratiquons ce comparatisme afin de rendre visible cette pluralité d'imaginaires consubstantiels à notre objet, les { API }.

Notre première hypothèse porte sur le chiffre comme signe d'écriture (II. 1). Nous montrerons qu'il rassemble au moins quatre modes de représentations imaginaires : la calculabilité permet de construire des modèles prédictifs des phénomènes physiques ou sociaux ; sa puissance d'abstraction radicalise un mouvement initié par l'écriture ; mais cette abstraction permet également la mise en relation avec des entités non-humaines ; il permet enfin l'encryptage d'information et devient le contrat d'une relation passée entre êtres humains.

297 Doueihi, Milad. *Op. cit.*, 2008.

298 Le Marec, Joëlle. *Op. cit.*

299 Turing qualifie sa machine d'« universelle ». Nous montrerons en quoi cette prétention est liée à sa façon d'écrire.

Notre seconde hypothèse porte sur la combinatoire : peut-elle être considérée comme une « { technologie intellectuelle[300] } » et dans ce cas, quelle forme de pensée spécifique produit-elle (II. 2) ? Nous répondons à la première question par l'affirmative et montrons qu'elle produit deux choses : une métaphysique de l'épuisement, soit des tentatives d'épuisement systématique des possibles combinatoires tout en postulant une entité qui échappe à cette combinatoire ; le modèle d'une méthode scientifique analytico-combinatoire.

Notre troisième hypothèse reprend les acquis de la seconde et revient sur le qualificatif d'« universel » que Turing donne à sa machine. Nous soutenons qu'il existe un universalisme informatique qui est à mettre en lien avec la dimension combinatoire de l'informatique, précisément par ce que la combinatoire fait partie, en Occident, d'un universalisme scientifique dont la machine de Turing est une des dernières réalisations (II. 3).

Si ces deux dernières hypothèses font un détour par rapport à la question des { API }, elles sont essentielles pour la suite de notre propos. Car une fois démontré que l'informatique, parce qu'elle est fondée sur un modèle d'écriture combinatoire, est porteuse d'un certain type d'universalisme, nous montrerons les effets de cet imaginaire sur les formes que les { API } permettent d'écrire (chapitre III).

300 Robert, Pascal. Qu'est-ce qu'une technologie intellectuelle ? *Communication & langages*. 2000, nº 123, p. 97-114.

« J'ai entendu dire que, du côté de Naucratis en Égypte, il y a une des vieilles divinités de là-bas, celle-là même dont l'emblème sacré est un oiseau qui s'appelle, tu le sais, l'ibis ; le nom de cette divinité est Theuth. C'est donc lui qui, le premier, découvrit le nombre et le calcul et la géométrie et l'astronomie, et encore le trictrac et les dés, et enfin et surtout l'écriture. »

Platon, *Phèdre,* 274c. Traduction Luc Brisson. Paris : Flammarion, 2012.

1 IMAGINAIRES DU CHIFFRE

L'écriture informatique est une écriture chiffrée. Des 1 et 0 qui composent le { codage } élementaire de tout { calcul } aux outils de mesure des clics et autres statistiques d'affichage, le chiffre et la mesure traversent tous les niveaux de l'écriture informatique, depuis la manière dont une machine calcule jusqu'aux formes que permet de produire cette machine. Les { petites formes } permises par les { API } n'échappent pas à la règle, puisqu'il est systématiquement possible d'assortir un bouton Facebook ou Twitter du nombre de clics sur ce bouton[301]. Là encore, le chiffre est présent, même si c'est sous une modalité différente des 0 et des 1 du { langage machine }.

API,
binaire,
calcul,
codage,
langage machine,
petites formes,

Voir glossaire tome II, p. 3-25

Cette dimension chiffrée fait aujourd'hui partie des nombreux discours sur le numérique. Qu'on pense aux tunnels de 0 et de 1 pour illustrer ce qu'est « la révolution numérique », qu'on pense à l'usage managérial, économique ou politique des statistiques de consultation d'une page[302] et plus largement aux fantasmes de la mesurabilité généralisée des activités humaines par les technologies numériques : traçage des parcours, santé réduite à une suite de mesures, *quantified self*[303], etc. Pour autant, parle-t-on du même chiffre dans ces différents cas ? Quelle est la différence ou les liens entre le { langage binaire } du { calcul } et les chiffres des statistiques ? En quoi traduisent-ils des rapports différents au monde et à la technique ?

Notre but dans cette partie est de prendre le chiffre comme lieu d'élaboration d'un imaginaire de la technique et de ses possibilités et de montrer comment ces imaginaires travaillent les formes produites par les { API } web. Il s'agit à la fois de démêler l'écheveau de ces représentations tout en acceptant leur multiplicité. Nous adoptons pour cela un regard anthropologique sur le chiffre en tant que signe d'écriture. Nous partons du principe que le chiffre est un type de signe, présent dès les débuts de l'écriture, qui a acquis au cours de son histoire certaines fonctions, certains pouvoirs spécifiques par rapport aux caractères alphabétiques ou idéographiques.

301 Voir annexe nº 4.

302 Grignon, Thomas. L'expertise communicationnelle au prisme de ses instruments. L'exemple de Google Analytics. *Les Cahiers du RESIPROC*. 2015, nº 3, p. 23-47.

303 Quantified Self, 2017. *Wikipedia*. [En ligne] [mis en ligne le 26 mai 2006] [dernière modification 6 mars 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Quantified_self.

Cette enquête sur le chiffre comme écriture montrera quatre points. Un premier contact avec notre corpus, où le chiffre sert principalement à mesurer le nombre de clics sur des boutons, rend tentant de voir dans ce dernier une emprise de la statistique et de la mesure de nos activités à des fins notamment de gouvernance politique, ce que certains nomment la « gouvernementalité algorithmique[304] » (II. 1. A). La fascination pour la mesure chiffrée est essentielle dans cette nouvelle forme de pouvoir. Cependant, nous verrons que c'est là confondre assez rapidement { calcul } et statistique, justement parce que tous deux utilisent le chiffre comme moyen de notation. Or, l'{ écriture-calcul } dont nous parlons n'est pas celle qui intéresse les statisticiens. Les deux ne se jouent pas au même niveau. Il faudra donc reprendre la tâche à un niveau plus fondamental autour du calcul et non plus seulement de la mesure ou de la statistique. Nous défendrons la thèse que le chiffre prolonge le mouvement d'abstraction de l'écriture, autour du principe de { convertibilité }, du principe de position et de l'invention du zéro (II. 1. B). Troisièmement, le chiffre sert des pratiques mystiques et magiques (II. 1 C.). Il ne permet pas seulement la mesurabilité du monde physique, il permet aussi d'entrer en relation avec des entités métaphysiques. Enfin, le chiffre permet d'{ encoder }, de dissimuler une information et à cette fin devient le substitut technique d'un lien contractuel passé entre êtres humains (II. 1. D.). C'est la fonction de cryptage, bien connue en informatique et qui est visible dans notre corpus à travers les « { jetons } » (*token*) d'identification, essentiels pour le bon fonctionnement d'une { API }.

API,
calcul, codage, convertibilité, écriture-calcul, formalisme, jeton

Voir glossaire tome II, p. 3-25

304 Rouvroy, Antoinette et Berns, Thomas. Le nouveau pouvoir statistique. *Multitudes*. 2010, nº 40, p. 88-103.

1 A. Chiffre et statistique

À première vue, on pourrait penser que la question du chiffre dans notre corpus se réduit à celle du dénombrement et de la statistique. « 1200 *likes* », « 8 000 retweets », statistiques de consultation incluses dans le fonctionnement des cartes Twitter… Si cette dimension est indéniablement présente, ce serait réduire le rôle du chiffre et ce serait peut être rapidement confondre cet usage du chiffre avec celui des 0 et des 1 fait dans l'{ écriture-calcul } d'une machine. Il faudra donc commencer par distinguer chiffre, calcul, dénombrement et statistique (II. 1. A. a). Ce ne sont pas les mêmes opérations, pas les mêmes enjeux, pas les mêmes usages du chiffre. Ceci fait, nous pourrons nous consacrer pleinement à la statistique, qui est une certaine manière d'aborder les médias informatisés, leurs effets politiques et culturels (II. 1. A. b). Nous montrerons que cette approche, notamment celle de la gouvernementalité algorithmique, n'est pas la nôtre : en assimilant le { calcul } au dénombrement, elle empêche de poser de manière claire la question de l'{ écriture-calcul } et réduit d'emblée le champ des possibles signifiants du chiffre[305] (II. 1. A. c)

a Chiffre, calcul, dénombrement, statistique

Nous avons jusqu'ici employé les termes chiffre, dénombrement ou calcul sans tenter de les définir, ou tout du moins de les distinguer. Or, ils ne sont pas équivalents. Les chiffres sont des signes, représentant la plupart du temps une quantité. Un chiffre peut être ordinal ou cardinal[306], comme dans notre système européen moderne, mais un chiffre peut aussi exprimer une quantité indéfinie comme « beaucoup[307] ». Si l'invention du chiffre permet le dénombrement, c'est-à-dire le recensement au sein d'un ensemble fini des constituants de cet ensemble par l'attribution d'une place (premier, deuxième, etc.), tout dénombrement n'est pas pour autant calcul. Le calcul est la manipulation { formelle } des chiffres ou de tout autre signe censé représenter des quantités ou des rapports mathématiques[308]. La statistique, si elle se base sur le dénombrement et le calcul, ne leur est pas pour autant synonyme. La statistique est la science qui mesure et manipule selon des lois mathématiques des données issues notamment de faits sociaux. C'est en quelque sorte le volet administratif et politique

305 Par exemple en évacuant tout les usages mystiques et religieux du chiffre qui seront montrés plus loin.

306 Un chiffre ordinal exprime un ordre, une succession comme « premier » ou « deuxième ». Un chiffre cardinal exprime une quantité, comme « un » ou « deux ».

307 Ifrah, Georges. *Histoire universelle des chiffres : l'intelligence des hommes racontée par les nombres et le calcul.* Paris : Robert Laffont, 1994 [1981], vol. 1, p. 30-31.

308 Des calculs mathématiques utiliseront des symboles autres que les chiffres, par exemple π ou ∞.

du dénombrement. Être en mesure de compter quelque chose, de mesurer un phénomène, c'est se donner les moyens de l'administrer et d'en régler l'usage dans une société[309]. Nous considérons le chiffre comme un type spécifique de signes permettant des opérations intellectuelles comme le calcul ou le dénombrement.

algorithme
calcul,
petites formes,

Voir glossaire tome II, p. 3-25

b L'approche statistique : une narration dominante ?

Cette distinction faite, nous pouvons nous concentrer sur le cas de la statistique. Pourquoi tant de { petites formes } permettent de compter le nombre de clics des internautes ? En quoi cela est lié aux médias informatisés et à leur manière de fonctionner ?

La statistique est sans doute un aspect majeur des technologies numériques. Historiquement, un des premiers calculateurs, le Harvard Mark I, naît d'un contrat entre IBM et l'Université Harvard[310]. IBM était alors spécialisé dans la mise au point de calculateurs automatiques à destination des entreprises et administrations américaines, notamment pour réaliser des { calculs } démographiques. Si l'on prend la séquence chronologique antérieure à Turing (1800-1940), il est clair que la statistique joue un rôle important dans l'histoire de l'informatique[311].

Un pan de la recherche contemporaine s'attache à comprendre d'un point de vue historique et politique les effets de cet usage des statistiques par les médias informatisés. La thèse de la « gouvernementalité algorithmique » cherche ainsi à comprendre la gouvernementalité contemporaine « [...] en tant qu'elle tente de dire et de prédire les comportements (et intentions) [...][312]. Pour ce faire, ce mode d'administration du pouvoir repose d'une part sur la statistique, c'est-à-dire la mesure et le traitement de données sur les individus, données récoltées massivement dans des contextes divers ; d'autre part sur le { calcul } comme technique de mise en corrélation de données comportementales permettant la prévision de ces comportements. Selon Rouvroy et Berns, les médias informatisés jouent un rôle non négligeable dans le déploiement de ce nouveau type

309 Porter, Theodore M. *Trust in numbers: the pursuit of objectivity in science and public life*. Princeton : Princeton University Press, 1995 ; Desrosières, Alain. *Gouverner par les nombres*. Paris : Presses de l'École des Mines, 2008.

310 Cohen, I. Bernard et Welch, Gregory W (dir.). *Makin' numbers: Howard Aiken and the computer*. Cambridge : MIT Press, 1999.

311 Gardey, Delphine. *Écrire, calculer, classer : comment une révolution de papier a transformé les sociétés contemporaines (1800-1940)*. Paris : La Découverte, 2008.

312 Rouvroy, Antoinette et Berns, Thomas. Détecter et prévenir : de la digitalisation des corps et de la docilité des normes. Dans : Lebeer, Guy et Moriau, Jacques (dir.), *(Se) gouverner. Entre souci de soi et action publique*. Francfort : Peter Lang, 2010, p. 158.

de gouvernementalité, puisqu'ils s'appuient sur la notion de « trace numérique[313] » et offrent une puissance de manipulation de ces traces – et donc de prévision des comportements – inégalée[314].

Cette problématique spécifique du pouvoir renvoie plus largement aux rôles politiques et culturels des { algorithmes[315] }. L'idée centrale de ce champ émergent de la recherche est qu'un { algorithme } n'est pas un simple expédient technique mais automatise un certain nombre de conceptions historiquement et socialement situées des pratiques qu'il organise. Un des exemples les plus connus est un { algorithme } d'examen des images publiées sur Facebook afin de déterminer si ce sont des images pornographiques. Cet { algorithme } censure bien évidemment les images pornographiques au sens strict du terme, mais aussi bien souvent des peintures de maître où une partie charnue apparaît. Cet exemple montre que cet { algorithme } agit en fonction d'un modèle culturel de la nudité. De la même façon, Dominique Cardon a montré comment le PageRank (l'{ algorithme } de classement des pages web de Google) s'appuie sur, tout autant qu'il reconduit, une notion de « popularité » ou de la « réputation » héritée directement de la sociométrie des années 1930 et de la scientométrie des années 1960[316]. Or, la statistique est essentielle au bon fonctionnement d'un { algorithme }, car les données statistiques sont la base sur laquelle l'{ algorithme } va s'appuyer pour modéliser et prédire des comportements. { Algorithmes } et statistiques vont donc, dans ces théories, de pair.

V | **4** | **D**
un algorithme n'est pas un simple expédient ….
Voir p. 511 pour la critique de cette conception

313 Pour un état de l'art détaillé et une critique de la « gouvernementalité algorithmique » sur ce point, voir Collomb, Cléo. *Un concept technologique de trace numérique*. Thèse de doctorat. Compiègne : Université de Technologie de Compiègne et Bruxelles : Université Libre de Bruxelles, 2016, p. 101-170.

314 Rouvroy, Antoinette et Berns, Thomas. Le nouveau pouvoir statistique. *Op. cit.*, p. 88-103.

315 Gillespie, Tarleton. The Relevance of Algorithms. Dans : Boczkowski, Pablo J. et Foot, Kirsten A. (dir.), *Media Technologies*. Cambridge : The MIT Press, 2014, p. 167-194.

316 Cardon, Dominique. Dans l'esprit du PageRank. Une enquête sur l'algorithme de Google. *Réseaux*. 2013, nº 177, p. 66-68.

c La statistique : un cas particulier de l'écriture informatique

Dans tous ces travaux, la statistique tient une place prépondérante, que ce soit comme technique de base d'une nouvelle forme de gouvernementalité ou comme modélisation des pratiques sociales. Pour ces raisons, il est tentant de faire de la statistique la porte d'entrée quant à l'utilisation du chiffre et du { calcul } dans les médias informatisés. Or, comme l'a montré Cléo Collomb[317], ces narrations tendent à devenir monopolistiques et à promouvoir l'idée d'une emprise de la technique sur nos vies. Il nous semble donc nécessaire de proposer d'autres narrations, ce qui signifie donner à ces phénomènes (l'usage de la statistique dans les médias informatisés) plus de richesse, les replacer dans une histoire plus longue.

algorithme, binaire, calcul, écriture-calcul formalisme

Voir glossaire tome II, p. 3-25

De plus, notre thèse ne porte pas exactement sur ce niveau du rôle culturel de l'informatique. Nous nous intéressons à un niveau plus fondamental de l'écriture informatique, celui de l'{ écriture-calcul } automatisée. Et au niveau de cette écriture, un ordinateur ne calcule pas au sens statistique du terme. Il ne mesure pas des « traces » de pratiques sociales. Il mesure des différences de potentiels électriques, différences qui sont la base d'une abstraction logique ({ langage binaire }), abstraction qui permet **dans un second temps** d'effectuer des { calculs } statistiques. La statistique est donc un emploi particulier de l'{ écriture-calcul } automatisée. C'est une spécification contextuelle de ces machines dites universelles. Penser qu'un ordinateur est un outil proprement statistique, ou qu'il est l'outil privilégié d'une « digitalisation de la vie même[318] », c'est confondre le calcul au sens statistique du terme (enregistrement et calcul de traces de pratiques sociales) et la calculabilité au sens logique du terme (manière de représenter formellement tout ce qui peut être dénombrable). Les « données » que manipulent un { algorithme } ne sont en aucun cas les mêmes choses que les symboles notés par une machine. Bien sûr, le { calcul } { formel } permet aisément de représenter des modèles statistiques, puisque le but du { calcul } { formel } est de pouvoir traiter tout ce qui ressort des mathématiques. Entendu que la statistique est un domaine des mathématiques, on comprend que { calcul } { formel } et statistique sont des domaines très proches. Ils sont malgré tout distincts et ne doivent pas être confondus.

317 Collomb, Cléo. *Op. cit.*, 2016.

318 Rouvroy, Antoinette et Berns, Thomas. Le nouveau pouvoir statistique. *Op. cit.*, p. 90-91.

C'est pour cette raison que l'approche des imaginaires du chiffre par la statistique est « tentante ». Cette approche produit des théories fertiles et séduisantes sur le rôle culturel et politique du { calcul } algorithmique. Il ne s'agit pas de nier cet aspect statistique. Mais ces théories nous semblent problématiques. D'abord parce qu'elles évacuent tout un pan irrationnel de l'usage des chiffres (usages mystiques, divinatoires…). Ensuite parce qu'elles tendent à assimiler { calcul } et statistique, entraînant une confusion quant à la manière dont les médias numérique calculent, c'est-à-dire écrivent. Et puisque notre interrogation porte sur le chiffre comme écriture, cette confusion potentielle est éminemment problématique. C'est pourquoi nous la considérons comme l'un des multiples imaginaires à l'œuvre dans les usages du chiffre *via* et par les médias informatisés. Elle tient une place importante dans les imaginaires du chiffre, mais parmi d'autres imaginaires que nous allons désormais explorer.

II | **1** | **c**
usages mystiques, divinatoires…
Voir p. 174

1 B Le chiffre comme abstraction : convertibilité, principe de position, et invention du zéro

Notre système de numérotation moderne repose sur la conjonction de trois opérations intellectuelles : l'abstraction de la représentation graphique, permettant la commensurabilité (II. 1. B. a) ; le principe de position (II. 1. B. b.), permettant de se limiter à un ensemble de dix signes graphiques (de 0 à 9) ; l'utilisation d'un zéro comme représentation graphique du vide (II. 1. B. c.). Nous montrerons comment ces trois éléments sont à l'œuvre dans l'écriture informatique : parce qu'ils permettent la conceptualisation même d'une écriture { binaire }, mais aussi parce que grâce à eux le chiffre renforce l'abstraction permise par l'écriture (II. 1. B. d.), abstraction dont nous avons montré dans le premier chapitre qu'elle est un des points centraux de la généalogie des { API }.

API, binaire, convertibilité

Voir glossaire tome II, p. 3-25

a Commensurabilité et convertibilité : prélude à l'équivalence formelle du calcul

La première de ces opérations est la « correspondance unité par unité[319] ». Elle est le résultat d'une double manipulation : faire correspondre un élément de la réalité à un objet ou à une marque ; utiliser ces marques pour dénombrer d'autres choses que celle initialement comptée. Les premières formes d'écriture calculatoire ont par exemple été trouvées sur des os, sous forme d'encoches faites dans la matière. L'hypothèse dominante est qu'à chaque entaille correspond une tête de bétail[320]. Premier système de correspondance et donc de représentation, suivie d'une deuxième abstraction, corollaire de la première : l'appariement qui « [...] supprime la distinction qui existe entre deux ensembles du fait de la nature de leurs éléments respectifs[321] ». Une fois abstraite la notion de quantité (« quinze entailles = quinze bœufs »), on peut conserver uniquement la première partie du signe et l'assigner à un autre signifié (quinze entailles = quinze jours). C'est ce qu'on peut appeler le principe de **commensurabilité**. La commensurabilité repose sur l'articulation de deux systèmes sémiotiques. Le premier système est la mesure initiale.

319 Ifrah, Georges. *Op. cit.*, vol. 1, p. 42.

320 Selon Ifrah, les chiffres romains et le terme même de *rationem putare* (calculer) proviennent de cette pratique plus ancienne. *Rationem* vient de *ratio* (rapport) et *putare* signifie « retrancher par excision ». Calculer, c'est construire des rapports par excision de matière sur un support, ici le bois. Ifrah, Georges. *Idem*, p. 469.

321 Ifrah, Georges. *Idem*, p. 45

Dans notre exemple : quinze entailles (niveau du signifiant) valent pour quinze bœufs (niveau du signifié). Le principe de commensurabilité, c'est prendre ce niveau du signifiant pour en abstraire la seule quantité (quinze), quantité que l'on va pouvoir attribuer à d'autres signifiés, pour autant que les deux signifiés mis en relation d'équivalence par un seul et même signifiant soient tout deux dénombrables.

C'est la base de l'idée de { **convertibilité** }, permise par l'abstraction du chiffre. En permettant de noter un dénombrement indépendamment de la chose dénombrée, le chiffre permet de noter de la même façon des choses différentes et de poser leur équivalence sous un certain aspect : celui du nombre. Dans l'exemple précédent, on peut dire qu'il y a autant d'entailles que de bœufs et l'on peut alors utiliser ces quinze entailles pour compter d'autres choses : d'autres animaux, une durée, une quantité de céréales, etc. Par calcul, on pourra alors poser l'équivalence de choses de nature différentes et en permettre le commerce ou l'échange. On trouve une remarquable application de ce principe avec l'invention et la diffusion de la monnaie. À certains égards, l'informatique s'appuie sur cette abstraction du chiffre[322] mais à certains égards seulement. Nous montrerons dans la partie suivante que certes, le principe de { convertibilité } du dénombrable[323], est un principe important de l'informatique, mais que c'est surtout la notion d'équivalence logique qui est un aspect plus déterminant que la seule { convertibilité }.

II | **3** | **B**
principe important de l'informatique
Voir p. 228

322 Herrenschmidt, Clarisse. *Op. cit.*

323 Tout ce qui peut être dénombré est posé sur un plan d'équivalence.

b Principe de position et optimisation du système combinatoire

La seconde de ces opérations, c'est le principe de position, né dans les milieux savants de Babylone au II^e siècle avant notre ère[324]. Dans notre système, le chiffre « 2 » ne vaut pas toujours deux. Il peut valoir, selon là où il est placé dans la numérotation, deux ou vingt ou deux cents ou encore vingt milliards. Dans le nombre « 222 », le premier 2 en partant de la droite vaut $2x10^0$, soit deux (2x1) ; le second 2 vaut $2x10^1$, soit vingt (2x10) ; le troisième vaut $2x10^2$, soit deux cents (2x100). On obtient donc le résultat deux cent vingt-deux par ajout des trois puissances : 2+20+200. C'est le principe de base. Notre système est ainsi dit en base décimale, car il repose sur des puissances de dix. Le nombre de symboles est équivalent à la base : en base dix, on utilise dix symboles (de 0 à 9), en base deux, ou système binaire on en utilise deux, par exemple 0 et 1. « 222 », en système binaire, s'écrit donc : « 11011110 ».

binaire, calcul, discret, porte logique

Voir glossaire tome II, p. 3-25

Valeur binaire	1	1	0	1	1	1	1	0
Équivalent en puissance de 2	$1x2^7$	$1x2^6$	$0x2^5$	$1x2^4$	$1x2^3$	$1x2^2$	$1x2^1$	$0x2^0$
Valeur décimale	128	64	0	16	8	4	2	0

En base deux, « 11011110 » revient en base décimale à faire 128+64+8+4+2, soit deux cent vingt-deux. On le voit, la base deux est beaucoup plus gourmande en espace : il y a besoin de plus de signes pour exprimer la même valeur. Cette base est également peu lisible pour des humains habitués culturellement à manipuler des chiffres en base dix. En revanche, pour une machine fondée sur une suite d'états physiques { discrets[325] } telle qu'un ordinateur, cette base est bien plus efficace pour le { calcul[326] }. Et si cette machine a vocation à être utilisée par des humains, alors on comprend la nécessité de convertir ces formes de { calcul } en base décimale, bien plus intelligible pour des humains, comme nous venons de le faire par le tableau ci-dessus[327].

I 3 C
une machine fondée sur une suite d'états physiques
Voir p. 150

L'intérêt du principe de position est de permettre l'expression de grandes valeurs numériques avec une faible quantité de signes graphiques. Il a partie liée à la combinatoire et plus précisément à son économie de

324 Ifrah, Georges. *Op. cit.*, vol. 1, p. 770.

325 Nous entendons par là le fait qu'une machine informatique, en tant qu'elle est construite sur le modèle conceptuel de Turing, fonctionne par une suite de sauts entre opérations élémentaires, chaque opération étant équivalente à un changement d'état physique dans l'organisation de la machine, par exemple une circulation différente du courant électrique dans ses composants.

326 Turing, Alan M. Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE) [1946]. Dans : Carpenter, B. E et Doran, R. W, *A.M. Turing's ACE report of 1946 and other papers*. Cambridge : MIT Press, 1986, p. 21.

327 On comprend alors la nécessité d'en passer par des architextes, c'est-à-dire des outils qui permettent de faire le pont entre l'écriture-calcul de la machine et l'écriture-texte de l'humain.

moyens sémiotiques. Un système combinatoire utilise un certain nombre d'éléments de base. L'intérêt de ces systèmes est de permettre un grand nombre de combinaisons avec peu d'éléments, rapport qui est déterminé par certaines règles de permutation. En liant au chiffre, selon sa position dans l'espace, une valeur liée à l'expression d'une puissance, on peut, avec dix unités élémentaires, composer une quantité infinie de nombres.

c Invention du zéro et représentation de l'absence

La troisième de ces opérations est l'utilisation du zéro pour représenter une quantité vide. Le zéro, mis au point entre le IV^e^ et V^e^ siècle après notre ère notamment par la civilisation maya et dans le système de numérotation indien[328], est intimement lié au principe de position. En effet, lorsque l'on utilise le principe de position, l'espace est signifiant. « 22 » ne veut pas dire la même chose que « 2 2 ». C'est pourquoi il est nécessaire de trouver un moyen de représenter graphiquement le vide, d'où la notation du « 0 ». Le zéro est donc lié par voie de conséquence au principe de position et témoigne d'une forte abstraction, constitutive de l'écriture[329] : représenter l'absence. Or, pour la machine, de quel vide parle-t-on ? 1 et 0, la base du { calcul } machinique, représente des différences de potentiel : 1 est la représentation du passage, dans un transistor, d'un courant électrique de cinq volts. Le 0 est la représentation de l'absence de courant électrique. Mais pourquoi pas six volts ? Et que se passe-t-il si la tension du courant est de deux volts et demi ? Le potentiel électrique étant de l'ordre du continu et non du { discret }, il faut déterminer des seuils intégrés aux composants électroniques. Ces composants – une { porte logique[330] } par exemple – vont, selon le courant reçu, bloquer (0) ou laisser passer (1) le courant. Mais la valeur de la tension n'est pas aussi nette que zéro ou cinq volts. Pour que la machine fonctionne, il faut déterminer un seuil faisant que le composant, en dessous de cinq volts, considère qu'il s'agit d'une valeur nulle. Il s'agit donc d'une représentation bien **conventionnelle** de l'absence, utilisée pour engendrer une combinatoire { binaire } (1/0), condition de fonctionnement des médias informatisés en vertu du principe de non-ambiguïté du signe informatique .

I 3 C d
condition de fonctionnement des médias informatisés
Voir p. 153

328 Selon Ifrah, la première occurrence du zéro moderne se trouve dans le *Lokavibhâga*, un traité cosmologique daté de 485. Le zéro y est encore nommé « vide », « atmosphère » ou encore *ruponaka* : « privé de la forme ». IFRAH, Georges. *Op. cit.*, vol. 1, p. 930.

329 DERRIDA, Jacques. *De la Grammatologie*. Paris : Les Éditions de Minuit, 1967.

330 Une porte logique est un composant électronique qui laisse passer ou pas le courant selon les règles de la logique booléenne. Une porte « *AND* » (« ET ») ne laissera passer le courant que si elle reçoit de la part de deux circuits une valeur positive – un courant de cinq volts.

d Écriture, chiffre et abstraction

calcul, convertibilité, écriture-calcul, formalisme, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

Par ces trois caractéristiques – { convertibilité }, principe de position et le zéro comme représentation du vide – on peut arguer que le chiffre, dans son fonctionnement sémiotique, radicalise le mouvement d'abstraction combinatoire de l'alphabet, dans le sens où il permet de manipuler formellement (c'est-à-dire sur le même plan logique : celui du commensurable) des éléments du monde physique ainsi convertis en des unités logiques[331]. Mais cette conception du chiffre ou du calcul comme abstraction et système généralisé d'équivalence est le résultat de la radicalisation de l'abstraction permise en premier lieu par l'écriture[332]. Non pas que le chiffre ou le calcul en tant que tels soient des technologies de l'abstraction, mais plutôt que le chiffre, en tant qu'écriture, participe à ce que permet le déploiement de l'écriture alphabétique en Occident : une abstraction et une manipulabilité des éléments de l'écriture à travers un jeu combinatoire et des règles de permutation.

Jack Goody nous fournit un exemple parlant de cette thèse. Lorsqu'il se rend au Ghana chez les LoDagaa, il se rend compte qu'ils maîtrisent des techniques complexes de calcul. Mais lorsqu'il demande à quelqu'un de lui montrer comment compter, la réponse qu'il reçoit est : « compter quoi ? ». Goody en conclut que « [...] les procédés opératoires sont de nature plus concrète dans les sociétés sans écriture[333] ». Ce qui ne veut pas dire qu'un LoDagaa soit incapable de se représenter une quantité abstraite, indépendamment de ce qui est calculé. Mais que les méthodes de calcul effectifs sont très liées au contexte. Au contraire, Goody observe que la scolarisation et l'apprentissage de l'écriture « [...] a tendance à entraîner une plus grande "abstraction", une décontextualisation du savoir[334] [...] ». Le principe d'équivalence est donc certainement à l'œuvre dans tout calcul, comme le soutient Ifrah, mais la généralisation de ce principe est le résultat de l'écriture et de l'abstraction qu'elle permet.

331 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans : Detienne, Marcel (dir.), *Les Savoirs de l'écriture en Grèce ancienne*. Lille : Presses Universitaires de Lille, 1988, p. 387-419.

332 Goody, Jack. *Op. cit.*, 2007 [1979], p. 50-54.

333 Goody, Jack. *Idem*, p. 52.

334 *Ibidem*.

On mesure alors l'étendue de l'espace dans lequel le { calcul informatique } fait penser. Si le chiffre permet l'abstraction, parce qu'il est un certain type d'écriture et que l'écriture, de manière générale, favorise l'abstraction, l'informatique telle qu'elle s'est développée en Occident étend ce mouvement. L'{ écriture-calcul } informatique exploite en effet le principe d'équivalence que nous venons de présenter : elle suppose l'équivalence { formelle } des unités du { calcul }, indépendamment de leur sens dans un contexte précis. De ce point de vue, l'informatique en tant que { technologie de l'intellect } construit un espace mental favorisant les procédures d'abstraction, parce qu'elle se trouve au point de jonction entre l'écriture en général, le chiffre en particulier et le calcul en tant qu'utilisation spécifique du chiffre.

I | 3 | A | c
indépendamment de leur sens dans un contexte…
Voir p. 139

1 C. Mystiques du chiffre

La statistique et l'abstraction constituent donc deux des nœuds constitutifs du chiffre comme signe d'écriture. Au-delà de ces usages rationnels, nous allons montrer qu'il existe aussi des usages mystiques et magiques du chiffre. Il est en effet un moyen de se représenter, pour un humain, l'irreprésentable est d'accéder à une entité supérieure. En ce sens, le chiffre révèle ce qui est caché, ou en tout cas en donne une mesure qui permet d'apprivoiser, ce qui échappe à nos sens. Le savoir de cette mesure, la maîtrise des chiffres et du calcul sont alors potentiellement la source d'un pouvoir sur ceux qui en ignorent les arcanes. Comme le rappelle Clarisse Herrenschmidt « [...] la graphie par chiffres indo-arabes fut longtemps tenue pour magique[335] » en Europe occidentale du moins. Cette sous-partie sera dédiée à montrer cet imaginaire mystique, à travers deux exemples : l'utilisation dans la Mésopotamie antique des chiffres pour figurer la hiérarchie des dieux et l'ordre cosmique (II. 1. C. a) ; l'utilisation du zéro dans les calendriers mayas comme représentation du vide, en fonction d'attendus autant sémiotiques que religieux (II. 1. C. b). Ces deux exemples nous permettront de revenir sur l'imaginaire du { binaire } dans l'informatique et de préciser ce que nous entendons en liant, par le biais du chiffre, mystique et informatique (II. 1. C. c)

binaire, convertibilité

Voir glossaire tome II, p. 3-25

a Le chiffre comme respect de l'ordre cosmique

Le premier exemple est issu de la civilisation mésopotamienne.

En plus des usages comptables et astronomiques du chiffre, il en fut trouvé sur certaines tablettes du VII^e^ siècle avant notre ère des usages mystiques, qui s'étendent selon Ifrah tout au long du second millénaire et jusqu'au premier millénaire avant notre ère[336]. Sur ces tablettes, des nombres particulièrement significatifs sont attribués aux dieux du panthéon mésopotamien. À Anu, le plus important des dieux, est attribué le nombre 60, nombre qui occupe une place essentielle en raison du système sexagésimal mésopotamien[337]. À Sin, le dieu lunaire, correspond le nombre 30 puisqu'il est « le seigneur du déroulement du mois[338] ». À Gibil et Nasku, compagnons de Shamash, le nombre 10, car Shamash étant tributaire du nombre 20, il est accompagné de deux fois dix (Gibil et Nasku).

335 Herrenschmidt, Clarisse. *Op. cit.*, p. 315.

336 Ifrah, Georges. *Op. cit.*, vol. 1, p. 384.

337 Soixante est, dans un système sexagésimal, la base de calcul de tous les autres nombres, les mésopotamiens utilisant le principe de position.

338 Ifrah, Georges. *Idem*, p. 384.

On peut penser qu'il s'agit là d'un jeu savant et sans incidence. On peut aussi suivre Ifrah et Bottero, en y voyant l'expression d'une pensée cosmologique. Selon Ifrah, l'univers céleste était vu par les mésopotamiens comme un univers « numérologiquement harmonieux[339] ». Il ne s'agit donc pas tant d'attribuer des chiffres aux dieux que de restaurer, à travers l'écriture, l'ordre céleste et les qualités de telle ou telle divinité. Le cas de Sin est exemplaire à cet égard : le principe de { convertibilité } (30 = jours du mois = Sin) permet de construire des relations d'identité entre choses du monde physique et divinités. Le chiffre participe donc, dans ce contexte, à entretenir une cosmogonie et une partition de l'univers entre les humains et les dieux, tout en assurant la relation entre les deux.

b Le zéro et le « fardeau du temps »

Un deuxième exemple, dans un contexte bien différent, est la notation du temps dans le calendrier maya[340] et plus particulièrement la notation du zéro. C'est un sujet relativement complexe, car grâce à l'étude des codex et des stèles de l'époque, on a pu déterminer que les mayas utilisaient au moins trois calendriers : le *Tzolkin*, calendrier liturgique de deux cent soixante jours ; le *Haab*, calendrier civil trois cent soixante-cinq jours et enfin le « compte long », manière de compter le temps depuis la création du monde – dans la cosmogonie maya – soit le 11 août 3114 avant notre ère. Ces calendriers sont tous utilisés dans un contexte religieux[341]. Pour ne pas nous apesantir, nous nous concentrerons sur le « compte long » et l'utilisation du zéro dans ce mode du représentation du temps.

Le compte long est, à une exception près, en base vingt et utilise un système pseudo-positionnel, avec un zéro représenté sous forme de petite coquille. Il y a neuf unités de bases qui permettent de compter, chacune étant une puissance de vingt sauf la troisième (c'est l'exception). Ainsi, le *kin*, unité de base, vaut un jour (1x200). Un *uinal*, unité de deuxième ordre (équivalent de notre dizaine) vaut vingt *kin*, c'est-à-dire vingt jours (1x201). Le *tun*, unité de troisième ordre, vaut dix-huit kin, c'est-à-dire trois cent soixante jours (1x201x18), l'équivalent de notre année solaire (Ifrah suppose que le décrochage dans la base vingt permet de faire correspondre le compte long et le calendrier civil *Haab*, qui répartissait

339 Bottéro, Jean. Mésopotamie : l'écriture, la raison et les dieux. Paris : Gallimard, 1987 [1984].

340 La civilisation maya est une civilisation mesoaméricaine, qui au maximum de son expansion occupait un territoire compris entre le Honduras, le Mexique et le Guatemala actuel. On date leur période « classique » entre 250 et 900 après notre ère.

341 On sait par ailleurs que les mayas utilisaient pour les calculs ordinaires des *quipus*, brins de tissus sur lesquels on faisait des nœuds pour chaque unité ou décimale d'un nombre. Voir Ifrah, Georges. *Op. cit.*, vol. 1, p. 170-174.

l'année en trois cent soixante jours et cinq jours « maudits », les *uayeb*, le tout faisant notre année calendaire occidentale de trois cent soixante-cinq jours). Ensuite, la base vingt reprend : un *katun*, unité de quatrième ordre, vaut sept mille deux cents jours (1x202x18) et ainsi de suite jusqu'à *alautun*, unité de neuvième ordre qui désigne un cycle de vingt-trois milliards et quarante millions de jours, soit soixante-quatre millions d'années (1x208x18).

À chaque ordre de grandeur (*kin*, *uinal*, etc.) est attribué un dieu et donc un glyphe céphalomorphique, c'est-à-dire le visage de profil du dieu. Pour signifier combien de ces unités devaient être prises en compte dans le calcul d'une date, les mayas utilisaient des glyphes céphalomorphiques d'une autre série de dieux. Ainsi, chaque glyphe est composé d'au moins deux visages : l'un pour dire l'unité de grandeur, l'autre pour en exprimer le nombre[342]. Pour écrire une date en compte long, on empile donc de bas en haut et de droite à gauche les glyphes céphalomorphiques, en descendant dans l'ordre des puissances. Avec ce système, on peut représenter des nombres longs par un empilement graphique relativement économe, même si cognitivement exigeant.

342 On trouve également sur ces glyphes un système de bâtons et de points (chaque bâton vaut 5 et chaque point 1) ce qui permettait très certainement de rendre plus lisible les quantités signifiées.

Fig. 6. Un exemple de « compte long » maya. Plaque de Leyde (découverte en 1864, vers Puerto Barrios, Guatemala). À gauche (face antérieure de la plaque), un souverain maya. À droite (face postérieure), un décompte en compte long. Le premier glyphe en partant du haut est le glyphe représentant le dieu Yaxkin. Les quatre glyphes qui suivent correspondent au compte long, chacun ayant un glyphe céphalomorphique de l'ordre de grandeur assorti sur la gauche du nombre d'unités (un bâton = 5, un point = 1). Le compte est donc, de haut en bas : huit baktun, quatorze katun, trois tun, un uinal, douze kin. Ce qui correspond à un cycle de un million deux cent cinquante-trois mille neuf cent douze jours depuis la création du monde : (8x 144 000) + (14 x 7200) + (3 x 360) + (1 x 20) + (12 x 1). La plaque est donc datée de l'an 320 avant notre ère.

Source : Shook, Edwin W. Tikal Stela 29. *Expedition*. janvier 1960, vol. 2, n° 2, p. 34.

Le zéro pose un cas particulier. Il n'y a en effet pas besoin d'avoir un signe spécial pour le zéro, le système n'étant pas strictement positionnel. Si je veux écrire un cycle de sept mille deux cent quarante et un jours, il me suffit de trois glyphes : un *katun* (7 200) mélangé avec le glyphe du chiffre 1 (1x7200) ; un *uinal* (20) mélangé avec le glyphe du chiffre 2 (2x20) et enfin un *kin* (un jour). Il n'y a pas besoin de représenter la puissance de troisième ordre, *tun* (trois cent soixante jours). Pourtant, la plupart des stèles et des calendriers représentent le dieu correspondant, accompagné d'un coquillage plus ou moins stylisé représentant le zéro. Pourquoi cela ? Selon Ifrah, cela est dû au contexte religieux du compte long, à la concep-

tion du temps par les mayas et à la spatialité de leur écriture. Le temps est un fardeau cyclique, que chaque dieu se passe d'année en année. Mais lorsque le fardeau est nul, lorsqu'un cycle recommence, il ne faut pas pour autant supprimer le dieu de la stèle. Non seulement cela serait prendre le risque de courroucer le dieu, mais en plus cela brise l'ordre régulier des puissances qui s'empilent de bas en haut : le plus haut *alautun* la puissance de neuf, en dessous la puissance de huit, ainsi de suite jusqu'à *kin* l'unité élémentaire. Ainsi le zéro permet de ne pas supprimer l'image du dieu, image qui prend une fonction quasi-ontologique – c'est le dieu même que l'on attaque – mais aussi de conserver une organisation canonique des signes sur la stèle. Comme le résume Ifrah, « [...] le souci d'esthétique et la crainte mystique, ainsi que le caractère solennel et les exigences d'une " mise en page " particulièrement soignée des stèles mayas firent donc de l'invention du zéro une nécessité absolue[343]. » Nous n'irions pas jusqu'à parler de « souci esthétique » ou de « nécessité absolue », mais il est clair que l'image du chiffre, en d'autres termes le signe le représentant, possède en ce contexte une puissance ontologique et qu'écrire un nombre en compte long est une manière de rendre hommage aux dieux tout en réinsérant cette écriture dans un cycle cosmique plus long.

343 Ifrah, Georges. *Idem*, p. 753.

c Une mystique de l'informatique ?

On pourrait, à la manière d'Ifrah, multiplier les exemples et offrir un début d'anthropologie comparée des chiffres et de leurs usages religieux. De la fascination indienne pour les énigmes cosmologiques impliquant de grands chiffres[344] à l'*Oudjat* égyptien (œil d'Horus arraché lors de son combat contre Seth), symbole religieux qui sert d'unité de comptage du grain[345], en passant par la vision de la « Mesure du Corps » dans la littérature juive de la *merkaba*[346], de nombreuses utilisations du chiffre ne tiennent pas de pratiques comptables ou statistiques. Elles sont liées à des pratiques divinatoires, mystiques, par lesquelles l'humain accède à certaines réalités intangibles.

Confiner l'écriture { binaire } des machines à de l'{ encodage } { formel }, c'est donc évacuer cette dimension mystique des chiffres et plus particulièrement du zéro. Au contraire d'une vision rationnaliste de l'informatique et de l'{ écriture-calcul } qui consisterait à n'y voir que le produit de la logique { formelle }, nous rappellons par l'anthropologie que cette écriture particulière s'enracine dans une culture bien plus large que la science moderne et contemporaine. Ce qui ne veut pas dire que nous défendons l'idée d'une mystique du numérique, ou que l'ordinateur soit une nouvelle forme de divinité. Nous affirmons simplement que la représentation du { code informatique } comme suite de 0 et de 1 puise dans des imaginaires anciens, multiples et qui n'excluent pas une forme de pensée tout autre que le contexte d'émergence du langage { binaire }.

binaire, codage, code, écriture-calcul, formalisme

Voir glossaire tome II, p. 3-25

344 Ifrah, Georges. *Idem*, p. 942.

345 Ifrah, Georges. *Idem*, p. 411.

346 La *merkaba* – terme hébreu voulant dire « chariot » – est une branche de la mystique juive. On trouve plus particulièrement dans les livres des *Heykhalot*, « *Livre des Palais* », écrits entre le Ve et VIe siècle, la description d'un voyage mystique passant par sept palais et aboutissant à la *Shiur Qomah*, la vision du Corps divin, aux dimensions fantastiques. Voir Scholem, Gershom. *Les grands courants de la mystique juive : la Merkaba, La Gnose, la kabbale, le zohar, le sabbatianisme et le hassidisme*. Paris : Payot, 1968 [1950]. Chapitre II : « La mystique de la merkaba et de la gnose », p. 53-94.

1 D. Chiffre et confiance

Le chiffre est un moyen de rendre secret, d'encrypter une information pour ne la rendre lisible qu'à ceux ou celles qui en ont le code (II. I. D. a). Le chiffre devient alors vecteur de confiance, substitut technique des liens que peuvent entretenir des humains entre eux, ce que nous montrerons en prenant l'exemple des *tokens* d'identification dans les { API } web (II. I. D. b).

algorithme, API, application, codage, développeur, hexadécimale, jeton, requête

Voir glossaire tome II, p. 3-25

a L'encryptage comme « technologie de la confiance »

L'encryptage consiste à masquer une information par la construction d'une version codée, grâce à des chiffres ou des caractères alphabétiques. Cet encodage peut faire l'objet d'une automatisation par un { algorithme }, rendant le code plus ou moins facile à décrypter. C'est tout l'enjeu de la cryptographie informatique : trouver des manières d'encrypter suffisamment robustes pour que les systèmes informatiques soient fiables afin de faire transiter des informations sensibles : identifiants bancaires, mots de passe, messages privés, etc.

Comme l'a montré l'historien des sciences Theodore Porter, le chiffre est une « technologie de la confiance[347] » (*technology of trust*). Porter s'intéresse à la science occidentale et aux usages politiques de la statistique. Il montre, dans son livre *Trust in numbers*, que l'usage du nombre et plus spécifiquement de la mesure (d'un phénomène physique ou social) contribue largement à établir un consensus sur la chose mesurée, voire à en établir l'existence sociale. Selon Porter, le chiffre est dans notre société un signe au fonctionnement sémiotique et social particulier, puisqu'il atteste de la réalité d'une chose. Si c'est mesurable, cela doit exister. Le chiffre et la mesure sont des « technologies de la confiance » : ils sont des marqueurs auxquels on se fie et autour desquels on peut fonder des politiques scientifiques, administratives, sociales, etc.

347 Porter, Theodore M. *Op. cit.*

b Le système des jetons dans les API web, ou la matérialité de la confiance par l'encryptage

Ce fonctionnement du chiffre comme technologie de la confiance, confiance renforcée par un encryptage préalable, est présent dans les { API } contemporaines sous la forme du *token*, ou « { jeton } d'identification ». Lorsqu'un { développeur } utilise l'{ API } de Twitter ou de Facebook pour écrire une { application }, il doit au préalable s'identifier personnellement auprès de ces plateformes, ainsi qu'identifier l'{ application } qu'il va créer. La plateforme lui délivre alors un « { jeton } », une clé { hexadécimale[348] } qui permet d'accéder à certaines informations, selon la nature de ce { jeton } et le mode d'identification utilisé. À chaque fois que le { développeur } va formuler une { requête } à l'{ API }, il devra fournir dans sa { requête } sa clé – son *token* – afin que la { requête } aboutisse. Il en va de même lorsqu'un utilisateur s'identifie auprès d'une { application }. Une clé est générée – celle de l'utilisateur – et utilisée par l'{ application } pour récupérer des informations sur l'utilisateur ou effectuer des opérations en son nom. Il s'agit donc en quelque sorte d'un système tripartite qui engage la plateforme, un { développeur } et les utilisateurs de l'{ application }. Ce dernier s'identifie, il reçoit une clé pour lui, ainsi que pour son { application } (premier niveau). Ensuite, quand un internaute utilise cette { application }, une clé est générée pour chaque utilisateur et est prise en charge par l'{ application } (second niveau). Il faut donc s'imaginer que chaque { développeur } possède à la fois sa clé personnelle et un jeu de clés utilisateurs, qu'il va pouvoir utiliser pour faire agir une { application } à leur place.

Le { jeton } d'identification est un système de **reconnaissance** et de **délégation** central dans la construction de la confiance par la plateforme. C'est une forme de **reconnaissance** par le système même d'{ encodage }. Lorsque le serveur reçoit la { requête } { API }, la suite alphanumérique est lue et authentifiée par la plateforme. La plateforme reconnaît donc la clé qu'elle a générée et, sous condition de cette reconnaissance, autorise certaines actions et l'accès à certaines informations. Le terme métaphorique de « clé » est éloquent : la plateforme est l'équivalent d'une serrure et le { jeton } l'équivalent d'une clé. Il faut en quelque sorte que la serrure « reconnaisse » les formes de la clé pour que la porte s'ouvre. Le { jeton } des { API } fonctionne sur le même principe.

348 L'hexadécimal est un système de notation en base seize, qui mélange les dix chiffres ordinaires et les six premières lettres de l'alphabet.

C'est ensuite une forme de **délégation** car l'utilisateur confie au dispositif une certaine puissance d'action (publier des textes, effacer du contenu, etc.). Il y a donc une confiance qui s'installe et se matérialise techniquement à travers le { jeton }. Mieux que matérialiser, le { jeton } se substitue à l'assentiment de l'utilisateur. Une fois l'autorisation donnée, le { jeton } tient lieu de cette autorisation. Il fait donc mieux qu'incarner un lien de confiance, il remplace la volonté de l'utilisateur.

API, codage, développeur, jeton,

Voir glossaire tome II, p. 3-25

De ce point de vue, le *token* d'identification est un moyen d'attester de certains liens de confiance, par un système de reconnaissance d'un code partagé. Il est bien une technologie de la confiance, mais au sens d'un objet qui vient remplacer tout autant qu'incarner des liens de confiance établis entre deux entités. En cela, le *token* s'inscrit dans une histoire plus longue des substituts techniques à la relation de confiance[349]. Il sert à matérialiser un contrat passé entre une entreprise (Facebook, Twitter…) et un { développeur }. Par sa forme même – la façon dont il est encrypté – il assure la reconnaissance mutuelle des parties engagées et permet l'accès à des données ou à des fonctionnalités. En cas de rupture des termes du contrat par une des parties, il est très facile d'agir sur le { jeton } afin de bloquer l'accès au { développeur } ayant trahi la confiance de la plateforme. Même s'il existe d'autres modalités pour construire cette confiance (charte graphique, recommandations d'ordre éthiques…), le { jeton } y joue un rôle central, précisément parce qu'il est la concrétisation sémio-technique de ces liens. Il est un objet sur lequel on peut agir en fonction de l'évolution du contrat passé entre la plateforme et les { développeurs }.

IV 4 A
charte graphique…
Voir p. 407

Le chiffre et plus particulièrement le chiffrement au sens d'un { encodage } par ces signes particuliers que sont les chiffres, sont dans ce cadre une manière de construire la confiance dans l'objet[350]. Parce que le { jeton } fait l'objet d'un encryptage efficace, il est fiable et peut prétendre attester de la relation passée. Les { API } web, par le biais de l'{ encodage } des { jetons } d'identification, mobilisent donc cet imaginaire du chiffre comme technologie de la confiance.

349 Qu'on pense par exemple au *symbolon* grec, analysé par Marcel Détienne comme l'un des moyens de tenir un discours vrai, ou en tout cas d'attester auprès d'autrui la vérité de son discours. Detienne, Marcel. *Les maîtres de vérité dans la Grèce archaïque*. Paris : Pocket, 1995 [Paris : Maspero, 1967].

350 Nous nous inspirons de l'approche de Loeve et Normand, qui ont montré comment certains objets techniques ne sont pas intrinsèquement dignes de confiance, mais que cette confiance se construit à travers des médiations et selon la vie triviale de cet objet. Loeve, Sacha et Normand, Mickaël. How to Trust a Molecule? The Case of Cyclodextrins Entering the Nanorealm. Dans : Zülsdorf, Torben B., Wienroth, Matthias, Coenen, Christopher, *et al.* (dir.), *Quantum Engagements. Social Reflections of Nanoscience and Emerging Technologies* Berlin : AKA Verlag, 2011, p. 195-216.

Au cours de cette enquête, nous avons pu cerner quatre imaginaires ou fonctions du chiffre. Certaines de ces fonctions sont étroitement liées à l'informatique et apparaissent directement dans notre corpus (le { codage } chiffré comme substitut de liens de confiance ; la mesure statistique). D'autres sont présentes de façon plus lointaine et rejoignent des dynamiques anthropologiques qui ont trait plus largement à l'écriture : renforcement de la pensée abstraite ; moyens de mise en relation avec des entités divines. En montrant cette pluralité, nous avons pu déjouer une approche rationnaliste des liens entre chiffre et informatique, en commençant par une vision statistique qui confine le chiffre à la seule mesure. En partant d'une remarque d'ordre factuel (l'écriture informatique utilise le chiffre) nous avons exploré la pluralité des imaginaires afférents. Attelons-nous désormais à une seconde caractéristique de l'écriture informatique : le fait qu'elle soit une écriture combinatoire.

Partant du fait que l'écriture informatique utilise largement le chiffre, nous avons interrogé plus en profondeur cette caractéristique. Au-delà du simple aspect statistique qui constitue une porte d'entrée quant au rôle du chiffre dans notre corpus, nous avons montré que les chiffres font appel à de nombreux imaginaires – parfois contradictoires – et remplissent différentes fonctions selon leur contexte d'utilisation. Parmi ces imaginaires et fonctions, nous en avons isolé trois. Le premier est lié à l'abstraction permise par l'écriture[351] et nous avons montré que certains fonctionnements du chiffre – principe de position, de { convertibilité }, invention du zéro – renforcent cette abstraction. Le second est tout autre et à trait aux usages mystiques et divinatoires du chiffre dans certaines cultures. Le troisième est la fonction du chiffre comme vecteur de confiance, à travers l'exemple de la cryptographie.

convertibilité

Voir glossaire tome II, p. 3-25

351 Mouvement d'abstraction par ailleurs relevé dans le chapitre précédent.

Il est temps à présent de s'intéresser à une seconde caractéristique de l'écriture informatique : le fait qu'elle soit une écriture combinatoire. Notre hypothèse pour cette sous-partie est que la combinatoire est une « { technologie intellectuelle } » qui produit des formes spécifiques de pensée. Pour démontrer cette hypothèse, nous commencerons par rappeler que la combinatoire est un trait constitutif de tout système d'écriture (II. 2. A). Mais nous irons plus loin, en démontrant que l'aspect combinatoire de l'écriture est en soi une { technologie intellectuelle } (II. 2. B). En ceci elle produit des formes de rationnalité spécifiques selon les contextes où elle est mobilisée. Parmi ces formes de rationnalité, nous nous demanderons ce que la combinatoire a produit de particulier en Occident. Nous proposerons deux pistes : une mystique de l'épuisement (II. 2. C) et un modèle analytique de méthode scientifique (II. 2. D). Ce dernier point nous permettra de montrer dans la partie suivante en quoi ce modèle de méthode est lié à un universalisme informatique.

API, calcul, écriture-calcul, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

2 LA COMBINATOIRE COMME TECHNOLOGIE INTELLECTUELLE. ENTRE SCIENCE DES ÉLÉMENTS ET ÉPUISEMENT DU POSSIBLE.

La seconde caractéristique de l'{ écriture-calcul }, telle que nous l'avons théorisée à partir du modèle de Turing, réside dans le fait que c'est une écriture combinatoire. Nous allons désormais nous concentrer sur cet aspect de l'{ écriture-calcul }, qui participe de la conception même d'une { API }. Cela va nécessiter de faire un détour par rapport à notre objet. Une première partie montrera que cet aspect combinatoire n'est pas propre à l'{ écriture-calcul } mais que c'est un trait constitutif de tout système d'écriture (II. 2. A). Nous démontrerons par ailleurs que la combinatoire est une { technologie de l'intellect } (II. 2. B) et permet donc de faire émerger des modes de pensée et des formes graphiques spécifiques. Nous prendrons comme exemple le *Yi King* chinois.

Une question surgit dès lors : quels sont les modes de pensée spécifiques que l'écriture combinatoire a permis de développer en Occident et dont l'informatique est tributaire ? Pour répondre à cette question, nous identifions deux de ces modes de « pensée combinatoire ». Le premier repose sur le fait que la combinatoire institue un rapport particulier entre le fini – mais dénombrable – des permutations et ce qui échappe radicalement à ces combinaisons (II. 2. C.). Il y a donc un jeu qui se crée, par le calcul des possibles, entre l'épuisable (ce qui peut être découvert par l'ensemble des combinaisons) et l'inépuisable ou l'infini. Le second, étroitement lié au précédent, est que la combinatoire de l'écriture fournit un modèle de pensée scientifique, fondé sur la décomposition de problèmes en unités élémentaires. L'alphabet grec est à ce titre un système d'écriture qui tire au mieux parti de cette propriété combinatoire (II. 2. D.). Ces deux modes de pensée combinatoire permettent d'expliquer en partie pourquoi les pratiques d'écriture où l'aspect combinatoire est fortement valorisé ont à voir soit avec la quête d'une méthode universelle de production de savoirs, soit avec la quête d'une totalité inaccessible à l'humain (Dieu, le monde, le langage, etc.).

Une fois établi ce lien entre l'écriture combinatoire comme { technologie intellectuelle } et l'émergence d'une certaine pensée scientifique, nous pourrons alors revenir dans une troisième partie (II. 3) sur l'universalisme de la machine de Turing, en formulant l'hypothèse suivante : « l'universalisme » annoncé de cette machine est un projet scientifique fruit du développement concomitant de la pensée scientifique et d'une pensée combinatoire. Ce développement concomitant se fait par l'entremise d'une certaine conception de l'écriture. Cela nous permettra ensuite dans toute la seconde partie de la thèse d'analyser les effets de cet universalisme scientifique combinatoire sur les { formes-textes } produites et mises en circulation par les { API }.

API, forme-texte, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

2 A. La combinatoire comme trait constitutif de l'écriture

Nous partons de la proposition théorique que tout système d'écriture est un système combinatoire[352]. Ou plus précisément encore : la combinatoire est un trait constitutif de tout système d'écriture, ce qui suppose que combinatoire et écriture ne sont pas équivalents mais que l'une (l'écriture) suppose l'autre (la combinatoire). Cette idée est difficile à prouver empiriquement (cela demanderait une étude comparée impossible à mener ici) mais elle est assez facile à comprendre si l'on repart de la définition de la combinatoire donnée dans la première partie de ce travail. La combinatoire est l'art des permutations possibles entre les éléments simples d'un ensemble. Un système combinatoire repose sur deux attendus : un nombre fini d'éléments, bien souvent les plus élémentaires possibles ; un ensemble de règles de combinaisons entre ces éléments afin de déterminer par le calcul l'ensemble des permutations possibles. Tout système d'écriture comporte en effet un nombre fini d'éléments simples : les idéogrammes, phonogrammes ou pictogrammes qui constituent les briques élémentaires du texte à venir[353]. C'est ce qu'on appelle, dans notre système occidental, l'alphabet. L'alphabet est la liste des unités élémentaires de notre système d'écriture. Tout système d'écriture comporte également des règles de permutation, qui donnent un certain sens à une combinaison donnée, en interdisent d'autres, etc. C'est l'équivalent pour nous de la grammaire, mais il existe bien d'autres façons d'autoriser ou d'interdire certains usages combinatoires de l'alphabet[354]. Tout écriture suppose donc une suite finie de symboles et des règles de permutation entre ces symboles.

I | 1 | A | a
un ensemble de règles de combinaisons…
Voir p. 84

Une fois acceptée cette proposition, nous insistons sur le fait que nous parlons de l'écriture **en tant que système** et non en tant que pratique située, dans un esprit et un corps agissant selon certains attendus culturels et historiques. Dans un premier temps, il s'agit donc d'abstraire le système de son contexte d'application : dire que la combinatoire est un trait constitutif de tout système d'écriture ne revient pas à dire que toutes les

352 Cette idée est notamment défendue par Anne Zali et Emmanuël Souchier, dans leur séminaire annuel « Chemins d'écriture » tenu à Paris-Sorbonne depuis 2014.

353 On peut objecter que certains idéogrammes ou pictogrammes ne sont pas des éléments « simples » mais sont déjà des éléments complexes. C'est tout à fait vrai, mais cela ne remet pas en cause le fonctionnement combinatoire de l'écriture. On ne fait alors que remonter d'un niveau vers les éléments les plus simples possibles, ce qui est l'exercice même d'une pensée analytico-combinatoire.

354 Qu'on pense par exemple aux mots tabous, qui ont une validité grammaticale mais sont interdits pour d'autres raisons : religieuses, politiques…

cultures où l'écriture tient un rôle important exploitent ce principe combinatoire. Il y a certains contextes, certaines pratiques, certaines traditions où l'aspect combinatoire de l'écriture est fortement mis en avant, il y en a d'autres où il est minoré. Par exemple, la linguistique moderne utilise largement le principe combinatoire de l'écriture pour en faire un modèle de la langue[355]. La combinatoire est donc une constante structurelle de toute écriture, mais ce qui varie c'est la **valorisation** et l'**exploitation** de la combinatoire. Ce jeu entre des constantes d'un système (combinatoire, « pensée de l'écran[356] », « image du texte[357] », implication du corps écrivant, utilisation d'un outil d'inscription…) et des variables définit ce qu'est une pratique d'écriture, mobilisant les deux aspects (système et contexte) en même temps.

355 La théorie de la « double articulation » développée par André Martinet est d'ordre combinatoire : on part d'une liste finie de phonèmes, pour arriver par des combinaisons à une liste quasi-infinie de morphèmes. Voir Calvet, Jean-Louis. Double articulation. Dans *Encyclopædia Universalis* [en ligne]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.universalis.fr/encyclopedie/double-articulation/.

356 Christin, Anne-Marie. *Op. cit.*

357 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

2 B. La combinatoire, une technologie de l'intellect

Si nous considérons que la combinatoire est un trait constitutif de tout système d'écriture et que la plupart des systèmes d'écriture connus (anciens ou contemporains) possèdent des pratiques combinatoires plus ou moins codifiées, on peut désormais se demander dans quelle mesure la combinatoire donne lieu – tout comme l'écriture – à des formes spécifiques de pensée et de notation graphique. En somme, il s'agit d'étudier la combinatoire comme { technologie de l'intellect }. Après avoir défini ce qu'est une { technologie de l'intellect } grâce à un état de l'art sur la question (II. 2. B. a), nous en tirerons deux conclusions principales : qu'il y a une matérialité de la pensée et une co-constitutivité de cette dernière avec ses outils (II. 2. B. b). Il y a donc des formes de pensées spécifiques aux outils mobilisés. Nous montrerons alors en quoi la combinatoire est une { technologie de l'intellect } (II. 2. B. c), prenant l'exemple du *Yi King* (II. 2. B. c).

algorithme, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

a Qu'est-ce qu'une technologie de l'intellect ?

Pascal Robert, dans son article « Qu'est-ce qu'une { technologie intellectuelle } ?[358] », dresse une généalogie de ce concept. Le terme de « { technologie de l'intellect } » est né dans les années 1970, autour de deux auteurs : Jack Goody dans son livre *The Domestication of the Savage Mind* publié en 1977 (traduit en 1979 sous le titre *La raison graphique*) et Daniel Bell en 1973 dans son livre de 1973 *The Coming of Post-Industrial Society*, traduit en 1976 sous le titre *La Naissance de la société post-industrielle*. Tous deux s'intéressent aux rapports entre les moyens de communication et les transformations culturelles ou cognitives dans une société donnée. Mais les enjeux de leur question varient légèrement, d'où une différence conceptuelle entre ce que Goody nomme « *technology of the intellect*[359] » et ce que Bell appelle « *intellectual technology*[360] ». Bell s'interroge sur la manière dont l'informatique peut aider à « […] gérer la complexité croissante de notre société[361] ». C'est pour cela que Bell insiste sur la notion d'{ algorithme }, qui permet d'automatiser le traitement de l'information et d'en optimiser la gestion. C'est une définition de la { technologie intellectuelle } à mi chemin entre l'ingénierie et le management, très liée à un outil spécifique : l'{ algorithme } informatique.

358 Robert, Pascal. Qu'est-ce qu'une technologie intellectuelle ? *Op. cit.*, p. 97-114.

359 Goody, Jack. *Op. cit.*, 2007 [1979], p. 48, p. 10 dans l'édition anglaise de 1977. Les traducteurs français ont choisi l'expression « technologie intellectuelle ».

360 Bell, Daniel. *The Coming of Post-Industrial Society; a Venture in Social Forecasting*. New York : Basic Books, 1973, p. 29.

361 Robert, Pascal. Qu'est-ce qu'une technologie intellectuelle ? *Op. cit.*, p. 98.

algorithme, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

Jack Goody comprend la { technologie de l'intellect } comme tout « moyen de communication » qui opère une transformation des modes de pensée[362]. L'enjeu n'est pas de comprendre les mutations de la société contemporaine, mais de proposer un cadre théorique qui prenne en compte les opérations matérielles dans l'évolution des structures de pensée, en premier lieu desquelles la naissance et l'adoption de l'écriture. Non seulement le concept de Goody est plus extensif que celui de Bell, mais il ne traite pas de la « gestion de la complexité ». La { technologie de l'intellect } au sens de Goody est donc bien plus un concept anthropologique que managérial.

En revanche, il est clair que certaines { technologies de l'intellect } (au sens de Goody) permettent cette gestion de la complexité (au sens de Bell). Une carte en est un bon exemple : elle permet de représenter un ensemble complexe et inaccessible directement à nos sens, la forme d'un continent par exemple. Cette opération de représentation, qui passe donc nécessairement par une médiation sémiotique, permet alors de « gérer » cette complexité que ce soit au sens administratif du terme (qu'on pense au cadastre d'une ville) ou au sens cognitif du terme (je peux me représenter ce qu'est être européen, africain, américain, etc.). C'est en ce sens que Pascal Robert définit la { technologie intellectuelle }, comme un « [...] outil régulé de gestion du nombre (de la complexité) opérant une traduction de l'évènement en document grâce à l'opération de conversion des dimensions[363] ». La définition de Pascal Robert est donc en quelque sorte une synthèse de celles de Goody et de Bell, ne se limitant pas à un { algorithme } mais incluant la question de la complexité comme centrale.

Par ailleurs, nous ne serions pas complet si nous ne mentionnions pas deux autres occurrences de termes proches de celui de « { technologie intellectuelle } », l'une chez Pierre Bourdieu et Jean-Claude Passeron en 1970, l'autre chez Stéphane Mallarmé autour de 1870. La première occurrence est relevée par Marie-Anne Paveau[364], dans une étude sociologique de Bourdieu et Passeron sur l'école. Les deux auteurs notent que « [...] l'institution scolaire relègue objectivement au dernier rang de sa hiérarchie l'inculcation méthodique des techniques matérielles et intellec-

362 Goody, Jack. *Op. cit.*, 2007 [1979], p. 48.

363 Robert, Pascal. Qu'est-ce qu'une technologie intellectuelle ? *Op. cit.*, p. 103.

364 Paveau, Marie-Anne. Bourdieu me manque. À propos de la technologie intellectuelle. Dans : *La pensée du discours* [en ligne]. Billet du 8 octobre 2010. [Mis en ligne le 8 octobre 2010] [Dernière mise à jour le 20 février 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://penseedudiscours.hypotheses.org/1218

tuelles du travail intellectuel et du rapport technique à ces techniques[365] ». Les auteurs suggèrent que l'activité intellectuelle ne saurait avoir lieu sans outils matériels (les « techniques matérielles et intellectuelles »), mais que cette matérialité de l'activité intellectuelle est dévalorisée par le système scolaire au profit d'une « intellectualité pure » et abstraite[366]. Mallarmé quant à lui écrit le terme de « { technologie intellectuelle } » dans des notes de travail consacrées à ses études sur le langage et plus précisément sur la linguistique naissante. Selon Mallarmé, le fait que la langue, par la linguistique, devienne à la fois **objet** et **instrument** de connaissance en fait une « { technologie intellectuelle[367] } » qui permet de classer les autres sciences. Là encore, la { technologie intellectuelle } de Mallarmé est proche des réflexions de Goody sur les liens entre langage, écriture, écriture du langage et naissance d'une pensée scientifique[368].

b Matérialité & co-constitutivité de la pensée

Que retient-on de ces différentes conceptualisations des « { technologies de l'intellect } » ?

– Que « l'activité intellectuelle n'est pas pure, et qu'elle est à la fois collaborative […] et distribuée dans les environnements humains et non humains[369] ». Il y a une **matéralité de la pensée** et mieux que cela : les opérations intellectuelles sont indissociables des outils matériels qui les permettent.

– Qu'il existe, au plan anthropologique, un **co-constitutivité de la pensée et des outils**. C'est ce que l'école de Compiègne a nommé la « thèse TAC[370] » : la technologie anthropologiquement constitutive / constituante.

365 Bourdieu, Pierre et Passeron, Jean-Claude. *La reproduction : éléments pour une théorie du système d'enseignement*. Paris : Les Éditions de Minuit, 1970, p. 153, note 21. Cité par Paveau, Marie-Anne. Dans : *op. cit* [en ligne].

366 On notera à quel point Bourdieu et Passeron sont proches de Goody sur cette articulation entre matérialisation et abstraction de la pensée, et sur la valorisation de cette abstraction dans certaines sociétés.

367 Mallarmé, Stéphane. Notes sur le langage, folio 10, cité par Ruppli, Mireille et Thorel-Cailleteau, Sylvie. *Mallarmé : la grammaire & le grimoire*. Genève : Droz, 2005, p. 62.

368 Goody, Jack. *Op. cit.*, 2007 [1979]. Chapitre 2, « Des intellectuels dans les sociétés sans écriture ? », p. 61-84. Sur ce sujet précis, on consultera le travail très approfondi de Sylvain Auroux sur la grammaire et la mise par écrit du langage et de ses règles : Auroux, Sylvain. *La révolution technologique de la grammatisation. Introduction à l'histoire des sciences du langage*. Liège : Mardaga, 1994.

369 Paveau, Marie-Anne. Bourdieu me manque. À propos de la technologie intellectuelle. Dans : *op. cit.* [en ligne]. Nous soulignons.

370 Steiner, Pierre. Philosophie, technologie et cognition. État des lieux et perspectives. *Op. cit.*, nº 53-54, p. 7-40.

L'humain ne crée pas d'outils comme application préalable d'une pensée abstraite, mais les représentations mentales se forment au fur et à mesure – et en même temps – que l'utilisation d'outils. Il n'y a donc pas d'humain sans technique, ni l'inverse. L'être humain s'humanise à travers la technique et la technique en retour modifie la façon d'être humain.

– Que par conséquent tout changement dans l'organisation matérielle du travail intellectuel (nouvelles méthodes, nouveaux outils, etc.) transforme ce travail intellectuel, ses formes et ses résultats. Il existe donc des modes de pensée spécifiques aux outils que nous utilisons.

c En quoi la combinatoire est-elle une technologie de l'intellect ?

algorithme, graphe, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

En quoi peut-on dire que la combinatoire est une { technologie de l'intellect } ? Est combinatoire tout système de permutations réglées entre un nombre fini d'éléments, ces permutations étant alors prévisibles par le calcul. Une { technologie de l'intellect } est un outil de gestion de la complexité, notamment par la transformation de cette complexité en des dimensions adaptées aux sens humains, outil qui produit des formes de pensée spécifiques.

Notre hypothèse est que la combinatoire joue précisément ce rôle : elle permet de réduire une complexité incertaine (parce que non réalisée ou trop grande pour pouvoir être appréhendée) à un calcul. Lorsque, pour prendre un exemple simple, nous pouvons déterminer en amont – avec l'aide de quelques règles simples – la totalité des résultats possibles d'un jet de deux dés à six faces, nous réduisons une complexité incertaine (le hasard du lancer) à un ensemble intellectuellement compréhensible que l'on peut répartir dans un tableau. Il n'y a pas besoin de lancer les dés pour savoir toutes les combinaisons possibles. Et surtout, une fois déterminé la règle pour déterminer ces combinaisons, rajouter un, deux, trois ou mille dés ne change pas radicalement la situation : on peut toujours déterminer l'ensemble des combinaisons possibles sans avoir jamais à les réaliser effectivement par un lancer de dés. Cela prend simplement plus de temps de noter toutes les combinaisons possibles, mais la règle reste identique. Le calcul combinatoire permet donc de se représenter un ensemble complexe, de l'appréhender spatialement et intellectuellement par une réduction de la complexité à des dimensions adaptées à l'être humain. En cela, la combinatoire est bien une { technologie de l'intellect }.

I 1 A b
on peut toujours déterminer...
Voir p. 85

On peut alors regrouper tout un ensemble de productions culturelles par leur dimension combinatoire et montrer le type de rationnalité que produit la combinatoire. C'est ce que fait Donald Knuth, dans l'introduction de *Combinatorics : Ancient and Moderns*, une histoire de la combinatoire d'un point de vue de l'histoire des mathématiques. Comme il le soutient, la combinatoire est certes une pensée mathématique au sens moderne du terme, mais elle est surtout une condition de possibilités de systèmes de pensée, de courants esthétiques, scientifiques… Elle a donc un rôle fondamentalement culturel. Le système de prosodies des chants védiques, fondé sur des combinaisons entre syllabes longues et syllabes courtes, a permis de mettre au point des { algorithmes } explorant toutes les combinaisons possibles, ainsi de pouvoir organiser une véritable poétique selon les rythmes adoptés. Toujours selon Knuth, la typologie grecque de la métrique en poésie est un autre exemple d'un système combinatoire élémentaire (syllabes longues /syllabes courtes) qui a eu une influence majeure sur l'émergence de formes culturelles comme la poésie shakespearienne, la musique d'Athanase Kircher[371] ou encore celle de Marin Mersenne (1588-1648) qui, dans son *Traité de la voix et des chants* (1636), met sous forme de notation musicale les sept cent vingt permutations possibles à partir des six notes de la gamme (do, ré, mi, fa, sol, la). À cet égard, la table des matières de *Combinatorics : Ancient and Moderns* est exemplaire. En commençant par un chapitre sur les chants védiques indiens du 1er millénaire avant notre ère, en passant par la combinatoire rabbinique du Moyen-Âge, nous finissons par les techniques plus modernes d'énumération et ce que l'on nomme la « théorie des { graphes[372] } ».

III **3** **A** **a**
graphes
Voir p.311

371 Knuth, Donald. Two thousand years of combinatorics. Dans Wilson, Robin et Watkins, John J. (dir.). *Op. cit.*, p. 7-8.

372 Un graphe, en mathématique et en informatique, est la représentation d'un objet ou d'un problème sous la forme de nœuds reliés par des flèches. On peut alors représenter en termes simples des systèmes complexes comme un ensemble de relations sociales ou les échanges énergétiques au sein d'un système, ou le nombre de mouvements possibles au sein d'un système. Cette même théorie des graphes est au cœur de l'API de Facebook et du concept de « graphe social », puisqu'elle permet de représenter de façon computable et manipulable l'ensemble des relations qu'entretient un membre du réseau social avec d'autres membres.

La combinatoire ainsi située dans l'histoire de l'écriture, on peut l'aborder en termes communicationnels et non strictement mathématiques. Que des modèles algorithmiques modernes se retrouvent dans des traités mathématiques indiens du XIII^e siècle est une chose, la vie sociale de ces traités en est une autre. Expliquer la permanence historique des procédés combinatoires par la pertinence ou l'exactitude de leurs calculs nous semble limité, car le fruit d'une conception rationaliste qui voit dans la véracité logique ou l'effectivité mathématique l'explication principale de la circulation d'une idée ou d'une méthode. Du point de vue communicationnel, c'est la question de la { technologie intellectuelle } qui nous intéresse. Une complexité (l'ordre de l'univers ; l'ensemble des notes possibles ; la totalité des savoirs) est rendu accessible et manipulable par un ensemble de règles (règles de permutations et de combinaisons, règles sociales autour des pratiques combinatoires) et cette complexité fait l'objet d'une transformation, d'un changement de forme.

technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

L'ordre du monde est alors donné à lire en un schéma de soixante-quatre hexagrammes ; l'ensemble des poèmes possibles en quelques feuilles de papier savamment découpées ; Dieu tient dans l'espace d'une page :

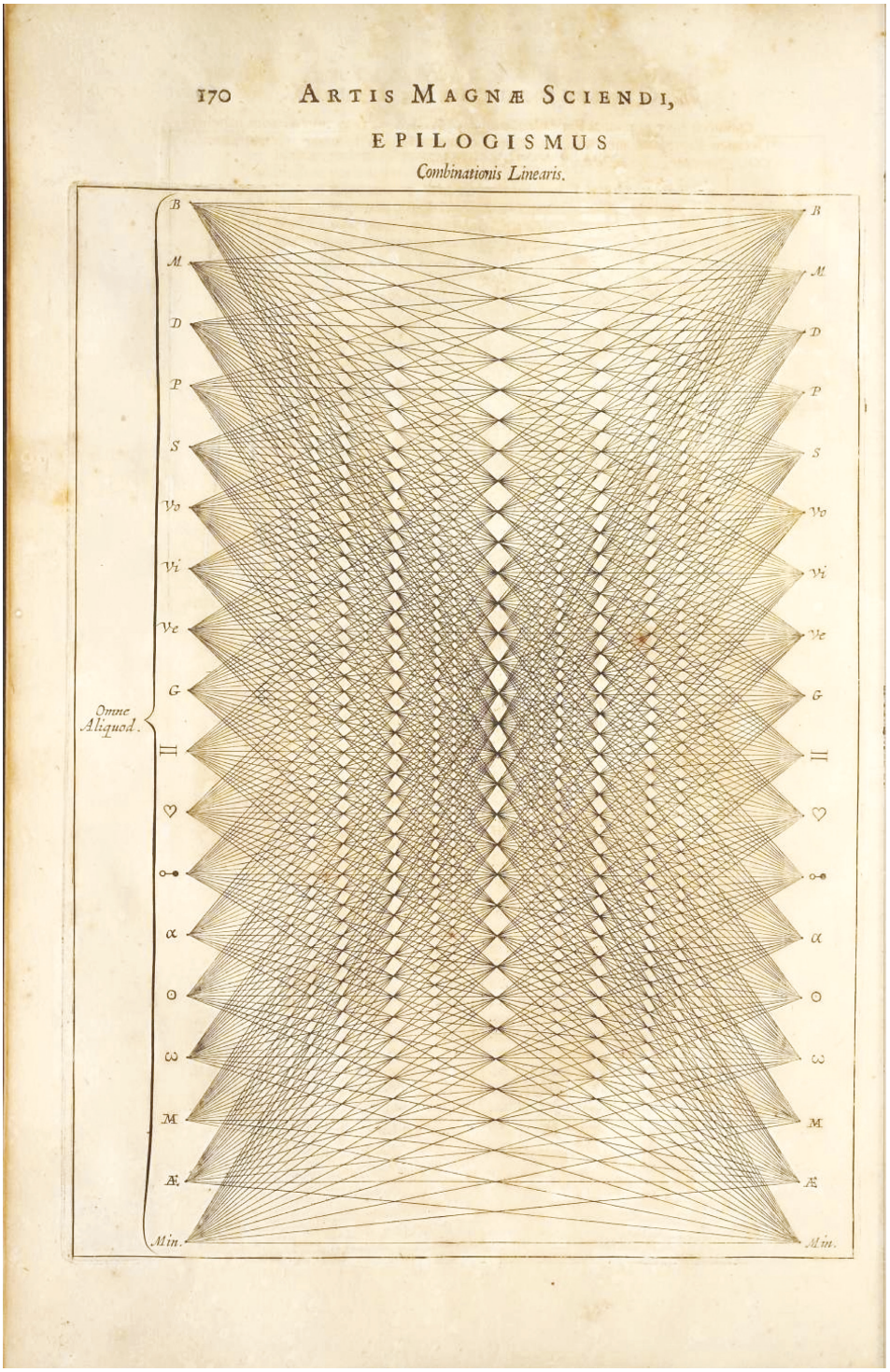


Fig. 7. Dieu dans l'espace d'une page… Source : KIRCHER, Athanase. *Ars Magna Sciendi Sive Combinatoria*. Amsterdam : Jansonius & Waesberge, 1669. Livre quatre, p.170. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://diglib.hab.de/drucke/6-3-quod-2f/start.htm?image=00196.

Cet exemple d'un système combinatoire particulièrement élaboré frappe, par sa qualité esthétique tout d'abord, mais aussi par sa dimension métaphysique. Kircher conclut le quatrième tome de son *Grand Art* sur la combinatoire. Les combinaisons relevées sur cette page ont vocation à lister, autant que faire ce peut, les attributs divins. Nous sommes dans l'exercice d'une { technologie intellectuelle }, qui permet d'appréhender par l'écriture une réalité qui échappe à nos sens ordinaires. Et c'est précisément parce que la combinatoire est une { technologie intellectuelle } qu'elle est fertile d'un point de vue culturel : elle organise et permet une pensée particulière du monde, ramené à une liste d'éléments permutables ; elle nourrit un rapport au langage permettant de faire émerger des formes esthétiques diverses : littérature sous contrainte, art poétique, etc.

technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

La combinatoire n'est donc pas qu'un sous-domaine des mathématiques, mais aussi une { technologie de l'intellect } qui produit de nombreuses pratiques d'écriture. Ces pratiques dépendent, comme toute écriture, d'un support ; elles reposent sur un système conventionnel de signes (hexagrammes, notation musicale, métrique des vers) ; elles s'inscrivent dans un contexte particulier qui leur donne une valeur sociale, esthétique, qui mobilise et construit des représentations du monde, des rapports de pouvoir, etc. Pour faire comprendre ce point, nous allons détailler l'exemple du *Yi King* (parfois orthographié *I-Ching*), ou *Livre des Mutations*, qui selon Knuth est l'une des premières formes de pensée combinatoire dont l'importance est significative dans la culture chinoise. Ce déplacement vers une tradition orientale pour soutenir notre argument sur la combinatoire comme { technologie intellectuelle } nous permettra ensuite de poser un certain nombre de questions quant à l'usage de la combinatoire dans un cadre occidental.

d Le *Yi-King :* un exemple d'une structure combinatoire aux enjeux politiques et métaphysiques

Le *Yi King* est un traité du canon confucéen, traité central dans la tradition chinoise depuis l'empire des Han (II^e^ – I^er^ siècle avant notre ère[373]). À travers soixante-quatre chapitres, correspondant à soixante-quatre hexagrammes, le traité décrit l'ensemble du monde et des changements à venir. Il est construit en partant des deux forces complémentaires et antagonistes, forces fondamentales de la pensée chinoise : le *Yin* et le *Yang*. Le *Yang* est représenté graphiquement sous la forme d'un trait continu (—) ; le *Yin* sous la forme d'un trait discontinu (- -). Par une première combinaison et en superposant les traits, on arrive à quatre figures possibles (*Yin* sur *Yin ; Yin* sur *Yang ; Yang* sur *Yin ; Yang* sur *Yang*). Par une seconde combinaison, en rajoutant un trait, on arrive à un ensemble de huit trigrammes, allant de trois traits pleins (☰) à trois traits discontinus (☷). Ces huit trigrammes, ou *BaGuà*[374] sont ensuite recombinés et superposés pour former les soixante-quatre hexagrammes qui composent le tableau complet du *Yi King*. Lors de la divination, qui peut se faire par un lancer de pièces de monnaie ou plus traditionnellement par la manipulation de cinquante tiges d'achillée millefeuille[375], on note d'abord la question posée. Ensuite, le devin manipule les tiges pour arriver à un total de trente-six, trente-deux, vingt-huit ou vingt-quatre tiges, ces numéros correspondant au quadruple de 9, 8, 7 et 6. Chacun de ces chiffres renvoie, par codage conventionnel, aux quatre figures de bases évoquées plus haut (*Yin* sur *Yin*, *Yin* sur *Yang,* etc.). Dans le cas d'un chiffre pair (8 ou 6), un monogramme *Yin* est tracé. Si le chiffre est impair, un monogramme *Yang* est tracé[376]. L'opération, répétée six fois, permet de produire un hexagramme. Le devin et le consultant pourront alors se reporter au chapitre correspondant à l'hexagramme et commencer le travail d'interprétation[377].

373 Cheng, Anne. Le corpus canonique confucéen. Dans Jacob, Christian et Giard, Luce (dir.). *Op. cit.*, p. 165. Si le canon date du deuxième siècle avant notre ère, on estime que le texte fut rédigé quelques cinq siècles plus tôt. Le « canon » est en fait un certain état commenté puis stabilisé du texte initial, le plus célèbre de ces commentateurs étant Confucius.

374 Du chinois *Ba* (huit) *guà* (figures de divination).

375 L'achillée millefeuille est une plante commune de l'hémisphère nord, dont la tige suffisamment rigide permet de telles utilisations.

376 Vandermeersch, Léon. Origine et évolution de l'achilléomancie chinoise. *Comptes rendus des séances de l'Académie des Inscriptions et Belles-Lettres*. 1990, vol. 134, n° 4, p. 951.

377 Pourquoi passer par quatre chiffres, (9, 8, 7 et 6) et non deux, si de toute façon la construction des hexagrammes passe par un choix binaire (pair/impair) ? Précisément parce que c'est un traité des *mutations* et qu'il est censé traduire l'évolution possible de la situation. Un monogramme « Yin sur Yang » est dit « Yin mutant » : c'est un Yin *tendant vers son inverse*, expliquant ainsi les mutations. Alors que le Yin sur Yin est dit un Yin « naissant », s'interprétant différemment.

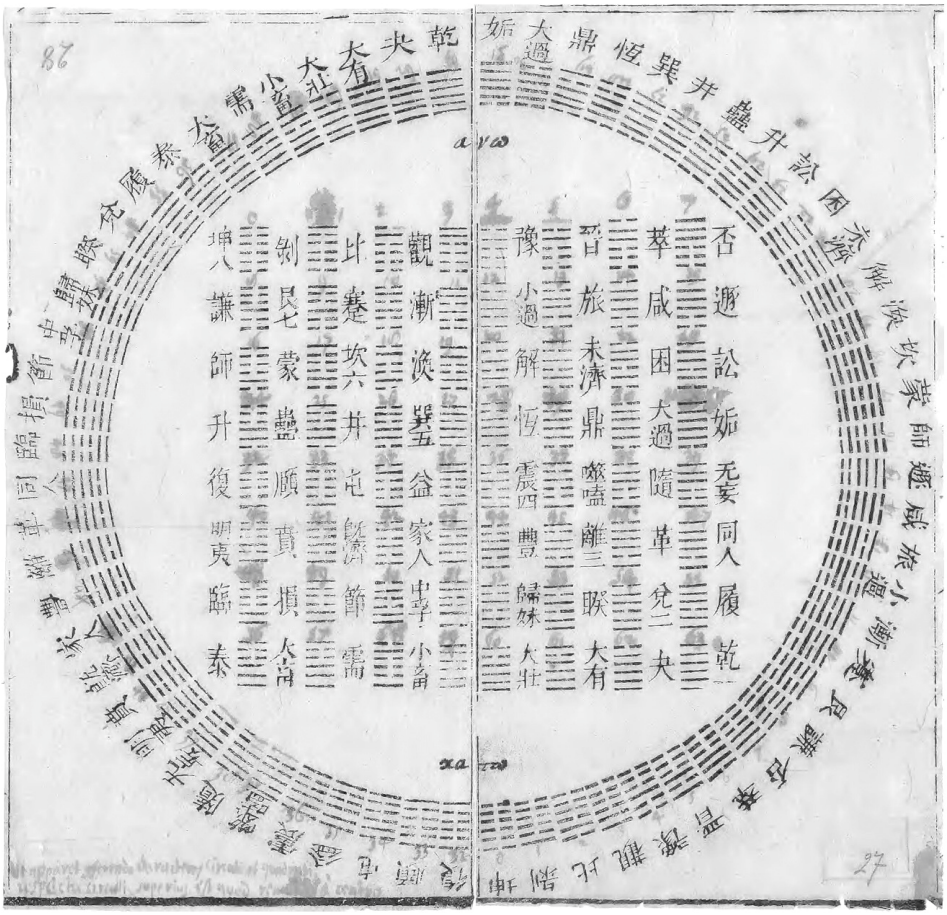


Fig. 8. Le *Yi King*, une technologie combinatoire de l'intellect. Tableau synthétique des soixante-quatre hexagrammes du *Yi King* ayant appartenu à Gottfried Whilelm Leibniz. Source : Perkins, Franklin. *Leibniz and China. A Commerce of Light*. Cambridge : Cambridge University Press, 2004, p. 117.

Le *Traité des Mutations* est l'une des premières structures combinatoires connues, mais son importance culturelle ne s'explique pas seulement par sa structure mathématique. Elle s'explique d'abord par le rôle politique et métaphysique qu'a joué le traité dans l'histoire chinoise. Comme le montre Anne Cheng, chaque traité des six Classiques chinois a un double rôle : cosmologique et politique, les deux étant liés. Certes, le *Yi King* est une partie d'un « [...] tout unifié , synthétique et cohérent à l'image du monde dont il se présente comme le double textuel [...][378] », mais il est surtout la justification livresque d'une structuration de la société idéale divisée en caste de bureaucrates. Parmi ces castes, le « Bureau de la Divination », qui tire son pouvoir du *Yi King*. Le sort du politique et du cosmologique est alors lié. Et quand la dynastie des Han entame en l'an 206 avant notre ère la canonisation des six Classiques, ils « [...] se présentaient et se représentaient à eux-mêmes comme les restaurateurs d'un ordre

378 Cheng, Anne. Le corpus canonique confucéen. Dans Jacob, Christian et Giard, Luce (dir.). *Op. cit.*, p. 170.

sociopolitique (mais aussi cosmique)[379] […] ». L'ordre textuel « double » l'ordre politique, le recouvre en tout point. Les divinations du Yi King ont dès lors une portée qui s'étend au sort de l'Empire chinois lui-même.

Son importance s'explique aussi par la dimension codée des hexagrammes obtenus et c'est cet aspect qui nous intéresse le plus. En raison de leurs usages divinatoires, à chacun des soixante-quatre symboles est associé une qualité, un nom ou un principe. Par exemple, l'hexagramme constitué uniquement de traits continus (pure *Yang*) correspond au Ciel, *qiàn* et à la qualité de donner, de créer ; l'hexagramme composé exclusivement de traits discontinus (*Yin*) correspond à la Terre, *kun* et à la faculté de recevoir[380]. On sait par ailleurs que le Ciel et la Terre sont deux pôles fondamentaux de la pensée chinoise[381]. Les hexagrammes du *Yi King* sont donc des symbolisations graphiques à la fois d'un ordre cosmique, de qualités morales, physiques, etc. C'est par ce jeu d'associations qu'un tel système combinatoire à vocation à décrire l'ensemble du monde et de ces mutations. En le couplant au langage et à sa polysémie, la liste fermée des soixante-quatre éléments devient ouverte à un jeu quasiment infini d'interprétations. Par ce codage, c'est-à-dire cette attribution arbitraire d'un ensemble de valeurs à une représentation graphique (un empilement de six traits continus), la combinatoire du *Yi King* devient un terreau culturel, une matrice permettant de générer des formes de savoirs, de rapports au monde, de pouvoirs, etc.

379 Cheng, Anne. *Idem*, p. 173.

380 Knuth, Donald. Two thousand years of combinatorics. Dans Wilson, Robin et Watkins, John J. (dir.). *Op. cit.*, p. 5. Pour un relevé complet de la table des significations élémentaires, voir Hexagramme, 2016. *Wikipedia*. [En ligne] [mis en ligne le 14 mars 2012] [8 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/Hexagramme.

381 Billeter, Jean-François. *Leçons sur Tchouang-Tseu*. Paris : Allia, 2015 [2002] ; Guénon, René. *La Grande Triade*. Paris : Gallimard, 2016 [1957], p. 15-18.

API, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

Cet exemple du *Traité des Mutations* achève notre démonstration : la combinatoire est une { technologie de l'intellect } qui utilise une propriété fondamentale de l'écriture mais qui produit des formes de pensée, des organisations sociales, des façons spécifiques de voir le monde. Mais cet exemple, aussi parlant soit-il, a aussi ses limites. En quoi est-il lié aux { API } ? Comme nous l'avons dit en introduction de cette partie, nous faisons un détour. Cet exemple nous a permis de montrer que la combinatoire agit comme un terreau culturel. Si les { API } sont les héritières d'une pensée combinatoire présente dès les débuts de l'informatique et si cette pensée est le produit de la combinatoire comme { technologie intellectuelle }, alors on peut faire l'hypothèse que l'informatique est le produit – occidental cette fois – de ce que la combinatoire produit comme façon de penser. Quelles sont donc ces structures de pensée ? Quel est le propre d'une « pensée combinatoire » et qui ne se retrouve pas déjà dans la raison graphique telle que pensée par Goody ?

I 3
une pensée combinatoire…
Voir p. 137

Nous soutiendrons qu'il y a au moins deux spécificités de la pensée combinatoire. La première est une métaphysique de l'épuisement (II. 2. C). Un système combinatoire repose sur l'épuisement des possibles par le calcul des combinaisons. En ce sens, la combinatoire institue un rapport bien spécifique entre le fini – le combinable – et ce qui échappe à ces combinaisons, c'est-à-dire l'infini. Notre hypothèse, en nous appuyant sur un texte de Deleuze[382], est que tout système combinatoire traite d'un domaine fini (les nombres, la nature, les noms possibles de Dieu) tout en postulant et recherchant une entité qui échappe à ce système : Dieu, l'incalculable, la totalité de ce qui peut être dit ou écrit, etc.). En ce sens, un système combinatoire est frappé d'un paradoxe fertile : chercher à traquer l'infini par des moyens finis. La seconde est une radicalisation de l'abstraction permise par l'écriture alphabétique et qui fait de l'écriture, en tant que manipulation combinatoire de lettres, une préparation – ou propédeutique – à un modèle de connaissance scientifique (II. 2. D).

382 Deleuze, Gilles. L'épuisé. Dans : Beckett, Samuel, *Quad et autres pièces pour la télévision*. Paris : Les Éditions de Minuit, 1992, p. 57-105.

2 C. Métaphysique de l'épuisement

« L'un des instincts humains le plus frappant est le besoin irrésistible de chercher des motifs récurrents[383] […] ». Voilà l'hypothèse forte posée par Ronald Graham dans l'avant-propos de *Combinatorics*. La combinatoire est une mise en ordre. Elle répond au besoin anthropologique d'organiser le monde, de le rendre familier à la pensée, habitable pour les corps, de conjurer la hantise que tout cela ne soit le produit du pur hasard. Elle est une forme de « domestication » d'une altérité avec laquelle nous devons, bon an mal an, composer.

La combinatoire est une mise en ordre, mais une mise en ordre qui passe par l'épuisement des possibles, la recherche systématique de toutes les permutations engendrables par le système. Mais dans le même temps, cet épuisement s'accompagne de la présupposition qu'il existe quelque chose qui échappe au système, quelque chose d'inépuisable (II. 2. C. a). C'est pourquoi les pratiques d'écriture combinatoire, notamment en Occident, ont été mobilisées dans des contextes mystiques et métaphysiques : elles permettent de faire le lien entre l'épuisement des possibles et la recherche d'une entité supérieure. D'arriver à l'inépuisable par l'application systématique d'une méthode de recensement exhaustif de tout ce qui peut être combiné (II. 2. C. b).

a Combinatoire, épuisement & exhaustivité

Il y a quelque chose dans la combinatoire qui dépasse le seul jeu mathématique ou langagier. Ce quelque chose, c'est ce que Gilles Deleuze essaye de saisir à travers la figure de l' « Épuisé ». L'épuisé, qu'il voit surgir dans le théâtre de Beckett, est celui qui s'adonne à la combinatoire, soit « […] l'art ou la science d'épuiser le possible, par disjonctions incluses[384] ». Or, épuisement et combinatoire sont deux faces d'une même pièce : on s'épuise à combiner, mais on combine parce qu'on est épuisé. Pourquoi ? Parce que « […] seul l'épuisé peut épuiser le possible, parce qu'il a renoncé à tout besoin, préférence, but ou signification[385] ». En effet, la combinatoire selon Deleuze est une activité qui a la particularité d'être sans but précis autre qu'elle-même : on combine pour combiner, pour faire la liste de tous les possibles. Ce faisant, les choses combinées sont vidées

383 « *One of the most compelling instincts that human beings have is the irresistible urge to look for patterns* […] ». Graham, Ronald, avant-propos. Dans Wilson, Robin et Watkins, John J. (dir.). *Op. cit.*, p. v.

384 Deleuze, Gilles. L'épuisé. Dans : Beckett, Samuel, *Op. cit.*, p. 61.

385 *Ibidem*.

de leurs significations, elles sont ramenées à un plan d'équivalence où ce qui importe est leur caractère combinable. Mais cette combinatoire n'a lieu qu'à l'intérieur d'un terrain commun, ce qui a pour conséquence que « […] tout se divise, mais en soi-même et Dieu, l'ensemble du possible, se confond avec Rien[386] […] ».

La combinatoire « épuise » : elle parcourt l'intégralité de l'existant et du possible. Ce faisant elle vide le monde et l'humain qui l'habite : l'exhaustif est lié à l'épuisement. Mais en même temps, c'est là tout l'intérêt du texte de Deleuze, cet épuisement par l'exhaustif suppose, ou entraîne, la poursuite d'une entité supérieure. La combinatoire est à la fois « […] l'échange indéfini des formulations mathématiques et la poursuite de l'informe ou de l'informulé[387] ». En d'autres termes, on cherche à travers des moyens finis (la liste des éléments combinables) à approcher l'infini. La combinatoire postule son au-delà, son extériorité, ce qui lui échappe. En quelque sorte, c'est parce que l'on épuise le fini par la combinatoire que l'on suppose un infini, parce que c'est désormais le seul sens possible.

386 Deleuze, Gilles. L'épuisé. Dans : Beckett, Samuel, *idem*, p. 60.

387 Deleuze, Gilles. L'épuisé. Dans : Beckett, Samuel, *idem*, p. 62.

b Épuiser le fini et espérer l'infini

Ce que nous permet de penser Deleuze, c'est que la combinatoire comme pratique d'écriture s'accompagne sur le plan symbolique d'une recherche d'ordre métaphysique, ce qui explique les liens historiques entre combinatoire et divination ou religion. On retrouve cet usage religieux de la combinatoire dans bien d'autres contextes, notamment dans la mystique juive du Moyen-Âge, autour de questions à mi-chemin entre linguistique et théologie.

La première trace dans la pensée juive d'une pensée combinatoire, soit l'étude des règles de permutation dans un ensemble fini, apparaît dans le *Sefer Yestsirah* (*Livre de la création*), traité théologique rédigé autour du VII^e^ siècle[388]. L'auteur, inconnu, y examine la question de la création, en lien étroit avec les vingt-deux lettres de l'alphabet hébreu. Il est ainsi dit dans le texte que pour créer les êtres du monde, Dieu a dessiné, pesé puis combiné l'ensemble des lettres et qu'ensuite en prononçant les mots ainsi formés, il a crée l'ensemble des êtres :

> *« Vingt-deux lettres, il les grava et les fit ressortir en les sculptant, il les pesa et en inversa la disposition. Par elles, il créa l'âme de toute formation (*yitsour*) et l'âme de tout discours (*dibbour*) destinés à être créés à l'avenir. […] Vingt-deux lettres fondamentales, fixées sur une roue comportant deux cent trente et un portails. Et la roue tourne vers l'avant et l'arrière.* […] *Comment les pesa-t-il et les inversa-t-il ? – Le* aleph *fut associé à toutes les autres lettres et toutes les autres lettres furent associées au* aleph. *Le* beth *fut associé à toutes les autres lettres et toutes les autres lettres furent associées au beth. Et ainsi la roue tourne encore et encore. L'ensemble de la formation et l'ensemble du discours est issu de ce Nom unique*[389]*. »*

Dans ce cadre de pensée, où le Verbe créateur opère dans un ensemble fini, la combinatoire permet à l'humain de dénombrer par quelques règles simples l'ensemble de la vie possible. Les commentateurs suivant du *Sefer Yestsirah* reprendront ces acquis mathématiques pour continuer le travail esquissé dans ce livre et l'étendre à d'autres types de problèmes, par exemple astrologiques. C'est ce que fera Abraham Ibn Ezra (1090-1167), rabbin espagnol qui voudra étudier les conjonctions possibles des sept planètes, au nom d'une pensée astrologique où les astres influencent le

388 Katz, Victor. Jewish combinatorics. Dans Wilson, Robin et Watkins, John J. (dir.). *Op. cit.*, p. 109.

389 *Sefer Yetsirah*, chapitre II, sections 2, 4 et 5. Traduction dans Idel, Moshé. *Le Golem*. Paris : Les Éditions du Cerf, 1992, p. 60-61.

caractère et la destinée des humains[390]. Là encore, les règles d'étude des permutations possibles au sein d'un ensemble fini se chargent d'enjeux théologiques, éthiques et cosmologiques qui dépassent le seul cadre des mathématiques[391]. Surtout, ces jeux d'associations et de permutations — dont la combinatoire est la recension des règles — permet la lecture des textes sacrés et une meilleure compréhension de la divinité. En d'autres termes, la combinatoire, en tant que { technologie intellectuelle }, nourrit une herméneutique, c'est-à-dire une méthode d'interprétation de textes ; en tant qu'épuisement, elle construit une figure de l'Incombinable (Dieu) comme extérieure aux règles de permutation.

calcul, *subroutine*, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

C'est cette dualité, ce quasi-paradoxe, qui peut expliquer l'utilisation de la combinatoire dans des pratiques liées aux phénomènes religieux ou divinatoires, en tout cas ce rapport particulier que tisse l'être humain, à travers l'élaboration de modèles combinatoires, avec ce qui lui échappe. En tant que ces pratiques combinatoires reposent sur l'élaboration de règles mathématiques visant à trouver tous les motifs possibles au sein d'un ensemble fini, la combinatoire se situe du côté du { calcul } et de l'algorithmie. En tant qu'en même temps, elle est liée à la quête de son extériorité, d'une unité totalisante mais qui ne se résume pas à l'ensemble fini des permutations, alors la combinatoire ouvre vers les pratiques divinatoires, tel que nous l'avons vu avec le *Yi King*.

Voici donc identifiée une première forme de pensée permise par la combinatoire. La seconde nous est inspirée par Mario Vegetti[392]. Nous soutenons avec lui que l'écriture telle qu'elle s'est développée en Occident à partir de son modèle platonicien se caractérise par la forte exploitation de son caractère combinatoire. Ce qui donne naissance à un modèle de connaissance scientifique – un modèle analytico-combinatoire – que nous identifions donc comme un second effet de la combinatoire en tant que { technologie intellectuelle }.

II 2 A
forte exploitation de son caractère …
Voir p. 189

390 Katz, Victor. Jewish combinatorics. Dans Wilson, Robin et Watkins, John J. (dir.). *Op. cit.*, p. 113. Cette thèse d'une correspondance entre plan astrologique et plan psychologique est d'ailleurs particulièrement répandue à l'époque et se retrouve dans les milieux hermétiques de la Renaissance. Voir Yates, Frances A. *The Art of Memory*. Londres : Random House, 2014 [Chicago : University of Chicago Press, 1966]. Chapitre 6, « *Renaissance memory: the memory theatre of Gulio Camillo* », p. 135-162.

391 Cette réflexion repose sur une séparation des champs du savoir qui est très certainement la conséquence d'une *épistémê* moderne, où religion et science sont clairement séparées. Il est vrai qu'avant l'époque moderne, la religion assumait ou chapeautait tout un ensemble de « disciplines » au sens moderne du terme, comme l'astronomie, les mathématiques, la logique, etc. Pour la notion d'*épistémê*, voir Foucault, Michel. *L'archéologie du savoir*. Paris : Gallimard, 1969.

392 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans : Detienne, Marcel (dir.), *op. cit.*, p. 387-419.

2 D. L'alphabet vocalique grec comme propédeutique combinatoire à une pensée scientifique

Si la première spécificité de la pensée combinatoire telle qu'elle s'exerce à travers l'écriture est une mystique de l'épuisement, la seconde est une radicalisation de l'abstraction de l'écriture. Radicalisation qui explique l'émergence d'un modèle analytico-combinatoire de pensée scientifique : traiter un problème en le découpant en unités simples qu'on va résoudre, puis à partir de là recomposer l'ensemble complexe. C'est ce même modèle que nous avons vu à l'œuvre dans la façon dont sont pensées les { *subroutines* } au début des années 1950 : elles résultent d'un découpage analytique des problèmes et ont vocation à être des éléments simples que l'on peut abstraire de leur contexte initial pour pouvoir les remobiliser dans d'autres problèmes.

I 2
la façon dont sont pensées les subroutines…
Voir p. 111

Ce modèle naît, à en suivre le philosophe italien Mario Vegetti, avec la théorie platonicienne de l'écriture. Il y voit en effet la naissance d'une méthode analytico-combinatoire de résolution des problèmes, l'écriture étant située exactement entre le jeu et la science, entre le sensible et l'intelligible (II. 2. D. a). En cela, l'écriture par sa dimension sensible (la lisibilité des caractères) est une initiation à une science combinatoire des éléments (II. 2. D. b), mais elle est aussi par sa dimension vocalique – le fait que toutes les lettres d'un mot soient écrites et non seulement ses consonnes – l'initiation à une pensée abstraite (II. 2. D. c), dans la lignée des théories de Jack Goody. Or, nous proposons l'idée que ce n'est pas l'écriture en tant que telle qui permet cette grande abstraction, mais bien plutôt l'exploitation de sa dimension combinatoire (II. 2. D. d)

a L'écriture selon Platon, entre jeu et science

Dans un article de 1988, Mario Vegetti étudie « les dynamiques de l'écriture chez Platon », c'est-à-dire la manière dont Platon – dans toute son œuvre et non pas seulement dans le mythe de son invention raconté dans le *Phèdre* – aborde la question de l'écriture. Sa thèse est que le « modèle scriptural » proposé par Platon est une :

> « [...] *infrastructure théorique d'une* épistémê *analytique et combinatoire : invariance des éléments premiers, possibilité d'y ramener les composés, règles de dérivation des composés des éléments*[393] ».

L'écriture alphabétique et vocalique grecque dans les écrits platoniciens est, à en suivre Vegetti, un « modèle de connaissance analytico-synthétique[394] » et donc un exercice préparatoire – une propédeutique – à l'exercice d'une certaine pensée scientifique sur ce modèle analytico-synthétique. Et ce, en vertu précisément de la dimension combinatoire de l'écriture grecque. Selon Vegetti, la conception platonicienne de l'écriture fournit, par son aspect combinatoire, la base conceptuelle d'une méthode scientifique particulièrement développée en Occident[395].

Comment Vegetti en arrive-t-il à cette thèse ? Il commence par rappeler que le statut de l'écriture est pour le moins ambigü chez Platon. Technique *pharmakôn* (à la fois poison et remède), l'écriture est surtout rattachée à une série d'autres inventions : l'arithmétique, la géométrie, l'astronomie, le jeu de dames, les dés et enfin les lettres de l'écriture (*grammata*[396]). Soit autant de disciplines ou d'activités qui « se basent sur des éléments simples[397] [...] » et qui « [...] en exploitent les vertus combinatoires[398] ». Platon situe donc l'écriture dans un ensemble plus large de pratiques combinatoires, dont les lettres font partie. Mais comment en font-elle partie ? L'écriture est « [...] à la frontière entre la série des jeux dont elle-même fait partie et celle des savoirs, porteurs de vérité dont elle est l'avant-courrier[399] ». L'écriture est donc une technique ambigüe parce que c'est une technique charnière : un jeu innocent de permutations peut fournir

393 Vegetti, Mario. *Idem*, p. 394.

394 Vegetti, Mario. *Idem*, p. 388-389.

395 Goody rejoint d'ailleurs Vegetti sur ce point, avec certaines différences qui seront explicitées quelques paragraphes plus loin.

396 Platon. *Phèdre*, 274c. Cité par Vegetti, Mario. *Idem*, p. 390.

397 *Ibidem*.

398 *Ibidem*.

399 Vegetti, Mario. *Idem*, p. 390-391.

un « modèle de connaissance heuristique[400] ». L'écriture est prise entre le jeu (les dames, les dés) et les savoirs scientifiques « nobles » (arithmétique, géométrie, astronomie).

b L'écriture comme science des éléments premiers

Pourquoi ce statut charnière ? Parce que l'écriture est le premier accès sensible à une combinatoire d'éléments simples. Platon n'ignore pas que l'écriture est avant tout une technique parmi d'autres et qu'elle doit donc faire l'objet d'un apprentissage. C'est d'ailleurs le premier apprentissage qu'il recommande aux enfants. C'est un savoir littéralement *primaire* : premier et fondamental que de savoir transformer un flux de parole en une suite de signes écrits[401]. Mais là où l'écriture n'est pas une technique comme les autres, c'est par sa caractéristique proprement combinatoire : la lettre (*gramma*) reste identique à elle-même malgré toutes les permutations. Elle est donc « [...] un *stoikheîon*, un élement, premier, simple et invariant de l'écriture[402] ». La lettre, envisagée comme élement simple, peut alors « [...] fonctionner comme paradigme de tout élément auquel peuvent être ramenées et duquel peuvent être dérivées des structures complexes[403] ».

Autrement dit, le *gramma* en tant que signe graphique, c'est-à-dire par son « image du texte[404] » fournit la représentation sensible de ce que peut être un savoir fondé sur la combinaison d'éléments simples. C'est une porte d'entrée sensible à des savoirs plus abstraits (la géométrie, à l'arithmétique et à l'astronomie). C'est une porte d'entrée car elle est fondée sur le même modèle analytico-synthétique.

400 Vegetti, Mario. *Idem*, p. 391.

401 Encore aujourd'hui, l'apprentissage de la lecture et de l'écriture est la pierre de touche de tout enseignement ultérieur.

402 Vegetti, Mario. *Idem*, p. 392.

403 *Ibidem*.

404 Souchier, Emmanuel. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

En quoi ce modèle est-il combinatoire ? Parce qu'il repose sur les deux opérations que nous avons identifiées comme centrales pour parler de système combinatoire : isoler une suite finie d'éléments simples (les *stoikheîa*) et régler les permutations possibles entre ces éléments. En appliquant ces règles à la suite d'éléments simples, on peut alors générer quasi-automatiquement (entendre, par simple application mécanique des règles) un grand nombre de combinaisons. La lettre, par son image, fournit la représentation sensible des *stoikheîa*. La deuxième étape, c'est d'apprendre la grammaire, soit la manière dont certaines lettres s'accordent entre elles[405]. Reconnaissance d'éléments premiers et invariants ; règles de permutation de ces éléments, voilà ce qui caractérise selon Vegetti « l'*épistémê* analytique et combinatoire » de l'écriture selon Platon[406].

c Vocalisme et abstraction

L'écriture chez Platon est donc une technique d'initiation, par le jeu de combinaisons des lettres, à un modèle de savoir scientifique reposant sur la combinaison d'éléments simples. Mais pourquoi cette écriture alphabétique grecque et pas une autre ? L'innovation de l'écriture grecque est l'inclusion, dans l'alphabet emprunté aux Phéniciens, des voyelles. Et ce sont les voyelles qui « [...] ont la fonction d'un lien sans lequel aucune combinatoire de lettres ne serait possible[407] [...] », ou en tout cas sans lequel le modèle combinatoire platonicien ne pourrait être possible. L'introduction des voyelles permet en effet de produire des combinaisons uniques, c'est-à-dire des mots différenciés. Avec un alphabet consonnantique, CHN peut désigner le mot chien, ou Chine, ou encore chaîne. Les voyelles permettent donc de générer des combinaisons les moins ambiguës possibles.

405 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans Detienne, Marcel (dir.), *op. cit.*, p. 394.

406 Vegetti va même jusqu'à évoquer page 393 l'aspect extensible et auto-reproducteur de l'écriture : connaître certaines combinaisons (par exemple un nom) permet de reconnaître n'importe quelle combinaison similaire. Mais il évacue assez rapidement ce point, notamment parce qu'il mène au paradoxe relevé par Platon dans l'*Euthydème* que, puisque tout texte est constitué de lettres, apprendre toutes les lettres et toutes les combinaisons revient à connaître tout texte possible. Nous avons vu que ce n'est pas le cas, car l'alphabet est un système combinatoire ouvert à une infinité de permutations, mais que la quête de la « totalité de ce qui peut être dit » est un imaginaire structurant de l'écriture. C'est pour cela que nous ne l'évacuons pas aussi rapidement que Vegetti.

407 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans Detienne, Marcel (dir.), *op. cit.*, p. 394.

Cette remarque n'a pas seulement une influence décisive sur la morphologie des mots. Avec une écriture vocalique, la parole peut être consignée au plus près, un lecteur peut s'appuyer uniquement sur l'aspect des mots pour en déchiffrer le sens, un texte peut donc être transmis dans le temps et dans l'espace sans qu'un lecteur ait besoin de se référer à l'auteur pour savoir quelle voyelle mettre dans tel ou tel mot. Si l'on peut reconnaître un mot à sa seule forme graphique, cela ouvre la voie à une mécanisation de l'écriture[408] mais cela promeut aussi une forte abstraction conceptuelle, comme le soutient Goody.

d Le modèle analytique platonicien, fruit de l'écriture… ou de sa dimension combinatoire ?

On peut en dernière analyse préciser cette théorie de Goody sur les liens entre pensée scientifique et écriture. Ce dernier soutient que « […] l'intérêt pour les règles du raisonnement ou pour les fondements de la connaissance semble bien naître […] de la formalisation des messages […] inhérente à l'écriture[409] ». Un peu plus plus loin, Goody étend cette idée à la science occidentale en général, dont le développement serait intrinsèquement lié aux techniques d'écriture[410].

Il faut bien évidemment entendre ici l'écriture comme consignation de la parole, soit le modèle platonicien analysé par Vegetti. Or, ce que montre justement Vegetti c'est que le socle théorique de la « formalisation des messages » et des savoirs n'est pas l'écriture en tant que telle mais sa propriété combinatoire. En d'autres termes, ce n'est pas l'écriture en elle-même qui est un facteur explicatif de l'émergence d'une pensée scientifique analytique et abstraite. C'est la valorisation et l'exploitation de son aspect combinatoire – ce que Vegetti souligne chez Platon – qui nous semble être un aspect plus déterminant. C'est pour cela que nous considérons qu'une pensée analytique est principalement l'effet non pas de l'écriture en général, mais plus specifiquement de la combinatoire en tant que { technologie de l'intellect }.

technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

408 Lassègue, Jean et Longo, Giuseppe. What is Turing's Comparison between Mechanism and Writing Worth? Dans Cooper, S. Barry, Dawar, Anuj et Löwe, Benedikt (dir.), *op. cit.*, p. 450-461.

409 Goody, Jack. *Op. cit.*, 2007 [1979], p. 97.

410 Goody, Jack. *Idem*, p. 107.

Voici donc identifié un second effet de la combinatoire comme { technologie intellectuelle } : le développement, autour de la théorie platonicienne de l'écriture, d'un modèle analytico-combinatoire de méthode scientifique. Cette méthode tire parti de la dimension combinatoire de l'écriture et en radicalise les possibilités d'abstraction. En montrant cela, nous achevons de tirer un fil qui nous conduit de la { formalisation } des { *subroutines* } à la fin des années 1940 jusqu'au mythe de l'écriture chez Platon. Ce fil, c'est celui d'un certain modèle de rationnalité et d'abstraction qui se contruit autour de l'écriture. C'est l'idée qu'il y a, par l'écriture, un développement concomitant de la pensée scientifique et des pratiques combinatoires et que ce développement est le produit propre de la combinatoire telle qu'elle s'est développée en Occident. L'informatique, en tant que science construite sur un type d'écriture combinatoire – l'{ écriture-calcul } – se situe dans cette lignée généalogique, dans ce fil qui relie combinatoire, écriture et pensée scientifique.

II 2 A
dimension combinatoire de l'écriture
Voir p. 189

écriture-calcul, **formalisme**, ***subroutine***, **technologie de l'intellect / intellectuelle**

Voir glossaire tome II, p. 3-25

La combinatoire n'est pas propre à la technique informatique, c'est une { technologie de l'intellect } inhérente à tout système d'écriture. En tant que { technologie de l'intellect }, elle permet la production de certaines formes de pensée. Nous avons relevé deux de ces formes : une technique d'épuisement du fini, par le calcul, qui est aussi une manière de postuler ce qui échappe à ce calcul : une entité infinie, bien souvent assimilée à une entité divine ; un modèle analytico-combinatoire de production de connaissance, autour de l'alphabet vocalique grec et de la théorie platonicienne de l'écriture. Pour des yeux contemporains, l'écriture combinatoire se situe donc toujours et en même temps entre deux eaux. Entre une logique « rationnelle », scientifique et la mise en relation avec ce qui échappe à cette rationnalité.

Si l'écriture combinatoire est le creuset d'un certain type de méthode scientifique, nous souhaitons désormais montrer que cet aspect de l'écriture combinatoire est à mettre en lien avec une troisième caractéristique de l'écriture informatique : l'universalisme. Nous faisons l'hypothèse que lorsqu'en 1937, Turing qualifie sa machine d'« universelle », il se situe dans une longue lignée propre à la science occidentale, celle des tentatives d'écriture combinatoire qui ont prétendu au statut de « méthode » ou de « langage » universel, censés unifier l'ensemble des savoirs humains, voire les humains entre eux. Afin de vérifier cette hypothèse, nous allons commencer par préciser ce que nous entendons par « universalisme numérique ». (II. 3. Introduction). Nous montrerons alors que la science occidentale est structurée autour de la recherche d'une « méthode universelle », d'une clé permettant de découvrir et de classer l'ensemble des savoirs. Cette recherche repose sur quatre piliers : une théorie du signe adaptée ; une vision unifiée du monde ; une méthode combinatoire d'exploration de ce monde ; un enjeu politique d'unification (II. 3. A.). L'informatique, en tant que science occidentale, n'échappe pas à cette quête mais la reconfigure à sa façon. Le { formalisme } hilbertien notamment, fournit une théorie du signe adéquate à une combinatoire universelle centrée autour du calculable (II. 3. B), combinatoire que Turing va ensuite mécaniser pour faire de l' « universalisme » une capacité d'imitation de tout ce qui est de l'ordre du calculable (II. 3. C). C'est donc le critère du calculable, qui fournit, en dernière analyse, le principe unificateur d'un monde que l'informatique peut permettre d'explorer et de modéliser, ce par l'entremise d'une écriture combinatoire { formelle } et abstraite.

formalisme,

Voir glossaire tome II, p. 3-25

3 RÉSURGENCES DE LA GRANDE CLÉ

Les pratiques d'écriture combinatoire ont mené en Occident à au moins deux types de pensée : la recherche d'un épuisement d'une totalité par la combinatoire (tout en supposant ce qui échappe à cet épuisement) et la mise au point d'une méthode analytico-combinatoire de résolution des problèmes. Ce que nous allons désormais montrer, c'est que l'informatique – parce qu'elle repose sur une écriture combinatoire – se situe dans cette histoire longue de la recherche d'une méthode scientifique. Plus précisément, c'est parce que Turing qualifie sa machine, dès 1937, d'« universelle » que l'informatique s'inscrit dans cet idéal scientifique occidental. Nous allons donc étudier avec plus de précisions ces liens entre informatique, combinatoire et universalisme, en faisant l'hypothèse que l'informatique est porteuse d'une forme d'universalisme parce qu'elle repose sur une écriture combinatoire. En somme, notre question est : en quoi l'informatique réactive à sa façon – combinatoire – le fantasme d'une écriture universelle ?

Il nous faudra procéder en plusieurs étapes. Montrer tout d'abord qu'il existe une prétention universaliste du numérique (II. 3. Introduction). Expliquer ensuite que l'informatique est une science et qu'en tant que telle, elle s'inscrit dans une longue lignée de tentatives pour trouver une méthode universelle. Nombre de ces tentatives ont en commun de s'appuyer sur une { formalisation } du langage et des procédés combinatoires (II. 3. A). C'est pourquoi nous réexaminons le projet de logique { formelle } d'Hilbert sur lequel s'appuie Turing à l'aune de ces prétentions universalistes (II. 3. B), en insistant sur les notions de calculable et d'équivalence { formelle }. C'est le calculable qui fournit un plan d'équivalence entre toutes choses, nœud de l'universalisme informatique. Enfin, nous montrons que le modèle de machine que Turing imagine en 1937 reprend ces attendus universalistes par le biais d'une combinatoire { formelle }, mais requalifie cette universalisme comme puissance d'imitation. Est universelle une machine qui peut imiter n'importe quelle autre machine, si le comportement de cette dernière peut être représenté sous une forme calculable (II. 3. C).

3 Introduction : un universalisme numérique ?

Milad Doueihi, lorsqu'il replace l'humanisme numérique dans les « trois humanismes » de Claude Levi-Strauss, insiste sur l'extension et l'expansion inédite d'une technique – l'informatique – tant et si bien qu'elle a vocation à englober l'ensemble des activités humaines. C'est cette prétention universaliste du numérique qui fait de ce dernier un phénomène similaire à une religion[411]. L'universalisme s'oppose donc ici à tout ce qui fait la particularité du contexte : spécificités géographiques, historiques. L'enjeu de l'humanisme numérique, c'est de proposer une méthode qui puisse réinsérer au cœur de cet universalisme la complexité des pratiques, toujours locales et situées.

À commencer par la technique informatique elle-même qui, comme le rappelle Milad Doueihi, est foncièrement occidentale : « [l]e numérique est certes un produit occidental, mais il est une réalité globale. Les modèles qui sous-tendent son fonctionnement sont tous ou presque tous dérivés de l'expérience occidentale[412] [...] ». Nous nous situons dans la prolongation de ces réflexions, mais en opérant une forme de retournement. Non pas utiliser l'humanisme numérique pour déjouer l'imaginaire universaliste, mais appliquer les méthodes de l'humanisme numérique à cet imaginaire et le resituer dans l'histoire scientifique occidentale : qu'est-ce qui, dans l'informatique, relève d'un universalisme plus fondamental de la science occidentale ?

411 « Je soutiendrai que, dans la période actuelle, la culture numérique est, de fait, la seule rivale de la religion en tant que présence universelle. » Doueihi, Milad. *Op. cit.*, 2008, p. 23.

412 Doueihi, Milad. *Op. cit.*, 2011, p. 39.

Lorsque nous parlons d'« universalisme informatique », nous nous situons au niveau d'un imaginaire, c'est-à-dire au niveau d'un certain objet discursif, d'une représentation de la technique qui en oriente l'utilisation. Cet imaginaire n'est jamais effectif techniquement, à quelque niveau que ce soit. Un { programme } n'est jamais « universel », au sens où il dépend toujours de la machine sur laquelle il est exécuté et parce que son utilisation se fait toujours en contexte. En revanche, il est important que cette représentation existe.

formalisme, programme

Voir glossaire tome II, p. 3-25

L'enjeu de la présente partie est d'explorer cet imaginaire de l'universalisme, en le replaçant dans une histoire de la science occidentale et montrant qu'il existe un lien historique entre combinatoire, notation { formelle } et universalisme. Notre hypothèse est que s'il existe une prétention universaliste du numérique, c'est parce que ce dernier tient de la technique informatique qui elle-même, en tant que science[413], hérite d'une quête de longue haleine de la science occidentale : la mise au point d'une méthode universelle, d'une « *Clavis Universalis*[414] ».

413 Milad Douehi distingue l'informatique comme technique, née dans les années 1930 autour de Turing, et le numérique, qui est l'extension de cette science à une industrie (celle du logiciel et du matériel) puis à une transformation culturelle plus générale sous l'influence de la technique. Doueihi, Milad. *Op. cit.*, 2013.

414 Rossi, Paolo. *Clavis universalis : arts de la mémoire, logique combinatoire et langue universelle de Lulle à Leibniz*. Grenoble : Millon, 1993.

3 A La combinatoire formelle dans la quête d'une méthode universelle : des enjeux religieux et politiques

L'exemple le plus éclatant de l'universalisme de la science occidentale est peut être la quête d'une « méthode » universelle. Cette méthode universelle, c'est l'idée qu'il existe, au-delà de la disparité manifeste des savoirs (disparité qu'incarnent par exemple les « disciplines » contemporaines), un principe unificateur, une seule et même règle à laquelle il suffirait de se fier pour arriver à découvrir infailliblement tout objet de connaissance.

Cette quête d'une *Clavis Universalis*[415] traverse l'histoire des sciences en Occident. Incarnée notamment par Raymond Lulle (*circa* 1232-1315), Giordano Bruno (1548-1600), Athanase Kircher (1602-1680) ou encore Gottfried Wilhelm Leibniz (1646-1716), elle connait des évolutions et des modulations importantes selon les lieux et les époques. Tantôt tentation, tantôt repoussoir, elle joue un rôle prépondérant dans la naissance d'une pensée encyclopédiste[416] et plus largement scientifique[417]. Nous étudierons plus particulièrement deux de ces tentatives de méthode universelle : l'art combinatoire de Raymond Lulle et les travaux de Leibniz sur une *caractéristique universelle* et sur la *Restitution Universelle*[418]. Pourquoi Lulle ? Parce qu'il constitue très certainement la pierre de touche de cette quête d'une méthode universelle[419] et que sa méthode, à la fois hautement formelle – au premier abord – dans sa construction et missionnaire dans ses enjeux, pose de manière aigüe les rapports entre combinatoire, { formalisme } et universalisme. Leibniz ensuite, principalement parce qu'il reprend les bases lulliennes mais en étend le principe combinatoire et surtout formel. Nous expliquerons la théorie de chaque auteur, sans pour autant les mettre sur le même plan. L'enjeu n'est pas de dire que Leibniz dit « la même chose » que Lulle et que tous deux préfigurent l'informatique contemporaine.

formalisme,

Voir glossaire tome II, p. 3-25

415 « Le terme *clavis universalis* fut employé, entre le XVI^e^ et le XVII^e^ siècle, pour désigner la méthode ou la science générale qui permettent à l'homme de saisir, au-delà des apparences phénoménales [...] la structure ou la trame idéale qui constitue l'essence de la réalité. » Rossi, Paolo. *Idem*, p. 13.

416 Eco, Umberto. *Op. cit.*, 2010.

417 Yates, Frances A. *Op. cit.*, 2014 [1966]. Chapitre 17, « *The Art of memory and the growth of scientific method* », p. 355-374 ; Rossi, Paolo. *Op. cit*.

418 Leibniz, Gottfried W. *Op. cit*.

419 « On peut affirmer sans craindre d'exagérer que la quête europénne de la méthode, racine de la pensée occidentale, commence avec Raymond Lulle. » Yates, Frances A. *Raymond Lulle et Giordano Bruno*. Paris : Presses Universitaires de France, 1999, p. 18.

L'enjeu est de dessiner les lignes de force d'un projet scientifique et politique, en montrant les points communs entre leurs pensées tout en les situant dans leurs contextes respectifs. C'est pourquoi nous commencerons par situer les textes que nous allons étudier (II. 3. A. a).

Ceci fait, nous montrerons que la recherche, par des moyens combinatoires, d'une « méthode » ou d'une « langue » universelle[420] » s'organise autour de quatre caractéristiques :

1 Une théorie du signe (II. 3. A. b)

2 Une vision unifiée du monde (II. 3. A. c)

3 Une méthode combinatoire d'exploration de ce monde (II. 3. A. d)

4 Une visée politique de concorde universelle (II. 3. A. e)

Ces quatre caractéristiques nous serviront de fil rouge pour analyser les théories de Lulle et de Leibniz, puis ensuite pour comprendre comment se situe Hilbert dans cette perspective de la quête d'une méthode universelle.

a Corpus choisi

Avant de passer à la théorie du signe chez Lulle et Leibniz, il faut présenter rapidement leurs travaux et dire sur quels textes nous nous appuyons. Traiter du lullisme est chose ardue car c'est une doctrine particulièrement circulante et polymorphe[421]. Raymond Lulle est un théologien et mystique catalan, qui développa à la fin du XIII^e^ siècle un *Grand Art* (*Ars Magna*), écrit vers 1272. Cet *Ars Magna* s'appuie sur une combinatoire qui permet, selon l'auteur, de produire un ensemble d'énoncés à propos de toutes choses du monde. L'*Ars Magna* est, au cours des siècles suivants, abondamment diffusé en Europe et devient l'une des principales méthodes d'enseignement scolastique. À la Renaissance, le lullisme se trouve sur deux fronts. D'un côté, il est connu et célébré par les néo-platoniciens italiens ou par les alchimistes, mais ce qui a tôt fait du lullisme un art « magique » dérivé de la Kabbale juive[422]. De l'autre côté, l'humanisme naissant condamne, pour son caractère « magique »,

420 Les deux sont intimement liées, comme nous le montrerons.

421 Au risque de l'anachronisme, nous pouvons dire que le lullisme est un remarquable « être culturel » au sens d'Yves Jeanneret. Voir JEANNERET, Yves. *Penser la trivialité. Volume 1, la vie triviale des êtres culturels*. Cachan : Hermès Science – Lavoisier, 2008, p.13.

422 Les liens entre la Kabbale et le lullisme sont complexes, notamment parce que l'utilisation à la Renaissance du lullisme a brouillé les pistes. Les deux courants sont contemporains. Selon Yates, ils procèdent d'un seul et même cadre de pensée hérité de Jean Scot Érigène, sans que l'on puisse dire que Kabbale et lullisme soient deux courants identiques. Voir YATES, Frances A. *Op. cit.*, 1999, p. 147-150 ; ECO, Umberto. *Op. cit.*, 2011 [2010], p. 509-511.

l'Art lulliste[423]. Cette condamnation se fera plus éclatante encore au XVII^e siècle, lorsque Descartes ou le Chancellier Bacon cherchent à élaborer de nouvelles méthodes pour guider l'entendement[424]. Le lullisme est alors un repoussoir. Moqué pour ses prétentions universalisantes, il est considéré comme une machine rhétorique, permettant de produire une multitude d'énoncés sans jamais rendre capable de parler à propos[425]. L'anathème sera tenace, même si – si l'on suit Paolo Rossi – une forme de résurgence du lullisme aura lieu à la fin du siècle, avec les travaux de Kircher puis de Leibniz.

Ce dernier se réclame de Lulle pour construire une méthode scientifique dans sa *Dissertatio de Arte Combinatoria* de 1666, puis dans ses travaux ultérieurs sur la combinatoire et le langage { formel[426] }. Des recherches de Leibniz sur la combinatoire, nous utilisons principalement des textes qui s'étendent de 1666 (date de la *Dissertatio*) à 1715 et de son opuscule sur la *Restitution Universelle* (*Apokatastasis Panton*). Les principales lettres sur la « caractéristique universelle » sont rédigées entre 1671 et 1680 puis en 1693, Leibniz rédige un autre opuscule traitant d'un problème similaire : *De l'horizon de la doctrine humaine*.

formalisme,

Voir glossaire tome II, p. 3-25

b Une théorie du signe

La première caractéristique de ces méthodes combinatoires à visée universelle est qu'elles s'appuient sur une théorie du signe.

Ainsi Lulle, pour sa méthode, utilise neuf lettres, de B à K[427]. Ces lettres sont les éléments de base de sa combinatoire et d'un encodage complexe. Lulle commence par déterminer neuf entités, ou « dignités absolues » qu'il rattache à chaque lettre : B pour la Bonté (*Bonitas*), C pour la Grandeur (*Magnitudo*) etc.[428] Ces neuf dignités sont les éléments à plus haute valeur dans sa combinatoire, ce sont les « valeurs absolues » des neuf lettres. Ces lettres peuvent être envisagées sous cinq autres rapports : comme question (A = « Est-ce que ? », B = « Quel ? », etc.), comme sujet de la proposition à venir (B = Dieu, C = Anges, etc.) et enfin comme vertus

423 D'autant plus qu'il est très proche – depuis les travaux de Giordano Bruno – des arts de mémoire là aussi fermement condamnés par Érasme ou Pierre de la Ramée.

424 Yates, Frances A. *Op. cit.*, 1999, p. 18.

425 Descartes, René. *À Beeckman* [en ligne]. Lettre du 29 avril 1619. [Consulté le 1^er septembre 2017]. Disponible à l'adresse : http://www.earlymoderntexts.com/assets/pdfs/descartes1619_1.pdf, page 17.

426 Rossi, Paolo. *Op. cit.*, p. 202-203.

427 L'alphabet d'alors ne fait pas de différence entre I et J. C'est pourquoi Lulle va jusqu'à K : B, C, D, E, F, G, H, I, K.

428 Pour le système complet, voir Eco, Umberto. *Op. cit.*, 2011 [2010], p. 512-513, et plus spécifiquement la figure 1 p. 513.

(B = Justice, C = Prudence, etc.) ou vices (A = Avarice, B = Gourmandise, etc.). Ainsi, la combinaison AB peut vouloir dire « La Bonté est grande », ou « Dieu est grand », ou encore la Justice est grande, selon le rapport que l'on envisage. Lulle commence donc par un véritable encodage[429]. Cet encodage, purement conventionnel, a pour fonction principale de faciliter la combinatoire par une économie de moyens sémiotiques (on peut représenter des propositions complètes en les abrégeant par des triplets de lettres comme ABD) et par une mémorisation facilitée de la table d'encodage[430].

Pour autant, ces lettres ne sont pas celles de la logique { formelle }, en cela que l'Art est censée donner accès aux choses mêmes :

> « *À la différence de la logique dite formelle,* [l'Art de Lulle] *a non seulement affaire aux mots, mais aux choses, et s'intéresse non seulement à la structure des discours, mais à celle du monde. Une métaphysique exemplariste ou un symbolisme universel est au départ d'une technique qui prétend pouvoir parler, à la fois et en même temps, de logique et de métaphysique, et énoncer les règles qui sont à la base des discours et celles qui structurent le réel*[431]. »

C'est la grande différence entre l'Art de Lulle et la logique { formelle } contemporaine ou même la combinatoire leibnizienne. Les signes manipulés par l'artiste lullien sont un accès direct aux choses : « [l]a logique présumée de Lulle n'est pas formelle ; c'est une rhétorique qui sert à exprimer une ontologie[432]. » L'encodage auquel procède Lulle est une abstraction commode pour représenter sa méthode (une rhétorique), mais cette méthode prétend traiter des choses mêmes (c'est une ontologie).

Leibniz reprend certaines bases de la combinatoire lullienne, mais en « vide » en quelque sorte le pouvoir de signification des signes. La *caractéristique universelle* qu'il veut mettre au point est un « calcul aveugle[433] » (*cogito cæca*). Chez Leibniz, « [...] la combinatoire devait générer des formes symboliques vides, non encore liées par un contenu. L'art devenait ainsi un calcul entre des symboles dépourvus de sens[434] [...] ». Le signe

429 La table d'encodage est reproduite en annexe nº 5.

430 Lulle, Raymond. *L'art bref*. Milan : Archè, 1987, p. 17. La véritable difficulté de l'Art tient à la mémorisation du code, d'où sa proximité avec les arts de mémoire de l'époque. Sur ce point, voir Rossi, Paolo. *Op. cit.*, p. 66-80. ; Yates, Frances A. *Op. cit.*, 2014 [1966]. Chapitre 8, « *Lullism as an art of memory* », p. 175-196.

431 Rossi, Paolo. *Op. cit.*, p. 55.

432 Eco, Umberto. *Op. cit.*, [2011] 2010, p. 538.

433 Eco, Umberto. *Idem*, p. 76.

434 Eco, Umberto. *Idem*, p. 557.

ainsi délié de tout contenu, ses possibilités s'abstraction s'en trouvent renforcées car il acquiert une capacité importante : celle de jouer un rôle vicariant. Il peut se référer à n'importe quel objet et traiter de n'importe quel sujet, de la géométrie au langage humain en passant par la musique[435]. L'important est que les règles de son utilisation soient claires et qu'il ne renvoie qu'à une seule pensée. Il peut alors faire l'objet d'un calcul combinatoire :

> *« […] Toute vérité, proposition ou thèse "doctrinale" peut et doit être formulée dans une langue quelconque, dont le seul caractère exigé est qu'elle soit intégralement et adéquatement représentable dans une écriture alphabétique[436] ».*

Pourquoi l'écriture alphabétique ? En raison de son aspect précisément analytique et combinatoire[437]. Chez Leibniz, un « caractère » est avant tout une « marque visible représentant les pensées[438] ». La caractéristique, son outil universel du savoir, est une certaine combinaison des pensées, c'est-à-dire une certaine organisation des caractères. À condition seulement que ces caractères soient élémentaires et en nombre finis, afin de permettre une combinatoire abstraite. L'écriture alphabétique fournit à Leibniz un exemple remarquable d'écriture pouvant faire l'objet d'une telle { formalisation }.

On le voit donc, la clé dans ces différentes théories du signe est la question de l'abstraction, qu'elle soit à des fins essentiellement mnémotechniques (Lulle) ou de { formalisation } (Leibniz). Mais la différence tient aussi à la vision du monde dans laquelle s'inscrit cette théorie du signe. Les lettres que l'on combine reflètent-elles ou non la structure du monde ?

formel

Voir glossaire tome II, p. 3-25

435 Leibniz, Gottfried W. *Op. cit.*, p. 151.

436 Leibniz, Gottfried W. *Idem*, p. 148.

437 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans : Detienne, Marcel (dir.), *op. cit.*, p. 387-419.

438 Leibniz, Gottfried W. *Op. cit.*, p. 148.

c Un principe unificateur

Les méthodes universelles combinatoires – c'est notre second point – reposent sur la vision d'un monde unifié par un principe. Si bien que les arts combinatoires ne sont alors pas des outils d'unification du monde, mais bien plutôt d'*exploration* de ce monde, en tant qu'ils sont modelés sur ce principe unificateur. C'est la condition de leur universalisme : parce qu'ils sont formés sur ce qui est commun à toutes choses, ils peuvent prétendre traiter tout type de questions.

C'est patent chez Lulle, qui s'appuie sur deux idées principales. La première – largement partagée à son époque – est que l'univers est le fruit d'une combinaison d'éléments simples et de la puissance créatrice des noms divins[439]. Par une méthode combinatoire qui s'appuie sur ces noms divins (les neuf « dignités absolues »), on peut alors comprendre l'ensemble de l'univers : « […] l'extension de l'Art ou de la méthode unique à toutes les branches du savoir est possible en vertu d'un présupposé "métaphysique" : celui d'un monde dans lequel se reflètent les idées de l'esprit qui a présidé à sa création et à sa mise en ordre[440] ». La seconde idée est celle de la « chaîne des êtres », ou « échelle de la création », héritée du néo-platonicisme[441]. C'est l'idée que tout être participe aux Dignités divines : tout être est une émanation, pourrait-on dire, d'un certain aspect de la divinité. Il existe donc une filiation entre une pierre, une plante, un mammifère, l'humain, ou un ange en ceci qu'il ont été créés par une même entité. La liste de ces aspects, c'est la liste des Dignités encodées par Lulle sous les lettres B à K. Si tout être participe aux Dignités, alors une méthode s'appuyant sur ces Dignités peut aller d'un être à un autre, puisqu'elle permet d'atteindre le point commun de tous les êtres. Elle peut donc s'appliquer indifféremment à une plante (botanique), à un animal (zoologie), à l'humain ou à la divinité même (métaphysique). C'est la base de l'universalisme combinatoire de Lulle : en modelant l'outil selon la structure échelonnée du réel, on peut par cette méthode en explorer les différents niveaux[442].

439 Postulat qui est la base théorique des trois grands monothéismes de l'époque, voir Yates, Frances A. *Op. cit.*, 2014 [1966], p. 179-180.

440 Rossi, Paolo. *Op. cit.*, p. 64.

441 Eco, Umberto. *Op. cit.*, 2011 [2010], p. 529 ; Yates, Frances A. *Op. cit.*, 2014 [1966], p. 181.

442 S'il nourrit l'extension universelle de l'Art, ce principe est également ce qui limite le nombre de combinaisons possibles. Comme le rappelle Eco, Lulle précise dans les règles d'utilisation de son Art que certaines combinaisons ne sont pas admises car elles vont à l'encontre de cette « nature » des choses. En somme, la foi en une conception de l'Univers ordonnée par Dieu agit comme garde-fou à la puissance de la combinatoire formelle : « Les principes de la foi et une cosmologie bien ordonnée (indépendamment des règles de l'Ars) doivent tempérer l'incontinence de la combinatoire ». Eco, Umberto. *Op. cit.*, 2010, p. 522. Il y a donc des critères extérieurs au système qui en limitent les possibilités d'expression, et ces critères sont éminemment tributaires du contexte dans lequel ce système combinatoire est utilisé.

Quid de la caractéristique leibnizienne ? Là encore, il faut revenir au rôle essentiel du langage chez Leibniz. Tout langage est une déformation, une diffraction – à la manière d'un miroir – d'une « source des choses[443] ». Cette déformation, conséquence des différences culturelles[444], est pourtant surmontable car « les langues [...] sont virtuellement équivalent[es] par leur capacité de représentation[445] ». C'est là un point essentiel de l'universalisme leibnizien que de postuler ce rapport entre « source » et « diffraction », par le langage de cette source mais que pour autant chaque langage peut représenter toute chose. L'important devient le choix du langage, c'est-à-dire le mode de représentation qui soit à même de représenter adéquatement toute chose du monde. C'est pourquoi Leibniz s'en tient à la « doctrine », c'est-à-dire

> *« [...] tout ce qui relève d'une connaissance transmissible [...] et à cet effet formulée dans une langue qui en rend l'intelligence en droit accessible à qui en comprend les termes. Elle enveloppe vérités de raisons et vérités de faits, sciences démonstratives et d'observation, qu'elles soient relatives aux substances comme aux phénomènes, aux esprits comme aux corps, aux idées comme aux mots[446]. »*

En adoptant un langage « qui rend l'intelligence » de tout ce qui est connaissance transmissible, ce langage étant celui que nous avons vu plus haut, alors Leibniz peut prétendre mettre au point une « [...] *science générale* [...] en mesure de dévoiler les raisons agissant dans le monde, et d'expliquer la structure de la réalité[447] ». L'art que met au point Leibniz est donc « universel » parce qu'il s'appuie sur une vision unifiée du monde, en tant qu'il existe une équivalence entre chose et pensée. On peut donc, par l'entremise d'une écriture comme notation de la pensée, retrouver cette équivalence dans une écriture adaptée.

443 Leibniz, Gottfried W. *Op. cit.*, p. 132.

444 C'est pourquoi, du moins chez Leibniz, le projet d'une « caractéristique universelle » et celui d'une « langue adamique » sont très proches. La langue adamique serait la « langue originelle », pré-Babel, qui est aussi la langue universelle puisque partagée par tous les êtres humains. Sur ce sujet, voir Eco, Umberto. *La quête d'une langue parfaite dans l'histoire de la culture européenne : leçon inaugurale faite le vendredi 2 octobre 1992, Collège de France, chaire européenne*. Paris : Collège de France, 1992.

445 Leibniz, Gottfried W. *Op. cit.*, p. 150.

446 Leibniz, Gottfried W. *Idem*, p. 140.

447 Rossi, Paolo. *Op. cit.*, p. 215.

Il faut donc renverser la perspective : un système combinatoire n'est pas l'outil d'unification d'un monde fragmenté, mais ne peut émerger qu'une fois ce monde unifié, que ce soit par la perfection de Dieu, par une structure mathématique ou par certaines lois physiques. Un système combinatoire est en revanche une méthode d'exploration de ce monde. C'est là notre troisième point.

d Une méthode d'exploration

Cette méthode d'exploration peut se réaliser différemment. Chez Lulle, les lettres ne sont pas disposées dans un tableau statique mais sur des roues concentriques et mobiles, pouvant générer plusieurs combinaisons par le mouvement. Ainsi, « portant les figures géométriques de l'Art […], l'« artiste » monte et descend sur l'échelle de l'Être, mesurant les mêmes proportions à chaque niveau[448]. ». Par l'Art, qui est la méthode d'investigation du monde, l'échelle des Êtres se monte et se descend. C'est ce qui fait du lullisme une tentative encyclopédique plus que mystique : l'enjeu n'est pas directement de « remonter » au niveau le plus élevé de la création par un mouvement univoque d'élévation, mais de parcourir l'ensemble de la création par une méthode combinatoire correctement inventée et appliquée.

Le parcours lullien n'est pas pour autant un calcul, au contraire de Leibniz. La mise en mouvement, chez ce dernier, se fait par le calcul, entendu comme suite d'opérations finies sur des symboles univoques. C'est par le calcul – et son langage idoine – qu'on peut déterminer l'ensemble des doctrines et de ce qui peut faire l'objet d'un savoir. Le calcul a par ailleurs cette propriété remarquable d'être en quelque sorte prédictif : « il suffit que la procédure existe pour ne pas avoir à l'employer : la logique du calcul dispense en effet du calcul[449] ». Le calcul permet de s'économiser le parcours effectif d'un ensemble, tant qu'il est possible de prouver que le parcours est possible. Ce que fait Leibniz quand il calcule « l'horizon de la doctrine humaine » : l'important est de prouver que ce nombre est fini, sans qu'il n'y ait besoin d'articuler effectivement toutes les phrases. Là encore, on voit comment Leibniz est sur la voie de { formalisation } d'une méthode universelle, ce qui est encore plus net chez Hilbert puis Turing, même si les attendus de cette { formalisation } sont chez eux fort différents.

formalisation

Voir glossaire tome II, p. 3-25

448 « *Bearing the geometrical figures of the Art, inscribed with their letter notations, the 'artista' ascends and descends on the ladder of being, measuring out the same proportions on each level.* » Yates, Frances A. *Op. cit.*, 2014 [1966], p. 181.

449 Leibniz, Gottfried W. *Op. cit.*, p. 126.

e Enjeux politiques des projets combinatoires

Enfin, quatrième point, à quoi sert cette exploration combinatoire ? Quel en est l'enjeu ? Nous avons dit plus haut que les méthodes universelles n'unifient pas le monde, mais s'appuient au contraire sur une vision d'un monde préalablement unifié. C'est vrai d'un point de vue du modèle épistémologique de ces méthodes. En revanche, elles sont un facteur d'unification sur le plan politique et religieux, si tant est que l'on puisse séparer les deux à l'époque. C'est un point sur lequel insiste Yates dans le cas de Lulle :

> *« La valeur principale de son Art résidait aux yeux de Lulle dans ses possibilités missionaires. Il croyait qu'un Art reposant sur des principes reconnus par les trois grandes religions, christianisme, judaïsme, islam, fournissait des arguments infaillibles en faveur de la conversion au christianisme. Cette vocation passionément missionaire était l'aspect essentiel de la vie et de l'œuvre de Lulle, le moteur de son inlassable promotion de son Art*[450]*. »*

L'universalisme de Lulle est indissociablement encyclopédique, théologique et missionnaire. En s'appuyant sur certains présupposés théologiques, il propose une méthode qui permet de produire du savoir, mais surtout de convertir juifs et musulmans au christianisme, par la supériorité de sa technique. Cette visée missionnaire est moins nette chez Leibniz, en tout cas pas en ces termes. En revanche, une « caractéristique universelle » permettrait, selon son inventeur, « [...] d'apporter, partout où elle parviendra, la vraie religion[451] [...] ». Plus largement, son projet scientifique est étroitement lié à un projet politique de concorde universelle, ce par l'entremise d'un langage commun[452].

450 Yates, Frances A. *Op. cit.*, 1999, p. 17.

451 Leibniz, cité par Rossi, Paolo. *Op. cit.*, p. 215.

452 Eco, Umberto. *Op. cit.*, 1992.

Nous avons ici montré que la quête d'une méthode universelle passe par une sémiotique (une théorie du signe), une physique (une vision du monde), une métaphysique (lien avec une entité supérieure) et un projet politique. Pour vérifier si l'informatique se situe bien dans la suite de ce projet scientifique, il faut se demander si ces quatre caractéristiques s'appliquent au modèle de Turing. À la question du signe, nous avons déjà donné quelques éléments mais nous allons préciser notre réponse en remontant au { formalisme } hilbertien qui constitue le fond théorique sur lequel Turing imagine sa machine. C'est le { formalisme } hilbertien qui donne à Turing une théorie du signe et une vision unifiée du monde par le { calcul }.

I | 3 | C
quelques éléments
Voir p. 150

calcul
formalisme

Voir glossaire tome II, p. 3-25

3 B Le formalisme hilbertien comme universalisme logique

La quête d'une méthode ou d'un langage universel repose en grande partie sur une vision unifiée du monde, mais aussi sur une théorie du signe. Pour comprendre comment ces deux dimensions se jouent dans le modèle de la machine de Turing, il faut remonter à l'« épistémologie formaliste[453] » sur laquelle se fonde Turing quand il qualifie sa machine d'« universelle[454] », car c'est cette épistémologie qui fournit une théorie du signe et une vision unifiée autour de ce qui est de l'ordre du calculable. C'est donc en reprenant quelques acquis du { formalisme } hilbertien que l'on pourra comprendre une partie de l'universalisme informatique et ses rapports avec une écriture combinatoire. Le premier de ces acquis est un renfermement – notable par rapport aux tentatives de Lulle et de Leibniz étudiées précedemment – de la logique { formelle } sur le calculable (II. 3. B. a). Le monde unifié dont va traiter cette logique, c'est le monde de ce qui peut être calculé, c'est-à-dire déterminé par une suite finie d'opérations soumises à certaines règles. Le second acquis est une théorie { formelle } du signe, qui doit être vidé de son sens contextuel pour pouvoir se prêter à des manipulations calculatoires (II. 3. B. b), ce qui permet de poser l'équivalence { formelle } de tout { calcul } (II. 3. B. c).

calcul, formalisme,

Voir glossaire tome II, p. 3-25

a Le renfermement du formalisme hilbertien

Hilbert pose au début du XX^e^ siècle son « programme » scientifique dans un contexte de crise. La géométrie, qui faisait jusque-là office de fondement des mathématiques, est remise en cause par les travaux en géométrie non-euclidienne. Il faut donc unifier les mathématiques autour d'une nouvelle discipline. C'est ce qu'Hilbert propose avec la logique { formelle }. L'hypothèse forte est que la preuve de toute proposition mathématique peut être démontrée *par des seuls moyens mathématiques* et non par des moyens externes, tels que la géométrie. La logique { formelle } est donc à la fois une unification des mathématiques et un renfermement de celles-ci sur elles-mêmes. Il y a en effet dans la logique hilbertienne une hypothèse de l'auto-référentialité des mathématiques : il n'y a pas

453 Lassègue, Jean. *Pour une anthropologie sémiotique ; recherches sur le concept de Forme symbolique*. Habilitation à diriger des recherches. Paris : Paris-Sorbonne, 2010, p. 38.

454 Rappelons que l'article de Turing *On Computable Numbers […]* est une réponse au problème de la décision posé par Hilbert et Ackerman dix années plus tôt et, qu'en cela, il s'inscrit dans ce projet scientifique et dans l'épistémologie formelle de la logique du début du XX^e^ siècle.

besoin de faire appel à d'autres domaines pour traiter les problèmes mathématiques. Incidemment, tout problème peut donc être prouvé par des moyens mathématiques, le moyen par excellence étant le { calcul }. On peut donc, c'est l'hypothèse d'Hilbert et la base de l'épistémologie formaliste, prouver une démonstration mathématique par le seul { calcul }.

Il y a une forme de vision du monde dans le { formalisme } hilbertien, en tout cas une décision épistémologique forte. Même si nous sommes loin des prétentions du Grand Art lullien, ou encore d'une « caractéristique universelle », l'enjeu est d'unifier le monde mathématique autour d'un langage : celui de la logique { formelle }. La grande différence avec les tentatives antérieures, c'est la portée de cette unification. Hilbert ne parle que des mathématiques et du { calcul }, il n a pas prétention à « mathématiser » l'ensemble des savoirs scientifiques ou à considérer le { calcul } comme une « méthode universelle » pour traiter tout type de problème. Il y a donc une réduction des prérogatives de la logique { formelle } autour du calculable. Ce qui est calculable constitue en quelque sorte le noyau dur de cette discipline, le type d'objets ou de phénomènes dont elle prétend pouvoir traiter.

b Le formalisme comme ascèse du signe

Cette décision épistémologique s'accompagne d'une décision sémiologique. Le renfermement des mathématiques sur elles-mêmes est tributaire d'un type d'écriture et d'une théorie du signe, ce qu'ont bien vu Jean Lassègue et Bruno Bachimont. Le premier parle d'« [...] un divorce strict entre les caractères employés et la signification dont ils [sont] porteurs[455] », le second d'une « ascèse du signe[456] ». Pour que l'on puisse prouver par le { calcul } une proposition mathématique, il faut se doter d'une écriture { formelle }, c'est-à-dire une écriture alphabétique, univoque, indépendante des langages naturels. Mais cette écriture est frappée d'un paradoxe :

455 Lassègue, Jean. *Op. cit.*, p. 40.

456 Bachimont, Bruno. *Op. cit.*, 2010, p. 156.

> *« La stratégie formaliste* […] *s'appu*[ie] *sur une thèse sémiologique forte selon laquelle, d'une part, toute pensée mathématique est susceptible de se projeter intégralement dans les signes qu'elle manipule et, d'autre part, qu'en rester au niveau de la seule intuition de ces signes est à la fois nécessaire et suffisant pour réfléchir de l'intérieur des mathématiques à leur nature propre. Ainsi, du point de vue sémiologique, le formalisme avait-t-il* [sic] *ceci de particulier de combiner deux attitudes qui semblaient, au premier abord, aller en sens contraire : la transparence du signe et la confiance absolue en son pouvoir de représentation*[457]*. »*

binaire, calcul, code, formalisme, logique booléenne

Voir glossaire tome II, p. 3-25

Transparence du signe, car la notation { formelle } est censée représenter directement les opérations de l'esprit. Par exemple, dans la notation de la { logique booléenne }, le signe « V » ne veut pas dire la lettre V mais l'opération « ou ». Et ce, indépendamment de l'image du texte : le { formalisme } fait disparaître l'épaisseur culturelle de la lettre, qui ne fait que représenter – idéalement encore une fois – une opération logique. Confiance en son pouvoir de représentation ensuite, puisque ce « V » veut *toujours* dire « ou », en dehors de tout contexte d'utilisation. Cette confiance résulte de la transparence du signe en logique. Si on peut se doter d'un inventaire pérenne et partagé des symboles logiques, c'est-à-dire d'un alphabet, alors on peut décrire les mathématiques par un langage non ambivalent et les unifier autour de ce langage.

457 Lassègue, Jean. *Op. cit.*, p. 41.

c Formalisme et universalisme : de l'importance de l'équivalence formelle

C'est par ce { formalisme } qu'on peut parler d'un « universalisme hilbertien », bien différent de celui de Lulle par son étendue. Cet universalisme, c'est ce que Bruno Bachimont appelle « l'idéalité computationnelle[458] ». Le { calcul } (*comput*, d'où le terme d'idéalité *computationnelle*) peut être idéalement réalisé sur n'importe quel support et sans que l'on ait à se poser la question du sens de ce { calcul } – ce qui le rend automatisable. D'un point de vue théorético-idéal en effet, un { calcul } dans le sens { formel } d'Hilbert peut être effectué sur du papier, de la cire, un support optique ou magnétique, sans qu'il y ait besoin d'interpréter les symboles utilisés à chaque opération. Le { calcul } est idéalement auto-référent, refermé sur lui-même. Le support ne doit pas avoir d'incidence décisive sur le résultat du { calcul }. C'est le niveau « théorético-idéal » du numérique, qui se caractérise par une « calculabilité universelle[459] ». C'est le niveau du { code } { binaire }, où « [...] tout contenu, dès lors qu'il est numérique ou numérisé, peut être réduit à un code calculable, manipulable[460][...] ».

Cette idéalité résulte d'une double abstraction : abstraction vis-à-vis de la matière et abstraction vis-à-vis du sens. Abstraction vis-à-vis de la matière car en théorie « [...] la même information peut être matérialisée sur des supports matériels distincts, tout en gardant ses propriétés informationnelles intactes[461] ». Une machine de Turing peut être construite en papier, en Lego ou avec des composants électroniques. Les informations qu'elle manipule peuvent être stockées sous forme de bosses, de papier perforé, de cire, voire d'empreintes accoustiques dans un liquide, tant que ces informations sont non ambivalentes et que la distinction entre le 0 et le 1 est claire. Il n'y a **aucune différence de nature entre ces différentes matérialités**.

458 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit.*

459 Crozat, Stéphane, Bachimont, Bruno, Cailleau, Isabelle, *et al.* Éléments pour une théorie opérationnelle de l'écriture numérique. *Document numérique*. 2012, vol. 14, nº 3, p. 18.

460 *Ibidem*.

461 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit.*

binaire, calcul, formalisme, format, interface, langage de programmation, matériel,

Voir glossaire tome II, p. 3-25

La différence est d'ordre quantitatif : le dénombrement des opérations – le temps de { calcul } – sera plus long. Les composants peuvent être plus chers à l'achat et à l'entretien, plus compliqués à utiliser… mais la machine n'en est pas moins « universelle » pour autant[462].

Abstraction vis-à-vis du sens ensuite car les symboles logiques sont « […] manipul[és] selon des règles applicables mécaniquement, sans ambiguïté, selon un procédé également effectif et réalisable par une machine[463] ». Les symboles utilisés lors du { calcul } ne veulent rien dire d'autre que leur fonction dans le { calcul }, ce que Lassègue nomme la « transparence du signe ». Pour quelqu'un d'attaché à la sémiotique, c'est une violence terrible imposée au signe. Jean Lassègue emploie le terme de « divorce », Bruno Bachimont celui d'« ascèse », tous deux suggérant une forme de violence ou en tout cas de contrainte. Mais c'est une contrainte créative, ou en tout cas fertile – c'est pour cela que le terme « ascèse » est préférable – car elle permet l'automatisation et l'extension *ad indefinitum* du { calcul }, par son aspect { formel } et indépendant du { matériel }.

Cette idéalité – par définition – n'existe jamais en tant que telle mais doit toujours être doublement incarnée : par le choix d'une machine, d'un { format } et d'un { langage de programmation } (ce que Bruno Bachimont appelle l' « implémentation matérielle[464] ») ; par le choix d'une { interface } qui rend les résultats du { calcul } visible à l'homme et lui donne une intelligibilité sémiotique. C'est à ce niveau que l'épaisseur culturelle du signe, objet d'étude privilégié des écrits d'écran, intervient. L'implémentation matérielle et l'interfaçage viennent en quelque sorte « engraisser » par une histoire, une culture, un milieu socio-économique, le signe ascétique de la logique hilbertienne. Les deux niveaux, idéalité et implémentation, existent en même temps dans tout écrit d'écran et l'imaginaire universaliste dont nous parlons se situe au niveau de l'idéalité computationnelle.

462 Ce point est essentiel pour comprendre la différence entre le niveau de l'idéalité computationnelle et celui de l'implémentation matérielle, et comment cette différence recouvre la différence entre l'informatique comme science et comme industrie. Du point de vue théorique de la science informatique, le calcul est formel et abstrait. Du point de vue de l'industrie, il est au contraire essentiel de poser la question de la réalisation matérielle de ce calcul (quelle machine, quelle mémoire, pour quelle vitesse de calcul, etc. ?) car cela va déterminer un modèle économique et des manières de s'approprier un objet technique.

463 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit*.

464 *Ibidem*.

Les rapports entre idéalité et implémentation posent en dernière analyse la question de la **représentation du { calcul }**, au sens de la convention d'écriture adoptée pour écrire et donner à lire le { calcul }. Le problème du { formalisme } n'est pas tant les opérations du { calcul } – la manipulation des informations { binaires } – mais le langage qui va être adopté pour décrire ce { calcul }. D'un point de vue { formel } – idéal –, un { calcul }, qu'il soit écrit en suivant les conventions d'Hilbert, de Church ou de Turing, est équivalent. Ce sont trois façons de représenter une opération, mais ce ne sont pas trois opérations différentes. Elles sont donc dites **équivalentes**. Si j'écris « A ou non A » ou « A ∨ ¬A », ce sont deux représentations équivalentes d'une même proposition, l'une en langage naturel, l'autre en logique propositionnelle. Ce sont deux matérialisations, deux implémentations d'un même { calcul } au plan « théorético-idéal ». La seconde peut être plus facilement automatisée, c'est son avantage en ce qui concerne son implémentation matérielle, mais d'un point de vue théorique, les deux propositions sont équivalentes.

Ce principe d'équivalence { formelle } est central pour comprendre l'universalisme informatique tel qu'il prend forme avec la première description d'une machine à calculer automatique. En effet, cette machine est « universelle », non pas en tant qu'elle permet de tout calculer[465] mais en tant qu'elle peut adopter le comportement de toute machine qui peut être *décrite* sous la forme d'instructions calculables. C'est ce que va introduire Turing par son modèle de l'imitation. Dans ce contexte, « universel » veut dire *imitable* et *réplicable* et ce, par les caractéristiques de l'écriture { formelle } hilbertienne.

465 Turing précise bien qu'il ne s'occupe que du dénombrable, ou des « nombres calculables », ce qui est un champ d'application certes large, mais bien délimité.

3 C Turing & la machine à imiter : l'extension du domaine du calculable

calcul, formalisme, logiciel, matériel, programme

Voir glossaire tome II, p. 3-25

I 1 D a
caractéristique fondamentale des ordinateurs…
Voir p. 97

La notion d'« universalité » apparaît deux fois chez Turing : dans son article de 1937 et dans celui de 1950 sur mécanisme et intelligence. Dans les deux cas, sa définition est relativement simple, mais redoutable d'un point de vue intellectuel. Est dite universelle une machine qui est capable d'imiter entièrement le comportement d'une autre. En informatique, un cas exemplaire est ce qu'on appelle l'« émulation ». Pour faire fonctionner un { logiciel } ancien ou non prévu sur la machine, un { programme } « imite » le système ou la machine qui permet de faire fonctionner le { logiciel } suranné. L'universalisme, entendu dans ce sens, est une caractéristique fondamentale des ordinateurs et de la scission entre { matériel } et { logiciel }, ce qui n'est pas sans incidence sur l'extension inédite de ces techniques et donc sur leur « universalisme » au sens ordinaire du terme. L'hypothèse de cette partie est donc que l'universalisme computationnel, initié par Turing, repose sur une extension du domaine du computable – tel que défini par Hilbert – à des domaines initialement non pris en charge par les médias informatisés.

Pour prouver cette hypothèse, nous reprendrons l'article *On Computable Numbers […]* de 1937, en portant une attention particulière à la notion de « machine universelle ». Cette notion s'appuie sur celle de « description » du comportement d'une machine (II. 3. C. a), puisqu'une machine universelle est celle qui est capable de reproduire le comportement d'une autre machine dès lors qu'il a fait l'objet d'une telle description. Cette puissance d'imitation est étroitement liée au principe d'équivalence { formelle } (II. 3. C. b) et au domaine du calculable, facteur d'unification qui se trouve – dans le cas de l'informatique – considérablement élargi (II. 3. C. c). Ainsi, nous pourrons confirmer notre hypothèse initiale : il existe un universalisme informatique fondé sur des méthodes combinatoires. Comme toute méthode universelle, elle s'appuie sur une théorie du signe (les symboles { formels }), une vision unifiée du monde (le calculable), une méthode d'exploration (le { calcul } automatisable) et une visée d'unification, en tout cas d'extension du principe d'unification – en l'occurrence le calculable.

a Le concept de « description standard » chez Turing

La première fois que Turing qualifie sa machine d'universelle, c'est dans *On Computable Numbers […]*, paragraphe 6 : « *The universal computing machine* » (« La machine à calculer universelle »). Ce paragraphe arrive juste après un paragraphe sur la description des configurations de la machine[466]. Turing pose un problème fondamentalement hilbertien : comment décrire ce que fait sa machine dans un langage qui permette de rendre de façon intelligible et sans équivoque ses opérations ? De cette question, Turing tire le concept de « description standard[467] » (*standard description*) ou « SD ». La description standard est un moyen alphabétique simplifié de noter la configuration de la machine, soit l'ensemble de ses opérations. Plutôt que de noter les détails de tous les { calculs }, on attribuera à une certaine suite d'instructions une lettre. On pourra alors décrire l'ensemble des opérations par une suite de type « DADDCRDAA […] ». L'ensemble forme ce que Turing nomme une « séquence calculable » (*computable sequence*), mais exprimée de manière ramassée. Encore une fois, il faut se rappeler qu'il s'agit là d'une *commodité graphique*. En vertu des principes du { formalisme } hilbertien, la forme DADDCRDAA est logiquement équivalente à la forme détaillée de cette séquence.

466 Une configuration est l'ensemble des états de la machine pour effectuer un calcul donné.

467 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 238.

Sur ces bases, Turing affirme : « Il est possible d'inventer une seule machine qui peut être utilisée pour calculer n'importe quelle séquence calculable[468]. » Pour cela, il suffit d'écrire la description standard d'une machine spécifique M pour qu'une machine M' (une machine universelle) puisse décoder cette description – traduire les caractères alphabétiques en instructions { binaires } – et ainsi reproduire le comportement de la machine M. Ce que Turing appelle dans son article de 1950 une faculté d'« imitation » : « La propriété spécifique des calculateurs numériques, le fait qu'ils puissent imiter n'importe quelle machine à états { discrets }, est décrit en disant que ce sont des machines universelles[469]. » L'universalisme de la machine de Turing est donc indissociable de l'{ écriture-calcul }. Si l'on peut écrire le fonctionnement d'une machine sous forme ramassée (la description standard) et finie (une liste d'opérations qui décrit intégralement le fonctionnement de cette machine), alors elle peut être imitée par une machine universelle.

binaire, calcul, discret, écriture-calcul, formalisme, logiciel, matériel, programmation, programme

Voir glossaire tome II, p. 3-25

b Universalisme et imitation : de l'importance de l'équivalence formelle

Cette universalité est fondée sur le { calcul } comme régime d'équivalence de l'ordre de l'imitable. Tout ce qui peut être représenté sous forme d'opérations finies et par une écriture { formelle } peut faire l'objet d'une exécution par une machine. Le { calcul }, entendu au sens de Hilbert, est ce qui unifie tout ce que peut faire un ordinateur. Tant et si bien que du point de vue { formel }, les différentes machines **spécifiques** imitées par un ordinateur **général** sont logiquement équivalentes : elles sont toutes calculables. La forme que doit prendre la description des opérations tient aux spécificités { matérielles } de la machine. Mais du point de vue « théorético-idéal », une machine dont on peut calculer le fonctionnement est logiquement équivalente à tout autre machine également calculable[470].

La notion d'universalité informatique est non seulement essentielle d'un point de vue théorique, mais elle l'est également du point de vue de l'implémentation matérielle de l'informatique. Car que veut dire, dans

468 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 239.

469 « *This special property of digital computers, that they can mimic any discrete-state machine, is described by saying that they are universal machines.* » Turing, Alan M. Computing Machinery and Intelligence. *Op. cit.*, p. 442.

470 Or, une machine de Turing est, par définition, une machine à états discrets entièrement programmable, c'est-à-dire dont tous les mouvements peuvent être décrits sous forme de calcul. On peut donc, en théorie, donner comme instruction à une machine de Turing d'en imiter une autre. La machine universelle est en ce sens aussi une machine auto-référentielle, et auto-reproductrice : elle peut écrire le fonctionnement d'une « autre » machine, identique à elle-même. Nous ne rentrerons pas plus en avant dans cette auto-référentialité qui pose le problème plus fondamental de « l'arrêt ».

ce contexte, « imiter » ? Imiter, c'est exécuter un { programme }. Programmer, c'est décrire le fonctionnement d'une machine en termes { formels } de façon à ce qu'ils puissent faire l'objet d'un { calcul }. Une fois cette description faite, la machine programmée devient exécutable par un ordinateur. Si, par exemple, il est possible de décrire mathématiquement la trajectoire d'une fusée ou d'un missile, alors il peut y avoir un { programme } qui simule, ou « imite » cette trajectoire. On peut donc employer la machine universelle – qui est aussi générale – dans le contexte spécifique de la balistique. Cette tension entre machine universelle et { programmation } spécifique est au cœur de la distinction entre le { matériel } et le { logiciel } : « [...] cette dualité fondamentale [entre *software* et *hardware*] distingue l'ordinateur, machine générale, apte à tous les calculs, et donc universelle, des autres machines dont le comportement est spécialisé dans une tâche particulière[471]. » Un ordinateur, parce qu'il est construit sur le modèle établi par Turing de « machine universelle », est une machine générale, du moins qui peut traiter tout ce qui est de l'ordre du dénombrable. Le { programme } est une écriture qui vient spécifier cette machine, en lui demandant d'adopter un comportement spécifique.

C'est le premier niveau de l'universalisme informatique : une machine tellement { formelle } et « abstraite » qu'elle est capable de simuler n'importe quelle entité qui peut être décrite d'un point de vue { formel }. L'écriture traverse de toute part cet universalisme : la machine de Turing n'est universelle que parce qu'elle s'appuie sur le modèle d'écriture d'Hilbert ; la description de ce qui doit être imité est là aussi un jeu d'écriture. Mais pour l'instant, c'est un universalisme limité dans ses perspectives et ses applications.

471 Jean-Claude Ganascia, cité par Herrenschmidt, Clarisse. *Op. cit.*, p. 406.

c Le calculable comme facteur d'unification

C'est la cybernétique qui va permettre l'extension du calculable à des dimensions inédites : « [l]a cybernétique [...] a permis de comprendre les systèmes physiques en termes d'information [...]. Cette vision permet de comprendre comment la matière physique est en fait un contenu, et que l'information est présente matériellement dans les objets qui nous entourent[472]. » Si l'on peut considérer les systèmes physiques comme des échanges d'information, que cette information peut être l'objet d'un { calcul }, alors on peut représenter ces systèmes sous une forme écrite et calculatoire, c'est-à-dire d'un { programme }. Il est alors possible de simuler ces systèmes grâce à un ordinateur. Ce qui est en jeu avec la cybernétique, c'est donc une extension sans précédent du domaine du calculable. Mais puisque le calculable est depuis Turing une { écriture-calcul } mécanisable, alors il s'agit en réalité d'une extension du domaine du scripturaire[473]. La question est en quelque sorte renversée. Il ne s'agit plus de se demander ce qu'est un « nombre calculable » comme le fait Turing en 1937, mais plutôt de se demander : qu'est-ce qui **n'est pas** calculable ? Et donc incidemment : qu'est-ce qui ne peut pas être représenté sous la forme d'une écriture { formelle } et combinatoire ? Le renfermement proposé par Hilbert s'est en quelque sorte renversé : après avoir isolé le noyau des objets de la logique { formelle }, voilà que ce noyau se voit attribué des prérogatives étendues. Le calculable identifié par Hilbert puis automatisé par Turing est en quelque sorte la matrice initiale de l'universalisme informatique.

calcul, écriture-calcul, formalisme, programme

Voir glossaire tome II, p. 3-25

On peut désormais répondre à la question qui nous occupait au début de cette partie. Sur quelle vision du monde unifié repose l'universalisme de la machine de Turing et de quel type est cet universalisme ? C'est le calculable qui unit tout ce que peut traiter une machine de Turing. L'informatique en tant que science s'occupe de tout ce qui est calculable, donc de tout ce qui peut être écrit algorithmiquement, c'est-à-dire en utilisant un langage { formel } qui décrive une liste finie d'opérations élémentaires. L'universalisme dont il est question dans ce contexte est avant tout une puissance d'imitation. Cette puissance est tirée du plan d'équivalence logique créé par le { calcul }. Si tout ce qui est calculable est logiquement équivalent – même si sémiotiquement différencié dans sa mise en forme – une machine capable de se porter à ce niveau élémentaire est dite universelle, en ceci qu'elle est capable d'imiter tout ce qui peut être calculable.

472 Bachimont, Bruno. *Op. cit.*, 2010, p. 154.

473 Là encore, entendons-nous : d'un certain modèle combinatoire et calculatoire de l'écriture.

On peut donc relire l'ensemble de l'histoire de l'informatique comme celle d'une extension du domaine du calculable. Hilbert avait formalisé la logique pour tenter d'unifier les mathématiques. Turing mécanise cette écriture { formelle } pour déterminer ce qui est de l'ordre du calculable. Mais nous sommes encore dans l'informatique comme science. La pensée cybernétique, ainsi que le développement industriel de l'informatique étend cette calculabilité à des objets qui ne sont plus du seul domaine de la logique ou des mathématiques : systèmes physiques, images, sons, livres, films, voire créativité ou intelligence[474]… Cette extension d'un projet scientifique initialement limité dans sa portée constitue pour nous le cœur de l'universalisme numérique. L'écriture traverse de toute part cette histoire. Le { formalisme } hilbertien est une forme d'écriture, tout comme la machine de Turing et ce qui est calculable veut dire : ce qui peut être écrit formellement comme une suite finie d'opérations. C'est donc un certain système d'écriture, abstrait et combinatoire, qui forme la base de l'universalisme informatique contemporain, que ce soit l'universalisme comme puissance d'imitation (au sens de Turing) ou au sens de l'application d'une « méthode universelle » – le { calcul } – renouant par là avec les tentatives passées de Lulle ou de Leibniz.

Notre hypothèse de départ est donc confirmée : il y a une forme d'universalisme informatique qui repose sur un certain type d'écriture combinatoire. Démontrer cette hypothèse nous conduit à mettre en lumière trois idées. Premièrement, que cet universalisme est le produit d'une certaine histoire de la science occidentale, dont l'une des lignes de force est la recherche d'une méthode universelle. Cette recherche s'articule autour de quatre points : une théorie du signe ; la vision d'un monde unifié ; une méthode d'exploration ; une visée de concorde politique ou religieuse. Nous avons vu par ailleurs comment le modèle de Turing reprend au moins les trois premiers. Deuxièmement, l'universalisme prend la forme particulière d'une puissance d'imitation par le { calcul }. Troisièmement, l'universalisme dont nous parlons résulte d'une extension des prérogatives de cette puissance d'imitation, c'est-à-dire d'une extension des phénomènes que nous pouvons modéliser comme étant calculables.

474 Turing n'a lui-même de cesse d'étendre les applications de sa machine. Et lorsqu'il écrit son article de 1950, il en vient à proposer une définition de « l'intelligence » qui soit formalisable, justement sous la forme d'un « jeu de l'imitation ».

d'une scie
combinat
de l'écritu
à une ind
du texte.

ence
toire
ure…
ustrie

API,
calcul,
écriture-calcul,
formalisme,
industrialisation,
programmation

Voir glossaire
tome II, p. 3-25

Arrivé au terme de cette première partie, nous avons démontré que les { API } constituent le dernier avatar d'un modèle combinatoire de l'écriture, modèle qui s'ancre dans l'histoire de l'informatique autour de ce que nous avons appelé l'{ écriture-calcul } automatique (chapitre I). Une analyse généalogique a ensuite montré que cet aspect combinatoire est un trait constitutif de l'écriture, que l'informatique exploite de façon particulière (chapitre II). En tant que science, elle reconduit un certain universalisme scientifique occidental historiquement lié aux pratiques combinatoires. Elle le reconduit en télescopant deux projets : la mécanisation du { calcul } par Turing et l'extension de ce qui est calculable par la cybernétique. Il y a donc une { industrialisation } d'un universalisme scientifique – s'appuyant sur sur une combinatoire { formelle } – qui s'étend à des pratiques culturelles comme l'écriture, au sens de production d'un texte. Pour analyser cela, il faut passer d'une conception historique de l'{ API } (l'{ API } comme objet technique historicisable) à une conception sémiotique des { API } : ce sont des outils d'écriture et de mise en forme des textes de réseau. En somme, il s'agit de passer d'une discipline scientifique fondée sur une théorie de l'écriture à une industrie du texte, les { API } web étant exactement à la jonction des deux. C'est un objet qui, par sa face informatique, reconduit une théorie { formelle } et combinatoire de l'écriture ; par sa face culturelle et son déploiement contemporain, il appartient à une industrie du texte.

C'est pourquoi ce chapitre inaugure une seconde partie consacrée au texte, c'est-à-dire aux différentes formes que les { API } permettent de produire et d'articuler en une entité matérielle visuelle et signifiante. L'enjeu est de voir comment les caractéristiques soulevées dans la première partie (combinatoire, abstraction, universalisme) travaillent les formes produites, comment les propriétés des { API } comme outils d'écriture participent des textes que ces outils permettent de produire. En somme, nous voulons montrer la pensée du texte présente dans les { API }.

Le problème sera envisagé sous trois aspects. Tout d'abord l'aspect éditorial, c'est-à-dire celui de la mise en forme du texte, en montrant que les { API } web promeuvent une vision modulaire et manipulable du texte, conçu comme autant de blocs réagençables (chapitre III). Ensuite, l'aspect économique et industriel, où nous montrerons que par cette modularité, les { API } s'inscrivent dans – tout autant qu'elles renforcent – une « { industrialisation } de la trivialité[475] » et notamment une { industrialisation } des textes et de leur circulation (chapitre IV). Enfin, un aspect « mythique » au sens certalien du terme. Dans quelle position subjective nous placent les { API } et plus largement la { programmation } comme écriture ? Quelle est « l'économie scripturaire[476] » de la { programmation } ? Comment cette économie traduit-elle des rapports de pouvoir qui se lisent dans les textes de réseau, notamment dans le rapport que nous, humains, entretenons avec les médias informatisés ? (chapitre V).

475 Jeanneret, Yves. *Op. cit.*, 2008.

476 De Certeau, Michel. *Op. cit.*, p. 195-224.

D'UN UNIVERSALISME COMBINATOIRE À LA MODULARITÉ & LA MANIPULABILITÉ DES FORMES-TEXTES.

III

En nous situant dans le cadre global d'une « culture anthologique » intégrée aux outils d'écriture du { Web } (III. Introduction), nous posons comme hypothèse de départ une tendance à la modularité du texte de réseau (III. 1). Les { *widgets* }, { formes-textes } permises par les { API } web, sont identifiés comme des vecteurs privilégiés de cette tendance. L'analyse de ces { *widgets* }, des lieux qu'ils occupent dans la page, amène à montrer que cette conception modulaire du texte s'articule à une logique de manipulation des modules de texte : réagencements, republication, réécritures, etc. Cette manipulabilité du texte – conçu alors comme agencement de blocs – est au cœur de ce qu'Illich appelle le « texte livresque[475] » mais trouve avec le { calcul } une dimension nouvelle qui ouvre au problème de l'éditorialisation, c'est-à-dire au sens que peuvent prendre ces différentes combinaisons et à l'outillage de ce travail éditorial (III. 2). Or, dans certains cas, cette éditorialisation est automatisée. Quels sont les enjeux de cette automatisation, à quoi sert-elle ?

API, calcul, forme-texte, Web, *widget*

Voir glossaire tome II, p. 3-25

475 Illich, Ivan. *In the Vineyard of the Text: a Commentary to Hugh's Didascalicon*. Chicago : University of Chicago Press, 1996 [1993].

API, architexte, discrétisation, documentation

Voir glossaire tome II, p. 3-25

À travers l'étude de deux exemples, nous montrerons que l'enjeu est de maximiser la manipulabilité des textes par une { discrétisation } de plus en plus fine de leurs blocs élémentaires (III. 3), { discrétisation } à laquelle participent pleinement les { API } contemporaines, notamment à travers leur { documentation }.

INTRODUCTION : NUMÉRIQUE ET « CULTURE ANTHOLOGIQUE » DU TEXTE

Ce passage de l'écriture au texte appelle quelques réajustements théoriques et introduit de nouvelles questions. Nous avons en effet traité jusqu'ici des médias informatisés en tant que machines écrivantes. Il s'agit désormais de les considérer comme des outils d'écriture. Quels sont les rapports entre médias informatisés et texte ? Comment conceptualiser ces derniers afin de comprendre le rôle qu'ils jouent dans la production des textes de réseau contemporains ?

La difficulté de ces questions tient au fait qu'il existe de nombreux travaux sur les rapports entre informatique et texte et particulièrement sur les textes de réseau (III. Introduction. A). Un état de l'art nous permettra de situer notre thèse entre deux courants. Compte tenu de notre problématique générale, nous retenons d'un côté la théorie des écrits d'écran et plus particulièrement le concept d'{ architexte } pour nous donner les moyens d'analyser sémiotiquement la pensée du texte présente dans les outils d'écriture. Compte tenu de notre corpus et de la forte présence de fragments circulants, nous retenons d'un autre côté l'hypothèse d'une « culture anthologique[476] » du numérique (III. Introduction. B). Nous pourrons alors formuler notre hypothèse principale pour ce chapitre : les { API }, en tant qu'{ architextes }, participent à l'établissement de cette culture anthologique, une culture qui s'hybride avec la composante informatique et combinatoire étudiée dans la première partie.

476 Doueihi, Milad. *Op. cit.*, 2011, p. 105-139.

A Texte et numérique

API, Web

Voir glossaire tome II, p. 3-25

Il existe de nombreux travaux sur les liens entre informatique et texte et plus précisément entre { Web } et texte. Selon certains, les premières structures hypertextuelles comme le { Web[477] } consacrent l'avènement d'une pensée post-structuraliste du texte[478] : dissémination du sens, non-finitude du texte, importance de la lecture comme co-écriture, auctorialité troublée si ce n'est absente… Cette appropriation de l'hypertexte par les théories littéraires a entraîné, entre autres, une mise en lumière de la notion de fragment[479], particulièrement opératoire pour penser les hypertextes. D'autres ont vu en l'hypertexte une structure adaptée à une nouvelle époque de l'écrit[480], à de nouveaux modes de pensée[481]. D'autres encore s'intéressent aux nouvelles formes de textualités permises par l'hypertexte[482], ou à ses modalités de lecture[483], que ce soit dans ses enjeux cognitifs ou économiques[484]. Les théoriciens des écrits d'écran ont, quant à eux et pour la plupart d'entre eux, concentré leur travaux sur les outils d'écriture du { Web } et la façon dont ils encapsulent certaines conceptions culturellement situées d'activités d'écriture : critique littéraire[485], écriture de l'intime[486], journalisme[487]… Au sein de ces travaux, nous nous situons à la croisée de deux courants. Compte tenu de notre problématique générale (située en Sciences de l'Information et de la Communication et plus particulièrement dans la sémiotique des écrits d'écran), nous nous intéressons aux modèles textuels produits par les outils d'écriture informatique comme les { API }. Par notre corpus

477 Attention, il ne faut pas confondre informatique, hypertexte et Web. L'informatique est la science née à la fin des années 1930 qui s'industrialise dans les années 1950. L'hypertexte est une forme d'organisation documentaire non-linéaire, qui s'appuie sur la technologie informatique, proposée par Ted Nelson en 1965. Le Web, né en 1991, est une mise en application du concept d'hypertexte sur le réseau Internet.

478 Landow, George P. *Hypertext: the convergence of contemporary critical theory and technology*. Baltimore : Johns Hopkins University Press, 1992.

479 Angé, Caroline. *Op. cit.* ; Angé, Caroline. Le fragment comme forme texte : à propos de *Fragments d'un discours amoureux*. *Op. cit.*, p. 23-34.

480 Bolter, David J. *Writing space: the computer, hypertext and the history of writing*. Hillsdale : L. Erlbaum Associates, 2001 [1991].

481 Clément, Jean. L'hypertexte, une technologie intellectuelle à l'ère de la complexité. Dans : Brossaud, Claire et Reber, Bernard (dir.), *Nouvelles technologies cognitives et épistémologie*. Cachan : Hermès Science – Lavoisier, 2007, [s. p.].

482 Rojas, Estrella. New figures of web textualities: from semio-technical forms toward a social approach of digital practices. *Archival Science*. 2009, vol. 8, nº 3, p. 227-246 ; Angé, Caroline. *Empreintes de l'hypertexte rétrospective et évolution*. Cachan : Lavoisier, 2011.

483 Boullier, Dominique, Ghitalla, Franck, Le Douarin, Laurence, *et al. L'outre-lecture : manipuler, (s')approprier, interpréter le Web*. Paris : Bibliothèque Publique d'Information, 2003.

484 Giffard, Alain. Des lectures industrielles. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *op. cit.*, p. 117-216.

485 Candel, Étienne. *Autoriser une pratique, légitimer une écriture, composer une culture : les conditions de possibilité d'une critique littéraire participative sur Internet : étude éditoriale de six sites amateurs*. Thèse de doctorat. Paris : Université Paris-Sorbonne, 2007.

486 Deseilligny, Oriane. Du journal intime au blog : quelles métamorphoses du texte ? *Communication & langages*. 2008, nº 155, p. 45-62.

487 Jeanne-Perrier, Valérie. Des outils d'écriture aux pouvoirs exorbitants ? *Op. cit.*, p. 97-131.

ensuite, qui contient de nombreuses formes fragmentaires, nous nous situons dans le courant de l'humanisme numérique proposé par Milad Doueihi qui voit dans la résurgence du fragment le signe de l'émergence d'une « culture anthologique ».

B La tendance anthologique du numérique

Milad Doueihi définit l'anthologie comme « une collection choisie »[488], mais surtout comme « [...] une économie au sens large du terme, de l'écrit. Elle inaugure des échanges, des déplacements, et rend possibles des formes de compétences autrement difficilement accessibles[489] ». L'anthologie est avant toute chose une forme littéraire : collection de morceaux choisis, le plus souvent les meilleurs d'une tradition et d'un corpus. Mais, par cette définition même, on voit comment l'anthologie engage déjà des rapports de pouvoir et une conception de la place de l'écrit dans une société : une anthologie permet de faire circuler des fragments d'œuvres sélectionnées (par qui ?) comme les « meilleures », demandant des modalités de lecture et d'écriture spécifiques. Il faut savoir instituer puis reconnaître l'autorité d'un texte, du compilateur (dont le statut doit lui aussi être institué comme différent de l'auteur), il faut produire des formes matérielles à même d'accueillir et de faire circuler ces anthologies, etc. L'anthologie engage toute la complexité des pratiques d'écriture, à la croisée de l'économie, des représentations de l'écrit, de son rôle social et communicationnel.

488 Doueihi, Milad. *Op. cit.*, 2011, p. 105.

489 *Ibidem*.

Tout le paradoxe historique de cette « tournure anthologique » contemporaine est qu'elle ne repose plus sur une « économie de la rareté » mais au contraire qu'elle se développe dans une économie d'abondance. L'informatique, le { Web } en particulier, permettent d'écrire à tout-va et de disséminer des textes avec une portée et à une échelle jusqu'ici inédites. Si l'anthologie est devenue « [...] la forme et le format par excellence de la civilisation numérique[490] », c'est justement, selon Milad Doueihi, parce qu'elle s'est transformée. D'une collection dictée par des contraintes matérielles, l'anthologie en est venue à représenter

API, architexte, forme-texte, industrialisation, petites formes, standardisation, Web, *widget*

Voir glossaire tome II, p. 3-25

> « [...] *des fragments conçus et formés pour la circulation et la transmission dans un environnement qui valorise une nouvelle manière de lire et d'écrire. Le fragment, ou toute pièce, tout document de n'importe quelle nature, est citable mais surtout il se livre à des formes d'intégration dans des outils d'écriture qui sont presque toujours des outils d'échange et de partage*[491]. »

La tendance anthologique du numérique est le résultat d'une convergence entre pratiques et outils d'écriture : les outils intègrent certains modèles textuels, par exemple l'anthologie ; les utilisateurs de ces outils retravaillent ce modèle anthologique. Surtout, le fragment est placé dans une économie nouvelle du partage et de l'identité, permise par la technologie. Entre pratiques et outils, il s'agit donc d'une forme d'{ industrialisation } de l'anthologie : cette dernière est préparée et intégrée dans des outils d'écriture standardisés et diffusés à échelle massive.

D'où l'intérêt de l'approche propre aux écrits d'écran et plus particulièrement du concept d'{ architexte }, soit « [...] les outils qui permettent l'existence de l'écrit à l'écran[492] [...] » en vertu de certaines conceptions culturelles de l'activité d'écriture. En portant notre regard sur les outils d'écriture du réseau (comme les { API }), nous pouvons voir en quoi ces outils sont porteurs d'une vision du texte participant à ce tournant anthologique. Il s'agit donc de se demander quelle est la conception du texte des { API } contemporaines, si les « { petites formes[493] } » sont les formes privilégiées de cette culture anthologique, comment elle est anticipée dans les { API } contemporaines, quelles en sont les répercussions sur le texte et plus particulièrement sur ce lieu fétiche du texte qu'est la page.

490 Doueihi, Milad. *Idem*, 2011, p. 106.

491 *Ibidem*.

492 Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Op. cit.*, p. 103.

493 Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 165-201.

1 UNE PAGE MODULAIRE

Si le numérique est porteur d'une culture anthologique et si les { API } participent à cette culture, on peut supposer que leur composante informatique joue un rôle dans les modalités effectives de cette culture anthologique. En d'autres termes, nous supposons que l'aspect computationnel des { API }, leur aspect combinatoire et calculatoire étudié dans la première partie, transforme la question du fragment et de l'anthologie. C'est pourquoi nous requalifions le fragment en « module[494] » et proposons l'hypothèse que l'une des principales conceptions du texte véhiculées par les { API } est celle d'une page modulaire, conçue comme un ensemble de blocs discrétisables, agençables et interchangeables. Les { *widgets* } sont identifiés comme l'un des vecteurs privilégiés de cette modularité. Ces { formes-textes } réactivent tout d'abord la notion fort ancienne d'ancrage (III. 1.A) : dans la page vont se déposer certains blocs. Mais le lieu n'est pas neutre dans cet ancrage. Les lieux de la page sont des lieux de mémoire (III. 1. B), ce qui permet d'expliquer en partie la progressive { standardisation } visuelle des textes de réseau. Dans cet ancrage, le cadre joue également un rôle prépondérant (III. 1. C) et une analyse plus fine des { *widgets* } montrera que ces modules ne sont pas des formes élémentaires mais sont eux-mêmes conçus comme des formes modulaires, enchâssées les unes dans les autres et divisibles en plus petites unités. La notion de modularité traverse donc toutes les échelles de la page, depuis le niveau le plus large (la page en entier) jusqu'au niveau microsémiotique des blocs qui la composent (III. 1. D).

494 Nous passons de la notion de « fragment » à celle de module, en raison de la dimension combinatoire de l'informatique. Cette notion nous semble plus à même de décrire une « mécanique » du texte plutôt qu'une esthétique de l'archipel, ce à quoi peut renvoyer la notion de fragment.

1 A Modularité, ancrage, & texte livresque : le texte au port de la page

API,
calcul,
code,
code source,
documentation,
HTML,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

Le premier aspect frappant des { *widgets* } est la notion d' « ancrage » ou d' « inscrustation », traduction du mot anglais « *embed*[495] ». On peut ainsi « incruster », « ancrer » dans une page des { tweets }, des vidéos, des statuts Facebook, des { *timelines* } (groupe de { tweets }), des descriptions de pages Facebook. Pour cela, il suffit de copier / coller dans le { code source } d'une page web un bout de { code } { HTML } permettant d'afficher sur cette page web un fragment de texte provenant de Facebook ou de Twitter. La plupart du temps, il suffit de cliquer sur une icône à côté d'une publication « ancrable » pour que Twitter ou Facebook génère automatiquement la ligne de { code } à utiliser. Il est également possible d'écrire soi-même cette ligne de { code }, auquel cas la { documentation } de l'{ API } de Facebook ou Twitter donne la syntaxe à respecter et quelques règles d'utilisation[496]. Mais pas seulement : cette { documentation } donne aussi de précieuses informations quant à la conception du texte qui sous-tend ces procédures d'ancrage.

495 *To embed* est un verbe anglais renvoyant à tout un champ lexical de l'intégration d'un élément exogène dans un support. Il peut vouloir dire « enfoncer dans du bois », noyer quelque chose dans du ciment ou encore plus spécifiquement sertir une pierre précieuse dans un écrin. Nous avons choisi la métaphore maritime de l'ancrage parce qu'elle fait référence à une conception du texte fort ancienne et que cette métaphore est reprise dans le vocabulaire du Web : on « navigue », on parle de « balises » (*anchor*, ancre) HTML, etc.

496 Voir annexe nº 6.

Fig. 9. Exemples de publications « ancrées » grâce à des API. De gauche à droite et de haut en bas : le module Facebook « Page » du site droit-finances.net ; un tweet d'@Akacha630 sur le site lemonde.fr ; une publication Facebook de la chaîne *BeIN Sports France* ancrée sur le site talkfoot.fr.

Sources : www.droit-finances.commentcamarche.net/faq/6848/preavis-de-depart-d-un-mois-location-modele. Capture réalisée le 8 juillet 2015 ; www.lemonde.fr/les-decodeurs/article/2015/07/16/ma-france-a-moi-le-mot-cle-qui-a-enflamme-twitter-le-14-juillet_4685117_4355770.html. Capture réalisée le 17 juillet 2015 ; www.letalkfoot.fr/video/ligue-des-champions/video/le-resume-de-barcelone-fc-seville-5-4. Capture réalisée le 13 août 2015.

En analysant la { documentation }, nous allons tout d'abord montrer que la notion d'ancrage s'appuie sur une vision du texte comme « contenu » qui vient se placer dans un « contenant » : la page (III. I. A. a). Dans ce jeu entre le texte et la page, les { API } prolongent une conception scolastique du texte, où le texte est une entité abstraite de la page et vient s'y ancrer, conception en place depuis le XII^e^ siècle (III. I. A. b) mais qui trouve avec le { calcul } et ses possibilités une nouvelle expansion (III. I. A. c).

a Des textes « ancrés » : résurgences du texte comme contenu et de la page comme contenant

Facebook présente les posts ancrés comme « [...] une manière simple de mettre des publications [...] dans le contenu de votre site ou de votre page web[497] ». Même simplicité vantée pour l'ancrage d'un module « page », en plus d'introduire l'idée que l'ancrage de ce module permet d'augmenter la valeur promotionnelle de la page concernée : « Le { *plugin* } "Page" vous permet de facilement ancrer et faire la promotion de n'importe quelle page Facebook sur votre site web[498]. » L'action de créer une forme texte est assimilée à une opération de publicité, amenant potentiellement de nouveaux internautes à consulter la page Facebook. Même discours, encore plus laudatif, chez Twitter. Un post ancré « [...] amène le meilleur contenu crée sur Twitter à l'intérieur de votre article ou de votre site web[499] ». Il peut contenir des « photos uniques » ou des « [...] prévisualisations de lien interactif pour mettre en lumière du contenu supplémentaire[500] ». Un { tweet } ancré contient tous les « éléments clés » de « l'expérience Twitter » que sont l'auteur, les mots-dièses, les mentions, permettant « [...] au public de votre site de se connecter avec la conversation globale qui a lieu sur Twitter[501] ».

code, documentation, *plugin*, tweet

Voir glossaire tome II, p. 3-25

Le premier atout de ces formes est qu'elles permettent de disséminer facilement, en dehors de leur contexte de publication initial, des fragments de textes issus de Twitter ou de Facebook. Leur second atout est qu'elles permettent d'agir sur ces textes, sans que l'internaute ait besoin de passer par Facebook ou Twitter[502]. La manière dont ces entreprises construisent à travers la { documentation } la valeur symbolique et technique de ces formes sera étudiée dans le prochain chapitre. Pour l'instant, concentrons-nous sur la conception du texte présentée dans ce discours.

D'après cette { documentation }, il est question de « contenu » : ce qu'on trouve sur Twitter est du contenu, tout comme ce qu'on trouve dans un

497 « [...] *a simple way to put public posts* [...] *into the content of your web site or web page* ». Facebook, documentation de l'API : developers.facebook.com/docs/plugins/embedded-posts. Capture réalisée le 29 juillet 2015.

498 « *The Page Plugin lets you easily embed and promote any Facebook Page on your website* ». Facebook, documentation de l'API : developers.facebook.com/docs/plugins/page-plugin. Capture réalisée le 29 juillet 2015.

499 « *An embedded Tweet brings the best content created on Twitter into your article or website* ». Twitter, documentation de l'API : dev.twitter.com/web/embedded-tweets. Capture réalisée le 14 juillet 2016.

500 « *An embedded Tweet may include unique photos* [...] *or interactive link previews to highlight additional content* ». Twitter, documentation de l'API : dev.twitter.com/web/embedded-tweets. Capture réalisée le 14 juillet 2015.

501 « [...] *helps your site's audience connect with the global conversation happening on Twitter* ». Twitter, documentation de l'API : dev.twitter.com/web/embedded-tweets. Capture réalisée le 14 juillet 2015.

502 Tout en générant des statistiques de lecture et de manipulation collectées par ces plateformes.

site ou une page web. Dans ce « contenu » peuvent être insérés d'autres contenus, du « contenu supplémentaire » comme une image dans un { tweet }. Ce contenu est inséré, « ancré », dans ce qu'on suppose être un « contenant » : la page. On pourrait dire qu'il est presque « fiché », comme on fiche un morceau de bois dans un autre. Mais ficher suppose que le morceau ne doit plus bouger, que l'ouvrage ainsi réalisé doit rester en l'état. L'ancrage, au contraire, suppose que le « contenu » s'intègre au contenant, sans pour autant qu'il en soit solidaire. Il peut partir, lever l'ancre à tout moment. Il suffit pour cela de supprimer une ligne de { code }, ou que le { tweet } ou statut soit supprimé par exemple. Dans cette conception, le texte est un ensemble de blocs de « contenus » que la page accueille plus ou moins temporairement. Ces blocs existent indépendamment de la page qui les accueille et peuvent circuler de page en page sans que cela ne remette en cause – idéalement – leur intégrité sémiotique, c'est-à-dire leur lisibilité et l'intégralité des informations qu'ils « contiennent ».

b Texte et ancrage : histoire d'une conception abstraite du texte

Cette conception est fort ancienne, puisqu'elle remonte au XII^e^ siècle. Elle est essentielle dans ce qu'Ivan Illich (1926-2002) nomme le « texte livresque[503] » qui est, selon lui, un cadre de pensée fondamental de l'Europe et plus largement de l'Occident depuis lors. Illich s'intéresse au rapport entre culture et technique[504] et plus spécifiquement au rapport entre modes de pensée, organisation sociale et pratiques d'écriture et de lecture. Dans son ouvrage publié au début des années 1990, il s'interroge sur les mutations du texte induites par l'informatique et pour cela il cherche à circonscrire la période précédente, justement en train de muter. Il la fait remonter au XII^e^ siècle, en étudiant un manuel de lecture : le *Didascalicon* de Hughes de Saint-Victor, composé dans le premier tiers du XII^e^ siècle. Dans ce manuel, lire est présenté comme une activité sacrée. Chaque mot, chaque lettre possède une consistance ontologique propre, intrinsèquement liée à la matérialité de la page. Page qui est, métaphoriquement, un jardin. Lire, c'est déambuler dans ce jardin, en cueillir les fruits et lentement les ingérer, les digérer. Suite à une foule de transformations tant sociales (passage des ordres monastiques à l'enseignement scolastique)

503 L'expression est reprise au critique Georges Steiner. Steiner, Georges. The end of bookishness ? *Times Literary Suplement*. 8-16 juillet 1988, p. 754.

504 Illich, Ivan. *Op. cit.*, p. 96.

que techniques (généralisation de l'emploi d'index, invention des livres portables, etc.), l'activité de lecture change de statut lors de ce siècle, le livre et le texte acquièrent un statut symbolique et métaphorique différent. Et un siècle plus tard, lorsque le théologien Bonaventure commente les travaux de son prédécesseur Hughes de Saint-Victor :

> « […] *le texte a déjà commencé à flotter au-dessus de la page. Il est en passe de devenir une sorte de vaisseau qui transporte, à travers l'espace séparant la copie de l'original, des signes remplis de sens ; Il mouille l'ancre ici ou là. En revanche, malgré cette dissociation entre le texte et la page, le texte tient sa position dans le livre. Le livre, en retour, devient métaphoriquement un port pour le texte, qui y décharge le sens et révèle ses trésors*[505]. »

Le « texte livresque » est une entité abstraite, quelque chose de « distinct du livre[506] », au contraire du texte monastique qui est ontologiquement lié à son support matériel. La notion même de « support » indique que nous sommes dans un cadre de pensée livresque, où le texte se « dépose », s'« ancre » dans telle ou telle page, qu'importe — idéalement[507] — la matérialité de cette page. Le sens dépendrait intégralement du texte et non de la page ou du livre. Mais pour exister, le texte a besoin de « s'ancrer » quelque part, d'être donné à lire. Il est alors rendu visible et lisible. Ainsi, le sens ne peut émerger qu'une fois le texte « ancré » en une page.

505 « […] *the text has already begun to float above the page. Its on its way to become a kind of vessel that ferries meaningful signs through the space separating the copy from the original; it drops anchor here or there. However, in spite of this dissociation of the text from the page, the text maintains its port in the book. The book, in turn, metaphorically stands as a harbor for the text where it unloads sense and reveals its treasures.* » Illich, Ivan. *Idem*, p. 118.

506 Illich, Ivan. *Idem*, p. 119.

507 Tout comme Illich, nous parlons du point de vue de la « métaphore centrale » (*root metaphor*) d'une époque et non du point de vue du texte concret dont les conditions de lisibilité et d'intelligibilité sont toujours déterminées par des facteurs matériels : éclairage, qualité du papier ou de l'écran, logiciel utilisé, etc.

c Texte livresque et informatique : le calcul comme prolongement de l'abstraction livresque

Illich considère – avec Steiner – que l'informatique signe la fin du livresque[508], ou en tout cas une mutation décisive du texte devenu « sans signification, absurde, sans auteur ; considéré comme une suite d'instructions[509] […] ». Il est formé de lettres semblables à des « fantômes qui apparaissent puis disparaissent[510] ». Notre thèse est un peu différente. Il est certain que l'informatique, envisagée comme technique d'écriture, introduit par le biais de la théorie de l'information une conception du texte comme une somme d'informations, un « contenu » abstrait réplicable à foison. Le fait que ce soit là confondre deux types d'information et les effets de cette confusion ont été étudiés par ailleurs[511]. Il est certain également que l'informatique se fonde tant techniquement que symboliquement sur ce que Bruno Bachimont appelle « l'idéalité computationnelle[512] » : un { calcul }, s'il est écrit formellement, peut s'effectuer sur n'importe quel support, à travers n'importe quelle configuration matérielle tant que c'est une machine universelle telle que la décrit Turing. De ce point de vue, effectivement, le texte – en fait les symboles manipulés par une machine – n'a pas d'auteur, pas de sens, tant et si bien qu'on peut, avec de tels prérequis, difficilement qualifier cela de « texte » voire d'« écriture ».

calcul

Voir glossaire tome II, p. 3-25

II | 3 | C | b
machine unvierselle telle que la décrit Turing
Voir p. 236

Cependant, les formes que nous observons se déploient à un autre niveau de l'écriture informatique. Elles ont un auteur (le profil Twitter ou Facebook), elles ont une signification potentielle, car elles puisent dans une mémoire sociale des formes. Ce ne sont pas des fragments anonymes, décontextualisés, abstraits, tout comme le texte livresque a besoin d'un ancrage dans la page pour exister. Nous sommes ici du côté de la matérialité de tout texte, de sa situation de lecture. Du côté de l'imaginaire du texte, ces formes ne vont pas à l'encontre du texte livresque mais en prolongent et radicalisent la tendance à l'abstraction et à l'ancrage. Cette radicalisation porte également sur les possibilités de circulation du

508 Et par livresque, encore une fois, il faut entendre un rapport au savoir qui dépasse la seule question du texte et de sa matérialité et qui inclut les pratiques d'écriture, ses institutions de transmission et de conservation.

509 « *The new "text" has no meaning, no sense, and no author ; it is conceived of as a command sequence* […] ». Illich, Ivan. *Idem*, p. 119, note n° 7.

510 « *arbitrary font shapes* […], *ghosts which appears and dissapears* ». Illich, Ivan. *Idem*, p. 119.

511 Jeanneret, Yves. *Y a-t-il (vraiment) des technologies de l'information ?* Villeneuve d'Ascq : Presses Universitaires du Septentrion, 2007 [2000].

512 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit.*

texte : ces blocs de texte peuvent circuler comme rarement auparavant. Par quoi s'opère cette radicalisation ? Par le couplage entre deux imaginaires sémiotechniques : celui du texte comme vaisseau, contenu pouvant s'ancrer n'importe où ; celui du { calcul } { formel } comme ce qui peut se passer d'une matérialité. En tant qu'écrits d'écran, ces formes rassemblent ces deux imaginaires. Ils s'appuient sur la puissance d'abstraction du { calcul } { formel } tout en reconduisant et amplifiant par cette dimension calculatoire la métaphore du texte comme ancrage.

API,
calcul,
documentation,
formalisme,
petites formes,
standardisation,
widget

Voir glossaire tome II, p. 3-25

Voilà donc identifiée la première couche symbolique de notre problème, condition de possibilité d'une conception du texte comme agencement combinatoire de modules comme les { *widgets* } et autres publications ancrées. Désormais, il faut entrer plus en avant dans le corpus et interroger la topologie de la page où s'ancrent ces modules. En effet, la page est l'une des dimensions constituantes du texte et de sa lisibilité, aussi abstrait que soit ce dernier. En cela, loin d'être un simple lieu de passage, la page et son organisation sont déterminantes dans le sens que peuvent prendre les formes qui s'y ancrent.

1 B Modularité, lieux de mémoire & mise en page : une géographie déjà bien installée

L'hypothèse d'une conception modulaire de la page nous a amené à reconsidérer à l'aune du { calcul } la métaphore de l'ancrage, métaphore constitutive du texte livresque tel qu'il s'est construit à partir du XII^e^ siècle. Voici le premier résultat de cette partie. Il permet de formuler une seconde hypothèse, conditionnée par la première. Si la page est un port où s'ancrent les différents { *widgets* }, il est possible que le lieu de l'ancrage ne soit pas neutre. Contre une pensée idéalisée du texte en dehors de toute matérialité, nous faisons donc l'hypothèse que l'espace de la page est constitutif dans la signification du texte et que cela peut expliquer la { standardisation } visuelle des écrits de réseau contemporains.

Pour vérifier cette hypothèse, nous tirons des exemples de notre corpus – constitué de divers exemples de vingt-quatre { petites formes } permise par les { API } ainsi que de la { documentation } qui les accompagne – et en analysons la répartition dans l'espace de la page entre seuils, centres et péritextes[513] (III. I. B. a). Il ressort de cette analyse un sentiment de déjà-vu : la répartition spatiale de ces formes est relativement standard. Ce qui nous amène à formuler l'idée que les lieux du texte sont des lieux de mémoire (III. I. B. b). Dans le cas spécifique des { API }, cela conduit à deux constats. Premièrement, que les lieux du texte sont des lieux de sédimentation de fonctions énonciatives (III I. B. c) ; deuxièmement que la page est un parcours et que les { petites formes } permises par les { API } intègrent dans cet espace-parcours la lecture mais aussi les commentaires, partages, etc. Ainsi, l'espace de la page numérique est un parcours qui donne à lire toutes les étapes d'une lecture idéale, de la lecture d'un « contenu » central aux gloses et commentaires périphériques (III. I. B. d).

513 Genette, Gérard. *Seuils*. Paris : Éditions du Seuil, 1987.

a Centres, seuils et marges du texte : lieux et fonctions des petites formes

Dans le corpus que nous avons rassemblé, deux formes ressortent : les formes qui constituent en elles-mêmes le texte principal, par exemple un { tweet } ancré dans un article de presse. Il fait partie du cœur de l'article, même s'il s'en distingue sémiotiquement. Au contraire, tout un ensemble de formes se trouvent dans les marges de la page : boutons, modules de commentaires… Ce sont des formes péritextuelles, en ceci qu'elles se distinguent assez nettement, notamment par les multiples cadres à l'écran, du texte principal. Il y a des pages hybrides, qui regroupent les deux types de formes. Il y a enfin une forme, le bouton *Connect*, qui échappe à notre typologie. Le bouton *Connect* permet d'autoriser un site à accéder à nos données Facebook afin de créer automatiquement un profil utilisateur. En tant que bouton, il semble être péritextuel, mais la plupart du temps il est mis en plein centre de la page, très visible, probablement pour déclencher un clic. Il permet d'entrer dans le site. C'est un seuil. Il est donc central, puisque passage obligé, tout en s'effaçant une fois franchi, puisque passage. C'est un cas assez atypique dans notre corpus, mais qui fournira un exemple particulièrement intéressant d'éditorialisation automatique.

API,
forme-texte,
tweet,

Voir glossaire tome II, p. 3-25

III 3 A
un exemple particulièrement intéressant…
Voir p. 218

Fig. 10. Exemple de forme ancrée au centre du texte.
Source : http://www.bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx.
Capture réalisée le 16 juillet 2015.

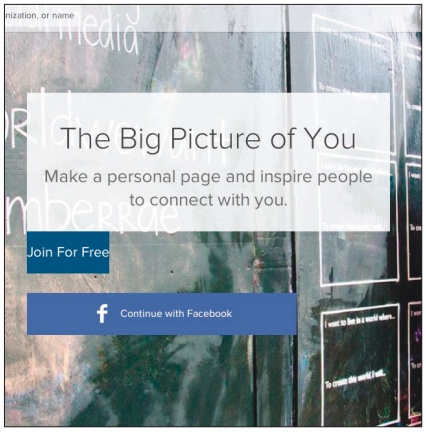


Fig. 11. Exemple de bouton-seuil.
Source : http://www.about.me.
Capture réalisée le 12 août 2015.

Fig. 12. Exemple de formes périphériques.
Source : http://www.slate.fr. Capture réalisée le 11 août 2015.

Cette distinction entre texte et « péritexte », héritée de Genette, est extrêmement classique mais permet d'entrer *a minima* dans le corpus qui, rappelons-le, est constitué de captures d'écran de { formes-textes } que permettent de générer les { API } de Twitter et de Facebook. Nous avons conscience du double problème que soulève cet emploi du concept genettien : d'abord que sa distinction entre texte, paratexte et péritexte s'applique facilement au texte livresque[514] mais peut-être moins facilement aux textes de réseau ; ensuite qu'elle reconduit une géographie du texte qui en est aussi une axiologie. Il y a « le » texte central et ses périphéries, celles-ci étant subordonnées à celui-là. Le risque est donc de reconduire malgré nous – et sans l'interroger – cette axiologie du texte. Reste que le vocabulaire employé par les plateformes (on « ancre » des « contenus »,

514 On considérera par exemple la couverture comme paratexte, les notes de bas de page comme péritexte, etc.

on fait des « boutons » de « partage » ou des « modules » de « commentaires ») appelle une culture où l'on sépare le texte principal de ses commentaires ou de sa circulation. Surtout, nous avons observé le plus souvent une disposition de la page où les boutons et autres modules occupent les marges ou les interstices de cette page, rejouant une organisation visuelle déjà fermement établie par neuf cents ans de culture livresque.

b Un air de déjà-vu… Les lieux du texte comme lieux de mémoire

standardisation, tweet, Web, *widget*

Voir glossaire tome II, p. 3-25

C'est en effet une forte impression de familiarité et de redondance qui se dégage en premier lieu du corpus. Les { *widgets* } de Facebook et Twitter – boutons divers, bloc de texte présentant les publications d'une page ou d'un profil, { tweet } ou vidéo ancrée – prennent place là où « on » les attend et par ce « on » nous entendons nos habitudes de lecture. Les boutons « Suivre » ou « *Like* » sont mis dans l'en-tête (*header*) ou en pied de page[515] (*footer*), voire à droite de la page, en marge du texte principal. Les boutons de partage d'un article encerclent parfois un texte mais, le plus souvent, ils se situent à la fin de celui-ci, retraçant un parcours où la dissémination de ce qui a été lu fait désormais partie de l'ordre du texte. Le « partage » est ce qui succède, dans l'ordre de la page, à la fin du texte. Il en va de même pour le module « commentaires[516] » qui, dans tout notre corpus, est placé en bas de la page, non pas en pied de page mais entre la fin du texte principal et le pied de page, comme faisant partie de ce texte tout en s'en distinguant nettement. Là aussi, le commentaire fait partie du texte. Tout deux partagent un espace similaire, même s'ils sont distincts. Plus généralement, les boutons se répartissent dans les espaces interstices de la page : entre des paragraphes, à droite du chapeau, dans des barres flottantes accompagnant le défilement de la page[517]…

515 Voir annexe nº 7.

516 Voir annexe nº 8.

517 Voir annexe nº 9.

Notre hypothèse est que ces « lieux du texte[518] » sont des « lieux de mémoire », première piste pour expliquer cette progressive { standardisation } visuelle du { Web[519] }. Les « lieux de mémoire » font partie de la tradition mnémotechnique des « arts de mémoire », tradition remontant à l'Antiquité grecque puis codifiée par les orateurs romains[520] dans le cadre de l'enseignement de la rhétorique et de la pratique judiciaire. Pour retenir un discours, les rhétoriciens et avocats recommandent de se représenter un lieu, un lieu connu ou un lieu imaginé, par exemple un temple ou un palais. Ce lieu doit pouvoir être parcouru dans un ordre bien précis et chaque station du parcours doit comporter un certain nombre de lieux, d'emplacements (*loci*). Par exemple, dans un temple, une pièce avec six alcôves ou six fenêtres. Dans chacun de ces *loci*, le rhétoricien place une image frappante, « agissante » (*imagines agentes*) en ceci qu'elle déclenche la remémoration des parties importantes du discours. L'exemple classique donné par Yates, tiré de l'*Ad Herrenium*, est celui d'un avocat devant retenir un discours pour défendre un individu accusé d'avoir empoisonné quelqu'un pour toucher un héritage. Le crime aurait été vu par de nombreux témoins (*testes* en latin). L'auteur de l'*Ad Herrenium* recommande en conséquence de s'imaginer un proche alité, malade, tenant en sa main gauche une coupe, en sa main droite une tablette et sur l'annulaire de cette main une bague incrustée de deux testicules de bélier. Cette image, frappante, permet de se remémorer la situation de départ du procès : un homme est mourant (l'alité), il a été empoisonné (la coupe) à cause d'une affaire d'héritage (la tablette). Il y a des témoins (*testes*, par homophonie avec *testiculos*, les testicules). Le parcours dans le bâtiment et d'autres images agissantes permettront de se rappeler du discours de l'avocat, le parcours correspondant à l'ordre (*ordinatio*) du discours et les images aux parties du discours.

518 Blanchard, Gérard. Textes, images et lieux de mémoire. *Communication & langages*. 1975, nº 28, p. 45-69. Pour une application des théâtres de mémoire à la presse quotidienne, voir Souchier, Emmanuël. L'espace de la presse quotidienne comme « théâtre de mémoire », forme instituante de la doxa contemporaine. Dans : Chevalier, Yves et Juanals, Brigitte (dir.), *Espaces physiques, espaces mentaux : identités et échanges*. Lille : Université Charles-de-Gaulle, 2007, p. 99-119.

519 Bien sûr, notre corpus représente une goutte microscopique dans un océan de pages et nous ne saurions présumer par là d'une tendance monolithique pour tout le Web. Nous cherchons simplement à expliciter et clarifier un certain nombre de redondances remarquées dans notre corpus, en estimant le rôle des outils d'écriture dans ces redondances. D'autres travaux ont par ailleurs remarqué une tendance similaire. Voir Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 165-201.

520 Il existe trois principaux traités d'arts de mémoire : le livre IV du *De Herrenium* (anonyme, 1er siècle avant notre ère), le *De Oratore* de Cicéron (83 / 84 avant notre ère) et l'*Institutio oratoria* de Quintilien (95 après notre ère). Pour une étude approfondie des arts de mémoire et de leurs transformations jusqu'à la Renaissance, voir Yates, Frances A. *Op. cit.*, 2014 [1966].

Les mnémotechniques antiques sont des systèmes complexes et qui peuvent aller de raffinement en raffinement par de savantes associations symboliques[521]. L'important est de bénéficier de la force syncrétique de l'image et de la puissance évocatrice du symbole pour faire tenir en un signe ramassé une foule d'informations. Ils s'articulent autour de ce couple fondamental *imagines / loci :* les images sont placées dans des lieux. L'image est effaçable à loisir, par exemple pour apprendre un nouveau discours. Le *locus* en revanche est immuable. Il est le support de projection de l'image[522]. Cicéron emploie la métaphore de la tablette de cire[523] : les lieux de mémoire sont la tablette, les images sont un équivalent des lettres que l'on trace. Cette métaphore est importante car elle nous fait comprendre un premier lien entre les arts de mémoire et l'écriture.

Ce lien n'est en effet pas évident, pour la raison suivante : les arts de mémoire antique fleurissent dans un contexte judiciaire, où l'oralité était largement préférée à l'écrit : c'est une composante essentielle du discours rhétorique, l'*actio* qui rend vivant le discours[524]. Pour nous, vivant dans une civilisation de l'écrit[525] et du livresque, ce système nous apparaît complexe. Il est plus simple d'écrire la liste des choses à retenir sur un bout de papier. Il est vrai que les arts de mémoire, en tant que stricte technique de mémorisation, ont cédé le pas à la notation écrite. Avec les progrès de l'alphabétisation et un accès facilité à des outils et des supports d'écriture, le besoin de s'imaginer des palais où placer des images suffisamment évocatrices pour se rappeler nos tâches de la semaine est moins prégnant. La relative disparition des arts de mémoire comme mnémotechnique ne veut pas dire en revanche que ce principe des *loci* et des *imagines* a disparu[526].

C'est le graveur, typographe et universitaire Gérard Blanchard (1927-1998) qui, suivant Yates, propose la thèse selon laquelle les arts de mémoire ont subsisté dans le livresque même et plus précisément dans la mise en page. Que font les metteurs en page ? Ils « [...] définissent, pour

521 Pourquoi par exemple des testicules de bélier ? Et pourquoi l'annulaire ?

522 Nous parlons ici de l'art de mémoire antique. L'agentivité des images va lentement évoluer vers un pouvoir quasi magique de talisman, notamment à la Renaissance, tandis que les *loci* vont eux aussi devenir mobiles, comme dans le système combinatoire de Raymond Lulle. Sur ces points, voir Yates, Frances A. *Idem*, p. 154. Et sur le lullisme, chapitre VIII, « *Lullism as an art of memory* », p. 173-198.

523 Cité par Yates, Frances A. *Idem*, p. 2-7.

524 Non pas que ce système soit indépendant d'une raison graphique, l'image de la tablette de cire le montre bien : on pense la mémoire comme un support d'écriture.

525 Nous entendons par « civilisation de l'écrit » une aire culturelle où l'écriture comme technique prend une place symbolique, économique, matérielle et cognitive très forte. Nos structures de pensée dépendent en grande partie de l'écriture, nous apprenons à lire et écrire à un très jeune âge, ce par une politique étatique, les moyens d'écriture sont relativement bons marchés et accessibles à tous, etc. Toutes ces médiations instituent l'écriture comme une technique fondamentale pour tout citoyen français et plus largement pour tout occidental moderne. Sur ce point, voir De Certeau, Michel. *Op. cit.* Chapitre X « L'économie scripturaire », p. 195-224. Pour l'importance politique et culturelle de l'alphabétisation, voir Illich, Ivan et Sanders, Barry. *ABC : l'alphabétisation de l'esprit populaire*. Paris – Montréal : L'Harmattan – Boréal, 1990 [1988]. Pour le rôle cognitif de l'écriture, voir Goody, Jack. *Op. cit.*, 2007 [1979].

526 La thèse principale de Yates est que les arts de mémoire ont eu une influence décisive à la fois sur l'esthétique picturale de la Renaissance et sur la naissance de la science moderne.

les messages à formuler, des “lieux à textes”. Dans l’espace de la page, ils déposent, ils disposent des textes. Ils vont, ce faisant, faire appel à un type de mémoire semblable mais infiniment plus fruste que celle qui était requise au Moyen-Âge[527] […] ». Une page, manuscrite, imprimée ou numérique, est un lieu de mémoire. Non pas au strict sens antique où l’on y mettrait des images frappantes pour se souvenir de choses importantes, mais au sens où elle consiste en un certain nombre de lieux (haut de page, pied de page, marges, coin inférieur droit, etc.) où sont placés des textes[528]. Qu’on songe par exemple à la grille de composition : la page est divisée en zones égales où vont se répartir les blocs de textes. Cette grille est connue depuis la composition des manuscrits hébraïques – on l’appelle alors « réglure[529] » –, elle est massivement utilisée dans la typographie suisse pour le dessin de lettres[530], elle est enfin la composition même de tout écran numérique. Un écran est en effet divisé en { pixels }, unité graphique minimale. Chaque { pixel } reçoit une valeur de couleur selon ce qui doit être affiché. Tout élément disposé sur une page web ou dans une { application } reçoit donc une certaine place que l’on mesure en { pixels[531] }.

application, pixel, Web

Voir glossaire tome II, p. 3-25

c Les lieux du texte comme lieux de mémoire : la sédimentation des fonctions énonciatives

Si la page est un ensemble de lieux, ces lieux ne sont pas disposés au hasard. Ils procèdent d’une culture visuelle et livresque, d’une mémoire des formes où est attribuée à chaque lieu une fonction. C’est pour cela que c’est un système de mémoire : à chaque lieu est associée une valeur au texte, ou une fonction d’énonciation. Dans un livre, par exemple, on sait par habitude que le bloc de texte du centre est le texte principal, qu’on retrouvera la pagination dans les marges, les notes en bas de page, etc. C’est là l’exercice d’une mémoire liée au lieu. De la même façon, même si c’est une mémoire bien plus récente, le { Web } est aujourd’hui suffisamment « stabilisé » visuellement pour qu’on puisse s’attendre à trouver en haut à gauche ou à droite de la page une icône permettant

527 Blanchard, Gérard. Textes, images et lieux de mémoire. *Op. cit.*, p. 50.

528 Ce passage de l’image au texte est permis par la dimension iconique de tout texte, sa « livrée graphique ». Voir Christin, Anne-Marie. *Op. cit.* ; Souchier, Emmanuel. L’image du texte : pour une théorie de l’énonciation éditoriale. *Op. cit.*, p. 137-145.

529 Dukan, Michèle. *La réglure des manuscrits hébreux au Moyen Age*. Paris : Éditions du Centre National de la Recherche Scientifique, 1988.

530 Blanchard, Gérard. Textes, images et lieux de mémoire. *Op. cit.*, p. 52.

531 Au-delà donc de la stricte page web, le support lui-même fonctionne sur un principe similaire : il est divisé en éléments immuables – des *loci* – auxquels vont être attribués différentes couleurs selon le contexte – l’équivalent des *imagines*.

de revenir à la page d'accueil du site. On sait par ailleurs que les menus déroulants qui se trouvent en haut de ladite page permettent de parcourir l'ensemble du site, alors que des menus situés en-dessous du *header* renvoient généralement à une navigation à l'intérieur d'un sous-ensemble du site[532], par exemple une rubrique spécifique d'un journal en ligne.

Ces habitudes de lecture sont le résultat de la sédimentation de pratiques de lecture et de mise en page. Et c'est cette mémoire des formes qui explique, en partie, pourquoi on trouve les mêmes boutons aux mêmes endroits. En d'autres termes : la { standardisation } visuelle progressive des pages web est en partie due au fait que ce sont des lieux de mémoire, où l'espace n'est pas neutre mais a une fonction précise dans l'énonciation et dans l'appréhension du texte. Par habitude de lecture et de composition, cette fonction se fixe progressivement. Reste à voir quelle serait dans notre corpus la disposition mémorielle des écrits de réseau et quelle part occupent les formes générées par les { API }.

API, standardisation, *widget*

Voir glossaire tome II, p. 3-25

Pour ce faire, nous avons synthétisé à partir de notre corpus une sorte d'archétype de page. Nous avons commencé par construire cette page à l'aide de fragments du corpus. Nous sommes ensuite monté d'un niveau d'abstraction pour formaliser – indépendamment des formes concrètes – les fonctions de chaque zone en termes d'énonciation. Où sont les blocs de textes ? Que permettent-ils par rapport à la page et / ou au site ? Qui assume le rôle d'énonciateur dans telle ou telle zone de la page ? Cela permet de voir une construction abstraite mais qui a l'avantage de présenter de façon claire une tendance dans la mise en page de sites web contemporains. Cette construction est, si l'on considère une page comme un lieu de mémoire, une organisation mémorielle. Après une analyse du système énonciatif à l'œuvre, nous verrons où se situent les boutons et { *widgets* } dans ce lieu de mémoire, pour en tirer quelques résultats quant à la modularité de la page et à l'énonciation qui s'en suit.

532 Voir annexe nº 10.

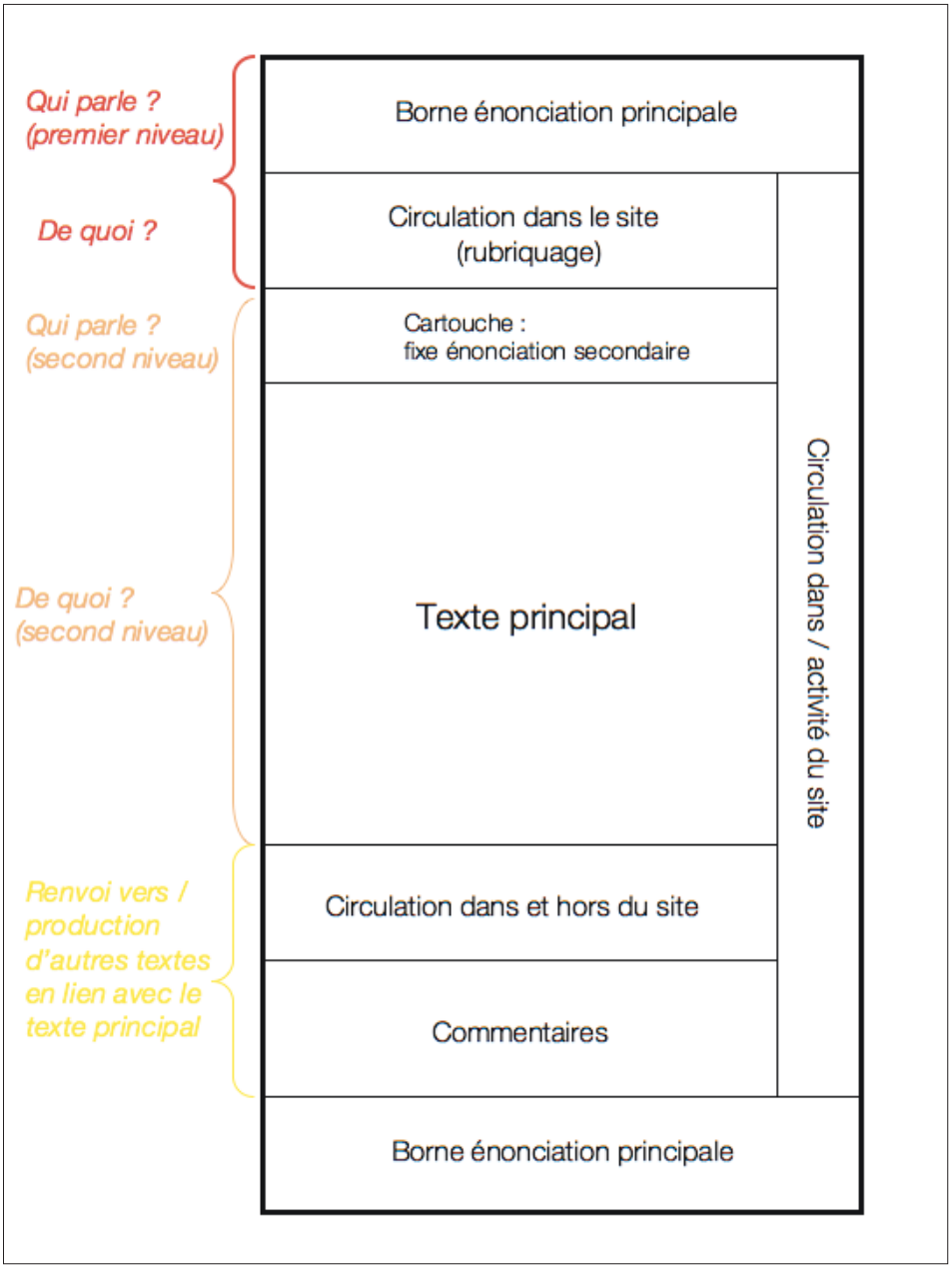


Fig. 13. Schéma canonique d'une page web.

On remarque en premier lieu une énonciation redoublée. Le *header* et le *footer* contiennent généralement des informations quant au site visité, par exemple lemonde.fr. On y trouve donc le logo du *Monde*, souvent une barre de recherche ou un bouton de connexion à un espace utilisateur… Tout ce qui se passe à l'intérieur de cette zone concerne le site en entier. C'est en quelque sorte un chapeau qui fixe d'emblée où l'internaute se situe et indique quelle est l'instance d'énonciation principale. Le *header* est la limite supérieure de cette énonciation, le *footer* sa limite inférieure. Ce qui se passe entre ces deux limites est de l'ordre d'une énonciation secondaire, à l'intérieur de la première. L'énonciation secondaire est bien

souvent définie par un cartouche avec un titre, le nom de l'auteur de l'article ou de la page, la date de publication éventuelle, etc. Par exemple, un article sur lemonde.fr ou une page du site south-park-streaming.com où l'on peut regarder un épisode de la série[533]. On ne peut dire que cet article ou cette vidéo sont « sur » ce site qu'en vertu de cette énonciation en inclusion : l'article a certes un auteur, mais il est pris en charge par une entité supérieure, symboliquement et spatialement : *Le Monde*[534].

Deuxièmement, la page s'organise autour d'un mouvement centripète. Les deux cadres inférieurs et supérieurs, souvent assortis d'un cadre sur le côté permettant la navigation dans le site, guident le regard au centre de la page où se trouve le texte principal (et aussi central). C'est ce texte qui fait la spécificité de la page, alors que les menus déroulants ainsi que le *header* et *footer*, restent immuables au cours de la navigation[535] et leur portée d'action est au niveau du site tout entier. Il y a donc une forme de spécification de l'activité : on se trouve sur le site x, à l'intérieur duquel nous lisons une page y, à l'intérieur de laquelle on peut trouver une image ou une vidéo z, etc. Cette spécification est lisible par un système spatial de cadres permettant l'imbrication des énonciations. Il y a là une première concaténation entre le spatial et le symbolique : le plus important, le texte propre à la page que nous lisons, est au centre de cette page. Ce centre peut être uniquement symbolique. Dans le cas d'une page avec une foule de commentaires qui s'étendent *ad indefinitum*, il est certain que le « centre » géographique de la page n'est pas le texte principal. Il n'en reste pas moins que les commentaires se définissent précisément par rapport à un texte initial, qui fait l'objet de ces gloses. Dans ce cas, le centre est symbolique plus que physique, mais la distinction entre texte central / textes périphériques reste opératoire.

533 Voir annexe nº 11.

534 Il peut arriver que cette énonciation soit à l'intérieur même de l'énonciation seconde, par l'inclusion de tweets ancrés par exemple. Le système se complexifie, mais la structure reste la même, à la manière de poupées russes.

535 Ce lien entre inclusion spatiale et spécification confirme une tendance plus large des médias informatisés : un article est *dans* lemonde.fr, qui est lui-même représenté *dans* un navigateur, lui-même inclus *dans* un système d'exploitation. Voir Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Op. cit.*, p. 102.

d Les lieux du texte comme lieux de mémoire : la lecture comme parcours

Si la page est un lieu de mémoire, comme le suggère Blanchard, alors lire, c'est parcourir un espace et un espace organisé dans un ordre précis[536]. Le parcours dans la page est une déambulation dans un ordre prescrit – et préécrit – de lecture, où chaque station, chaque zone, chaque point d'arrêt a une fonction déterminée. Sur ce point, il existe une continuité historique et culturelle entre les arts de mémoire, le texte livresque, les structures de lecture de la presse quotidienne[537] et les textes de réseau contemporains : tous utilisent une certaine topologie de la page comme espace de mémoire.

petites formes,

Voir glossaire tome II, p. 3-25

Mais les { petites formes } amènent une nouvelle donne. La spécificité de notre corpus tient en effet à la façon dont l'organisation spatiale est en fait l'organisation d'un parcours de lecture auquel sont adjointes certaines pratiques sociales : partage, commentaires… C'est là notre troisième point, dont la démonstration s'appuie sur la troisième zone identifiée sur notre schéma. Juste en-dessous du texte principal, on trouve bien souvent une zone regroupant les boutons de partage, des liens thématiques sur l'article lu (*tags*) renvoyant à d'autres articles sur les mêmes thématiques, des liens vers d'autres sites, etc. Il ne s'agit plus de circulation à l'intérieur du site, par exemple à l'aide de rubriques, comme dans les zones supérieures. Il s'agit d'un renvoi vers d'autres textes (liens internes et externes) qui sont supposés être en rapport avec le texte principal. De la même façon, les modules de commentaires sont bien souvent situés juste en-dessous de cette zone : ils outillent la production de textes au sujet du texte principal. Commentaires, liens internes ou externes, boutons de partage… sont donc liés entre eux par le fait qu'ils se définissent en référence au texte principal situé juste au-dessus, c'est-à-dire ce qui vient d'être lu[538]. Si nous envisageons la page comme un espace, mieux que cela, comme un parcours, il faut alors comprendre la mise en page comme la mise en ordre d'un parcours de lecture. On se rend sur un site x (*header* et *footer*) pour lire une page y (zone centrale) et ensuite la partager, la commenter ou être dirigé vers d'autres pages (zone inférieure). La spécificité de

536 Yates insiste sur l'importance du parcours dans le lieu de mémoire : on passe d'un lieu à un autre, dans l'ordre dans lequel on veut se souvenir des choses importantes et de leur enchaînement. Le parcours est partie intégrante du système mnémotechnique.

537 Souchier, Emmanuël. L'espace de la presse quotidienne comme « théâtre de mémoire », forme instituante de la doxa contemporaine. Dans : Chevalier, Yves et Juanals, Brigitte (dir.), *op. cit.*, p. 99-119.

538 Nous présupposons une lecture allant de bas en haut. Cette présupposition n'est pas un postulat arbitraire. Nous supposons que l'ordre de lecture d'une page web suit l'ordre de lecture occidental. De plus, nous nous appuyons sur une analyse du texte, qui donne à lire par son agencement une certaine situation « idéale » de lecture.

notre corpus tient dans cette dernière étape : les pratiques anthologiques (partage, favoris, etc.) ou de gloses sont incluses comme faisant partie de la lecture. Les boutons « J'aime » ou « Partager », en se glissant dans cet espace, équipent et intègrent l'acte de lecture. En d'autres termes, les « industries médiatisantes[539] », grâce aux formes que permettent de produire leurs { API }, s'immiscent dans ce parcours de la lecture et contribuent à en modifier la teneur et les enjeux[540]. La lecture « idéale » – celle qui est construite dans les pages web de notre corpus – est une lecture dont les pratiques anthologiques sont l'aboutissement, confirmant par là l'argument de Milad Doueihi d'une « culture anthologique » spécifique au numérique car intégrée et outillée à même la page.

API, petites formes, *widget*

Voir glossaire tome II, p. 3-25

La lecture est donc un parcours dans un espace et les pages web contemporaines – *via* les { API } – intègrent non seulement le texte principal mais aussi ses reprises, commentaires… en somme son devenir anthologique. La mise à jour d'une structure canonique, bien qu'abstraite, permet de faire voir les principales étapes de ce parcours qui, à force de répétitions, intègre nos habitudes de lecture et, par effet d'inertie, contribue à standardiser l'aspect formel des pages web. Reste une question : dans ce parcours, où se trouvent les { *widgets* } ? Nous avons mentionné qu'on en trouve souvent dans la zone inférieure, ce qui est vrai – en tout cas pour les boutons et autres formes péritextuelles. Mais on peut également les trouver dans le *header*, à côté du nom de l'auteur de la page, dans le *footer*… On les retrouve en fait disséminés un peu partout sur la page. La question devient alors vite un paradoxe : comment ces formes usinées pour être partout identiques acquièrent-elles une valeur sémiotique différenciée ? Qu'est-ce qui permet de faire la différence entre un bouton « J'aime » placé en haut d'une page et le même bouton situé en bas d'un article au milieu de la même page ? La réponse à ce paradoxe se trouve dans le jeu, essentiel, des cadres qui permettent de différencier au sein de la page des zones d'énonciation différentes.

539 Jeanneret, Yves. *Op. cit.*, 2014, p. 640-646.

540 Nous rejoignons sur ce point les analyses d'Alain Giffard sur l'industrialisation de la lecture. Giffard, Alain. Des lectures industrielles. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *Op. cit.*, p. 117-216.

1 C Modularité & cadrage : le cadre comme opérateur de contextualisation[541]

Nous voici arrivés à l'un des croisements possibles entre une culture et une industrie du texte. Si la page est devenue – au fil d'une longue élaboration historique – un lieu de mémoire où s'ancrent des formes sémiotiques, les formes de notre corpus sont aujourd'hui des modèles usinés pour être formellement identiques malgré leur ancrage en un lieu donné. Qu'est-ce qui distingue un bouton « J'aime » Facebook d'un autre bouton, que ce soit entre deux pages différentes, ou (cas encore plus radical) du même bouton mis deux fois dans la même page ? Qu'est-ce qui permet de construire sémiotiquement une différenciation entre ces deux boutons ? L'hypothèse que nous allons défendre dans cette sous-partie est que c'est l'espace qui permet cette différenciation et plus particulièrement la délimitation de l'espace par le cadre. En couplant une sémiotique du cadre (III. I. C. a) à la théorie de l'énonciation éditoriale, nous verrons que dans le cas des écrits d'écran, le découpage de l'espace de l'écran est d'ordre techno-sémiotique : il permet de visualiser les différentes couches logicielles effectives dans la production d'un texte (III. I. C. b). En portant cet acquis au niveau des { API } et des formes de notre corpus, nous pourrons alors démontrer notre hypothèse de départ : c'est bien le cadre et le découpage de l'espace qui permettent de contextualiser et de différencier sémiotiquement les boutons et autres { petites formes } produites par les { API } (III. I. C. c).

a Sémiotique du cadre

Le cadre a été notamment étudié par AnnetteBéguin-Verbrugge, qui le définit comme la condition phénoménologique de l'apparition du sens : « [l]e texte donné à la lecture, quel que soit son support, est toujours en partie déterminé pas ses contours et ses entours[542] ». Faisant fond sur la « pensée de l'écran[543] » qu'Anne-Marie Christin voit à l'origine de tout système d'écriture, Annette Béguin-Verbrugge montre que le découpage de l'espace, que ce soit celui d'une page, d'un écran, ou d'une ville, instaure un espace d'apparition possible de sens, par la

541 La présentation qui suit des travaux d'Annette Béguin-Verbrugge est une reprise – légèrement modifiée – de notre mémoire de recherche. Goyet, Samuel. *Op. cit.*, p. 28-30.

542 Béguin-Verbrugge, Annette. *Images en texte, images du texte : dispositifs graphiques et communication écrite*. Villeneuve-d'Ascq : Presses Universitaires du Septentrion, 2006, p. 12.

543 Christin, Anne-Marie. *Op. cit.*

simple définition d'un dedans et d'un dehors. Par cette définition, au sens étymologique de poser des limites, se traduit un geste éditorial voire poétique, une prise de pouvoir sur le sens. Le cadre est donc une décision sémiotique essentielle et consubstantielle à l'écrit : sans cadre, pas de sens possible. Le cadre circonscrit dans l'espace du visible un espace spécifique de potentialité du sens : « Sur un même support, cadres, marges et contours divers servent à organiser les relations entre éléments hétérogènes, à les intégrer dans le même empan perceptif et à les impliquer dans le même processus de construction inférentielle[544]. »

Sur ces bases, Annette Béguin-Verbrugge définit trois fonctions sémiotiques du cadre : la fonction indexicale, la fonction partitive et la fonction relative. Le cadre est un index en ceci qu'il renvoie à son dedans, soit l'espace d'apparition du sens. Dans cette opération, le cadre désigne donc le lieu où le sens peut advenir, où le regard devrait se poser. Il est à la fois un index et une conduite du regard. Ainsi, « [...] l'introduction d'un cadre ou d'une bordure constitue donc une instruction forte, voire parfois une véritable injonction de lecture[545] ».

Mais l'injonction du cadre est plus qu'une conduite du regard. Elle est aussi dans le statut de l'énoncé qui est ainsi mis en valeur. En effet, « [...] la limite contient en elle-même un présupposé sur l'unité et le statut de l'information qu'elle circonscrit[546] ». Instaurer un cadre, c'est de fait organiser un texte et indiquer que les éléments qui sont compris dans ce cadre font sens ensemble, font sens commun. Le cadre est donc l'opérateur qui assure le passage dans l'organisation du visible, entre la cohésion (la proximité spatiale de deux énoncés ou ensembles de signes) et la cohérence (une unité de sens). C'est la deuxième fonction du cadre chez Béguin-Verbrugge : la « fonction partitive[547] ».

Surgit alors un paradoxe. En effet, le cadre assure dans le même temps la séparation et le lien, il « isole tout en instaurant une relation[548] ». En ce sens, il est un opérateur de distinction et d'articulation des énoncés entre eux. C'est pourquoi cette deuxième fonction est extrêmement liée à la troisième qui est la fonction « relative », où le cadre « [...] permet d'entrer dans le jeu des relations logiques entre les sous-énoncés, eux-même

544 Béguin-Verbrugge, Annette. *Op. cit.*, p. 28.

545 Béguin-Verbrugge, Annette. *Idem*, p. 70.

546 *Ibidem*.

547 Béguin-Verbrugge, Annette. *Idem*, p. 71.

548 *Ibidem*.

cadrés, qu'il inclut[549] ». La fonction relative du cadre, c'est sa capacité à instaurer une relation – charge au lecteur de la qualifier – entre deux éléments qui appartiennent à la même zone encadrée. Cette dernière fonction se complexifie, puisqu'elle inclut un niveau second, étant donné que les énoncés mis en relation sont dits ici cadrés. La fonction relative serait donc en quelque sorte, si l'on suit cette définition, une méta-fonction du cadre. Or, nous avons montré que la fonction partitive reposait également sur une mise en relation d'éléments *a priori* hétérogènes. On peut donc considérer que la fonction relative est intimement liée à la fonction partitive et qu'elle n'est pas réservée à un niveau supérieur de cadrage.

b Cadre et énonciation éditoriale dans le cas des écrits d'écran : une spécification techno-sémiotique.

Cette théorie du cadre peut être couplée à la théorie de l'énonciation éditoriale[550], qui soutient que l'ensemble des acteurs participant à la mise en forme d'un texte et à sa « livrée graphique » – typographes, metteurs en page, imprimeurs – participent à son énonciation. Si la mise en page est une forme d'énonciation qui participe à la polyphonie de tout texte, alors le cadre est un des outils principaux de cette énonciation puisqu'il organise les énoncés, guide le regard, « dit » quelque chose du texte qu'il donne à lire. Les cadres définissent donc des types d'énonciation[551]. À partir de ce couplage entre théorie des cadres et énonciation éditoriale, on peut alors adjoindre la spécificité des écrits d'écran comme des outils toujours « encadrés » où chaque niveau enchâsse les énonciations entre elles et spécifie la tâche accomplie.

549 Béguin-Verbrugge, Annette. *Idem*, p. 93.

550 Souchier, Emmanuel. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

551 Souchier, Emmanuël. Histoires de pages et pages d'histoire. Dans : ZALI, Anne, *L'Aventure des écritures : la page*. Paris : Bibliothèque Nationale de France, 1999, p. 19-55.

Pour comprendre ce dernier point, il faut repartir de l'idée fondamentale de Turing que l'ordinateur est une machine « universelle », au sens bien particulier d'une puissance d'imitation d'autres machines dont le comportement peut être calculable. Ce rapport entre machine universelle et imitation d'une machine particulière s'observe dans la matérialité même des écrits d'écran : l'enchâssement des cadres à l'écran est une progression de l'universel au particulier, de la couche technique la plus générale à la plus particulière.

II 3 C
une puissance d'imitation d'autres machines
Voir p. 234

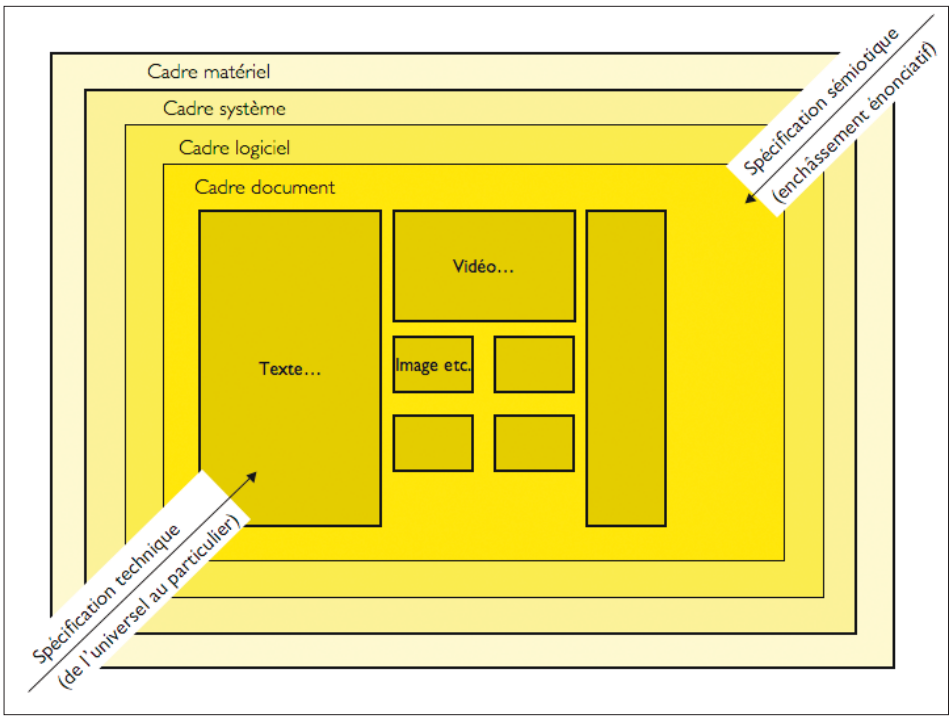


Fig. 14. Enchâssement des cadres dans les écrits d'écran et spécification techno-sémiotique.

Ce schéma reprend et prolonge la typologie de cadres établie par Emmanuël Souchier[552]. Le cadre { matériel } indique là où l'{ interface } va apparaître. C'est le cadre le plus « universel » car virtuellement toutes les opérations de la machine peuvent y être représentées[553]. Cette universalité repose sur la scission, construite au fil de l'histoire de l'informatique, entre le *hardware* comme agencement matériel et le *hardware* qui peut s'exécuter de façon idéalement indépendante de cet agencement matériel. Le second cadre est le cadre système : il spécifie un peu plus quelles activités vont pouvoir être menées et quels { logiciels } peuvent être exécutés. Il y a là une première différenciation sémiotique – on peut facilement

code, **code source**, **HTML**, **interface**, **logiciel**, **matériel**, **système d'exploitation**, **tweet**

Voir glossaire tome II, p. 3-25

I 1 D
de façon idéalement indépendante de cet agencement …
Voir p. 96

552 Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Op. cit.*, p. 100-105.

553 On pourrait objecter que la marque de l'écran est un signe distinctif qui spécifie déjà quel type de système d'exploitation est installé. C'est en partie vrai car il est possible d'installer MacOS X ou GNU/Linux sur un ordinateur non fabriqué par Apple, il est possible d'émuler un système d'exploitation à l'intérieur d'un autre, etc.

reconnaître un bureau sous MacOS X d'un bureau sous Windows ou GNU/Linux – aussi bien que technique : les mêmes raccourcis clavier ne déclencheront pas les mêmes actions selon le cadre système. Ensuite, le cadre { logiciel }, qui remonte encore d'un cran dans la spécification de l'activité menée avec un ordinateur. Selon que ce cadre est un navigateur web, un traitement de texte ou un lecteur vidéo, les pratiques ne seront pas les mêmes et l'{ interface } non plus. De la même façon, la même commande ne déclenchera pas nécessairement la même action selon le cadre { logiciel }. Il y a donc une spécification qui englobe les deux aspects des écrits d'écran : l'aspect sémiotique et l'aspect technique. Enfin, le cadre document est là où apparaît ce qui est actuellement lu par un utilisateur : une page web, un { logiciel } de messagerie, une vidéo… C'est dans ce cadre que l'activité est la plus spécifique, la plus déterminée. Cet emboîtement peut d'ailleurs se prolonger au sein même du « cadre document ». C'est le cas lorsqu'une page web inclut un bloc de texte, une vidéo et / ou un { tweet } par exemple. De ce point de vue, la modularité des pages que nous observons est une tendance plus générale des écrits d'écran.

Cet enchâssement est techno-sémiotique, c'est-à-dire que les rapports d'inclusion et de dépendance des différents cadres entre eux ont tout à la fois à voir avec le fonctionnement de la machine et le fonctionnement culturel des signes à l'écran, la manière dont ils sont signifiants pour un humain. Techniquement, le { tweet } ou la vidéo ancrés dépendent de la page dans lequel ils sont ancrés (cadre document), laquelle dépend du navigateur qui en interprète le { code } { HTML } (cadre { logiciel }), lequel dépend du { système d'exploitation } qui en compile puis en exécute le { code source } (cadre système), { système d'exploitation } dont l'exécution dépend en retour de l'initialisation de la machine (cadre { matériel }). Le feuilletage à l'écran traduit ces rapports d'interdépendance, même si tous ces niveaux fonctionnent en même temps. D'un point de vue sémiotique, l'enchâssement de différents niveaux traduit les différentes activités de lecture et d'écriture à l'œuvre, du plus générique au plus particulier. Et si la disposition d'une page est aussi une affaire d'énonciation, alors cet enchâssement est également une imbrication de différents niveaux d'énonciation qui composent la polyphonie d'un écrit d'écran. Dans une page web, quand on prête une attention fine à ce qui apparaît, c'est bien sûr l'auteur de cette page qui « énonce » quelque chose.

petites formes, standardisation, système d'exploitation,

Voir glossaire tome II, p. 3-25

Mais c'est aussi l'ensemble des entités (navigateur, { système d'exploitation }, etc.) qui permet l'affichage de cette page, dont le feuilletage relevé plus haut permet une première approche[554].

La position dans l'espace est donc différenciante, tant d'un point de vue technique que sémiotique. Qu'est-ce que cela veut dire ? Que l'efficacité énonciative et technique d'un signe dépend de sa position dans l'écran et du cadre dans lequel il se situe[555]. C'est l'une des clés de compréhension de ce que nous observons dans notre corpus. Les boutons se répartissent un peu partout sur la page, qui prend l'allure d'une composition de modules standards. Mais cette { standardisation } est en quelque sorte complétée par la position du module dans la page qui va en déterminer la fonction énonciative.

554 Le fait qu'il existe des expédients techniques pour limiter justement les différences d'affichage selon le logiciel ou le matériel utilisés est une composante essentielle des écrits d'écran mais qui prend acte en quelque sorte de cette part polyphonique de tout écrit d'écran, en essayant de lutter contre ce qui est vu comme un désagrément.

555 D'où l'on voit que le couple *loci / imagines* est ici réarrangé : c'est la conjonction entre une image – un bouton « J'aime » – et un lieu qui produit un effet spécifique. Le même bouton en un lieu différent n'aura pas le même effet. Le *locus* n'est pas neutre dans l'agentivité de l'*imago*.

c Cadrage et API : la contextualisation des « petites formes » industrielles

Une fois compris cela, on peut reprendre l'analyse des { petites formes } et comprendre le jeu qui s'institue entre leur apparence formelle standardisée et l'espace dans lequel elles se situent.

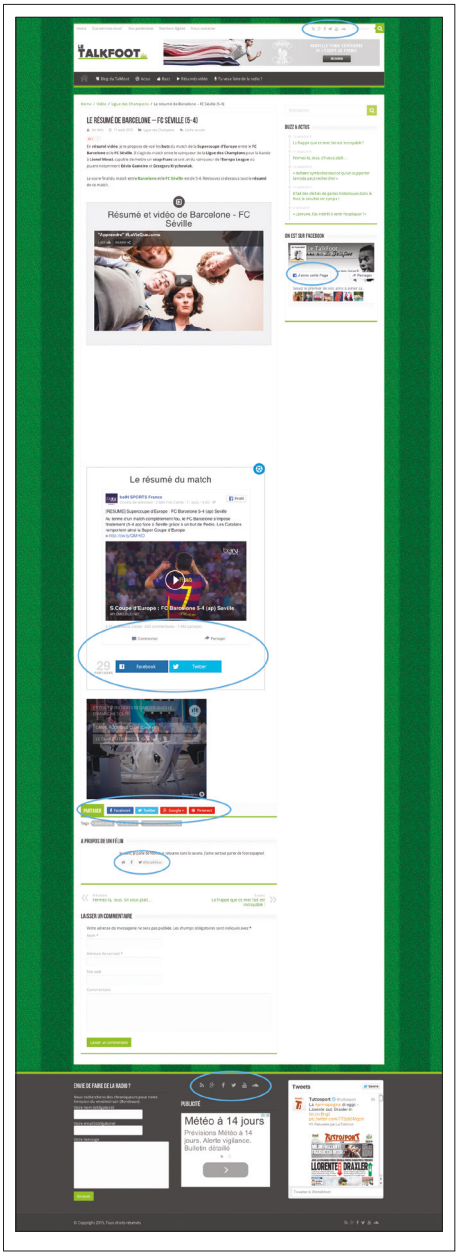


Fig. 15. Zone de répartition des petites formes dans une page web. Capture d'écran du site talkfoot.com, réalisée le 13 juillet 2015.

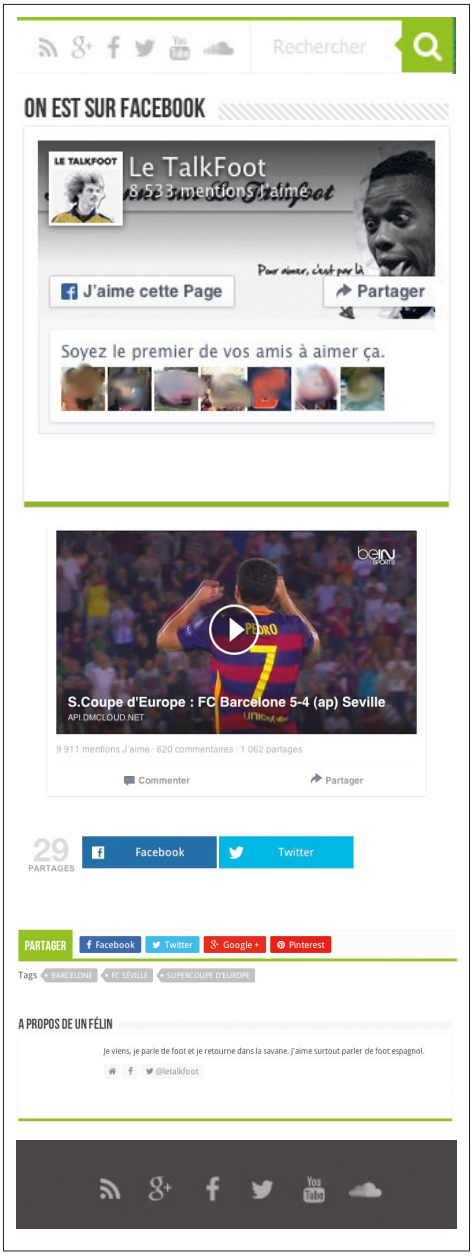


Fig. 16. Détails des différentes petites formes au sein d'une même page. Capture d'écran du site talkfoot.com, réalisée le 13 juillet 2015 (détails).

Sur les captures d'écran ci-dessus, on repère au moins six zones où sont insérés des boutons de partage ou d'assentiment (figure 15, cercles bleu). Ces zones sont disséminées un peu partout sur la page et, la plupart du temps, les boutons se ressemblent. Qu'est-ce qui fait donc que le même bouton « Partager » ou « J'aime » ne déclenche pas la même action ? Et qu'est-ce qui nous permet de dire en un coup d'œil que si nous cliquons sur le bouton « F » grisé en haut à droite de la page, nous « aimerons » la page Facebook du site www.talkfoot.com, alors que si nous cliquons sur le bouton « F » bleuté sous l'article, nous diffuserons la vidéo sur notre profil Facebook ? La charte graphique joue bien sûr un rôle, ainsi que l'aspect proprement linguistique du message : un même « F » assorti de « J'aime cette page » ou de « Partager » spécifie bien l'action que va déclencher le { signe passeur }. Mais ces deux aspects ne sont pas suffisants. Nous en voulons pour exemple la flèche assortie du mot « Partager », qui est identique en dessous du lecteur vidéo ou dans le cadre de droite « On est sur Facebook ». Pour autant, la première fait republier la vidéo, la seconde fait republier un lien vers la page Facebook de talkfoot.com.

API,
code,
code source,
discret,
discrétisation,
HTML,
petites formes,
signe passeur,
widget

Voir glossaire tome II, p. 3-25

Ce qui différencie l'effet techno-sémiotique de ces deux flèches, c'est leur cadrage. Ce cadrage est aussi une place donnée dans la page, qui nous permet de comprendre à quoi renvoie le signe. La mise en page et le cadrage sont donc des actions de **contextualisation** d'une forme usinée pour être la plus « neutre » possible. Un { *widget* } est un module textuel abstrait, fait pour être disséminé de pages en pages, indépendamment du contexte de publication. C'est donc une forme qui affiche une certaine « neutralité », au sens où elle doit pouvoir se prêter à de multiples réappropriations et republications. C'est pourquoi, finalement, rien ne ressemble plus à un bouton « J'aime » qu'un autre bouton « J'aime » ou « Tweeter » : ce sont des formes qui sont faites pour se répéter à l'identique tout en étant utilisables dans différents contextes. La mise en page et le cadrage sont des manières de recontextualiser ces formes, de leur donner un sens particulier et différencié.

Pour comprendre cette « neutralité » et cette recontextualisation, on peut les comparer à la différence entre verbe transitif (qui admet un complément d'objet) et verbe intransitif. Cliquer sur un bouton nous fait « aimer » et « partager ». Mais qu'est-ce que nous « aimons » et « partageons » ? Un bouton, sous sa forme abstraite d'un bout de { code } { HTML } à copier / coller dans le { code source } d'une page web, est l'équivalent d'un verbe rendu intransitif. Ce bouton permet d'« aimer » et de « partager », sans que soit encore décidé l'objet de cet amour et de ce partage[556]. Le cadrage et la mise en page servent à redonner un objet à ces verbes, à les rendre à nouveau transitifs : on « aime » telle page, on « partage » tel article. Ce pouvoir de recontextualisation par l'espace est dû à la force sémiotique du cadre, mais aussi à la page, conçue comme un lieu de mémoire où chaque *locus* donne aux *imagines* placées en lui une certaine fonction énonciative.

Nous sommes ainsi arrivé à déterminer certaines modalités sémiotiques effectives d'une culture anthologique du texte : des modules standards et remobilisables – les { *widgets* } – s'ancrent dans une page et acquièrent par cet ancrage une fonction d'énonciation. Ils acquièrent également une signification différenciée selon leur emplacement dans la page, malgré le fait que ces formes soient visuellement identiques.

Reste la question de la { discrétisation }, que nous avions suggérée dans l'hypothèse de départ de ce chapitre. Si les { API } participent, avec les { *widgets* }, à l'établissement d'une culture anthologique du texte par la mise à disposition de modules combinables, ces modules sont, de fait, { discrets }. Ce sont autant d'unités insécables du texte lisible à l'écran. Mais une rapide inspection de notre corpus montre que cette { discrétisation } va plus loin : chaque { petite forme } peut être envisagée comme une combinaison de formes plus élémentaires. Ce qui pousse à interroger la notion même de { petites formes } et le niveau de granularité choisi dans l'analyse.

556 En quoi l'on voit que ces formes textuelles sont celles privilégiées par les « industries médiatisantes » qui sont, selon Yves Jeanneret, des industries qui se « [...] désintéresse[nt] des contenus et enjeux de la communication [...] » au profit de la seule circulation. Jeanneret, Yves. *Op. cit.*, 2014, p. 10.

1 D Modularité & poupées russes : des petites formes dans des petites formes… jusqu'à quel point ?

API,
code,
code source,
discrétisation,
documentation,
forme-texte,
HTML,
petites formes,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

Nous avons jusqu'ici pris pour argent comptant les « { petites formes } » que sont publications ancrées, { tweets }, boutons, etc. Ces { petites formes } standardisées composent la grammaire d'une page modulaire et acquièrent une efficacité technique et sémiotique par la place qu'elles occupent dans cette page. Mais il apparaît assez vite que ces formes sont plus complexes qu'il n'y paraît : certaines sont composées de plusieurs { petites formes }, d'autres sont des éléments isolés de { petites formes } qu'on pensait par ailleurs formellement stables. La métaphore de la « grammaire » peut donc laisser penser que la liste des { *widgets* } serait la liste des unités de base d'un langage visuel, alors qu'il n'en est rien. Au-delà des { petites formes }, il y a des macro-formes, des micro-formes, en somme des niveaux multiples de compositions et d'agencements sémiotiques.

Dans le cadre de notre hypothèse générale d'une conception modulaire du texte, nous montrons dans cette sous-partie que la { discrétisation } des formes joue un rôle important pour établir cette vision modulaire. En étudiant la { documentation }, on comprend que certaines formes sont des macro-formes (III. 1. D. a), mais aussi plus radicalement qu'on peut isoler au sein des { petites formes } des formes encore plus élémentaires, des micro-formes, que l'on va alors pouvoir assembler en montant en niveau de complexité (III. 1. D. b). Une page web est donc proposée comme un assemblage de modules plus ou moins élémentaires, selon l'échelle ou le degré de granularité choisi. Ce qui ouvre à la question du lien entre { discrétisation } et manipulabilité des { formes-textes } (III. 1. D. c), question qui sera explorée plus en profondeur dans notre partie suivante.

a Les macro-formes

Quand il en va des { *widgets* } et de ce que l'{ API } permet de produire, certaines formes sont effectivement simples : un { tweet } ou une publication Facebook. En revanche, certaines formes sont complexes au sens où elles sont composées de plusieurs { tweets } ou de plusieurs statuts. Elles sont ce qu'on pourrait appeler des macro-formes. C'est le cas des { *timelines* } ancrées.

Fig. 17. Le tweet ancré, exemple de macro-forme. Capture d'écran du site www.lemonde.fr-les-decodeurs-article-2015-07-16-ma-france-a-moi-le-mot-cle-qui-a-enflamme-twitter-le-14-juillet_4685117_4355770.html (détail), réalisée le 17 juillet 2015.

Fig. 18. La timeline, exemple de macro-forme. Capture d'écran du site timeshighereducation.co.uk (détail), réalisée le 17 juillet 2015 à 19h11.

À gauche, un { tweet } issu du site des « décodeurs » sur lemonde.fr. À droite, une { *timeline* } intégrée dans le site du *Times Higher Education*. La { *timeline* } (TL) affiche les derniers { tweets } publiés sur le compte Twitter du journal. Ces deux formes sont considérées par Twitter comme des { *widgets* } et sont créées en copiant / collant quelques lignes de { code } { HTML } dans le { code source } d'une page web. Mais l'une (la TL) est composée d'un assemblage de l'autre (le { tweet }) et les effets sémiotiques en sont bien différents. Dans le cas du { tweet } ancré, il se repère d'abord par le mince filet gris qui l'entoure et qui regroupe le texte du

{ tweet }, les trois images qui le composent, la date de publication, la photo de son auteur, accompagnée d'un nom d'utilisateur (« chatchat ») et d'un identifiant Twitter (« @Akacha630 »). Se trouvent enfin à l'intérieur du cadre un bouton « Suivre », permettant de s'abonner aux autres { tweets } de cet utilisateur, ainsi que trois boutons permettant respectivement de répondre au { tweet } (flèche vers la gauche), republier le { tweet } (deux flèches formant une boucle) et de mettre le { tweet } en favoris (petite étoile). Voilà les principaux éléments qui composent cette « { petite forme } » particulière. Le filet gris permet d'isoler le { tweet } du texte de l'article, tout en liant les deux. C'est une frontière qui illustre la fonction relative du cadre : mettre en relation les parties d'un texte tout en établissant une claire différence entre ces parties. Ici, pas de doute : un { tweet } est intégré dans un article et l'on ne peut confondre le { tweet } et le corps de l'article. On notera également que le { tweet } occupe toute la largeur de la colonne de texte, confirmant par là qu'il est une partie de l'article du monde.fr.

API,
application,
code,
code source,
documentation,
petites formes,
programmation,
script,
signe passeur,
***timeline*,**
tweet

Voir glossaire tome II, p. 3-25

La { *timeline* } est un autre cas d'« ancrage » dans un site, mais aux attendus un peu différents. Elle est composée d'une multitude de { tweets }, organisés en ordre antéchronologique, tout comme la page de profil du *Times Higher Education* sur Twitter. Chaque { tweet } contient les mêmes informations que le { tweet } ancré : nom d'utilisateur, photo, pseudonyme, contenu du { tweet }, etc. Le { signe passeur } *expand* (« étendre », « déplier ») permet d'afficher le reste de ces informations : boutons de republication, image éventuelle… Là encore, ce sont des cadres qui organisent cet enchâssement énonciatif. La { *timeline* } est encadrée d'un filet gris, avec une barre de défilement sur la droite. À l'intérieur de ce cadre, quatre { tweets } sont mis en relation comme étant les fruits d'un même auteur : le compte Twitter du *Times Higher Education*[557]. Chaque { tweet } est séparé par un trait gris assurant la fonction relative du cadre : séparer tout en mettant en relation. La modularité ne s'exerce donc pas seulement au niveau de la page, mais à l'intérieur même des { petites formes }. Si l'on s'en tient au cadre extérieur, une { *timeline* } peut être considérée comme une seule et même { petite forme }, une unité du point de vue sémiotique. La même forme, examinée du point de vue des cadres intérieurs, fait éclater cette unité au profit de l'unité du { tweet }, qui devient la { petite forme } élémentaire. Mais nous pouvons encore remonter d'un niveau…

557 Le fait que, dans ce cas précis, trois de ces tweets soient des retweets, des republications, complexifie la polyphonie à l'œuvre car Twitter présente les retweets comme la rediffusion telle quelle du tweet original, avec la signature de son auteur initial. Il y a donc quatre voix différentes ici, toutes prise en charge par le *Times Higher Education*.

b Les micro-formes

Twitter, dans la { documentation } de son { API }, présente la notion de « *web intents* ». Un *intent* – qu'on pourrait traduire en français par « intention » – est en { programmation } une information passée entre deux entités (deux { applications }, deux pages web…) où l'une demande à l'autre de faire une certaine action. Dans le cas de Twitter, une petite étoile est un *web intent :* un bouton mis sur une page web qui permet de demander à Twitter de mettre en favoris un { tweet }. On pourrait donc appeler cela des « boutons d'intentionnalité »[558]. Cette formule n'étant pas très heureuse, gardons le terme *intents*.

Les *intents* sont un moyen d'effectuer certaines actions sur Twitter : publier, mettre en favoris, répondre, retweeter, s'abonner aux publications d'un utilisateur et ce sans être sur le site Twitter. Ils sont présentés par Twitter comme « […] rendant possible aux utilisateurs d'interagir avec du contenu de Twitter dans le contexte de votre site, sans devoir quitter la page ou donner d'autorisation à une { application } simplement pour cette interaction[559] ». En effet, pour faire fonctionner ces *intents*, il suffit de copier / coller deux lignes de { code } dans le { code source } d'une page : un { script } (widgets.js) et un lien précisant le type d'action à effectuer et sur quel { tweet }. Par exemple, la ligne de { code } `<a href ="https://twitter.com/intent/retweet?tweet_id=954890776453198733">Retweet<a/>` signifie qu'il y a dans la page web le mot « Retweet », que ce mot est un lien (`<a></a>`) renvoyant vers le site twitter.com (contenu entre guillements anglais après le signe égal). Ce lien permet de demander la republication (`/intent/retweet`) d'un { tweet } précis (`?tweet_id=954890776453198733`). Potentiellement, un *intent* permet donc de faire de n'importe quel mot ou image, de n'importe quelle partie d'une page web, un moyen de publier sur Twitter. C'est la structure technique de production de micro-formes de la culture anthologique.

Bien sûr, cela n'a pas grand sens de faire d'un mot ou d'une image au milieu d'une page le moyen de répondre ou de republier un { tweet }, même si c'est techniquement possible. Ces *intents* sont contextualisés, donnés à lire de telle manière que l'action à laquelle renvoie le { signe passeur }

558 D'aucuns critiqueraient ici un anthropomorphisme latent du terme : la machine ne « veut » pas, elle n'a pas d'intention au sens humain du terme. Il est clair qu'il n'est pas question ici de volonté mais de l'échange d'informations entre deux systèmes techniques, échange sémiotisé de telle manière qu'il laisse penser que la machine veut, ou que la technique est la simple réalisation d'une volonté humaine, niant la médiation techno-sémiotique. Ce point sera dûment traité dans cette thèse, de façon plus globale.

559 « […] *make it possible for users to interact with Twitter content in the context of your page without leaving the page or having to authorize an new app just for the interaction* ». Twitter, documentation de l'API : dev.twitter.com/web/intents. Capture réalisée le 14 juillet 2015.

soit immédiatement intelligible. C'est pourquoi Twitter recommande d'utiliser ses icônes « officielles » : petite étoile pour la mise en favoris, flèches pour le retweet, etc[560]. Ce sont ces mêmes icônes qui apparaissent dans tout { tweet } ou vidéo ancrée. Autrement dit, la « { petite forme } » qu'est le { tweet } est en fait techniquement et sémiotiquement scindable en ces « microformes » que sont les *intents :* boutons « Suivre », « Mettre en favoris », « Répondre », « Retweeter ». En revanche, ces boutons ne sont pas cadrés comme étant des éléments distincts du { tweet }. Ils sont donnés à lire comme des parties indissociables de la { petite forme } :

API, développeur, documentation, forme-texte, petites formes, *timeline*, tweet,

Voir glossaire tome II, p. 3-25

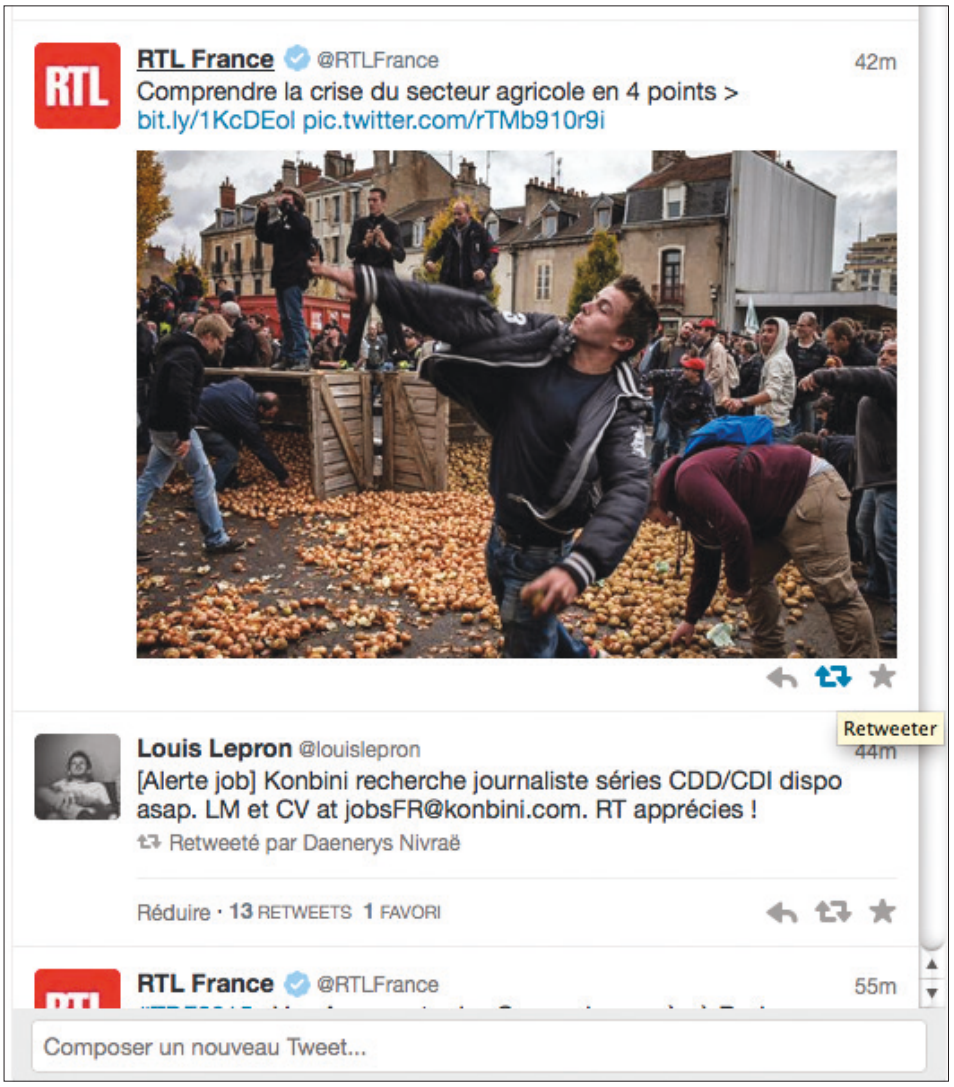


Fig. 19. L'enchâssement des micro-formes dans une macro-forme. Capture d'écran de la page www.canalplus.fr/c-cinema/c-ceremonie-des-cesar-sur-canal/pid5499-livetweet-ceremonie-cesar-2015.html (détail). Capture réalisée le 17 juillet 2016 à 18h38.

560 Voir annexe nº 12.

La figure 19 ci-dessus montre l'enchâssement des trois { petites formes } vues jusqu'ici : une { *timeline* } (macroforme) ancrée dans le site de *Canal+* contient un { tweet } ({ petite forme }), publié par RTL, qui contient lui-même trois icônes permettant d'agir sur twitter.com, trois *intents* donc (microformes). Les cadres permettent de déterminer sans ambiguité les deux premiers niveaux d'enchâssement, en revanche les trois *intents* sont simplement glissés sous la photo, sans cadre apparent. Ils font partie du { tweet }, ne sont pas des formes indépendantes de ce dernier. Alors que techniquement, elles le sont et sémiotiquement, elles peuvent l'être. Il s'agit donc ici d'une décision éditoriale automatisée par Twitter, une décision de mise en forme du texte. Cette décision porte sur le degré de granularité de la modularité de la page : sémiotiquement, l'élément « basique », le plus simple, n'est pas l'*intent* mais le { tweet }. Il y a donc, chez Twitter, par le biais d'une forme comme le { tweet } ancré, une certaine axiologie des { formes-textes } et de leur degré de complexité. Un { tweet } est le module de base de l'écriture des pages web, même s'il est techniquement divisible en plus petites unités.

c Une discrétisation qui ouvre à la question de la manipulabilité

Pourquoi alors détailler dans la { documentation } de l'{ API } de Twitter le fonctionnement des *intents ?* Pourquoi donner la possibilité de manipuler de telles microformes ? Pourquoi aller aussi loin dans la modularité, jusqu'à un niveau tellement élémentaire ? Parce que cela permet d'augmenter les possibilités de manipulation et de personnalisation des techniques de publication proposées par les plateformes. Lorsque Twitter explique ce qu'est un *intent*, il donne les moyens à des { développeurs } externes de ne pas se contenter d'une forme préfabriquée mais de composer à leur tour des formes permettant la publication sur Twitter. L'enjeu n'est pas tant de disséminer des formes que de disséminer des moyens de publication sur la plateforme. Pour réaliser cela, les { petites formes } et microformes sont des outils essentiels, par leur caractère modulaire et combinatoire. Et cette modularité est rendue d'autant plus grande que ce sont des formes **réductibles**. Un { tweet } peut être inséré dans une { *timeline* } sans perdre de sa lisibilité, il peut être intégré à même un article ; un bouton « répondre » peut être mis à la fin d'un article, au sein d'une { petite forme } elle-même incluse dans une macroforme, etc.

API, développeur, *widget*

Voir glossaire tome II, p. 3-25

C'est cette « puissance abbréviative[561] » des { *widgets* } et de leurs composants qui permet leur circulation et leur progressive routinisation dans l'économie textuelle des pages web contemporaines. En pouvant ainsi muter, se réduire ou s'agrandir, changer d'échelle, ces écritures augmentent leur capacité à s'emboîter, à se combiner, à être mobilisées dans des contextes multiples et hétérogènes. En retour, leur dissémination produit sur les pages web contemporaines des textes modulaires, composés de blocs réagençables.

Mais pas uniquement. Comme nous l'avons vu avec les *intents*, il y a dans le niveau de granularité des formes sémiotiques un autre enjeu, corollaire à leur dissémination, mais bien différent. Cet enjeu, c'est celui de leur réemploi par des { développeurs }, en d'autres termes, de leur **manipulabilité**. Il est donc temps d'analyser les rapports entre { *widgets* } et manipulabilité et plus largement entre le texte, son abstraction et ses outils de manipulation.

561 Souchier, Emmanuël. Histoires de pages et pages d'histoire. Dans : ZALI, Anne, *op. cit.*, p. 19-55.

L'hypothèse que les { API } web propagent une conception modulaire des textes de réseau nous a conduit à établir plusieurs points importants. Premièrement, que cette tendance est liée historiquement à la naissance du texte livresque et à une métaphore structurante de l'ancrage que l'on retrouve dans les { API }. Mais contre une pensée abstraite du texte, nous montrons que dans notre corpus, l'espace où s'ancrent les modules de texte dans la page est structurant à deux titres. D'abord parce que la page est un lieu de mémoire où se sédimente une certaine organisation visuelle et énonciative. Ensuite parce que le découpage de cet espace par le cadre permet de différencier sémiotiquement des formes textuelles usinées pour être identiques malgré leur usage répété. En revanche, une analyse plus précise de ces formes a montré qu'elles ne sont pas des modules élémentaires, mais qu'elles peuvent être divisées en plus petites unités.

API, calcul, discret, discrétisation, documentation, formalisme, *widget*

Voir glossaire tome II, p. 3-25

Ce qui nous conduit à l'idée que la { discrétisation } des formes est une tendance au moins aussi importante que la modularité de la page et que cette { discrétisation } sert une plus grande manipulabilité de ces formes. Ce lien entre manipulabilité, { discrétisation } et abstraction du texte est là encore présent dans le texte livresque, mais renforcé par les propriétés du { calcul } informatique et son { formalisme } qui reposent la question de l'éditorialisation (III. 2. A). Ce lien mis en évidence, nous en étudierons les attendus dans le cas des { *widgets* }, formes faites pour être transformées et qui portent en elles les outils de cette réappropriation (III. 2. B).

2 MANIPULATION ET ÉDITORIALISATION

L'une des visions du texte promues par les { API } étudiées est un texte modulaire, composé de blocs réagençables et recombinables entre eux. Cette modularité, permise par la technique informatique, s'appuie sur — tout autant que renforce — un processus d'abstraction du texte entamé au XII^e^ siècle. Prouver ce premier point nous a conduit à suggérer un lien entre la modularité du texte et ses possibilités de reprises, d'adaptions, de transformations… en somme ses différentes manipulations. Nous faisons désormais l'hypothèse que les { API } web, à travers leur { documentation } et par les formes dont elles permettent l'existence, participent à augmenter la manipulabilité des textes de réseau.

Pour éclairer cette hypothèse, nous reviendrons sur les travaux d'Illich, car l'abstraction progressive va pour lui de pair avec la mise au point de techniques de manipulation du texte : index, gloses, marques de citations… Autant d'outils d'organisation, de découpage, d'articulation du texte qui en augmentent les réutilisations possibles. Mais là encore, il faut interroger la part spécifique du { calcul } dans cette nouvelle économie du texte. Les travaux de Bruno Bachimont et un retour au { formalisme } hilbertien – cette fois en mettant l'accent sur sa théorie du signe – permettront de tisser le lien entre calculabilité et manipulabilité du texte. Si en effet les { API } prolongent la vision d'un texte manipulable, il faut prendre en compte la modification du support d'écriture, qui se caractérise par le { calcul }. Cette calculabilité du texte est aussi une { discrétisation } qui renforce sa manipulabilité, mais au prix d'une « ascèse du signe ». Ce qui pose l'enjeu de la rééditorialisation : la recomposition de textes intelligibles à partir d'une masse d'unités { discrètes } (III. 2. A.). L'étude des { *widgets* } montrera ensuite cette dynamique entre { discrétisation }, manipulation et rééditorialisation. Ce sont des formes qui ne font pas que disséminer du texte et s'ancrer dans des pages, mais qui portent en elles les possibilités de leur manipulation (III. 2. B.), constituant une sorte d'« appel à l'écriture » pour les lecteurs et entretenant la production de nouveaux textes.

2 A D'Illich à Bachimont : manipulations, théorie du support et « ascèse du signe »

Pour commencer à démontrer notre hypothèse, nous suivrons un mouvement similaire à notre partie précédente. Nous avons en effet tout intérêt à historiciser la notion de texte afin de comprendre comment s'est formé un lien entre abstraction et manipulabilité du texte. C'est Ivan Illich qui a formulé l'idée qu'avec la lecture scolastique, le texte ancré dans la page est l'image visible d'une pensée en exercice. Découper dans ce texte des unités qu'on peut réarranger, c'est ainsi entrer dans le jeu rhétorique de la pensée et de son ordonnancement (III. 2. A. a). De ce point de vue, l'abstraction du texte accompagne la manipulabilité de ses éléments. Un second argument est que le { calcul } au sens informatique du terme, parce qu'il repose sur une conception ascétique et { discrète } du signe, augmente également la manipulabilité des textes numériques et donc par extension des textes de réseau (III. 2. A. b). { Discrétisation } et fragmentation qui conduisent à la question de l'éditorialisation (III. 2. A. c), c'est-à-dire à la recontextualisation des éléments préalablement discrétisés pour permettre le { calcul }.

calcul, discret, discrétisation, écriture-calcul

Voir glossaire tome II, p. 3-25

a Le texte livresque comme consignation et *ordinatio* de la pensée

La séparation progressive entre le livre et le texte au XII^e^ siècle, décrite par Illich, est concomitante à la mise au point d'un certain nombre de techniques et à la modification de la valeur sociale et symbolique de la lecture et de l'écriture.

Parmi ces transformations, celle qui nous intéresse désormais est le passage de l'écriture comme technique d'enregistrement de la **parole** à une technique d'enregistrement de la **pensée**[562]. Ce passage est notamment le fait de transformations des techniques de présentation et de manipulation du texte, parmi lesquelles l'index alphabétique[563], la multiplication des gloses enchâssées dans le texte principal, la généralisation des marques de citation[564] et des titres de chapitres[565]. Le texte n'est alors plus la consi-

562 Illich, Ivan. *Op. cit*. Chapitre 6 : « *From Recorded Speech to the Record of Thought* », p. 93-114.

563 Illich, Ivan. *Idem*, p. 102-104.

564 Illich, Ivan. *Idem*, p. 98.

565 Illich, Ivan. *Idem*, p. 100.

gnation d'un discours qu'il s'agirait « d'écouter » à travers la page mais la mise en espace d'une pensée, d'articulations logiques qui peuvent être découpées, manipulées, comparées, etc[566].

Une nouvelle importance est donnée à la vision, au détriment de l'écoute : le texte se voit avant de s'entendre. Ce primat du regard, couplé à une abstraction du texte, contribue à faire de l'organisation de la page une action rhétorique, une *ordinatio*. Le découpage des paragraphes, la taille de l'écriture, les résumés et les chapitrages permettent de reconnaître en un coup d'oeil les différents arguments du texte : « Le lecteur situe immédiatement l'endroit où le contradicteur ou *adversarius* a eu son mot à dire. Grâce au marqueur visuel, la tâche de percevoir l'*ordinatio* de l'auteur passe de l'oreille interne à l'œil, du rythme du son à un nouvel espace artificiel[567]. » La page devient organisation visuelle d'une pensée, « [...] la représentation visuelle d'un argument bien pensé[568] ». Une telle conception est indissociable des techniques de visualisation et de manipulation du texte devenue entité agençable à la surface de la page. Que l'informatique, parce qu'elle est construite sur l'{ écriture-calcul }, prolonge et renforce cette représentation du texte comme *ordinatio*, voilà ce que nous allons désormais démontrer[569].

b Calcul et manipulabilité : une double abstraction du signe

Ivan Illich s'intéresse au texte livresque, dont il voit commencer la fin avec les technologies numériques. Notre objet est le texte numérique de réseau. Ce texte se caractérise entre autres par son aspect calculatoire. En quoi cette dimension renforce-t-elle l'abstraction et la manipulabilité de ce texte ?

566 C'est tout l'enjeu du passage, chez Illich, de la lecture monastique à la lecture scolastique.

567 « *The reader immediately recognizes where the tempter or adversarius has been given his say. The visual marker shifts the task of perceiving the author's ordinatio from the inner ear to the eye, and from the rythm of sound to a new factitious space.* » Illich, Ivan. *Idem*, p. 99.

568 « [...] *the visual representation of thought-through argument* ». *Ibidem*.

569 Il faut ici rappeler que nous parlons d'une « métaphore centrale » (*root metaphor*) de l'écriture et de la page. Il y a bien évidemment plusieurs façons d'écrire, de lire ou de composer une page, ce dont nous préviens Illich citant Bonaventure de Bagnorea (*circa* 1217-1274) page 106 d'*In The Vineyard of the Text*. C'est aussi ce que nous apprend l'histoire de la lecture et des pratiques lettrées. Nous ne nions pas la diversité des pratiques. Nous nous situons sur le plan d'une représentation dominante, orientée tout autant qu'orientant les pratiques. Le XII^e^ siècle est un moment de stabilisation d'une telle représentation dominante. Pour l'histoire des pratiques lettrées, voir Chartier, Roger. *L'ordre des livres : lecteurs, auteurs, bibliothèques en Europe entre XIV^e^ et XVIII^e^ siècle*. Aix-en-Provence : Alinea, 1992 ; Jahjah, Marc. *Les marginalia de lecture dans les « réseaux sociaux » du livre (2008-2014) : mutations, formes, imaginaires*. Thèse de doctorat. Paris : École des Hautes Études en Sciences Sociales, 2014.

calcul, discrétisation, écriture-calcul, formalisme, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

Dans ses travaux sur la « raison computationnelle », Bruno Bachimont propose l'idée selon laquelle les techniques numériques, en tant qu'outil d'écriture, sont (entre autres) des techniques de { discrétisation[570] } et que cette { discrétisation } permet une plus grande manipulabilité des écrits. Pour cela, il repart des travaux en anthropologie et en philosophie sur la question de la technique[571] et plus particulièrement sur l'écriture[572]. La technique est anthropologiquement constituée et constituante[573]. Ce qui signifie que l'outil est à la fois crée par l'humain – constitué – mais il est à la fois une extériorisation de certaines facultés humaines et que ces facultés en sont modifiées en conséquence – l'outil est constituant. Il y a donc une forme de co-construction entre humain et technique : la technique ouvre des possibilités cognitives et physiques à l'humain, l'humain peut explorer ces possibilités pour créer de nouveaux outils.

Parmi les techniques, certaines externalisent des fonctions cognitives comme la mémoire ou l'orientation spatiale. Elles sont ce que Goody a nommé des « { technologies de l'intellect[574] } », ce que Bruno Bachimont appelle encore des « techniques de la pensée[575] ». L'écriture fait partie de ces dernières. Si l'écriture est une { technologie de l'intellect }, alors une modification significative dans le système d'écriture (appareillage technique, rôle social…) peut entraîner une modification significative des structures de pensée et des manières de faire sens. Attendu que « toute connaissance repose sur une inscription qui en est la condition de possibilité[576] », alors le support d'inscription, les manipulations qu'il permet ou autorise, seront autant de possibilités de voir émerger des formes inédites de connaissance. C'est ce que Bruno Bachimont appelle la « théorie du support[577] » : le rôle actif et constitutif du support d'écriture dans la nature des opérations intellectuelles effectuées et des connaissances développées.

II | 2 | B | a
l'écriture fait partie de ces dernières.

Voir p. 191 pour un état de l'art sur le concept de « technologie intellectuelle »

570 En mathématique, on appelle ensemble discret un ensemble dont les parties sont séparées les une des autres. Par exemple, les nombres entiers sont un ensemble discret. On « saute » de 1 à 2 sans qu'il n'y ait de valeur intermédiaire. L'alphabet est un ensemble discret de lettres. Le discret s'oppose au continu. Par exemple, pour représenter les couleurs, on peut dire qu'un nuancier Pantone est discret alors qu'un spectre optique est continu.

571 Leroi-Gourhan, André. *Op. cit.*, [2008] 1964 (vol. 1), 1991 [1964], (vol. 2) ; Stiegler, Bernard. *La technique et le temps*. Paris : Galilée, 1994-1996, 2 vol.

572 Goody, Jack. *Op. cit.*, 2007 [1979] ; Goody, Jack. *La logique de l'écriture : aux origines des sociétés humaines*. Paris : Armand Colin, 1986 ; Derrida, Jacques. *Op. cit*.

573 C'est l'idée de base de la « thèse TAC » (Technologie Anthropologiquement Constitutive / Constituante). Voir Steiner, Pierre. Philosophie, technologie et cognition. État des lieux et perspectives. *Op. cit.*, p. 7-40.

574 Goody, Jack. *Op. cit.*, 1979, p. 10.

575 Bachimont, Bruno. *Op. cit.*, 2010, p. 16.

576 Bachimont, Bruno. *Idem*, p. 106.

577 Bachimont, Bruno. *Idem*, p. 121-127.

Ce que le numérique apporte comme changement significatif dans l'économie de l'écriture, c'est précisément le caractère novateur des modalités d'inscription. La principale caractéristique du support numérique, depuis le modèle défini par Turing en 1936, est l'{ écriture-calcul }. Ce { calcul } repose sur deux aspects qui nous intéressent ici en premier lieu : l'évacuation du sens des symboles manipulés et la { discrétisation } des opérations.

I | 3 | C | d
écriture-calcul
Voir p. 153

L'évacuation du sens est un résultat de l'héritage formaliste de l'informatique. Ce { formalisme } est à entendre en un sens hilbertien : un raisonnement mathématique peut être conçu comme la manipulation de certaines traces visibles (A, B, Σ), à la seule condition que ces traces soient considérées exclusivement du point de vue logique, comme des « types » :

> *« Faire des mathématiques reviendrait ainsi à coucher des traces noires sur la feuille blanche, et à manipuler ces traces, c'est-à-dire les réécrire, en fonction de leur type. Ainsi doit-on traiter toutes les traces représentant la lettre "a" comme pouvant être soumise aux même manipulations puisqu'elles possèdent le même type*[578]*. »*

L'abstraction logique ne s'oppose donc pas à la matérialité de la trace, bien au contraire. C'est parce qu'on peut reconnaître une récurrence dans les formes qu'on peut ramener toutes les formes similaires à une seule et même lettre, à un seul et même type. Dans un second temps, chaque symbole est considéré indépendamment des articulations avec les autres mais aussi indépendamment d'une signification arrêtée : un raisonnement logique peut arriver à des conclusions sur A et non-A, sans qu'on ait besoin de savoir à quel signifié précis renvoie le signifiant « A » (le « type » logique). Sous ces conditions, un raisonnement mathématique est arithmétisable, c'est-à-dire formulable comme un { calcul }.

II | 3 | B
un raisonnement mathématique est arithmétisable
Voir p. 228

578 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 197.

Il y a donc dans le numérique, en ce qui concerne les signes d'écriture :

> « […] *une double indépendance vis-à-vis du sens. D'une part, ils sont définis indépendamment les uns des autres, ce sont des primitives ; d'autre part, ils ne possèdent en eux aucune signification particulière. La seule chose qu'on demande véritablement aux signes d'un tel alphabet, c'est d'être distinct les uns des autres sans ambiguïté*[579] […] ».

calcul, code, discret, discrétisation, JavaScript, langage de programmation, langage machine, programme, variable

Voir glossaire tome II, p. 3-25

Bruno Bachimont parle ici d'une tendance de fond des outils numériques et il en va de tous les niveaux d'écriture de ces outils. Un 1 ou un 0, la base de tout { langage machine }, est effectivement la transformation en symbole calculable d'une variation physique mesurable (présence ou absence de courant électrique par exemple) sans pour autant que cette transformation dépende de la nature de la variation physique mesurée. Mais cette remarque vaut pour d'autres niveaux de { langages } informatiques. Quand, en { JavaScript } par exemple, on déclare une { variable[580] } par l'expression `var Exemple = 42`, « Exemple » aurait pu s'appeler « Conclusion » ou « Kangourou », cela n'aurait rien changé au déroulement du { programme }, du point de vue de l'effectivité machinique du { calcul } en tout cas. Les seuls termes inutilisables sont les mots « réservés », en l'occurrence ici « `var` ». Il est en effet clair qu'appeler sa { variable } `Var` est problématique puisque le navigateur interprétant le { programme } pourrait confondre l'action de déclarer une { variable } (le premier `var`) avec le nom de cette { variable } (le second `var`). Les recommandations faites de choisir des noms de { variables } « clairs », ne le sont pas pour des raisons qui tiennent à l'efficacité pour la machine mais à la lisibilité pour l'humain. C'est évidemment un aspect essentiel, mais qui ne concerne pas, ou plus, la même face du signe informatique : du côté technique, peu importe le nom tant qu'il n'est pas confondu avec autre chose. Du côté sémiotique, celui de l'humain et de son interprétation, il est plus facile de proprement rédiger son { code } en évitant les nomenclatures ambiguës. Mais l'ambiguïté est dans ce cas un simple effort cognitif à fournir pour l'être humain. Au contraire, la machine ne peut décider seule d'une situation ambiguë, elle s'arrêterait dans son { calcul }. Dans ce cas, la non-ambiguïté des termes et l'évacuation de la question de la signification est d'une importance fondamentale, puisqu'elle engage le bon fonctionnement de la machine.

II | 3 | B | c
sans pour autant que cette transformation dépende…

Voir p. 231 la notion d'« idéalité computationnelle »

579 Bachimont, Bruno. *Op. cit.*, 2010, p. 155.

580 « Déclarer une variable » revient à faire retenir à l'ordinateur une information associée à un mot. Par exemple `var x=3` veut dire que dans ce programme, la machine saura que x est l'équivalent du chiffre 3. On dit alors qu'on « assigne » la valeur 3 à la variable x.

Cette double abstraction, cette « ascèse du signe » qui permet au { calcul } de « [...] s'abstraire de toute signification pour se rapporter à une pure manipulation mécanique sur des signes vides de sens[581] » est aussi une { discrétisation }, définie comme :

> *« [...] l'opération selon laquelle un contenu est inscrit en un langage constitué d'unités discrètes indépendantes les unes des autres pouvant être manipulées dans le cadre d'opérations strictement syntaxiques, c'est-à-dire de manière algorithmique ou numérique[582] ».*

Les signes informatiques sont des primitives, les *elementa* de la combinatoire à venir. Ils sont donc interprétables en tant que symboles indépendamment des autres signes. C'est cette division des contenus en unités indépendantes qui va renforcer les possibilités de manipulation des écrits numériques.

I | **3** | **C** | **c**
les elementa *de la combinatoire à venir*
Voir p. 152

Pourquoi ce lien entre { discrétisation } et manipulation qui seraient, à en suivre Bruno Bachimont, quasiment synonymes[583] ? Là encore il faut se situer dans le strict domaine du { calcul } et de l'écriture numérique : l'évacuation de la question de la signification des symboles permet d'isoler les unités d'un { calcul } (par exemple une suite de 0 et de 1, ou une combinaison particulière) et de les considérer comme des unités désolidarisées les unes des autres. Cette désolidarisation, qui est aussi une { discrétisation }, permet de poser ces unités sur un plan d'équivalence : elles sont calculables. Entendre : elles peuvent être le produit de multiples recombinaisons, manipulations, permutations. C'est pourquoi, par l'entremise de la calculabilité, { discrétisation } et manipulation sont « réciproques » : ce qui est { discret } est calculable formellement et ce qui est calculable formellement est recombinable à l'envie. Que ces manipulations soient mécanisables, donc effectuées par une machine comme un ordinateur, est une conséquence de la formalisation du { calcul } telle qu'on l'entend ici[584].

581 Bachimont, Bruno. *Idem*, p. 156.

582 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 198.

583 « Toute discrétisation est par définition une numérisation. Il faut donc comprendre que ce qui est discret est manipulable et que, réciproquement, ce qui est manipulable est discret. » *Ibidem*.

584 La manipulation étant décorrélée d'une interprétation, c'est-à-dire d'une production contextuelle du sens des signes qui sont intrinsèquement plurivoques, le calcul peut être fait par une machine. Bachimont, Bruno. *Idem*, p. 197-198.

c Ascèse, désorientation, rééditorialisation

API, discret, formalisme

Voir glossaire tome II, p. 3-25

Cette expansion d'un ascétisme sémiotique entraîne dans un premier temps une désorientation, une perte de sens[585]. La manipulation et l'assemblage, par des machines, de symboles { formels } ou (à des niveaux plus proches de l'humain) de documents peut produire l'effet d'une masse textuelle dont l'humain peut difficilement faire sens. Une fois les contenus discrétisés, calculés et rendus ainsi automatiquement manipulables, se pose donc la question de la réorganisation. Comment agencer de manière intelligible ces contenus ? Bruno Bachimont, dans son texte de 1999, prend l'exemple du dossier patient dans les hôpitaux, qu'il confronte à une structure hypertextuelle. La lecture de ce dossier, ses annotations et réécritures, se fait en fonction du patient et du diagnostic à établir. Elle est, comme toute lecture, éminemment contextuelle et consiste en une réappropriation et un réarrangement des matériaux documentaires disponibles[586]. La structure hypertextuelle, si elle peut être déroutante dans un premier temps, permet aussi des configurations inédites qui peuvent amener de nouvelles interprétations, des diagnostics plus précis. C'est pourquoi il est nécessaire, pour réinjecter du sens et des possibilités d'interprétation, de fournir des outils de manipulation et de recontextualisation. En équipant les possibilités de manipulation – et donc de production de sens – ouvertes par la technique, on peut éviter d'en subir l'effet désorientant. C'est tout l'enjeu de « l'éditorialisation » :

> *« Si la fragmentation permet l'explosion du contenu en unités arbitraires, la recombinaison a tendance à recontextualiser les contenus de manière arbitraire. Progressivement, les outils de gestion audiovisuelle ne permettront pas seulement de retrouver les contenus et de les rejouer dans leur intégralité, mais ils proposeront aussi de sélectionner des parties pour en faire des ressources pour d'autres productions. Autrement dit, on passe de l'indexation, qui a pour but de retrouver un contenu, à une éditorialisation, qui a pour but de produire de nouveaux contenus à partir d'éléments pris arbitrairement*[587] […] ».

585 Le terme de désorientation est emprunté à Bernard Stiegler. Bruno Bachimont parle plutôt de « désémantisation » ou de « perte de sens ». Voir Stiegler, Bernard. *Op. cit.*, 1996 ; Bachimont, Bruno. *Op. cit.*, 2010, p. 160-161.

586 Manipuler, en ce sens, est « […] la condition de possibilité pour l'interprétation et la compréhension » comme le soutient Bruno Bachimont près de dix ans après ce travail sur le dossier patient. Bachimont, Bruno. *Idem*, p. 148.

587 Bachimont, Bruno. *Idem*, p. 161.

Bruno Bachimont prend l'exemple de l'audiovisuel comme illustration d'une tendance plus fondamentale de fragmentation / recombinaison / éditorialisation. La particularité du numérique comme écriture calculatoire est de fragmenter les contenus en unités { formelles } { discrètes }. Cette fragmentation maximise les possibilités de recombinaisons, mais ces possibilités sont tellement grandes qu'il est très difficile pour un humain d'en tirer du sens. D'où la nécessité de dispositifs d'éditorialisation, qui permettent d'organiser les combinaisons possibles. Il ne faut plus seulement permettre de retrouver des contenus (problème d'indexation) mais il s'agit de donner les moyens de créer de nouvelles combinaisons, en fonction des contextes de lecture et d'interprétation. C'est ce passage de l'indexation à l'éditorialisation qui permet de ne pas en rester à l'effet désorientant du numérique mais de créer les conditions de possibilité de nouvelles manipulations et donc de nouvelles productions de sens.

Ce tryptique fragmentation / recombinaison / éditorialisation est donc particulièrement important dans ce que l'informatique fait au texte. Quels sont les effets de cette tendance structurante du numérique sur les formes-textes que les { API } permettent de créer ?

2 B Boutons de Panurge et « appel à l'écriture » : des *widgets* comme formes manipulables

API,
discret,
documentation,
tweet,
Web,
widget

Voir glossaire tome II, p. 3-25

Nous soutenons que le triptyque fragmentation / recombinaison / éditorialisation se joue à deux niveaux dans les { API } web. Au niveau tout d'abord des { *widgets* } à proprement parler, qui sont des formes de l'assentiment encourageant les manipulations : partage, favoris, ancrage, etc. (III. 2. B. a) Au niveau ensuite de la manière dont ces { *widgets* } sont construits informatiquement : ce sont des complexes sémiotiques divisibles en plus petites unités dont la { documentation } de l'{ API } fournit la liste, cette divisibilité permettant des manipulations plus fines. Par cette manipulabilité, nous montrons que ces formes constituent un « appel à l'écriture » : elles se prêtent particulièrement à la reprise, à l'adaptation, à la republication et embarquent avec elles les outils de ces manipulations (III. 2. B. b). Nous aurons ainsi montré que les { API } web participent à augmenter la manipulabilité des textes de réseau : parce qu'elles permettent de produire des formes { discrètes } et recombinables, mais aussi parce que ces formes intègrent les possibilités de leurs manipulations.

a Les formes de l'assentiment

Il faut commencer par rappeler une évidence : les { API } que nous étudions (Facebook et Twitter) sont celles de plateformes de réseaux socionumériques. Ces plateformes reposent en grande partie sur des « profils » composés de fragments textuels (des « statuts », { tweets }, ainsi que des photos ou articles partagés). Les { API } de ces plateformes sont donc intimement liées à la manipulation et à la circulation de ces fragments. Pour s'en convaincre, il suffit de regarder rapidement l'ensemble des { *widgets* } proposés par Facebook et Twitter et de les classer selon leur fonctions communicationnelles. Ces { *widgets* } peuvent être divisés en quatre catégories principales. La première regroupe les { *widgets* } qui permettent **l'assentiment** à un contenu ou un individu. Ce sont par exemple les boutons *Like* (« Aimer », Facebook), *Follow* (« Suivre », Facebook et Twitter), *Favorite* (« Mettre en favori », Twitter). Ils permettent par un clic de manifester ses sentiments positifs vis-à-vis d'une page ou d'un individu. On peut aimer ce que publie cet individu, ou choisir de suivre ses

futures publications – avec le bouton *Follow*[588]. La deuxième regroupe les { *widgets* } qui permettent de faire circuler des publications : bouton *Send* qui permet d'envoyer un lien comme message privé à un ami (Facebook), bouton *Share* permettant de publier un lien sur son profil (Facebook et Twitter), *Feed dialog* (Facebook) ou encore bouton Retweet (Twitter) qui permet de **republier** en l'état le { tweet } d'un autre utilisateur. Une troisième catégorie comprend les { *widgets* } qui permettent de publier du contenu sur d'autres sites que Facebook ou Twitter. On y trouve toutes les fonctions d'**ancrage** : *embed posts*, *embed videos*, *embed tweets,* etc. On peut aussi y inclure la fonction « Page » de Facebook, qui permet de placer dans un site un cadre où apparaissent les dernières publications de son profil. Enfin, une quatrième catégorie de { *widgets* } regroupe ce qu'on pourrait appeler les fonctions de **glose** : commentaires Facebook inclus directement dans un site, réponse (*reply*) sur Twitter… Ce sont des outils qui permettent de commenter des publications existantes.

Assentiment, circulation, publication, commentaires… autant d'outils d'écriture et de réécriture des fragments amenés à circuler sur le { Web }. Il s'agit donc bien d'outils de manipulation et de rééditorialisation des textes qui participent plus largement à la tendance structurante du numérique relevée par Bruno Bachimont, elle même s'appuyant sur une conception abstraite du texte née au XII^e^ siècle. Ceci étant, une analyse plus serrée des formes sémiotiques évoquées globalement jusqu'ici permet d'approfondir cette piste de travail.

588 La question n'est pas ici de déterminer si ce terme de « *like* » est pertinent ou pas, ou quels sentiments il traduit chez l'utilisateur. Notre position est ici communicationnelle : un certain dispositif sémiotechnique (le « profil ») permet de qualifier une action technique (un clic sur une icône) comme la manifestation d'une approbation individuelle (« j'aime cette page / cet article »), en vertu justement des vertus communicationnelles du profil qui permet de faire le lien entre le clic et un individu donné. Pour plus de détails sur cette construction identitaire du profil, voir Georges, Fanny, Seilles, Antoine, Artignan, Guillaume, *et al. Sémiotique et visualisation de l'identité numérique : une étude comparée de Facebook et Myspace* [manuscrit en ligne]. 2009. [Mis en ligne le 25 août 2009] [consulté le 1^er^ septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-00410952 ; Gomez-Mejia, Gustavo. *Op. cit.* ; Gomez-Mejia, Gustavo et Candel, Étienne. Signes passeurs et signes du web : le bouton *like*, ou les ressorts d'un clic. Dans : Barats, Christine, *Manuel d'analyse du Web en sciences humaines et sociales*. Paris : Armand Colin, 2013, p. 141-146.

b Un « appel à l'écriture »

petites formes, signe passeur, tweet, *widget*

Voir glossaire tome II, p. 3-25

Au sein des { *widgets* } eux-même, on constate une forte mise en avant des fonctionnalités de manipulation :

Fig. 20. Les formes de l'assentiment : le bouton « J'aime » (1). Capture d'écran du site www.slate.fr (détail). Réalisée le 11 août 2015.

Fig. 21. Les formes de l'assentiment : le bouton « J'aime » (2). Capture d'écran du site cinema.jeuxactu.com (détail). Réalisée le 11 août 2015.

Fig. 22. La réécriture des formes de l'assentiment. Capture d'écran du site www.facebook.com (Profil personnel, détail). Réalisée le 11 août 2015 à 23h46.

Les fenêtres flottantes des figures 20 et 21 apparaissent lorsque l'on clique sur le bouton « J'aime ». La figure 22 apparaît sur notre profil Facebook quand on choisit d' « ajouter un commentaire[589] ». On remarque d'emblée

589 Du reste, la mention « x personne aime y page » (sur le même modèle que « Samuel Goyet aime Filmsactu.com ») apparaît également sur le profil quand on clique sur le bouton « J'aime » d'une page web.

que l'action promise par le { signe passeur[590] } – déclarer « aimer » un site web – se double d'une panoplie d'autres { signes passeurs } permettant de commenter ce « J'aime ». On peut écrire ce pour quoi on a aimé, d'autres personnes peuvent commenter à leur tour… Nous pourrions même, en vertu de la mise en abîme permise par ces { petites formes }, aimer que nous aimons filmsactu.com (figure 22, icône « J'aime ») ! Graphiquement, l'action initiale est vite reléguée dans des zones marginales de la page. Le cadre qui apparaît lors du clic sur l'icône du pouce levé occulte cette dernière et, par la fonction indexicale[591] du cadre, attire l'attention des internautes vers la zone de texte fraîchement apparue.

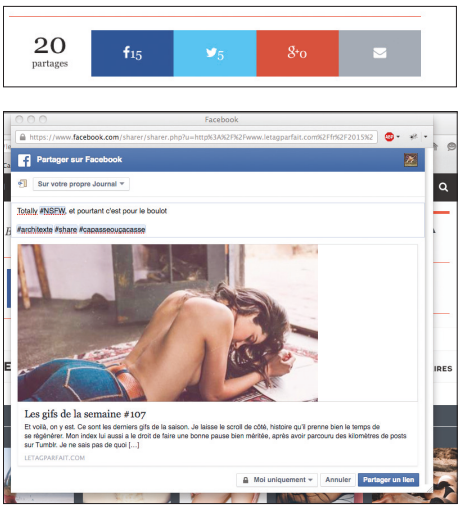


Fig. 23. Un « appel à l'écriture » dans les petites formes (1). Captures d'écran du site www.letagparfait.com (détails). Réalisées le 11 août 2015.

Fig. 24. Un « appel à l'écriture » dans les petites formes (2). Capture d'écran du site www.bigbrowser.blog.lemonde.fr (détail). Réalisée le 16 juillet 2015.

Cet « appel à l'écriture » qui passe par les multiples moyens de manipulation des textes se remarque dans d'autres formes plus complexes.

La figure 24 est un { tweet } ancré dans un article du blog « Big Browser » hébergé par lemonde.fr. Les deux captures de la figure 23 sont tirées du site letagparfait.com. La capture du haut est la barre de boutons de « partage » de la page en question. La capture du bas est la fenêtre qui s'ouvre dans un navigateur quand on clique sur le bouton Facebook, celui le plus à gauche de la barre de partage, visible par l'icône « F» bleu foncé. Dans ces deux modalités de republication, le contenu principal (les images) occupe la place centrale. En revanche, on est frappé par la multiplicité de

590 Le concept de signe passeur, théorisé dès 1999, désigne un certain type de signe présent dans les écrits d'écran. Les signes passeurs permettent d'afficher un nouveau texte ou de transformer le texte existant. Ce sont donc des signes effectifs, qui possèdent une puissance agissante sur le texte et qui préfigure leurs effets à travers leur apparence même. Voir Gomez-Mejia, Gustavo et Candel, Étienne. Signes passeurs et signes du web : le bouton *like*, ou les ressorts d'un clic. Dans : Barats, Christine, *op. cit.*, p. 141-146 ; Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Op. cit.*, p. 100.

591 Béguin-Verbrugge, Annette. *Op. cit.*

API,
code source,
développeur,
industrialisation,
petites formes,
signe passeur,
tweet,
widget

Voir glossaire tome II, p. 3-25

{ signes passeurs } permettant de commenter ou de refaire circuler le lien ou le { tweet }. Parmi ces signes, le { tweet } ancré, qui reprend – sous la photo – la structure canonique d'un { tweet } : texte d'accompagnement, photo de profil de l'auteur[592] accompagnée d'un bouton « Suivre », puis sous le texte les boutons pour répondre au { tweet }, republier le { tweet } en l'état (retweet) ou le mettre en favoris (petite étoile). Tout est fait pour garantir la reprise et la circulation d'un { tweet } au sein de Twitter et la fonction d'ancrage du { tweet } permet de favoriser cette manipulation en dehors du contexte original de publication.

Le cas du *sharer* (module de partage) Facebook est plus intéressant. Il permet de publier un lien sur son profil personnel mais pas seulement. Le lien est d'abord l'objet d'une prévisualisation, avec une illustration définie par les { développeurs } dans le { code source } de la page. Surtout, le *sharer* permet de régler finement les modalités de partage : on peut assortir le lien d'un texte, on peut publier ce lien dans notre journal, dans le journal d'un ami ou comme message privé (figure 23, menu déroulant en haut à gauche, « Dans votre propre journal »). On peut enfin régler le degré de publicité du lien. Ici, nous avons décidé que nous seul pourrons voir la publication (rangée de bouton du bas, premier en partant de la gauche, avec l'icône du petit verrou : « Moi seulement »). Tous ces réglages définissent les modalités de republication et donc les manipulations possibles du lien hypertexte. Il s'agit bien d'outils d'éditorialisation en ceci qu'il est question de pouvoir définir les conditions précises de recontextualisation d'un texte.

592 Qui est déjà un lien vers son profil Twitter.

Cette forte présence d'outils d'éditorialisation au sein même des { petites formes } est une constante de notre corpus[593]. Il s'agit d'un phénomène assez complexe car hybride. Un premier aspect du phénomène tient à la dynamique propre au numérique fragmentation / recombinaison / éditorialisation. Cette dynamique travaille les formes présentes à l'écran : les { *widgets* } sont tout autant des outils de dissémination que des outils de manipulation de fragments textuels. Ils équipent, par des possibilités de commentaires ou de réglages de la publication, la recontextualisation de ces fragments. Cette recontextualisation repose en partie sur un « appel à l'écriture » : jeux de cadres, omniprésence de { signes passeurs } permettant de commenter la publication ou renvoyant vers son auteur originel… C'est le second aspect de ce phénomène : les { API } ne produisent pas que des formes qui promeuvent une vision modulaire de la page. Ces formes sont faites pour appeler d'autres textes : commentaires, réponses… Elles comprennent donc en elles le principe d'une glose circulante. Cette glose, c'est le troisième aspect, est essentielle aux plateformes de réseaux sociaux dont la notion de lien social repose en grande partie sur le partage et la mesurabilité de fragments identitaires polyphoniques[594]. L'appel à l'écriture permet donc de nourrir la plateforme, d'entretenir la bonne tenue des liens identitaires. Les { API } web de notre corpus participent en ceci au phénomène de conscription[595] décrit par Gustavo Gomez-Mejia et ont partie liée avec une { industrialisation } des formes de l'identité[596].

593 Voir annexe nº 13.

594 Doueihi, Milad. *Op. cit.*,2011, p. 88-89.

595 La « conscription » désigne le fait que dans les dispositifs du Web contemporain, l'identité « s'écrit ensemble » : un individu est défini par la mise en relation avec d'autres individus. La liste (x amis aiment aussi…) ou le répertoire de visages (« Vous connaissez peut-être… » / « *Followers you know* » sur Twitter) sont des formes privilégiées de la conscription. Voir Gomez-Mejia, Gustavo. *Op. cit.*, p. 201-203.

596 Nous ne traiterons pas plus loin ce thème, n'étant pas la problématique centrale de notre thèse.

API, calcul, discrétisation, écriture-calcul, *widget*

Voir glossaire tome II, p. 3-25

Les { API } web, en tant qu'elles appartiennent à une histoire du texte et de l'écriture, participent donc à une dynamique ancienne d'abstraction et de manipulabilité des textes. Les propriétés spécifiques de l'{ écriture-calcul }, notamment sa { discrétisation }, renforcent cette dynamique. Dans le cas des { API } web, cela se manifeste par des formes non seulement particulièrement amenées à se transformer et à se rééditorialiser, mais aussi par le fait que ces formes embarquent les outils de cette transformation éditoriale par le commentaire ou encore la republication.

Mais ces manipulations sont encore opérées par un être humain. Elles sont en partie automatisées, certes (on l'a vu avec l'exemple du *Tag Parfait* et du choix de la photo), mais c'est l'internaute qui décide quoi et comment republier. Que se passe-t-il quand ce travail éditorial est entièrement automatisé et exécuté par une machine ? Et qu'est-ce que cela peut nous dire sur le phénomène de rééditorialisation qui semble au cœur de la logique textuelle des { API } ?

Les textes de réseau appartiennent à une histoire longue du texte où l'abstraction va de pair avec des outils de manipulation d'un texte devenu modulaire. Le { calcul } informatique, en tant qu'il repose sur une { discrétisation }, renforce cette manipulabilité du texte. L'analyse des { *widgets* } de notre corpus nous a permis de montrer cette tendance à la manipulabilité : les { *widgets* } intègrent en eux les possibilités de leur manipulation et de leur rééditorialisation. Ce sont des formes écrites pour appeler à la réécriture.

API, calcul, discrétisation, forme-texte, petites formes

Voir glossaire tome II, p. 3-25

Mais la { discrétisation } pose le problème de l'éditorialisation : une fois une forme découpée en unités élémentaires et insignifiantes pour permettre le { calcul }, comment redonner du sens à ces unités ? Comment les réassembler en formes sémiotiques intelligibles ? Et comment cette { discrétisation } a partie liée avec la tendance à la manipulabilité des formes ? Pour interroger ce lien et répondre à ces questions, il faut étudier un cas particulier d'éditorialisation : l'éditorialisation automatique par des machines. Parfois, la recomposition de formes sémiotiques est assumée automatiquement par Twitter ou Facebook et leurs { API } respectives permettent cette génération automatique. L'étude de deux cas (III. 3 A. ; III. 3. B.) nous permettra de mettre en lumière le rôle structurant de la { discrétisation } dans l'éditiorialisation des textes de réseau par les { API }. Une { API } met à disposition les unités élémentaires des { petites formes } des textes de réseau, cela permettant une plus grande manipulabilité et personnalisation de ces formes. Ainsi, nous montrerons que les { API } intensifient ces liens entre manipulabilité et { discrétisation } : plus une forme est discrétisable, plus elle est manipulable et plus elle peut donc se prêter aux réécritures en tout genre (III. 3. C). Pour les industries contemporaines du texte, fournir, grâce à une { API }, le niveau le plus fin de granularité devient alors un enjeu éditorial et économique : c'est permettre la réappropriation et la circulation des données à travers des { formes-textes } faites pour cela.

3 DE LA DISCRÉTISATION À L'ÉDITORIALISATION AUTOMATIQUE

Si certaines { formes-textes } générées par les { API } permettent de ré-éditorialiser des publications comme nous venons de le voir, certaines prennent complètement en charge cette fonction éditoriale en l'automatisant. C'est notamment le cas des formes-seuils comme le bouton « *Facebook Connect* ». Les exemples ci-dessous sont tirés du site senscritique.com, un site de critique culturelle.

III 1 B a
formes-seuils
Voir p. 260

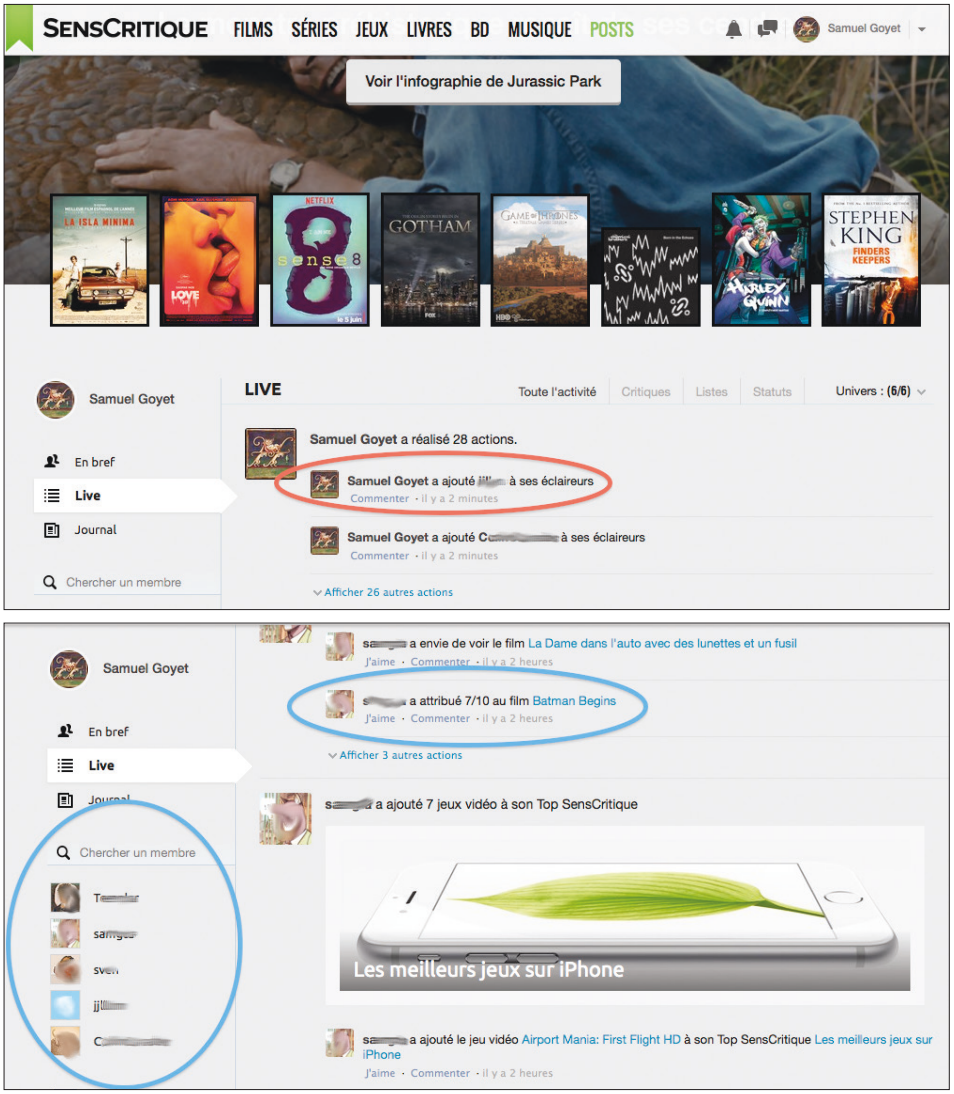


Fig. 25. API et éditorialisation automatique (1) : le cas Sens Critique. Captures d'écran du site www.senscritique.com (détails). Réalisées le 07 août 2015.

Nous avons utilisé la fonctionnalité « Se connecter avec Facebook » qui permet d'économiser la peine de se créer un compte utilisateur auprès de Sens Critique. Lorsque l'on clique sur le bouton « Commencez maintenant » (figure 25, capture du haut, bouton central bleu), une fenêtre avec la charte graphique de Facebook s'ouvre et nous demande d'autoriser Sens Critique à récupérer les informations de notre profil Facebook[597]. Une seconde fenêtre permet ensuite de régler le degré de publicité des publications à venir, dans la veine de ce que nous avons vu plus haut. Enfin, c'est ce qui nous intéresse maintenant, Sens Critique crée un profil utilisateur avec notre photo de profil Facebook, nous annonce que nous avons réalisé une petite trentaine d' « actions » comme « ajout[er] J*** à [nos] éclaireurs »

597 Pour l'étude de cette forme contractuelle qui règle l'échange de données entre deux logiciels ou plateformes, voir notre partie sur le *token* d'identification en II. 1. D. b. Nous nous concentrons désormais sur les effets éditoriaux de cet échange de données.

(entourées en rouge). Plus loin, nous voyons que d'autres utilisateurs du site, au demeurant dans notre cercle de contacts Facebook, ont réalisé des actions similaires (entourées en bleu). Or, nous n'avons pas fait autre chose qu'autoriser Sens Critique à accéder à nos données Facebook *via* l'{ API } de la plateforme. Le profil a été généré automatiquement par des machines[598].

API, tweet

Voir glossaire tome II, p. 3-25

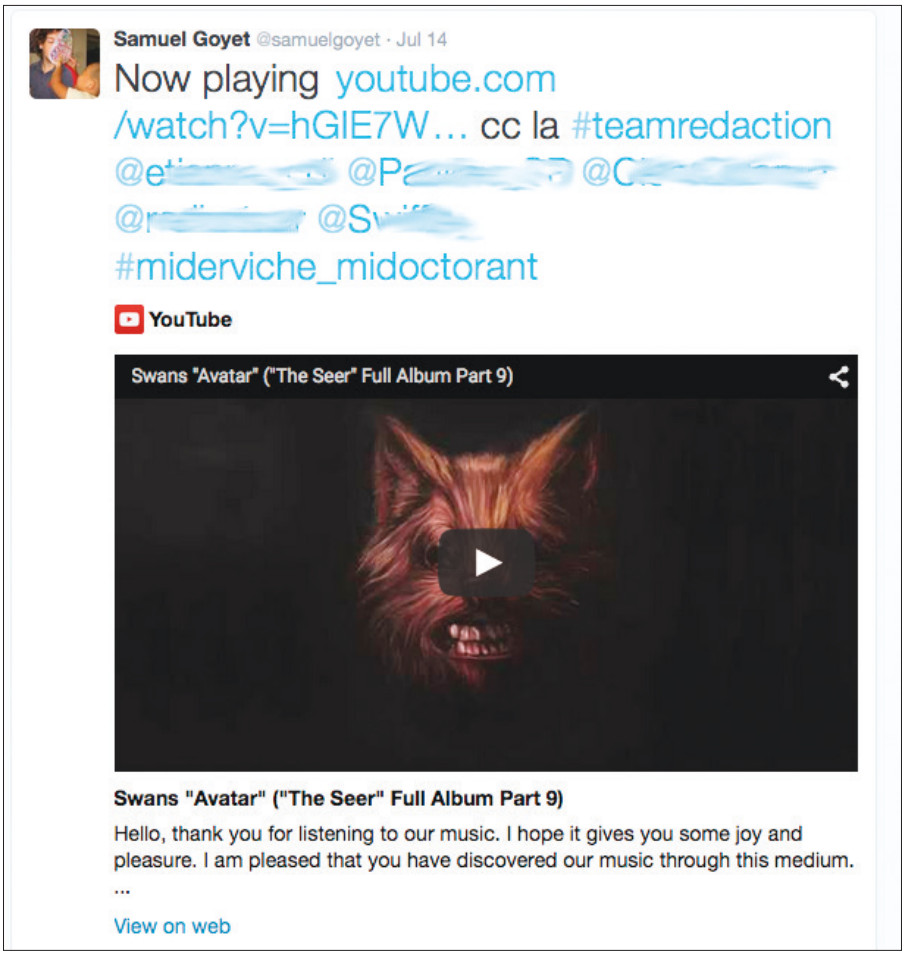


Fig. 26. API et éditorialisation automatique (2) : le cas des cartes Twitter. Capture d'écran du site twitter.com (profil personnel, détail). Réalisée le 18 juillet 2015 à 16h01.

Le { tweet } ci-dessus est une vidéo YouTube publiée sur Twitter. Il illustre le principe de la « carte », un principe de design web qui consiste à synthétiser les informations essentielles d'un sujet en une forme graphique condensée. On y trouve un lecteur vidéo, un titre, le site source, un descriptif ainsi qu'un lien vers YouTube. Tout ce que nous avons fait, en tant qu'internaute, c'est copier-coller le lien de la vidéo dans Twitter et l'assortir des mots : « *Now playing* [lien] cc la #teamredaction @[nom d'utilisateurs Twitter] #miderviche_midoctorant ». Tout le reste de la mise en page est automatisé et pris en charge par Twitter.

598 Voir également la plateforme about.me qui repose sur le même principe. Voir annexe nº 14.

API,
code,
développeur,
discret,
discrétisation,
documentation,
graphe,
graphe social,
interface,
petites formes,
tweet,
Web

Voir glossaire tome II, p. 3-25

Dans ces deux cas, parmi d'autres[599], une page ou une { petite forme } est crée par des machines sans intervention humaine autre qu'un clic de l'internaute. Ce constat nous permet de formuler l'hypothèse de cette sous-partie : les { API }, par le biais du { code informatique }, automatisent l'éditorialisation des textes de réseau. Et par éditorialisation, nous entendons avec Bruno Bachimont[600] une composition textuelle par recombinaison d'éléments { discrets }. Cette recombinaison s'appuie sur la façon dont les { tweets } et les profils Facebook sont construits : comme un agencement d'unités plus élémentaires, non nécessairement visibles quand on s'en tient aux formes sémiotiques vues jusqu'ici. L'étude successive des deux exemples ci-dessus – le bouton *Connect* de Facebook sur le site Sens Critique (III. 3. A) et la génération automatique par Twitter d'une « carte » (III. 3. B.) – nous permettra de comprendre comment se joue précisément l'éditorialisation automatique. Nous montrerons notamment que ce que l'{ API } rend public dans sa { documentation }, ce sont les unités élementaires de ces formes, fournissant par là les conditions d'implémentation technique d'une telle éditorialisation automatique. En étudiant d'encore plus près la { documentation }, nous montrerons que le lien entre { discrétisation } et manipulation est non seulement confirmé, mais intensifié : plus un texte est discrétisable, plus il est manipulable (III. 3. C.). Ce qui amène au système tant symbolique qu'économique qui accompagne cette recherche d'une { discrétisation } maximale. L'étude de la manipulation et de la rééditorialisation automatique *via* les { API } nous mènera ainsi au modèle économique propre à ce maillon de l'industrie contemporaine du texte que sont les { API } web, modèle qui fait l'objet du chapitre suivant.

599 Voir annexe nº 15.

600 Bachimont, Bruno. Nouvelles tendances applicatives : de l'indexation à l'éditorialisation. Dans : Gros, Patrick (dir.), *op. cit.*, p. 313-326.

3 A Sens Critique : de l'*Open Graph* comme vecteur d'une éditorialisation automatique

Pour comprendre le fonctionnement de ces cas d'éditorialisation comme recombinaison d'éléments { discrets }, il faut commencer par en expliquer le fonctionnement technique, qui repose sur le « { graphe social } » de Facebook (III. 3. A. a). Dans le cas de Sens Critique, un site intègre dans sa page d'accueil la fonction *Facebook Login*, qui permet de créer un profil utilisateur en utilisant les données Facebook de cette personne. Pour cela, Sens Critique utilise le { graphe social } de Facebook mais adapte et requalifie selon ses objectifs communicationnels propres les éléments du { graphe }. Dans le cas de Sens Critique, la rééditorialisation fonctionne donc sémiotiquement comme une requalification des données issues du { graphe social }, requalification qui se fait à travers l'{ interface } du site Sens Critique (III. 3. A. b).

a Le graphe social : une cartographie du Web

Le { graphe social } (*Social Graph*) est inspiré de la théorie des { graphes } en mathématique et en informatique. Particulièrement développée depuis le début du xx^e^ siècle[601], cette théorie consiste en l'étude et la construction de { graphes }, soit la représentation de systèmes (biologiques, urbains, informatiques…) sous la forme graphique d'entités (ou « nœuds ») reliées par des liens. Le { graphe social } est présenté en 2010 au cours de la conférence F8[602], conférence annuelle tenue par Facebook à l'attention d'un public de { développeurs } internes et externes. L'idée est la suivante : Facebook peut être représenté comme un ensemble de nœuds (un profil personnel, une page de marque, un groupe, un statut, une image, une vidéo, etc.) reliés par des liens. Quand un utilisateur « aime » une page, il crée une connexion dans le { graphe } : x « aime » la page y. Il est important de bien dissocier dans un premier temps le concept du { graphe } et sa manifestation sémiotique. Le { graphe } permet de représenter n'importe quel type de relation, pas nécessairement d'assentiment comme l'exemple que nous venons de donner.

601 Beineker, Lowell et Wilson, Robin. Modern graph theory. Dans : Wilson, Robin et Watkins, John J. (dir.), *op. cit.*, p. 331-352.

602 Facebook. *f8 Conference Keynote Speech – Recorded 21st April 2010* [captation vidéo en ligne]. Conférence donnée à Palo Alto le 21 avril 2010. [Mis en ligne le 21 avril 2010] [consulté le 1^er^ août 2017]. Disponible à l'adresse : http://original.livestream.com/f8conference/video/pla_e7a096b4-3ef9-466d-9a37-d920c31040aa.

API,
balise,
côté client,
côté serveur,
discret,
documentation,
formalisme,
graphe,
graphe social,
HTML,
interface,
métadonnée,
script,
Web

Voir glossaire tome II, p. 3-25

Si nous devenons amis avec une personne, c'est une autre relation qui se crée dans le { graphe }, si nous publions un statut, *idem*. De manière générale, cela signifie que toute action sur Facebook peut prendre une forme cartographique dont la représentation syntaxique serait : x [une action] y. Par exemple : Samuel publie une photo ; Marion joue à tel jeu ; Alexandre est à Bonifaccio, etc[603]. Le { graphe social } est complété la même année (2010) par le protocole *Open Graph*, qui permet « [...] à n'importe quelle page web de devenir un objet enrichi dans un graphe social[604] ». Concrètement, le protocole *Open Graph* est un ensemble de { balises } { HTML }, reconnues par les principaux navigateurs, qui permet de représenter une page web sous forme de { métadonnées }. Ces { métadonnées } peuvent ensuite être intégrées dans un { graphe }. La portée politique de ce protocole est non négligeable : si toute page web, qui comporte les { métadonnées } adéquates, peut être représentée dans un { graphe }, cela signifie qu'elle peut être, en théorie, intégrée dans le { graphe social } propre à Facebook. L'*Open Graph* est en quelque sorte une extension du principe de { graphe social } de Facebook à l'ensemble du { Web[605] }.

Le { graphe social } suppose donc que toute action sur Facebook et sur le { Web } peut être comprise comme une action d'une entité élémentaire (nœud A) sur une autre (nœud B). Cette représentation est bien { **formelle** } (elle ne se préoccupe guère en principe de la nature de l'action et de son sens dans un contexte précis), { **discrète** } (elle traite des unités indépendantes les unes des autres, les nœuds) et **combinatoire** : tous les nœuds peuvent être combinés entre eux par le biais d'actions. La *Graph API* est l'{ interface } qui permet « d'explorer » le { graphe }, c'est-à-dire de passer d'un nœud à un autre en voyant quelles sont les relations construites entre les deux[606], mais aussi de construire des relations entre différents nœuds.

603 Voir annexe nº 16.

604 « *The Open Graph enables any web page to become a rich object in a social graph.* » Open Graph Protocol, page d'accueil. 2017 [en ligne]. [Dernière modification le 27 juillet 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://ogp.me/. Développé par Facebook, l'*Open Graph Protocol* est désormais géré par l'*Open Web Foundation*.

605 Les premières minutes de l'intervention de Mark Zuckerberg à la conférence F8 de 2010 vont dans ce sens : tout sur Facebook peut être compris comme une action dans un graphe social, mais plus largement toute action sur le Web peut être intégrée au graphe. Voir Facebook. *Op. cit.*

606 Voir annexe nº 17.

b L'éditorialisation selon Sens Critique : une requalification sémiotique de la structure du graphe

Nous pouvons désormais revenir à Sens Critique. Le bouton « Commencez maintenant » déclenche un { script } { côté client } qui permet d'identifier le site auprès de Facebook en utilisant la fonction *Facebook Connect* décrite dans la { documentation } de l'{ API[607] }. Une fois identifié, une fois que l'utilisateur a accepté que Sens Critique accède à ses données, le site est mis en relation avec le { graphe social } de l'utilisateur. D'autre { scripts } prennent le relai { côté serveur } et explorent ce { graphe } afin d'en retirer les informations pertinentes auxquelles ils ont accès. En voyant que Sens Critique crée automatiquement une liste de contacts reprenant celle de nos amis Facebook, on peut supposer qu'un { script } récupère notre liste d'amis, la compare à la liste des utilisateurs de Sens Critique et ne garde que nos amis Facebook qui sont aussi inscrits sur Sens Critique

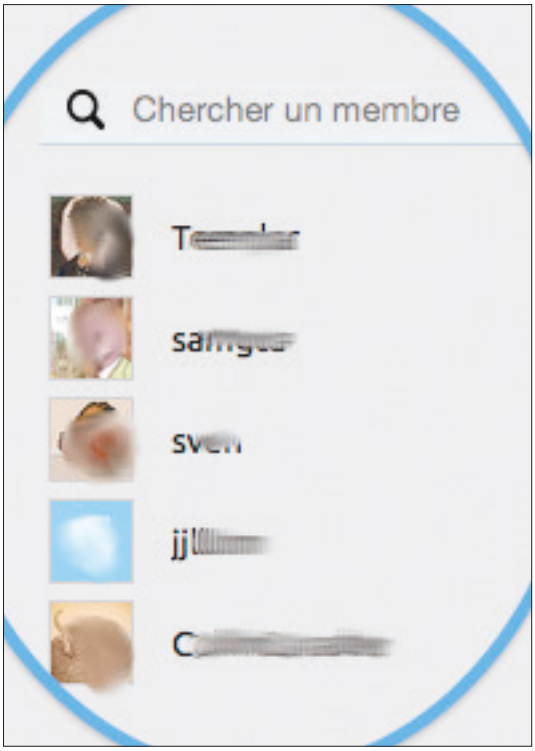


Fig. 27. La requalification automatique des « amis » en « éclaireurs ». Capture d'écran du site www.senscritique.com (détail). Réalisée le 7 août 2015.

607 Voir annexe nº 18.

application,
balise,
code,
discret,
format,
graphe,
interface,
petites formes,
programme,
script,
tweet

Voir glossaire tome II, p. 3-25

D'un point de vue techno-sémiotique, qui croise le fonctionnement du { code } et les formes qu'il engendre, le site part de la structure { discrète } et combinatoire du { graphe } pour ensuite requalifier, au niveau de la page web, les relations récupérées dans le { graphe }. Les « amis » Facebook deviennent des « membres » ou des « éclaireurs », comme dans la phrase générée automatiquement : « Samuel Goyet a ajouté [nom d'un contact Facebook] à ses éclaireurs ». La base logique est la même (un nœud relié à un autre) mais le contexte et les imaginaires du lien social sont différents. L'éditorialisation consiste à transformer une structure mathématique – le { graphe } – en une structure sociale, où le choix des termes et des catégories témoigne d'une certaine conception de la critique culturelle (dans le cas de Sens Critique), où des membres en « éclairent » d'autres et font dévouvrir des références ignorées.

L'automatisation du processus éditorial est donc une requalification de la structure du { graphe } de Facebook dans le vocabulaire propre au site utilisant les données de ce { graphe }. Le { script } qui génère la page est bien dans l'exercice d'une fonction éditoriale : il manipule des éléments { discrets } (nœuds et relations) et les compose comme texte. Il les insère dans un nouveau contexte, ici une { interface } web, appelant de nouvelles interprétations. Encore une fois, cette recontextualisation est entièrement automatisée. Elle est préparée en amont par des { programmes } informatiques et des { formats } de données adéquats, elle est déclenchée par l'utilisateur (lorsque nous donnons notre approbation pour que Sens Critique accède à nos données), mais l'exploration du { graphe }, la manipulation de ses éléments et leur textualisation dans une page web sont faites par notre navigateur.

3 B Les cartes Twitter : un exemple du lien entre discrétisation et éditorialisation automatique

Le fonctionnement des cartes Twitter est un peu différent. Une « carte » s'affiche dans un { tweet } contenant un lien (vers un article, une vidéo, une { application } mobile, etc.). Plutôt que de simplement afficher le lien sous forme texte, Twitter crée un petit chapeau introductif avec un titre, un texte descriptif, une image, éventuellement un lecteur vidéo si le lien partagé est une vidéo. Cette { petite forme } n'est pas crée par l'internaute, elle est automatisée par Twitter. Elle est crée à partir d'un autre dispositif technique : le protocole *Open Graph* (III. 3. B. a), qui permet de décrire par des { balises } les différents éléments présents dans une carte. Ce sont ces { balises } que Twitter scanne et rassemble en une forme sémiotique – la carte – qui puise dans une mémoire sociale des formes pour rendre cet assemblage culturellement lisible. Ainsi dans le cas des cartes Twitter, l'éditorialisation automatique se caractérise comme recomposition sémiotique d'une fragmentation technique (III. 3. B. b).

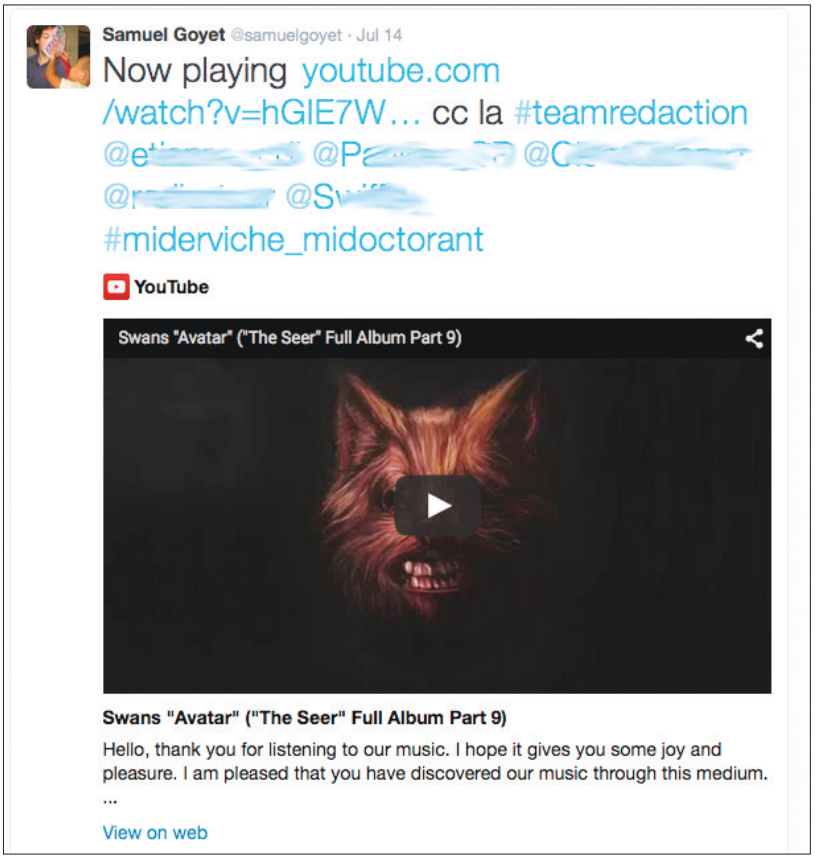


Fig. 28. API et éditorialisation automatique : le cas des cartes Twitter. Capture d'écran du site twitter.com (profil personnel, détail). Réalisée le 18 juillet 2015 à 16h01.

a Les cartes Twitter : un ancrage dans le protocole *Open Graph*

API,
balise,
code,
code source,
métadonnée,
petites formes,
tweet

Voir glossaire tome II, p. 3-25

L'{ API } de la plateforme permet de comprendre comment fonctionnent les cartes. Pour que Twitter puisse générer une carte, il faut inclure en amont dans le { code source } de la page amenée à être partagée certaines { métadonnées }, dont certaines sont rédigées suivant la syntaxe propre à Twitter, d'autres suivant la syntaxe du protocole *Open Graph*[608]. Si nous reprenons l'exemple de la vidéo YouTube plus haut, voici un extrait du { code source } de la page hébergeant la vidéo que nous avons partagée sur Twitter :

Fig. 29. L'écriture de l'*Open Graph*, condition de l'automatisation des petites formes. Extrait du code source de la page www.youtube.com-watch?v=hGIE7WaTuMg. Capture réalisée le 18 juillet 2015.

Nous avons retenu les dernières lignes de la { balise } `<head>`[609]. À chaque ligne correspond une { balise } et plus précisément une { métadonnée }. Les lignes qui concernent Twitter sont les suivantes :

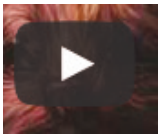
```
<meta content="player"  name="twitter:card">
<meta content="@youtube" name="twitter:site">
<meta content="http://www.youtube.com/watch?v=hGIE7WaTuMg"
name="twitter:url">
<meta content="Swans "Avatar" ("The Seer" Full Album Part
9)" name="twitter:title">
<meta content="Hello, thank you for listening to our music.
I hope it gives you some joy and pleasure. I am pleased
that you have discovered our music through this medium…"
name="twitter:description">
<meta content="https://i.ytimg.com/vi/hGIE7WaTuMg/maxresde-
fault.jpg" name="twitter:image">
[…]
<meta content="https://www.youtube.com/embed/hGIE7WaTuMg"
name ="twitter:player">
<meta content="1280" name="twitter:player:width">
<meta content="720" name="twitter:player:height">
[…]
</head>
```

608 Ce qui permet donc à Facebook de générer automatiquement un lecteur vidéo semblable à celui de Twitter.

609 Cette balise est faite pour accueillir les métadonnées de la page, c'est-à-dire des informations qui ne seront pas nécessairement affichées par le navigateur interprétant le code source.

Chaque { balise } est délimitée par un chevron ouvrant (<) et fermant (>). Une { balise } équivaut à une ligne. Le navigateur interprète le { code } de gauche à droite et de haut en bas, ligne par ligne. Le mot `meta` signale que les informations contenues dans la { balise } sont des { métadonnées }, le mot `content` précise le contenu de cette { métadonnée }. Enfin, le paramètre `name` spécifie le type de { métadonnée } selon la syntaxe préconisée par Twitter. Par exemple, la troisième ligne de cet extrait est une { métadonnée } qui contient une URL.

L'ensemble des { balises } permet de décrire dans une syntaxe lisible par Twitter de quel type de page il s'agit et quel type de carte générer. Quand cette page est publiée sous forme de lien sur Twitter, la plateforme lit le { code source } de la page et les informations ci-dessus. Elle se sert de ces informations pour mettre en forme le { tweet } et créer la { petite forme }. Voici le tableau de correspondance entre les lignes de { code } et les éléments du lecteur vidéo.

Ligne de code	Explication du code	Forme générée
`<meta content="player" name="twitter:card">`	Indique à Twitter qu'une carte peut être crée	—
`<meta content="@youtube" name="twitter:site">`	Indique le site source de la carte (YouTube)	YouTube
`<meta content="http://www.youtube.com/watch?v=hGIE7WaTuMg" name="twitter:url">`	URL de la page	—
`<meta content="Swans "Avatar" ("The Seer" Full Album Part 9)" name="twitter:title">`	Titre de la vidéo	Swans "Avatar" ("The Seer" Full Album Part 9) Swans "Avatar" ("The Seer" Full Album Part 9)
`<meta content="Hello, thank you for listening to our music. I hope it gives you some joy and pleasure. I am pleased that you have discovered our music through this medium…" name="twitter:description">`	Texte de présentation de la vidéo (écrit par celui qui l'a mis en ligne)	Hello, thank you for listening to our music. I hope it gives you some joy and pleasure. I am pleased that you have discovered our music through this medium. …
`<meta content="https://i.ytimg.com/vi/hGIE7WaTuMg/maxresdefault.jpg" name="twitter:image">`	Vignette illustrant la vidéo	
`<meta content="https://www.youtube.com/embed/hGIE7WaTuMg" name ="twitter:player">`	Indique à Twitter de créer un lecteur vidéo	
`<meta content="1280" name="twitter:player:width">`	Définit la largeur du lecteur, en pixels	—
`<meta content="720" name="twitter:player:height">`	Définit la hauteur du lecteur, en pixels	—

Ce tableau permet de comprendre les liens qui se tissent entre le niveau technique (le { code informatique }) et le niveau sémiotique de cette { petite forme }. Le { code } opère une { discrétisation } des informations composant une vidéo, { discrétisation } qui a comme mètre étalon la ligne de { code }. À chaque ligne, un élément de la vidéo. Le lecteur tel qu'on peut le voir sur Twitter est le résultat d'une combinaison de ces différents éléments. C'est l'illustration du couple { discrétisation } / manipulation théorisé par Bruno Bachimont : chaque carte Twitter est le produit d'une manipulation d'éléments { discrets }. Cela confirme la forte dissociation entre la perception d'une unité d'un point de vue sémiotique (on voit un « lecteur vidéo » dans un « { tweet } ») et sa composition technique éclatée.

code, discret, discrétisation, petites formes, signe passeur tweet

Voir glossaire tome II, p. 3-25

b L'éditorialisation automatique : fragmentation technique, recomposition sémiotique et mémoire des formes

Un examen plus approfondi de ces « synthèses toutes faites[610] » permet donc de comprendre l'action éditoriale de Twitter. La plateforme combine des éléments pour produire une { petite forme }, qui est la recomposition sémiotique d'informations fragmentées dans leur forme technique. Cette recomposition fait appel à une certaine mémoire des formes, qui permet de rendre intelligible aux lecteurs ce qu'ils ont sous les yeux : un « lecteur » vidéo. Et si nous cliquons sur ce « lecteur », la vidéo se déclenchera[611]. L'histoire des formes médiatiques joue donc à la fois le rôle de moule, pour organiser la recomposition des unités { discrètes }, mais aussi de garde-fou contre l'éventuelle désorientation que cette { discrétisation } peut entraîner. Les unités { discrètes } manipulées par la machine (un titre, une URL, une image) peuvent en théorie être assemblées n'importe comment, mais ce serait contre-productif, car illisible, dans la pratique. Le regard du lecteur humain est informé par une culture du texte, une culture de ce qu'est une vidéo et de comment la lire. La forme recomposée automatiquement que nous venons de voir puise à la fois dans cette mémoire des formes et la perpétue. Elle se sert de cette mémoire pour être lisible, la transforme d'une certaine manière – en la discrétisant – mais la reconduit, ce sans quoi on ne saurait à quoi renvoie ce { signe passeur }.

610 Nous reprenons les mots de Michel Foucault à propos de sa méthode : « Il faut remettre en question ces synthèses toutes faites, ces groupements que d'ordinaire on admet avant tout examen. » Foucault, Michel. *Op. cit.*, p. 32.

611 Pour une analyse historicisée des formes d'écoute en ligne et notamment des analyses de ces formes particulières des lecteurs audio, voir Gras, Stéphan-Eloïse. *L'écoute en ligne : Figures du sujet écoutant et mutations des espaces musicaux sur Internet*. Thèse de doctorat. Paris : Paris-Sorbonne, 2014 ; Heuguet, Guillaume. *De la musique sur les médias informatisés : axiologies culturelles et systèmes d'usages*. Thèse de doctorat. Paris : Paris-Sorbonne, en cours.

API,
appel,
code,
développeur,
discret,
discrétisation,
documentation,
littératie,
objet,
petites formes,
tweet,
widget

Voir glossaire tome II, p. 3-25

L'action éditoriale du { code informatique }, c'est donc précisément d'allier une mémoire sociale des formes et la structure { discrète } des données informatiques, structure dont la { documentation } de l'{ API } fournit dans le cas présent la grammaire. C'est une action éditoriale en ceci qu'elle donne sens à une foule d'éléments disparates en les liant par une mise en page, par une mise en texte : le { code } détermine la hauteur et la largeur du lecteur, la place respective des éléments les uns par rapport aux autres… En vertu de cette mémoire sociale des formes et de son rôle essentiel dans la recomposition effectuée par Twitter, nous soutenons donc que la machine ne fait pas que « manipuler ». Elle intègre dans son fonctionnement des modèles culturels. Elle recontextualise les différents fragments et leur donnent un « sens », par le choix de certaines organisations visuelles[612].

III **2**
des formes autorisant la manipulation des textes de réseau
Voir p. 289

Non seulement les formes que les { API } permettent d'écrire sont donc bien souvent des formes autorisant la manipulation des textes de réseau, mais de surcroît ces formes sont elles-mêmes le produit d'une éditorialisation, parfois automatisée. Cette éditorialisation s'appuie, comme on vient de le voir, sur une { discrétisation } : tout { *widget* }, tout bloc de fonctionnalité est une combinaison d'éléments plus « élémentaires » que le { *widget* } même. Ce qui est en jeu dans la { discrétisation }, c'est finalement la granularité de ces éléments, leur degré de précision pourrait-on dire. Or, l'une des fonctions principales de la { documentation } d'une { API }, c'est de fournir la liste détaillée des éléments de base qu'un { développeur } peut utiliser, la grammaire des *elementa* des { petites formes }. Pour comprendre en quoi les { API } web participent à la manipulabilité des textes de réseau, il est donc indispensable d'analyser comment, à travers cette { documentation }, les { petites formes } sont découpées en éléments { discrets }.

612 On pourrait bien sûr objecter que ce n'est plus là le travail de l'informaticien mais du designer d'interface. On pourrait surtout objecter plus fondamentalement que la machine ne fait que suivre des instructions de mise en forme rédigées par des humains et qu'elle ne peut donc « donner sens » à quoi que ce soit. Cette question essentielle sera traitée dans cette thèse (V. 1. C et V. 4), car il en va de notre rapport même à l'écriture pour et / ou avec des machines.

3 C La documentation comme grammaire des *elementa*

Cette grammaire prend le plus souvent la forme d'un tableau qui présente la manière dont sont construits informatiquement les { objets } fondamentaux de la plateforme. Par exemple, dans le cas de Twitter, un { tweet }, un utilisateur ou dans le cas de Facebook un statut, un profil, etc. Ce tableau permet à des { développeurs } externes de comprendre comment fonctionne la plateforme et les { objets } qu'ils vont être amenés à manipuler dans leur { code }. Pour rendre notre explication plus claire, nous nous concentrerons sur la page de la { documentation } de Twitter qui détaille la composition d'un { tweet[613] }. Il existe d'autre pages de la { documentation } décrivant d'autres { objets }, il existe des pages similaires dans la { documentation } de l'{ API } de Facebook. Le principe reste toutefois le même : décrire avec la plus fine granularité possible les { objets } informatiques auxquels donne accès l'{ API } (III. 3. C. a) afin d'en optimiser les futures utilisations par des { développeurs }. Ce qui nous conduira à l'idée que les { API } web, en proposant une { discrétisation } maximale des formes qu'elles permettent de produire, en maximisent également les manipulations possibles (III. 3. C. b). Le rapport entre { discrétisation } et manipulabilité n'est alors plus seulement un rapport inhérent à l'informatique, mais il est radicalisé : plus une forme est discrétisée, plus elle est manipulable et plus elle va pouvoir circuler. Ce qui a des effets sur la représentation d'une { petite forme } et la { littératie nécessaire pour la produire et la manipuler : la { discrétisation } s'accompagne d'une inflation scripturaire qui permet une plus grande personnalisation, mais demande plus de savoir-faire technique (III. 3. C. c).

a Anatomie d'un tweet

La page dédiée au { tweet } se trouve dans l'« aperçu général » (*overview*) de la { documentation } de l'{ API }. Il s'agit en effet d'un niveau assez générique, puisque le { tweet } va se retrouver dans des usages plus spécifiques de l'{ API } : { *widgets* }, { appels } au serveur pour récupérer des données… Comme toutes les pages de la { documentation }, celle-ci commence par une description de l'{ objet }, un exemple, puis le détail des composants de cet { objet }. Le { tweet } est décrit comme « […] le bloc

613 Voir annexe nº 19.

application, JSON, langage de programmation, objet, tweet, valeur booléenne

Voir glossaire tome II, p. 3-25

de construction de base de toutes les choses Twitter[614] ». Vient ensuite ce qu'on peut faire avec un { tweet } : l'ancrer, y répondre, le mettre en favoris, l'enlever de la liste des favoris ou l'effacer. Le { tweet } est donc d'emblée présenté comme un module combinable ouvert à différentes manipulations : ancrage, réponses, etc. Enfin, le *Field guide* (Guide de terrain / guide pratique[615]) détaille la liste des propriétés d'un { tweet }. Cette liste contient trois colonnes : le nom de la propriété ; son type ; sa description, souvent assortie d'un exemple. Ces propriétés peuvent être classées en deux grands ensembles : les propriétés nécessaires et les propriétés accidentelles. D'un point de vue technique, il est assez facile de trier l'ordre du nécessaire et de l'accidentel. Une propriété nécessaire est celle dont la valeur ne peut être nulle (*nullable*). Au contraire, une propriété accidentelle peut être nulle. Rappelons que nous parlons ici du { tweet } d'un point de vue technique, c'est-à-dire d'un { tweet } représenté en { JSON[616] }. À chaque propriété du { tweet }, par exemple sa date de publication, est attribuée une valeur. Si le champ « `created_at` » contient la valeur « `Thu Aug 25 17:44:03 +0000 2016` », cela signifie que le { tweet } a été publié le 25 août 2016 à 17h44.

Il y a donc des propriétés dont la valeur peut être nulle, d'autres pas. Parmi celles-ci : la date de création, le numéro d'identification (exprimé sous forme de numéro (*integer*) et de texte (*string*), une source (selon que le { tweet } ait été publié en utilisant la page twitter.com ou l'{ application } mobile par exemple), un contenu textuel et un utilisateur ayant écrit le { tweet }. En d'autres termes, un { tweet } a nécessairement :

– Un auteur (champ *user*) ;
– Un numéro d'identification (champ ID) ;
– Une source (champ *source*) ;
– Un texte (champ *text*) ;
– Une date de publication (champ *date*).

614 « *Tweets are the basic atomic building block of all things Twitter.* » Twitter, documentation de l'API. : http://dev.twitter.com/overview/api/tweets. Capture réalisée le 14 juillet 2015.

615 *Field guide* est un jeu de mot sur le double sens de *field*. *Field* peut vouloir dire « champ » au sens de « champ de texte » ou « champ sémantique ». *Field guide* renvoie spécifiquement à l'orientation, au guide pour ne pas se perdre lors d'une randonnée par exemple. La documentation de l'API joue de ce double sens : un tweet est composé de plusieurs « champs » ; en même temps la documentation permet de ne pas se perdre, elle est un guide pour utiliser Twitter.

616 *JavaScript Object Notation*, un dérivé du JavaScript qui permet de décrire de façon structurée des objets informatiques : un tweet, une photographie, etc. Dans ce langage, chaque élément d'un objet doit avoir un « type ». Un élément peut être un chiffre (*integer*), une chaine de texte (*string*), une suite d'objets (*array*) ou encore une valeur logique booléenne (*true / false*).

Les autres caractéristiques d'un { tweet } sont optionnelles. Elles peuvent à leur tour être réparties en quatre catégories :

– Les traces de manipulation du { tweet } (si le { tweet } a été retweeté, combien de fois, si le { tweet } est un retweet ; si le { tweet } est un favori ou s'il a été mis en favoris par quelqu'un, si le { tweet } est une réponse à un autre { tweet }, etc.). Il y a en tout quatorze champs dans cette catégorie.

– La présence ou non d'autres { objets } dans le { tweet } : si le { tweet } contient le nom d'un utilisateur, un lieu géographique, etc. Il y a quatre champs dans cette catégorie.

– Les données de localisation du { tweet } (*coord*, abbréviatif de *coordinates* : coordonnées) et la langue de rédaction du { tweet } (*lang*).

– L'évaluation politico-morale du contenu du { tweet }. Cette catégorie contient quatre champs : *possibly_sensitive*, { valeur booléenne } qui indique si le contenu du { tweet } est possiblement offensant, typiquement le cas d'images pornographiques ; *withheld_copyright*, { valeur booléenne } qui indique si le contenu du { tweet } tombe sous le coup de la loi états-unienne sur le copyright ; *withheld_in_countries*, un code géographique de deux lettres qui indique dans quel pays le contenu du { tweet } enfreint des lois sur le copyright ; *withheld_scope* peut avoir comme valeur *tweet* ou *user* et sert à préciser si c'est le { tweet } ou son auteur qui enfreignent la loi.

b Une maximisation du couple discrétisation – manipulabilité

Plusieurs enseignements sont à tirer de cet exemple. Tout d'abord, ce que nous appelions la forme « technique » d'un { tweet } est en fait sa représentation dans un { langage } précis, le { JSON } et ne s'oppose en rien à des considérations culturelles. Dans l'ensemble des données qui composent un { tweet }, il y a une prise en compte du contexte spatial – localisation de la publication –, linguistique – langue de publication– et même juridique, selon les législations locales. Second point : la très forte présence de champs décrivant les manipulations passées du { tweet }. Un { tweet } est donc défini en partie par les relations qu'il entretient avec d'autres textes. Il porte les marques de sa circulation : nombre de retweets, citation, réponses aux { tweet } originel, etc.

API,
application,
calcul,
code,
développeur,
discrétisation,
documentation,
forme-texte,
HTML,
littératie,
objet,
programme,
tweet,
widget

Voir glossaire tome II, p. 3-25

Ce qui constitue un paradoxe quant aux rapports entre { discrétisation } et manipulabilité. Au premier abord, cette analyse tend à confirmer notre hypothèse, dans la continuité d'Illich et de Bruno Bachimont, que l'abstraction du texte s'accompagne d'un développement des outils de manipulation de ce texte et que la { discrétisation } propre à l'informatique renforce ce couplage entre abstraction et manipulabilité. En revanche, Bruno Bachimont défend l'idée que le signe informatique a ceci de particulier qu'il est, certes, manipulable, mais qu'il ne porte pas les traces de ses manipulations[617]. Pourquoi donc cette mémoire de la circulation dans la { documentation } de l'{ API } ?

À ce paradoxe, plusieurs solutions s'offrent à nous. Tout d'abord, même si nous analysons ici une représentation informatique d'un { tweet }, nous sommes à un niveau d'écriture bien éloigné de celui dont parle Bruno Bachimont. Mais cette solution est insuffisante, car si le { calcul } est bien le « noème[618] » du numérique, il devrait pouvoir se retrouver à tous les niveaux des médias informatisés. Il faut en fait renverser la question : si les marques de manipulation n'apparaissent **théoriquement** pas, elles peuvent malgré tout être conservées, mémorisées, au prix d'un lourd expédient technique pour pallier à ce manque constitutif[619] du { calcul }.

Pourquoi alors, dans ce contexte précis, les concepteurs de l'{ API } ont choisi de mettre en avant ces marques de manipulation ? Il faut pour répondre à cette question repartir de la dimension pratique de la { documentation } d'une { API }. À quoi sert–elle ? À expliquer à des { développeurs } extérieurs à Twitter la structure des données de base de la plateforme : un utilisateur, un { tweet }, etc. Ces { développeurs } pourront ainsi, lorsqu'ils utiliseront l'{ API } pour récupérer des données auprès de Twitter et les utiliser pour leur propre { application }, comprendre le { code } qui leur est renvoyé par la plateforme. L'enjeu est pédagogique mais aussi pratique : plus la { documentation } sera précise, plus l'{ objet } sera défini par plusieurs paramètres, plus il sera réexploitable dans d'autres contextes. Par exemple, la présence du nombre de retweet dans la { documentation } de l'{ API } donne la possibilité d'écrire un { programme } qui ne récupère que les { tweets } ayant été retweeté plus de x fois. Et on peut imaginer croiser ce critère avec les coordonnées géographiques des { tweets } ou la présence d'un mot particulier dans le { tweet }, etc.

617 « Le propre du calcul est que le contenu ne porte pas sur lui les traces de sa manipulation [...] ». Bachimont, Bruno. *Op. cit.*, 2010, p. 158.

618 Bachimont, Bruno. *Idem*, p. 156-158.

619 Qu'on pense aux historiques des modifications, les journaux consignant les changements lors de mises à jour ou de *bugs* de logiciels, etc.

Une granularité maximale des { tweets } permet donc de maximiser en retour le nombre de combinaisons possibles et la réutilisation des données dans d'autres contextes que Twitter. Ces pages de { documentation } des { API } montrent en dernière analyse une radicalisation de la tendance relevée par Bruno Bachimont. Non seulement le { calcul } consacre la { discrétisation } et la manipulabilité des { formes-textes }, mais ce couplage est intensifié : plus une { forme-texte } est discrétisable, plus elle peut être manipulée et recomposée. Plus on peut remonter à des unités élémentaires (au-« deçà » du { tweet } : la liste de ses propriétés), plus on peut manipuler ces unités et personnaliser leur utilisation. Par « personnaliser leur utilisation », il faut entendre : écrire des { applications } utilisant les données issues de Twitter et donc replacer ces données dans de nouveaux contextes, leur donner un sens en les manipulant et en les recontextualisant.

c Granularité du code : une inflation scripturaire au service de la manipulabilité

En revanche, un niveau maximal de { discrétisation } d'une forme sémiotique a des conséquences sur les modes de représentation de cette forme et sur la { littératie } nécessaire pour la produire. En refaisant le chemin des { *widgets* } à la { documentation } de l'{ API }, on se rend en effet compte que les boutons et autres { tweets } ancrés sont finalement des formes assez imprécises, au sens où elles ne donnent pas accès à un haut degré de granularité. Ce sont des formes usinées, relativement généralistes, qui sont difficilement personnalisables. Mais en retour elles sont extrêmement simples à reproduire, puisqu'il suffit de copier / coller du { code } { HTML }. Au contraire, la { documentation } de l'{ API } permet d'accéder à un degré de précision très fin, mais où règne aussi une forme d'indétermination. Elle ne permet pas de créer aussi facilement des { formes-textes }, mais les possibilités de manipulation sont bien plus grandes. Entre le { code } { HTML } permettant d'ancrer un { tweet } dans une page et la page décrivant la composition informatique d'un { tweet }, l'{ objet } représenté est d'un point de vue logique le même : un { tweet }.

API,
application,
code,
discrétisation,
forme-texte,
HTML,
JSON,
requête,
standardisation,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

Ce qui se joue entre ces deux modes de représentation, c'est une variation dans le degré de { discrétisation } du { tweet }. Mais ce degré de { discrétisation } est aussi une variation dans le degré de { standardisation } de la { forme-texte } – du plus standard au plus personnalisé – ; dans le degré de compétence requise – du simple copier / coller à la { requête } { API } *via* une { application } authentifiée – et enfin dans l'économie spatiale de l'écriture : le { *widget* } est une forme sémiotiquement ramassée alors qu'un même { tweet } représentée en { JSON } prend une place considérable.

Pour s'en convaincre, voici une { *timeline* } ancrée représentée respectivement comme { *widget* }, comme { code } { HTML } et enfin dans la console[620] de l'{ API } de Twitter.

Fig. 30. Représentation d'une *timeline* sous forme de *widget*.
Capture d'écran du site www.bnf.fr/fr/acc/x.accueil.html (détail). Réalisée le 17 juillet 2015.

620 La « console » est un outil mis à disposition par Twitter qui permet de visualiser et de tester les requêtes formulées au serveur *via* l'API et qui permet donc de récupérer des données de Twitter.

Fig. 31. Représentation d'une *timeline* au format HTML. Extraits du code source de la page www.bnf.fr/fr/acc/x.accueil.html (détail). Captures réalisées le 17 juillet 2015.

```
[
  {
    "created_at": "Fri Aug 26 09:17:42 +0000 2016",
    "id": 769101504452161500,
    "id_str": "769101504452161537",
    "text": "RT @ParisGamesWeek: #SaveTheDate : Du 31/08 au
02/09, @laBnF met à l'honneur le #jeuvidéo ! Plus d'infos sur
> https://t.co/snfr4StNXx",
    "truncated": false,
    "entities": {
      "hashtags": [
        {
          "text": "SaveTheDate",
          "indices": [
            20,
            32
          ]
        },
        {
          "text": "jeuvidéo",
          "indices": [
            80,
            89
          ]
        }
      ],
      "symbols": [],
      "user_mentions": [
        {
          "screen_name": "ParisGamesWeek",
          "name": "Paris Games Week",
          "id": 180730848,
          "id_str": "180730848",
          "indices": [
            3,
            18
          ]
        },
```


Fig. 32. Représentation d'une *timeline* au format JSON. Capture d'écran du site https://dev.twitter.com/rest/tools/console (détail). Capture réalisée le 26 août 2016[621].

621 Dans la constitution initiale de notre corpus, nous avions oublié de faire une capture d'écran de la console Twitter. C'est pourquoi cette capture d'écran a été faite un an après les autres. Le tweet représenté n'est donc pas le tweet de la capture de gauche, mais la structure est la même.

La différence entre ces façons de représenter un { tweet } réside dans le degré de manipulabilité et d'abstraction de ces formes, dans les différentes cultures du texte mobilisées et dans la masse textuelle affichée. La figure 30 est relativement ramassée, elle permet de visualiser dans une fenêtre en inclusion dans une page les derniers { tweets } de la Bibliothèque Nationale de France. La figure 31 est un extrait du { code source } de la page d'accueil du site de la BNF. À chaque { balise } `<li>` correspond un { tweet }. En cliquant sur la flèche grisée de droite, on peut « déplier » la { balise } et voir toute l'arborescence du { tweet }. Enfin, la figure 32 est un { tweet } de la BNF récupéré en faisant une { requête } { API } à Twitter. Le résultat est en { format } { JSON }. Ce qu'on voit n'est qu'une petite partie du { tweet } : son texte principal, qui contient deux hashtags (#SaveTheDate et #jeuxvidéo) ainsi que le nom d'un utilisateur de Twitter, « ParisGameWeek ». Ces deux dernières formes de présentation ({ code source } et console) se caractérisent par une nette inflation scripturaire. Alors que la { petite forme } fait montre d'une grande économie visuelle, le { code source } ou { JSON } de la même forme prend une place très importante, car toutes les informations d'un { tweet } sont écrites dans leur intégralité[622]. Cette exhaustivité fait appel à une culture du texte plus spécialisée, mais permet également une plus grande manipulabilité des { tweets } : on sait précisément quelles informations sont présentes, on peut donc en choisir certaines, pas d'autres… Cette utilisation du { code } est en fait presque plus proche de l'artisanat que de l'industrie : on peut manipuler un { tweet } et ses différentes composantes de manière très fine, alors que le { *widget* } n'autorise que certaines adaptations mineures : couleur de fond, nombre de { tweets } affichés, etc. Plus la forme est compacte, plus elle est pré-fabriquée et peu maniable. Elle n'en est par contre que plus circulante, car plus lisible en ceci qu'elle requiert une culture moins spécialisée. Une forme plus étendue, comme l'{ HTML } ou le { JSON }, demande des savoir-lire spécialisés mais arrive à un niveau de { discrétisation } qui permet une plus grande manipulabilité des { tweets }.

API,
balise,
code,
code source,
discrétisation,
format,
HTML,
JSON,
petites formes,
requête,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

622 Même si, par souci de lisibilité, la mise en page permet de « ramasser » l'écriture. L'indentation des balises, l'équivalence entre une balise et un tweet permet de rendre cette forte discrétisation lisible, sans sacrifier à la granularité fine permise par la calculabilité des formes.

API,
calcul,
discret,
discrétisation,
forme-texte,
objet,
tweet,
widget

Voir glossaire tome II, p. 3-25

L'hypothèse de départ de ce chapitre est donc finalement confirmée : les { API } web de Facebook et Twitter participent à l'établissement d'une culture anthologique du texte. Elles y participent en propageant une conception modulaire du texte de réseau : une page web est dissociable en modules remobilisables et personnalisables selon les contextes d'utilisation. Les { *widgets* } sont ces modules, dont l'{ API } structure l'accès. La première partie de ce chapitre a montré comment cette conception modulaire du texte reprend la métaphore ancienne du texte comme ancrage et les modalités précises selon lesquelles les { formes-textes } s'ancrent dans une page web. La seconde partie a permis de faire le lien entre cette modularité, la { discrétisation } du { calcul } propre à l'écriture informatique et la manipulation des formes textuelles. Nous avons notamment montré que les { API } web contemporaines s'ancrent dans un double héritage. Dans les formes qu'elles permettent de créer, elles prolongent l'essor des outils de manipulation du texte liés à son abstraction. Par leur dimension calculatoire et informatique, elles renforcent ces moyens de manipulation par une forte { discrétisation } des textes mis en circulation, { discrétisation } couplée à une éditorialisation : une recomposition des éléments { discrets } qui produit une forme lisible. Cette { discrétisation } / éditorialisation se joue à deux niveaux : d'un côté les { *widgets* } sont des formes { discrètes } et donc manipulables, circulantes. De l'autre, elles sont à leur tour discrétisables, découpables en unités plus élémentaires. La troisième partie a démontré que les { API } web contemporaines intensifient le couple { discrétisation } / manipulation : plus une { forme-texte } est { discrète }, plus elle est manipulable. C'est pourquoi le découpage du { tweet } va très loin dans le niveau de détail. Un tel découpage se signale par sa grande manipulabilité. On peut alors organiser plus librement les différents élements constitutifs d'un { objet } informatique – en l'occurrence un { tweet } – même si ce gain en manipulabilité se paye par une forte inflation scripturaire qui demande des savoir-lire plus spécialisés. D'où l'intérêt d'une interrogation sur le système de valeurs, tant économiques que symboliques, qui organise cette conception du texte et cette quête d'une { discrétisation } maximale.

ÉQUIPER, FAIRE DÉSIRER, VALORISER, STANDARDISER LA CIRCULATION. API WEB ET « INDUSTRIE DES PASSAGES »

IV

L'approche sémiotique adoptée dans le chapitre précédent a montré que les { API } mettent en circulation une pensée modulaire du texte, fondée sur une { discrétisation } de ses éléments afin d'en maximiser les manipulations possibles. Mais à quoi sert cette pensée modulaire ? Ou plutôt qui et quoi sert-elle ? Attendu que les { API } sont des outils liés à une industrie de l'écriture (l'informatique contemporaine), les { formes-textes } permises par l'{ API } s'insèrent dans un système économique plus large. Quel est donc l'économie des { API } ? Comment cette écriture modulaire s'industrialise-t-elle ?

API, discrétisation, forme-texte, standardisation

Voir glossaire tome II, p. 3-25

Notre hypothèse est que les { API } sont des éléments essentiels d'une « industrie des passages[623] ». Il faut que les textes circulent et certaines entreprises tirent parti de cette circulation en l'équipant (instrumentation), en la valorisant (instrumentalisation) et en contrôlant l'aspect des formes sémiotiques la permettant ({ standardisation }).

623 Jeanneret, Yves. *Op. cit.*, 2014, p. 644, 670, 681.

API, application, développeur, forme-texte, interopérabilité, petites formes, standardisation

Voir glossaire tome II, p. 3-25

Pour montrer cela, nous commençons par repartir des principes des { API } web vus dans les chapitres précédents : délégation, appel à l'écriture, mise à disposition de blocs précodés. Nous lions ces trois principes au triptyque instrumentation ; instrumentalisation ; { standardisation } (IV. Introduction), ce qui nous permet de poser les jalons des trois parties suivantes. Premièrement, les { API } équipent la circulation en déléguant l'écriture (IV. 1.). L'instrumentation ne suffit toutefois pas : encore faut-il éveiller le désir d'écrire (IV. 2.). Ce passage par la question du désir nous amène à considérer que c'est le clic de l'internaute sur les { formes-textes } qui est rendu désirable, car présenté comme un acte de sociabilité. Comment passe-t on d'une action technique (le clic) à une action sociale (le partage) ? En d'autres termes, comment les { API } sont-elles instrumentalisées (IV. 3.) ? Nous faisons l'hypothèse que cette instrumentalisation se joue autour de l'{ interopérabilité }, les { API } étant alors utilisées comme vecteurs privilégiés de la circulation des textes de réseau. Mais cette circulation augmente nécessairement les phénomènes d'altération de ces textes, attendu que toute circulation est altération. Et si Facebook et Twitter tirent profit de la circulation, l'altération peut représenter un danger en ceci qu'elle compromet la lisibilité des textes. D'où l'idée que la { standardisation } des { formes-textes } est un antidote à leur altération (IV. 4.). Cela nous amènera à décrire précisément en quoi les { API } web sont le lieu d'une « pratique lettrée[624] », c'est-à-dire le lieu de définition d'une forme canonique du texte, valorisant certains acteurs, en excluant d'autres. Les bases de notre dernier chapitre seront ainsi posées : la fonction éditoriale des { API } web a des implications sémio-politiques, au sens où, dans la mise en forme du texte, sont choisies les voix du texte mises en exergue et celles qui sont ignorées ou invisibilisées.

624 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 12.

INTRODUCTION
UN « CAPITALISME TEXTUEL CONTEMPORAIN » ?

Le présent chapitre a pour ambition de rendre compte de la façon dont les { API } web participent au capitalisme contemporain. Le cadre général de réflexion est donc à la fois sémiotique et économique : comment des outils d'écriture comme les { API }, ainsi que les { petites formes } qu'ils permettent de produire, s'insèrent dans un système de production de valeur, tant économique que symbolique ? Pour répondre à cette question, il faut se doter de quelques outils théoriques à même de nous aider à penser le lien entre formes sémiotiques et économie. Repartons donc de quelques caractéristiques des { API } web, afin de voir quelles seront les théories pertinentes pour nous.

A Des trois logiques de l'API

Premièrement, une { API } repose sur un principe de délégation d'écriture : on donne – le plus souvent gratuitement – les moyens d'écrire des { applications }, des sites web, selon l'{ API } utilisée. Facebook et Twitter ont fondé une partie de leur développement économique sur ce principe : donner la main à des { développeurs } externes qui vont par leur activité enrichir la plateforme les ayant équipés ainsi que disséminer des formes qui permettent d'écrire des pages web. C'est ce que les théoriciens du « capitalisme cognitif » appellent des « externalités positives » : des activités éparses et infimes (« aimer », « partager », cliquer…) sont équipées puis confiées à des acteurs externes, qu'une plateforme va centraliser, calculer, exploiter. Yann Moulier-Boutang appelle cela la « pollinisation[625] » : l'ensemble des micro-activités sociales que certains acteurs industriels – Facebook, Twitter, Google et consorts – captent en partie et exploitent. Cette activité équipée gratuitement mais qui est – en retour – exploitée par les plateformes fournissant l'équipement, fait des internautes des « abeilles » du capitalisme cognitif[626]. Le marché est le suivant : nous vous donnons gratuitement les moyens d'écrire, de lire,

625 Moulier-Boutang, Yann. La pollinisation humaine à l'ère numérique. *Labyrinthe*. 2014, nº 40, p. 125-128.

626 Collomb, Cléo. *Google, une entreprise de mise en foule de la multitude* [manuscrit en ligne]. 2015. [Mis en ligne 11 janvier 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-01253751/document. *Working paper or preprint*.

de mener tout un lot d'activités culturelles ; en revanche votre activité est mesurée, tracée et fait l'objet d'une exploitation[627]. On peut donc penser que la délégation propre aux { API } est une façon à la fois d'équiper et d'exploiter l'activité pollinisatrice de la multitude[628].

III 2 B b
des formes faites pour motiver l'écrituree

Voir p. 300
la notion d'appel à l'écriture

API, développeur, forme-texte, industrialisation, polychrésie, *widget*

Voir glossaire tome II, p. 3-25

Deuxièmement, les { API } produisent des formes faites pour motiver l'écriture. On peut donc supposer qu'elles visent à provoquer un certain désir. Que serait un « capitalisme cognitif » équipant l'activité scripturaire sans désir de cette activité ? Il ne suffit donc pas d'équiper l'écriture pour ensuite l'exploiter, il faut rendre cette écriture désirable. C'est ce que l'économiste Frédéric Lordon appelle une *epithumè* : un « régime de désir[629] » sur lequel se construit le capitalisme. Lordon se pose la question suivante : qu'est-ce qui pousse certaines personnes à mettre leur force de travail au service d'autres ? Sur quoi se construit cet « enrôlement » ? Pour y répondre, l'auteur considère qu'il faut articuler économie et anthropologie, cette dernière étant une théorie des affects prenant part dans les décisions économiques. Cette anthropologie, Lordon la tire de la théorie des affects chez Spinoza. Son argument peut être résumé ainsi : tout être humain possède un *conatus*, un désir de mener les activités qui lui permettent de « [...] poursuivre ses objets de désir[630] ». Le capitalisme contemporain est fondé sur l'enrôlement de ces désirs d'agir individuels, au service d'un « désir-maître » qui est le *conatus* de l'employeur. Dans notre cas, on peut se demander par quels moyens une { API } construit la désirabilité de ses usages, enrôlant ainsi des *conatus* extérieurs (ceux des { développeurs } et internautes).

627 Les thèses du *digital labor* sont assez proches de celles du capitalisme cognitif, même si les tenants du *digital labor* maintiennent opératoire – au moins pour l'interroger – la notion classique de travail telle que pensée depuis Marx. Voir Cardon, Dominique et Casilli, Antonio A. *Qu'est-ce que le digital labor ?* Bry-sur-Marne : Institut National de l'Audiovisuel, 2015.

628 Hardt, Michael et Negri, Antonio. *Empire*. Paris : Exils, 2000.

629 Lordon, Frédéric. *Op. cit.*, p. 73.

630 Lordon, Frédéric. *Idem*, p. 17.

Troisièmement, les { API } permettent de structurer l'accès à des données ou à des ressources : l'internaute ou le { développeur } ont accès à des données, à condition de respecter un certain nombre de contraintes et d'impératifs[631]. Pour les plateformes publiant des { API }, cela permet de mettre en circulation ces données à travers des { formes-textes } comme les { *widgets* }. Il s'agit toujours d'un équipement et d'une délégation, mais cette fois envisagés comme profitant à la circulation des textes. Or, cette fonction des { API } est essentielle dans le contexte contemporain où la circulation est devenue productrice de valeur tant économique que symbolique[632]. Facebook et Twitter, parce qu'elles équipent la circulation et fondent leur modèle économique sur elle, s'insèrent dans ce contexte économique plus global. Ce sont des « industries médiatisantes[633] », au sens où elles équipent et tirent parti de la circulation des textes sur les réseaux. Nous entendons par circulation – ou trivialité selon les termes d'Yves Jeanneret – le fait que les idées dans une société donnée ne naissent pas de nulle part, mais passent par des médiations matérielles et sémiotiques (des textes, des images…). Ces idées sont par conséquent en permanence modifiées, réappropriées dans cette circulation, ce qu'Yves Jeanneret appelle la « { polychrésie[634] } » des êtres culturels. Sur ces bases, il fait l'hypothèse d'une « { industrialisation } de la trivialité[635] » : la circulation des textes est aujourd'hui devenue une source de pouvoir économique ou politique et fait l'objet d'une exploitation et d'une rationalisation. D'où des industries médiatisantes comme Google, Facebook ou Twitter qui participent à cette { industrialisation } de la trivialité.

631 Fauré, Christian. La pharmacologie des API. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *op. cit.*, p. 244-245.

632 Lee, Benjamin et LiPuma, Edward. Cultures of Circulation: The Imaginations of Modernity. *Public Culture*. 2002, vol. 14, nº 1, p. 191-213.

633 Jeanneret, Yves. *Op. cit.*, 2014, p. 10.

634 La polychrésie est définie par Yves Jeanneret comme la propriété des objets communicationnels à « […] faire l'objet de constantes réappropriations et à être ainsi [pris] dans un large spectre de logiques sociales différentes ». Jeanneret, Yves. *Op. cit.*, 2008, p. 84.

635 Jeanneret, Yves. *Idem*, p. 237-240.

B L'hypothèse d'une industrialisation de la trivialité

API, industrialisation, petites formes

Voir glossaire tome II, p. 3-25

L'avantage de cette hypothèse est qu'elle pose comme centrale la question des médiations sémiotiques. C'est pourquoi, au vu de nos objets et de notre problématique – les { API } comme outils permettant l'écriture de { petites formes } – nous considérons Facebook et Twitter comme des « industries médiatisantes », qui participent par leurs { API } à un mouvement d'{ industrialisation } des écritures[636]. Les { API } seraient en quelque sorte la cheville ouvrière de cette { industrialisation }, un dispositif particulièrement représentatif de cette dynamique.

Par { industrialisation }, nous entendons à la suite de Pierre Mœglin un processus qui regroupe au moins trois phénomènes, ou trois logiques : un « [...] recours à des systèmes techniques faisant, partiellement ou totalement, l'économie de la force et du temps de travail humain [...] » ; des « [...] méthodes d'organisation et de gestion [...] » qui permettent de faciliter et d'optimiser le fonctionnement des systèmes techniques ; enfin l'élaboration d'un « [...] état d'esprit enterpreunarial [...][637] » par lequel les deux processus décrits plus haut sont non seulement acceptés mais encouragés. Ces trois logiques (technologisation, rationnalisation, idéologisation[638]) définissent le processus d'{ industrialisation }. Dans notre cas, l'{ industrialisation } des écritures concerne donc le fait que les { API } outillent l'écriture des pages web (technologisation), qu'elles définissent une forme canonique du texte pour en permettre la reproduction (rationnalisation) et que cette reproduction matérielle est valorisée comme produisant une valeur économique ou symbolique (idéologisation). Elles servent ainsi une « industrie des passages[639] » contemporaine.

Non pas que ce phénomène d'{ industrialisation } des écritures soit nouveau – il existe dès l'imprimerie – mais les développements techniques et économiques contemporains lui donnent une nouvelle ampleur et de nouveaux enjeux.

636 Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 168.

637 Les trois citations ci-dessus sont tirées de Mœglin, Pierre. À la recherche de l'industrialisation du tutorat à distance. *Distances et savoirs*. 2005, vol. 3, nº 2, p. 251.

638 Yves Jeanneret propose à la suite de Mœglin le triptyque « instrumentation, standardisation, idéologisation ». Les différences entre les deux formulations sont assez minimes, mais ce chapitre utilisera principalement la terminologie d'Yves Jeanneret. Voir Jeanneret, Yves. *Op. cit.*, 2014, p. 10-11.

639 Jeanneret, Yves. *Idem*, p. 644.

Il convient ici de faire une remarque sur les liens entre circulation et éditorialisation. Si, comme nous cherchons à le démontrer, les { API } sont des vecteurs essentiels de mise en circulation des textes de réseau, cette circulation est toujours un problème d'éditorialisation. Attendu que toute circulation est altération[640] et qu'« [...] il n'est pas de texte qui, pour advenir aux yeux du lecteur, puisse se départir de sa livrée graphique[641] », alors il n'y a pas de circulation des textes qui ne soit pas une éditorialisation, c'est-à-dire une construction de la lisibilité du texte selon des conceptions historiques et culturelles. L'éditorialisation est bien en ce sens une altération : un texte ne « circule » pas au sens où il se réplique à l'identique d'un contexte de publication à un autre.

640 Jeanneret, Yves. *Op. cit.*, 2008, p. 87.

641 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 138.

En revanche, les pratiques éditoriales fixent un seuil de répétabilité[642] du texte, qui fait que, dans une culture donnée, on considère que telle occurrence d'un texte n'est pas la même que telle autre, ou au contraire que c'est le « même » texte que l'on lit, qu'il soit édité au Seuil ou chez Robert Laffont par exemple. Cette affirmation est fausse d'un point de vue purement phénoménologique[643], elle n'en est pas moins vraie si l'on se situe au niveau du seuil de répétabilité acceptable d'un texte. Dans notre cas, quand nous faisons l'hypothèse que les { API } encouragent la circulation des textes de réseau, nous faisons l'hypothèse qu'elles participent à fixer un seuil très bas de répétabilité des { formes-textes }. C'est le « même » { tweet } ou la même publication qui se « réplique » de page en page : telle serait l'idéologie du texte promue par les { API } web. Or, c'est que nous apprend Christian Jacob, il ne s'agit pas ici d'une négation de l'éditorialisation. Bien au contraire : une grande partie du travail éditorial consiste à fixer ces bornes du répétable, à assurer la reproduction de ce qu'on estime être la même chose. Il s'agit donc, paradoxe de toute médiation efficace, d'un travail éditorial qui tait son nom, qui s'invisibilise pour permettre aux textes de réseau de circuler et de s'éditorialiser au fil de cette circulation. Ainsi, ce que nous analysons dans cette partie, c'est l'idéologie contemporaine du texte qui consiste à fixer un seuil de répétabilité bas des { formes-textes } de réseau, au prix d'un lourd travail éditorial et au service d'une { industrialisation } de la circulation.

API, forme-texte, industrialisation, interopérabilité, standardisation, tweet

Voir glossaire tome II, p. 3-25

IV 1
d'un travail éditorial qui tait son nom…
Voir p. 343

642 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 19.

643 C'est tout l'intérêt de la théorie de « l'énonciation éditoriale » et plus largement de la bibliographie matérielle, que de permettre l'analyse des différentes versions d'un texte en les considérant comme des textes disctincts, aux enjeux et aux attendus différents.

Fort de ce cadre théorique, nous pouvons reformuler la question d'ouverture de ce chapitre : comment les { API } participent à une { industrialisation } de la trivialité, c'est-à-dire à une instrumentation, à une instrumentalisation et à une { standardisation } de la circulation ? Nous répondrons en quatre temps, autour de quatre hypothèses qui sont autant de reformulations des dynamiques d'{ industrialisation } (instrumentation/instrumentalisation/{ standardisation }) selon trois caractéristiques des { API } web (délégation/{ interopérabilité }/altération).

i) Les { API } participent à une { industrialisation } de la trivialité en **équipant et déléguant l'écriture** de façon à favoriser la circulation des { formes-textes } (IV. 1.) ;

ii) Les { API } participent à une { industrialisation } de la trivialité en **suscitant le désir de la circulation** (IV. 2) ;

iii) Les { API } participent à une { industrialisation } de la trivialité en **instrumentalisant** les { API } par le biais de l'**{ interopérabilité }** (IV. 3) ;

iiii) Les { API } participent à une { industrialisation } de la trivialité en **standardisant** les { formes-textes } contre les risques d'**altération** sémiotique (IV. 4).

1 UNE LOGIQUE DE DÉLÉGATION : DONNER LA MAIN ET ÉQUIPER LA CIRCULATION

La présente partie s'attache au thème de la délégation et plus précisément à la délégation scripturaire. Nous entendons par là le fait de donner la main, à travers des { logiciels } ou plus largement des { architextes }, à un utilisateur ({ développeur } ou internaute) afin qu'il puisse lui-même écrire avec et/ou pour une machine. Les { API } web fonctionnent sur ce principe de délégation : elles fournissent les moyens à des { développeurs } extérieurs d'utiliser des jeux de données et des formes préconçues pour concevoir leur propre { programme } ou leur propre page web. C'est un modèle à la fois technique et économique : l'{ API } est une structure d'accès à des données et en même temps elle permet d'extérioriser à moindre frais le développement d'{ applications } utilisant les données propriétaires de certains acteurs industriels, dans notre cas Facebook et Twitter.

API, application, architexte, code, développeur, forme-texte, HTML, logiciel, programme, *widget*

Voir glossaire tome II, p. 3-25

Devant ce constat, notre hypothèse est que si, à travers cette délégation se joue un équipement de la circulation, cet équipement n'est pas neutre : il participe à valoriser la circulation des { formes-textes }. En d'autres termes, une { API } ne fait pas que donner la main : elle la guide aussi afin d'assurer une plus grande circulation des textes de réseau.

Comment les { *widgets* } et autres { formes-textes } produites par l'{ API } équipent et servent la circulation des textes de réseau ? Nous répondons à cette question en deux temps. D'abord en étudiant le cas des « cartes » Twitter et en montrant qu'il y a, par le { code } { HTML }, un travail éditorial de préécriture de la circulation (IV. 1. A). Les { développeurs } mettent en forme les textes en prévoyant leur circulation dans différents contextes éditoriaux et cette mise en forme tire parti de la { discrétisation } permise par l'informatique. Nous montrons ensuite que cette préécriture ne se limite pas au { code } { HTML } mais intègre tous les niveaux de l'écriture architextuelle (IV. 1. B) témoignant en dernière analyse d'une idéologisation de la circulation au sens de Pierre Mœglin. Nous aurons ainsi montré que les { API } servent à instituer un certain état d'esprit vis-à-vis du texte produit : il doit évidemment apparaître et être lu, mais il doit aussi – et surtout – circuler et se répliquer dans d'autres environnements techniques et éditoriaux.

1 A Les cartes Twitter : une préécriture de la circulation…

API, code source, petites formes, tweet

Voir glossaire tome II, p. 3-25

Qu'est-ce qui, dans la manière dont une { API } donne la main, sert une circulation des textes de réseau ? Une première réponse nous est donnée avec les cartes Twitter. Les cartes sont ces { petites formes } générées par Twitter qui permettent, lorsqu'un { tweet } contient un lien, de ne pas afficher seulement le texte du lien mais également le titre de la page à laquelle renvoie le lien, un chapeau introductif et éventuellement une photo illustrant ce lien.

Fig. 33. La carte Twitter : un exemple d'optimisation de la circulation. Capture d'écran de twitter.com (détail, profil personnel) réalisée le 18 avril 2014 à 00h05.

Dans cet exemple, l'utilisateur @etienne_c**** publie un { tweet } avec le texte : « J'apprends aussi que Raymond Aubrac est mort le 10 avril. » assorti d'un lien hypertexte issu du monde.fr. C'est la seule chose qui a été écrite par un humain. Tout le reste est généré automatiquement par Twitter. Ce qui nous intéresse est la partie inférieure du { tweet }, encadrée de violet. Un logo (celui du journal *Le Monde*) est accompagné des mots « Le Monde ». Nous avons vu par ailleurs comment fonctionne sémiotiquement cette carte, comment elle puise dans une mémoire des formes et comment les cadres articulent différents niveaux d'énonciation. Dans le cadre de l'hypothèse générale de cette partie, nous allons désormais nous demander comment cette carte est techniquement produite et ce que cela nous dit sur sa propension à circuler. Sur quels éléments Twitter s'appuie-t-il pour la générer ? Comment sait-il qu'il faut afficher tel texte,

III 1 B c
comment elle puise dans une mémoire des formes…
Voir p. 265 pour l'énonciation dédoublée.

tel titre ? Pour répondre à ces questions et vérifier notre hypothèse, il faut sortir de twitter.com et aller voir du côté du { code source } du site vers lequel renvoie le lien, c'est-à-dire ici le monde.fr.

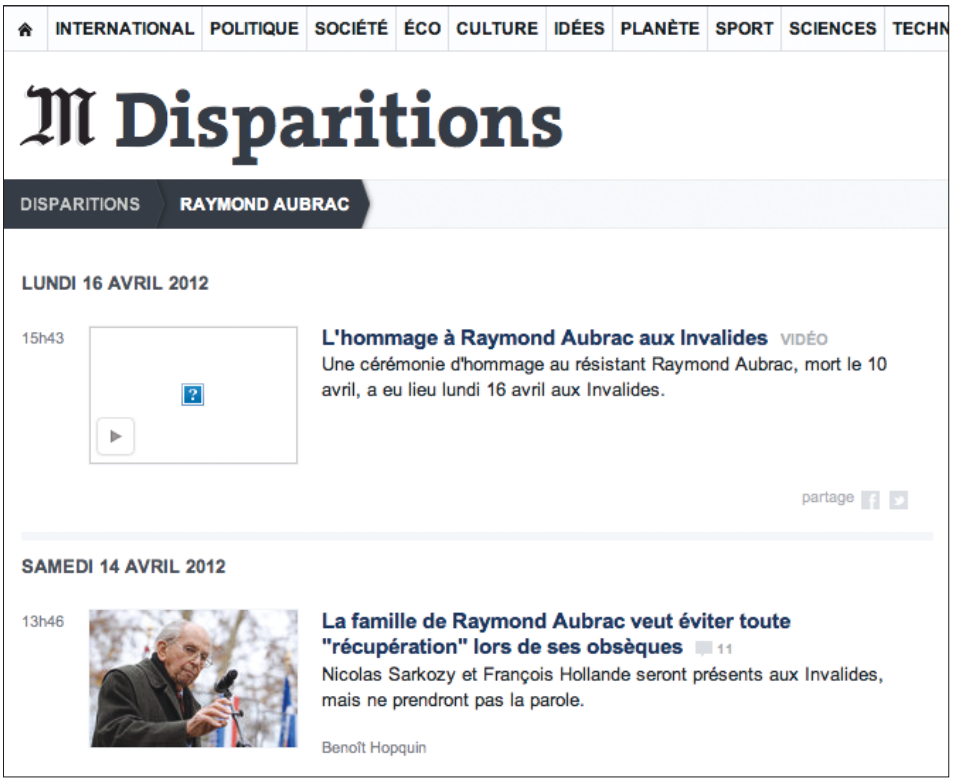


Fig. 34. Une préécriture de la circulation (1) : une page du *Monde*…
Capture de la page lemonde.fr/raymond-aubrac (détail), réalisée le 18 avril 2014.

```
<meta property="og:description" content="Raymond Aubrac - Découvrez gratuitement tous les articles, les
vidéos et les infographies de la rubrique Raymond Aubrac sur Le Monde.fr.">
```

Fig. 35. Une préécriture de la circulation (2) : …et le code correspondant. Capture d'écran du code source de la page lemonde.fr/raymond-aubrac (détail), réalisée le 18 avril 2014.

III 3 B a
la discrétisation des éléments sémiotiques…
Voir p. 316

En figure 34, la page du monde.fr vers laquelle renvoie le { tweet } d'@etienne_c***. En figure 35, un extrait du { code source } de cette page au { format } { HTML }. Nous revoyons la { discrétisation } des éléments sémiotiques que nous avions déjà notée à propos des lecteurs YouTube sur Twitter : à chaque ligne de { code } correspond un élément de la carte Twitter :

API,
balise,
code,
code source,
développeur,
discrétisation,
format,
forme-texte,
HTML,
industrialisation,
métadonnée,
programmation,
standardisation,
tweet

Voir glossaire tome II, p. 3-25

Fig. 36. Correspondance entre formes et codes source (1) : Twitter…
Capture d'écran de twitter.com (détail, profil personnel), réalisée le 18 avril 2014 à 00h05.

```
<meta property="og:description" content="Raymond Aubrac - Découvrez gratuitement tous les articles, les
vidéos et les infographies de la rubrique Raymond Aubrac sur Le Monde.fr.">

<meta property="og:title" content="Raymond Aubrac : Toute l'actualité sur Le Monde.fr.">

<meta name="twitter:card" content="summary">
<meta name="twitter:site" content="@lemondefr">
```

Fig. 37. Correspondance entre formes et codes source (2) : … et *Le Monde*. Captures d'écran du code source de la page lemonde.fr/raymond-aubrac (détail), réalisée le 18 avril 2014.

Lorsqu'un lien est publié sur Twitter, la plateforme scanne les { métadonnées } de la page à laquelle renvoie le lien et vérifie si la { balise } { HTML } `<meta name = "twitter:card">` y est présente, ce qui indique à Twitter de créer une carte. Les { développeurs } ayant écrit la page lemonde.fr/raymond-aubrac ont donc prévu en amont que cette page serait republiée sur des sites comme Twitter. En d'autres termes, l'analyse du { code source } montre qu'il y a, à travers la { programmation }, l'exercice d'un travail éditorial qui anticipe et prépare la circulation des { formes-textes }. C'est ce que nous appellons une **préécriture de la circulation**[644] : une page

644 Étant entendu que la circulation est un avatar de l'éditorialisation : tout texte qui circule est donné à lire, différemment selon les contextes où il apparaît et selon les différentes entités impliquées dans la construction de sa lisibilité. Lorsque nous parlons d'une préécriture de la circulation, nous n'impliquons pas que le texte se réplique « à l'identique » partout où il apparaît. En revanche, nous soutenons qu'il y a aujourd'hui, à travers le code informatique, une action éditoriale qui vise à réaliser cet idéal d'un texte identique à lui-même malgré ses différentes instanciations.

web est écrite de façon à ce que soit facilitée sa reprise par des industries médiatisantes. Elle est écrite selon la syntaxe propre à ces industries, ici Twitter. Avec cette syntaxe, nous sommes à l'articulation précise entre deux logiques de l'{ industrialisation } de la trivialité : d'un côté une instrumentation (Twitter fournit à travers son { API } les moyens techniques de la circulation) et de l'autre une { standardisation }, puisque la syntaxe à respecter fait que la carte Twitter va être répliquée à l'identique — peu importe qui republie le lien.

1 B … intégrée dans toutes les couches des écrits de réseau

Cette préécriture est-elle pour autant limitée à Twitter ? À travers l'exemple d'un { tweet } du magazine *Wired* et de l'analyse du { code source } de l'article auquel renvoie le { tweet }, nous montrerons qu'elle s'étend à tous les acteurs industriels impliqués dans la circulation du texte (IV. 1. B. a) et qu'elle peut même intégrer les outils de production des textes de réseau (IV. 1. B. b). En cela, nous montrons que les { API } sont le maillon d'un outillage de la circulation des textes de réseau. Mais cet outillage est aussi une optimisation de cette circulation. C'est pourquoi les { API } web sont l'un des vecteurs d'une « idéologisation » de la circulation (IV. I. B. c), confirmant ainsi notre hypothèse de départ.

a La prise en compte de toute la chaîne de distribution du texte

Voici un autre exemple de carte Twitter, concernant cette fois un lien vers un article du magazine *Wired*, lien publié sur Twitter par le compte @itis_ul.

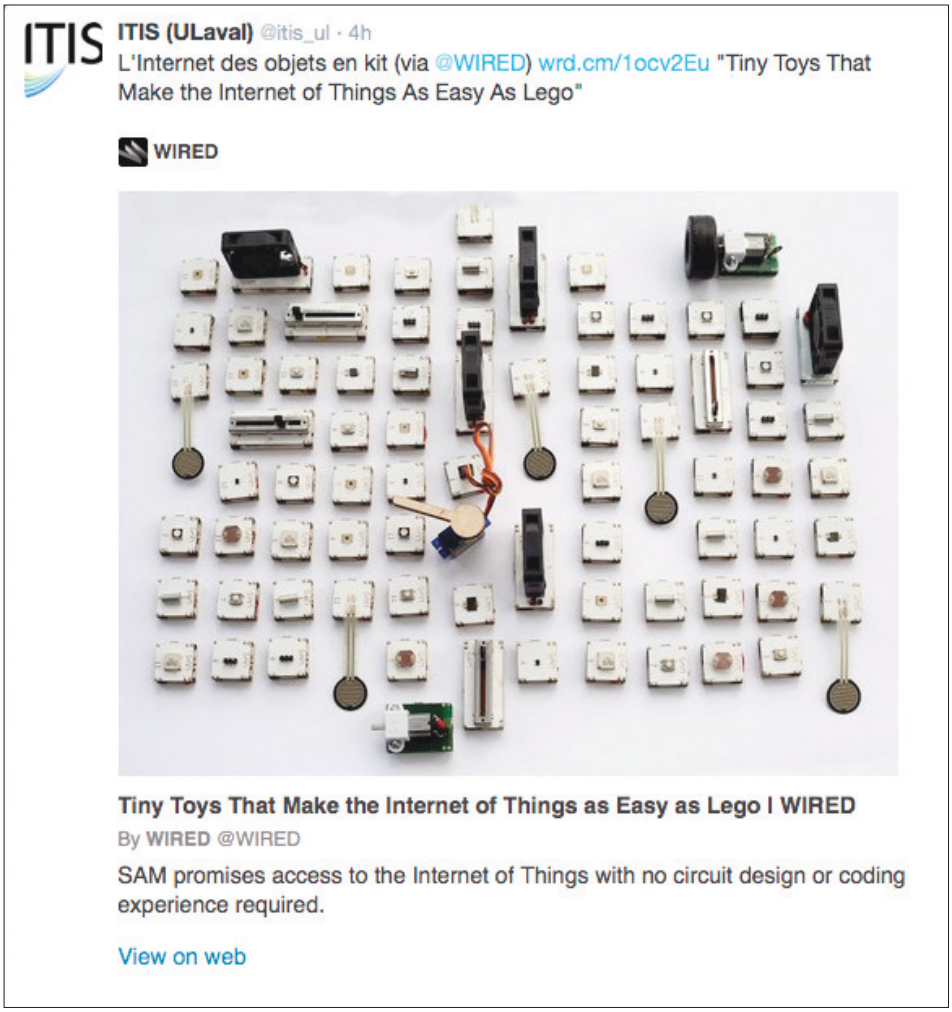


Fig. 38. Un article circulant sur les réseaux socionumériques sous forme de « carte ». Capture d'écran de twitter.com (détail, profil personnel) réalisée le 16 octobre 2014 à 19h37.

Fig. 39. Le code informatique qui permet cette circulation. Code source du site wired.com, détail. Capture réalisée le 16 octobre 2014 à 19h38.

Sur la figure 38, nous voyons la carte Twitter avec le lien vers l'article en question. La figure 39 est une capture du { code source } de la page de l'article sur wired.com. On retrouve des { balises } similaires à l'exemple précédent. Mais si l'on prête attention à la mise en page du { code }, on peut repérer des sortes de paragraphes, délimités par des { commentaires } (en vert : `<!-- Facebook -->` ; `<!-- WP_HEAD -->`). À chaque paragraphe correspond une industrie médiatisante : Facebook, Twitter, MS/Windows 8 (la boutique d'{ application } de Microsoft), Apple et Google. Dans chaque paragraphe, on trouve les mêmes informations, mais adaptées à la syntaxe spécifique de chacun de ces acteurs économiques. Cela permet à cet article et plus largement à l'{ application } mobile du journal *Wired*, d'apparaître de la même façon sur toutes les plateformes où le texte peut circuler : Facebook, Twitter, *app store*, etc[645]. Le ou les { développeurs } ayant écrit cette page ont donc intégré dans son { code source } les différentes modalités de ses rééditorialisations. Il y a là une anticipation de la trivialité qui passe par des pratiques d'écriture – la { programmation } – qui se déploient dans un contexte économique et industriel de valorisation de la circulation.

application, balise, code, code source, commentaire, développeur, programmation, système d'exploitation

Voir glossaire tome II, p. 3-25

C'est toute la chaîne de distribution et de diffusion de la presse qui est anticipée dans le { code source } de cet article de *Wired*. Parce qu'il est amené à être lu sur plusieurs supports (mobile, tablette…), qu'à chaque support correspond un certain { système d'exploitation } (Apple iOS, Android…), qu'il peut être également lu *via* des { applications } spécialisées, vendu sur des *app stores* qui eux aussi ont leur mot à dire sur les articles que publie

645 Nous reviendrons sur ce point dans les parties à venir.

Wired, anticiper la circulation d'un article nécessite de prendre en compte toute cette chaîne éditoriale dans le { code source } de la page. Dans ce contexte, le { code informatique } devient, *via* la syntaxe des { API } et les formes que ces dernières génèrent, un vecteur scripturaire de l'industrie des passages : les { développeurs } ayant écrit ces pages web avaient en vue la circulation de ces pages et l'optimisation de cette circulation.

API,
architexte,
balise,
CMS,
code,
code source,
commentaire,
développeur,
format,
HTML,
langage de programmation,
logiciel,
métadonnée,
plugin

Voir glossaire tome II, p. 3-25

b Une optimisation intégrée jusqu'aux outils d'écriture du Web

Cette optimisation peut concerner toutes les couches des écrits de réseau, comme le montre l'exemple suivant, tiré d'un article du site www.frenchweb.fr republié sur le compte Twitter éponyme.

Fig. 40. Un article du site frenchweb.fr republié sur Twitter. Capture d'écran du site twitter.com (profil personnel, détail). Capture réalisée le 16 octobre 2014 à 19h25.

Fig. 41. La circulation des textes intégrée à toutes les couches d'écriture du Web.
Code source du site frenchweb.fr (détail). Capture réalisée le 16 octobre 2014.

De prime abord, nous voici en face d'une carte Twitter comme les autres. En revanche, le { code source } de la page nous met sur la piste d'un nouvel élément. Si l'on observe là encore les { commentaires }, on peut y lire en ligne 41 `<!– WP HEAD […] –>` puis en ligne 43 `<!– This site is optimized with the Yoast WordPress SEO plugin v1.6.3 […] –>`. Les lignes 44 à 64 contiennent des { métadonnées } au { format } *Open Graph* qui permettent à des plateformes comme Twitter de générer des cartes et autres { petites formes }.

Ces deux lignes (41 et 43) indiquent que pour générer le { code } qui suit (lignes 44 à 64), un { logiciel } tiers a été utilisé : le { *plugin*[646] } Yoast pour Wordpress[647]. L'optimisation de la circulation intègre donc directement les outils d'écriture des écrits de réseau. Wordpress est en effet un *Content Management System* ({ CMS }) ou « système de gestion de contenus » : un { logiciel } facilitant l'écriture de pages web. C'est un cas exemplaire d'{ architexte } et donc de délégation de l'écriture[648] : un { CMS } permet l'écriture d'une page web, notamment en proposant de se passer de la maîtrise du { langage } { HTML }. Le { *plugin* } Yoast est un { logiciel } qui s'ajoute à Wordpress et prétend optimiser la gestion du référencement des pages écrites sous Wordpress par les différents moteurs de recherche.

Parmi ces optimisations, Yoast prend en charge la rédaction des { métadonnées } d'une page et donc des { balises } `<meta>` où se trouvent les

646 Un *plugin* est un logiciel complémentaire qui s'installe sur un autre logiciel pour y ajouter une fonctionnalité.

647 Il est d'ailleurs fort probable que ce code ait été généré automatiquement par ce *plugin* et non écrit directement par un humain, mais on ne peut pas l'affirmer à partir du seul code source de la page.

648 Jeanne-Perrier, Valérie. Des outils d'écriture aux pouvoirs exorbitants ? *Op. cit.*, p. 97-131.

informations nécessaires à la génération d'une carte Twitter[649]. Il apparaît donc que la circulation d'une page, ainsi que les « { petites formes } » de cette circulation (les cartes), font aujourd'hui l'objet d'une optimisation, au même titre que le référencement par les moteurs de recherche. Et cette optimisation passe par la mise au point d'outils – tels le { *plugin* } Yoast – qui standardise et participe à industrialiser la circulation des textes de réseau. Contrairement aux deux exemples précédents, qui tiennent encore de pratiques d'écriture artisanales, voici que l'{ industrialisation } de la trivialité est intégrée à même les outils d'écriture du { Web }. Les moyens d'anticiper la circulation sont exportés (*via* des { *plugins* }) vers d'autres outils d'écriture (les { CMS }) : ils sont en quelque sorte « embarqués », disséminés dans d'autres { architextes }.

API,
architexte,
balise,
CMS,
code,
code source,
développeur,
documentation,
industrialisation,
petites formes,
***plugin*,**
Web

Voir glossaire tome II, p. 3-25

c Une « idéologisation » de la circulation ?

Avec Pierre Mœglin, on peut proposer l'hypothèse qu'il s'agit là d'un processus d'idéologisation de l'industrie des passages. L'idéologisation est, chez Mœglin, l'une des composantes du processus d'{ industrialisation }. Il la définit comme « [...] l'avènement d'un état d'esprit, ou à ce que faute de mieux, j'appellerai "une mentalité entrepreneuriale", privilégiant l'utilisation de tous les moyens humains et techniques pour concourir au rendement et à la productivité[650] ». L'idéologisation, c'est lorsque les ressorts principaux d'un système économique (rendement, productivité, dans notre cas, circulation) intègrent les valeurs des acteurs de ce système. Ici, les { balises } dans le { code source } semblent indiquer que des { développeurs } ont intégré l'idée qu'il fallait que la page circule et qu'il fallait utiliser les moyens techniques adaptés pour optimiser cette circulation. De la même façon, l'inclusion de modules complémentaires dans les { CMS } pour optimiser le référencement par Google et autres industries médiatisantes montre que les outils d'écriture intègrent cet « état d'esprit » : le texte **doit** circuler et son contenu est optimisé en ce sens.

En ce qui concerne l'idéologisation, nous en restons à un niveau d'hypothèse, car pour la vérifier il faudrait aller au-delà de notre corpus et interroger les { développeurs } ayant écrit ces pages. En revanche, nous pouvons observer que ce sont bien tous les circuits de l'écriture informatique qui sont concernés par cette préécriture de la circulation. L'{ indus-

649 Yoast. Yoast SEO for WordPress plugin. Dans : *Yoast. SEO for* everyone [en ligne]. 2003-2017. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://yoast.com/wordpress/plugins/seo/#title-description.

650 Mœglin, Pierre. À la recherche de l'industrialisation du tutorat à distance. *Op. cit.*, p. 252.

trialisation } de la trivialité ne concerne alors plus seulement l'{ API } et sa { documentation }, mais toute la chaîne de production du texte amené à circuler, depuis les industries médiatisantes (Twitter, Facebook…) jusqu'aux outils d'écriture du { Web } ({ CMS }). D'un point de vue méthodologique, cette { industrialisation } peut se lire à travers le { code informatique }. Le { code } porte en effet la marque de son environnement technique, économique et éditorial : on y lit les différentes médiations du texte ({ CMS }, { *plugin* }, syntaxe de l'{ API }…) qui concourent à en optimiser la circulation.

L'hypothèse de départ de cette partie est donc confirmée. Dans certain cas, la délégation de l'écriture et son équipement ne sont pas neutres : si les { API } web donnent la main, elles la guident en même temps vers une plus grande circulation des textes produits. Donner la main est toutefois inutile si celui à qui l'on donne l'outil n'en fait rien. Il faut que l'outil donne envie d'écrire. Dans notre cas, il faut donc que les { API } éveillent le désir de la circulation, de la réappropriation, ce qu'on pourrait appeler le **désir du trivial**.

L'analyse des cartes Twitter et du { code source } correspondant ont montré une préécriture de la circulation qui s'étend à toutes les couches d'écriture des textes de réseau. L'outillage de la circulation proposé par les { API } web ne peut donc être qualifié de neutre en ceci qu'il embarque un certain « état d'esprit » visant à optimiser la circulation des textes.

API, code source, polychrésie

Voir glossaire tome II, p. 3-25

Mais cet état d'esprit ne suffit pas. Rien ne sert de proposer un outil si cet outil n'est pas utilisé. Il faut rendre l'utilisation désirable. C'est pourquoi la délégation de l'écriture des { API } web se couple à ce que l'économiste Frédéric Lordon nomme une *épithumè* : un régime de désir. Il faut rendre les { API } désirables. Partant de ce principe, nous faisons l'hypothèse que les { API } sont rendues désirables par l'exhibition de leur { polychrésie }.

I 1 E b
masquage d'information
Voir p. 102

API, application, développeur, documentation, forme-texte, masquage d'information, polychrésie, *widget*

Voir glossaire tome II, p. 3-25

Partant de cette hypothèse, nous isolons trois moyens par lesquels l'{ API } est rendue désirable. Un premier moyen est le principe du « { masquage d'information[651] } », similaire au *peep show* en ceci qu'il éveille le désir par un jeu du révélé et du caché, tout en protégeant le désiré du désirant (IV. 2. A). Le second moyen est l'instauration de liens de confiance entre les parties concernées : plateformes, { développeurs }, internautes (IV. 2. B). Ces liens sont construits par des médiations sémiotiques. Nous en avons relevé quatre dans notre corpus : le *token* d'identification ; la monstration de la « santé » de l'{ API } ; les fenêtres d'autorisation d'accès aux données personnelles ; les recommandations de la plateforme demandant aux { développeurs } d'être de « bons partenaires ». Le troisième moyen est l'utilisation de formes exhibant la { polychrésie } du dispositif (IV. 2. C). Nous en identifions trois : le portail ; la liste ; le guide.

L'analyse de ces formes infirmera notre hypothèse de départ : ce n'est pas la { polychrésie } des utilisations par les { développeurs } qui est exhibée, mais c'est le clic de l'utilisateur sur les { formes-textes } qui doit être provoqué par ces formes. Les { *widgets* } et autres boutons doivent déclencher le clic, le rendre désirable, ce parce que le clic est devenu « partage », ouvrant à la question de l'instrumentalisation des { formes-textes }.

651 Parnas, David L. On the Criteria to Be Used in Decomposing Systems into Modules. *Op. cit.*, p. 1053–1058.

2 SUSCITER LE DÉSIR DU TRIVIAL

Si les { API } fonctionnent sur une délégation de l'écriture et si cette délégation est aussi une instrumentalisation de la circulation des écrits, reste que donner la main ne suffit pas. Il faut encore donner l'envie d'écrire, d'utiliser les données de l'{ API }, de produire des textes et de les faire circuler. On peut donc penser que la viabilité économique d'une { API } web – et ses usages aujourd'hui généralisés – reposent en partie sur le fait qu'une { API } construit un désir d'écrire. Comment analyser ce désir ?

C'est tout le mérite de l'économiste Frédéric Lordon d'avoir élaboré une théorie de l'économie capitaliste qui lie désir et délégation, à travers l'idée du « faire faire ». Selon lui, le capitalisme repose sur un certain agencement des désirs individuels – le *conatus* que Lordon reprend à Spinoza. Un entrepreneur désire quelque chose, selon son *conatus*. Il va enrôler d'autres *conatus*, d'autres puissances d'agir, au service de son désir, qu'il va récompenser par le salariat et diverses gratifications symboliques. Il y a donc un enrôlement des désirs d'agir au service d'un « désir-maître » qui fixe l'orientation des désirs enrôlés[652]. Cet enrôlement passe par une structure de désir[653] – une *épithumè* – présente dans chaque entreprise capitaliste en ceci qu'elle fonctionne sur un « faire faire » à d'autres (les employés) ce qu'un individu désire.

Les { API } web sont une modalité de ce « faire faire ». Elles sont une charnière articulant désir-maître et désirs individuels. En effet, tout le principe de la délégation d'une écriture (donner la main) repose sur l'éveil d'un désir d'écriture. On donne la main à des externalités ({ développeurs }, start-up) afin qu'ils accomplissent leurs désirs d'agir (développer une { application } ou un site web). Mais ces désirs d'agir prennent place dans le cadre du désir-maître, fixé en partie par la { documentation } et ses contraintes[654]. De ce point de vue, la logique de délégation n'est pas seulement l'incarnation technique d'un modèle économique. Elle est un

652 Lordon, Frédéric. *Op. cit.*, p. 54.

653 Lordon, Frédéric. *Idem*, p. 73.

654 Par le jeu du feuilletage architextuel, on ne peut parler de désir « totalement » maître : les entreprises ouvrant l'API se soumettent à d'autres contraintes, comme la syntaxe HTTP, les standards en vigueur, la loi, etc. Il ne faut pas entendre « désir-maître » comme un règne sans partage. Il y a de l'hétérogène dans ce désir, qui est l'hétérogénéité induite par les médiations avec lesquelles les humains doivent composer.

API, application, documentation, logiciel, masquage d'information, polychrésie, programmation

Voir glossaire tome II, p. 3-25

certain agencement des désirs, elle participe à ce que Lordon nomme l'*epithumè* capitaliste. À partir du moment où une activité scripturaire comme la { programmation } est aussi une activité économique (l'industrie du { logiciel } et le modèle économique des plateformes comme Facebook et Twitter) et que cette activité économique repose sur l'optimisation de la circulation, alors on peut penser que cette écriture s'adosse à une économie des désirs. Elle devient un objet d'enrôlement : il s'agit de « faire écrire » des { applications } ou des sites web par des personnes extérieures à l'entreprise qui publie son { API }, de rendre désirable la trivialité des textes. La question devient alors : comment une { API } organise ce régime de désir ? Comment devient-elle objet de désirabilité ? Comment motive-t-elle les désirs d'écrire et de faire circuler ?

Notre hypothèse pour cette partie se situe à la croisée de cette problématisation des { API } comme modalités d'un « faire faire » et de l'idée d'une industrie des passages. Si la valeur se crée par la réappropriation et la circulation et si cette production est adossée – comme le suggère Lordon – à un régime de désir, alors c'est la réappropriation même qui devient désirable. Montrer tout ce que l'on peut faire avec un dispositif, c'est en construire la désirabilité. D'où notre hypothèse : les { API } web construisent un désir d'écrire en exhibant leur { polychrésie } et, en vertu des propriétés spécifiques des { API }, cette exhibition se fait selon le modèle du *peep show*. Cette hypothèse sera étudiée en trois temps. D'abord en montrant qu'une { API } masque et protège les données d'une plateforme, mais autorise quelques « œillades » bien encadrées, à la manière d'un *peep show* (IV. 2. A) ; ensuite qu'une { API } établit des liens de confiance permettant de la rendre désirable (IV. 2. B) ; enfin nous étudierons la { documentation } de ces { API } comme lieu privilégié de cette exhibition de la { polychrésie } (IV. 2. C).

2 A Regarder sans toucher : le *peep show* comme modèle de l'*épithumè* des API contemporaines

Quels sont les principes d'un *peep show*, ce dispositif particulier qui consiste à regarder, l'œil collé à un judas, un spectacle aujourd'hui le plus souvent érotique[655] ? Nous en isolons trois dimensions constitutives :

– la construction d'une position de regardant non-regardé : on peut voir sans être vu ;

– la protection du désiré de l'accomplissement des velléités du désirant : on ne peut pas toucher ni intervenir sur ce qu'on regarde ;

– l'incomplétude structurelle du regard du désirant : le dispositif fait que celui qui regarde ne peut pas tout voir de la scène qui se déroule.

Au moins deux de ces trois caractéristiques s'appliquent aux { API } web. Premièrement, les { API } proposent un accès structuré à des jeux de données et en cela elles protègent tout en donnant accès (IV. 2. A. a). Deuxièmement, le principe du { masquage d'information }, essentiel au fonctionnement des { API } web, est cette fois envisagé du point de vue de la construction du désir : en masquant certaines informations, on construit une incomplétude structurelle aux données, les rendants ainsi désirables (IV. 2. A. b).

I | 1 | E | b
masquage d'information
Voir p. 102

a Masquer pour mieux protéger

Comment appliquer ces trois points aux { API } web contemporaines ? La donnée et son accès jouent un rôle central. Premièrement, les données ou blocs de fonctionnalités qui sont rendus disponibles par l'{ API } sont protégés. Leur accès n'implique pas un transfert de propriété : les données continuent à être détenues par Facebook ou Twitter et elles ne peuvent être modifiées dans leur structure. Une { API } ne donne donc qu'une possibilité de consultation des données, mais ne menace en rien leur structure interne du côté des serveurs de Facebook ou de Twitter. Le désiré (la donnée) est protégé des velléités du désirant (un internaute

655 Le *peep show* est apparu dans les foires et les cirques au XIX^e^ siècle. On y projetait à l'origine des films courts ou des images, pas nécessairement érotiques ou pornographiques. Ce n'est que plus tard, dans les années 1970, que la forme actuelle du « *live peep show* » érotique est apparue. Lorsque nous parlons du *peep show* dans le cas des API, nous ne parlons pas d'un désir au sens psychanalytique, ce que pourrait laisser penser cette analogie Nous entendons le désir au sens de Lordon relisant Spinoza, c'est-à-dire une certaine orientation de la puissance d'agir.

API, appel, application, développeur, logiciel, masquage d'information, programmation, requête

Voir glossaire tome II, p. 3-25

ou { développeur } voulant utiliser l'{ API }). À cet égard, l'{ API } est similaire au judas du *peep show* : il permet de jeter un coup d'œil mais crée également une coupure avec ce qui est vu. Une { requête } { API } est une de ces « œillades » : on peut voir comment sont faites certaines données, mais pas dans leur intégralité et sans que cela ait une incidence directe sur les données sources. Les manipulations qui sont faites au niveau de la page web ou de l'{ application } n'affectent pas les données telles qu'elles sont stockées. Cela présente un intérêt évident pour les plateformes : ne pas mettre en péril ce qui constitue un élément essentiel de leur modèle d'affaires. Pour les { développeurs } ou internautes utilisant l'{ API }, cela permet de faciliter l'usage des données sans devoir connaître comment fonctionne tout le système. Ce double intérêt (protection / facilité d'utilisation) est au principe du { masquage d'information[656] } (*information hiding*), essentiel dans une approche modulaire de la { programmation } et dans les { API } contemporaines[657].

I 1 E b
masquage d'information
Voir p. 102

b Masquer pour mieux éveiller le désir

Plus que cela, nous soutenons que l'*information hiding* ({ masquage d'information }) joue un rôle primordial dans l'instauration d'une désidérabilité des { API }. Selon Galloway, l'histoire du { logiciel } peut être construite autour d'une tension entre le reflet (*reflection*) et l'offuscation (*obfuscation*[658]), entre ce qui est montré et ce qui est caché. Dans la { programmation } modulaire, le { masquage d'information } (*information hiding*) est une procédure qui masque la façon de fonctionner d'une partie du système, précisément pour en faciliter la compréhension et l'usage de la partie explicite, « représentée ». L'*information hiding* agit dans les { API } non seulement du point de vue pratique (utilisation facilitée) mais aussi du point de vue affectif : cela participe à construire la désirabilité des données et des modules mis à disposition, en jouant de cette dynamique entre le caché et révélé. En somme, une { API } cache et protège ce qui doit l'être, mais laisse visible juste ce qu'il faut pour attiser la curiosité. Tout comme un *peep show* excite en autorisant les regards sans que l'on ne puisse toucher à la personne regardée, une { API } éveille le désir de la donnée en en révélant certains aspects, sans que cette révélation menace son intégrité.

656 Parnas, David L. On the Criteria to Be Used in Decomposing Systems into Modules. *Op. cit.*, p. 1053–1058.

657 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*, p. 4.

658 Galloway, Alexander R. Language Wants To Be Overlooked: On Software and Ideology. *Journal of Visual Culture*. 2006, vol. 5, nº 3, p. 323-324.

En résumé, une { API } est bien un lieu d'élaboration de « sentiments intenses[659] ». Elle organise une économie du désir, une *epithumè*. Pour qualifier cette epithumè, la métaphore du *peep show* fonctionne au moins sur deux aspects : le fait que l'on puisse voir les données et leur structure sans pour autant en disposer librement (sans pouvoir y « toucher » – métaphoriquement parlant) ; le fait que l'on n'ait accès, par la structure du dispositif, qu'à une partie des données et qu'une grande partie reste cachée. Le seul point sur lequel notre métaphore achoppe, c'est celui de l'anonymat : faire un { appel } à l'{ API } requiert une identification préalable. Mais sur les deux autres points, nous avons vu que l'{ API }, par sa structuration, cherche à créer du désir auprès des { développeurs } externes. Du désir envers quoi et à quelle fin ? Du désir envers les données ou modules mis à disposition par l'{ API }. Et ce, afin que ces modules ou données soient le plus réutilisés possible.

II | 1 | D | b
une identification préalable
Voir p. 181 sur le jeton comme substitut matériel de la confiance

Ce premier point établi, on peut formuler une seconde hypothèse. En quoi une { API } serait-elle désirable si elle n'était pas fiable ? C'est-à-dire : si elle ne donnait pas tous les signes de son bon fonctionnement technique et du fait qu'elle va nous donner accès aux données requises ? En d'autres termes, le désir de l'utilisation est aussi construit par la confiance que l'on peut témoigner envers un dispositif technique. Il y aurait donc des façons d'instaurer des liens de confiance envers une { API } web et ces liens participeraient à construire la désirabilité du dispositif.

659 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op.cit.*

2 B Instaurer des liens de confiance

API, application, code, développeur, hexadécimal, jeton

Voir glossaire tome II, p. 3-25

Si l'{ API } est censée instaurer la désirabilité des données, la condition première est sans doute de créer une relation de confiance entre les trois parties intéressées : la plateforme ; un { développeur } écrivant une { application } ou une page web ; un utilisateur de l'{ application } ou lecteur de la page web. Une plateforme donne l'initiative d'écriture une { application } à un { développeur } ; un utilisateur donne l'autorisation à une { application } d'écrire à sa place.

Cette tripartition crée des rapports complexes de confiance. Si un { développeur } n'a pas « confiance » dans la fiabilité de la plateforme et de son { API } – si, par exemple, il craint que les conditions d'accès aux données changent du jour au lendemain – difficile d'utiliser l'{ API } sans réticence. Idem du côté de Facebook ou Twitter : la plateforme met en jeu une part de son image de marque en donnant les rênes de l'écriture à des { développeurs } externes. Si ces derniers écrivent des { applications } malveillantes, c'est la plateforme qui risque d'en pâtir. Enfin du côté de l'utilisateur, il s'agit de montrer les signes de la confiance qu'il peut accorder tant à la plateforme qu'à l'{ application } qu'il utilise : si une { application } sur Facebook récupère et utilise certaines de ses informations sans le prévenir, cela risque d'écorner la confiance dans le dispositif.

En commencant par préciser comment la confiance envers un objet technique se construit par des médiations sémiotiques (IV. 2. B. a), nous analyserons ensuite ces médiations dans le cas de notre corpus. Elles prennent quatre formes (IV. 2. B. b) : le { jeton } d'identification ; le concept de « santé » de l'{ API } ; l'autorisation d'accès aux données ; les recommandations données aux { développeurs }.

a Qu'est-ce qu'avoir confiance dans une API ?

Tout le problème de la notion de confiance est qu'elle est souvent définie comme exclusivement humaine. Faire confiance, c'est croire qu'un autre être humain va agir de telle manière, sans que l'on puisse en avoir la garantie[660]. On distingue alors la confiance de la fiabilité. On peut dire qu'un système technique est « fiable » en ceci qu'il aura un comportement prévisible s'il ne faillit pas. La fiabilité peut être garantie, par tout un lot de médiations : connaissances techniques, labels, expériences passées, etc[661]. Bien souvent, cette distinction entre confiance et fiabilité, mène à une forme d'anthropocentrisme de la confiance qui pousse à en exclure les objets techniques[662].

Or, la confiance passe aussi par des expédients techniques, ce que montrent bien Loeve et Normand dans le cas des nanotechnologies. La seule efficacité d'une molécule pharmaceutique n'explique pas sa diffusion et adoption massive. Ce qui l'explique, c'est que cette molécule est instituée par des médiations comme objet de confiance. En d'autres termes, nous faisons également confiance aux objets et cette confiance est construite communicationnellement. Un objet n'est jamais digne de confiance en tant que tel[663]. Il est institué comme fiable et comme objet en lequel on peut faire confiance. Or, l'{ API }, par sa dimension protocolaire et contractuelle[664], nécessite une confiance entre tous les acteurs impliqués. Par quelles médiations une { API } est donc instituée comme objet de confiance ?

b Les quatre formes de la confiance

Dans notre corpus, ces médiations prennent quatre formes. La première, c'est le { jeton } d'identification : l'utilisateur fait confiance à l'{ application } et lui délègue ses possibilités d'écriture. Une clé { hexadécimale } tient lieu de rapport de confiance. Le chiffrement agit comme facteur de fiabilité tout autant que de confiance par l'imaginaire du chiffre qu'il mobilise : l'internaute postule que ce { code } ne peut pas

660 À cet égard, la confiance est liée à la notion de risque et d'incertitude.

661 Ici, la notion d'incertitude n'intervient donc plus de façon structurelle : une erreur de fonctionnement peut être corrigée.

662 Loeve, Sacha et Normand, Mickaël. How to Trust a Molecule? The Case of Cyclodextrins Entering the Nanorealm. Dans : Zülsdorf, Torben B., Wienroth, Matthias, Coenen, Christopher, *et al.* (dir.), *op. cit.*, p. 203.

663 Par exemple parce qu'il est fiable et remplit un rôle attendu.

664 Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Op. cit.*

être décrypté[665]. Cette forme de la confiance rejoint plus généralement l'ensemble des pages de la { documentation } qui invitent à utiliser un protocole de sécurité fiable pour protéger l'{ API } et le transfert de données. Elle se construit sur un imaginaire de la sécurité et des technologies de cryptage informatique.

II 2 C
un imaginaire de la sécurité…
Voir p. 203

La seconde, c'est ce que Twitter nomme la « santé » de l'{ API } (*API health status*), mais qu'on retrouve également chez Facebook sous le nom de « *status dashboard* ». C'est un tableau qui montre si les différents services de l'{ API } de Twitter fonctionnent correctement à travers un code iconique élémentaire : cercle vert ; cercle orange ; croix rouge[666]. On peut donc avoir une vision synoptique et actualisée dans le temps de l'efficacité de l'{ API } de Twitter. Cette efficacité est mesurée à l'aune du temps de réponse des serveurs, exprimé en millisecondes. Il s'agit pour la plateforme d'attester de la **fiabilité** technique de l'{ API } : les { développeurs } comptent sur la plateforme pour fonctionner sans heurts. D'où des outils de surveillance de cette performance technique comme ces tableaux.

API, application, développeur, documentation

Voir glossaire tome II, p. 3-25

La troisième de ces formes, c'est celle qui apparaît quand un internaute veut autoriser une { application } à utiliser des informations récupérées sur Twitter ou Facebook. C'est la petite fenêtre demandant « Autorisez-vous l'application x à accéder à [telles informations][667] ? ». Ces indications font partie des recommandations (*guidelines*) que les { développeurs } externes doivent suivre : afficher clairement ce à quoi l'{ application } accède et quels pouvoirs l'utilisateur lui donne. C'est peut être la forme contractuelle la plus explicite. Il s'agit de construire la confiance entre les utilisateurs et l'{ application }, en affichant quels pouvoirs il lui délègue.

Quatrième forme enfin, l'ensemble des recommandations ou *guidelines* que les plateformes donnent aux { développeurs }. Nous prendrons l'exemple de Twitter. Pour leur { API }, quatre documents remplissent cette fonction : les *developer policy* (« ligne de conduite pour les développeurs ») ; le *developer agreement* (« accords avec les développeurs ») ; les *display requirements* (« exigences d'affichage ») et enfin les *geo guidelines* (« recommandations pour l'API geo »). Dans ces quatre documents, l'entreprise détaille quelques règles à respecter, ce sans quoi une { ap-

665 La technologie utilisée pour générer la clé est *OAuth*, technologie standard et dont le nom synthétise les enjeux de la confiance : il s'agit de faire un « serment » (*oath*) par une « authentification » (*Auth*), c'est-à-dire de se jurer une forme de fidélité, qui ne passe plus la parole comme c'est le cas initialement du serment mais par l'écrit et le chiffre. Voir Benveniste, Émile. *Le vocabulaire des institutions indo-européennes. Tome 1. économie, parenté, société*. Paris : Les Éditions de Minuit, 1969. Chapitre 8, « La fidélité personnelle », p. 103-122.

666 Voir annexe nº 20 et 21.

667 Voir annexe nº 22.

plication } pourra ne plus être autorisée à récupérer des données *via* l'{ API }. Le *developer agreement* est un contrat au sens le plus classique du terme, où sont détaillés les droits et obligations des { développeurs } externes. La *developer policy* est, elle, une version plus souple des attentes de Twitter vis-à-vis de ceux auxquels elle délègue du pouvoir. En plus de consignes quant à l'image du texte à respecter, la *developer policy* tisse tout un ensemble de valeurs morales, incitant à établir entre { développeurs }, Twitter et les utilisateurs des rapports de bonne entente. Par exemple, dans la section 3 « Respecter le contrôle et la vie privée des utilisateurs », il est demandé de montrer clairement, dans l'{ application }, comment cette dernière va utiliser les informations que lui confie l'utilisateur[668]. De manière générale, Twitter reprend une rhétorique de la transparence : il faut « montrer » (*display*) ou « révéler » (*disclose*) « clairement » (« **Clearly** *identify your service* », nous soulignons) ce que fait l'{ application } avec les données qu'elle utilise. Mais la section la plus intéressante est la section 6, qui explique comment être « un bon partenaire envers Twitter[669] ». La notion de « bon partenaire » est de prime abord très floue. Sur quels critères peut-on juger un « bon » partenaire ? Cette notion est extrêmement souple et traduit effectivement un rapport de confiance au sens le plus hasardeux du terme : Twitter compte sur les { développeurs } pour ne pas faire n'importe quoi avec les données qu'ils utilisent[670]. Mais lorsqu'on examine en détail ce que Twitter entend par « être un bon partenaire », il s'agit principalement de respecter les autres recommandations (sous-section a et d), ne pas diffuser les données de Twitter à des tierces parties (sous-section b). Il est également interdit de monétiser les données, de profiter de l'{ API } pour surveiller les performances de Twitter ou pour utiliser ces données à des fins publicitaires « en dehors de Twitter[671] ». Ce sont donc des consignes très précises, au contraire de la notion floue de « bon partenaire ». Cette section illustre la tendance plus large de ces documents : une ambiguïté entre des recommandations souples (être un « bon » partenaire, être « transparent ») et certaines très détaillées.

IV **4**
consignes quant à l'image du texte à respecter
Voir p. 405

668 Sous-section c et d, voir annexe nº 23.

669 « *Be a Good Partner to Twitter* », voir annexe nº 24.

670 Flou qui rappelle le fameux « *Don't be evil* » de Google, phrase emblématique de la firme pour affirmer leur mission. Voir Goyet, Samuel. Google Glass au regard de son API : visions & régulations d'une innovation technique [en ligne]. Communication au colloque international *Digital Intelligence*, Nantes, 17-19 septembre 2014. [Mis en ligne le 8 décembre 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.univ-nantes.fr/medias/fichier/paper_36___s_goyet___google_glass_1415432706656.pdf.

671 « [...] *to target users with advertising outside of the Twitter platform* [...] », voir annexe nº 24 section 6. e. v.

Quelle est la relation de confiance construite par cette ambiguïté ? Elle est à l'image d'une « entente cordiale » entre deux parties qui ont besoin mutuellement l'une de l'autre. D'un côté, Twitter (et Facebook, dans une moindre mesure) a bâti son modèle économique sur le développement d'{ applications } tierces. L'entreprise a donc besoin du travail de { développeurs } externes et autorise une certaine souplesse. Dans le même temps, elle doit contrôler son image de marque et affirmer sa mainmise sur son produit (les données), d'où les importantes restrictions. L'ensemble maintient un équilibre où la confiance est en constante renégociation, parce que construite sur une ambiguïté et une dépendance mutuelle[672].

API, application, développeur, documentation, objet, polychrésie

Voir glossaire tome II, p. 3-25

Mais cette approche de la confiance ne suffit pas. L'établissement de celle-ci, même de manière ambiguë, n'est qu'une première étape. Il faut encore rendre désirable ce qu'on peut faire avec l'{ API } : utiliser des données, des modules préécrits à ses propres fins. C'est-à-dire construire le désir de l'{ API } par une mise en avant des possibilités d'écriture qu'elle offre. Il est donc temps désormais d'étudier l'hypothèse principale de notre partie : en quoi l'exhibition de la { polychrésie } constitue-t-elle l'un des leviers de désirabilité des { API } web ?

672 Il existe bien sûr d'autres formes de construction de la confiance, par exemple dans les discussions internes aux communautés de développeurs, en dehors de Twitter et donc de notre corpus.

2 C Portail, liste, guide… La maximisation des affects désirants par le miroitement des possibles ?

Nous avons jusqu'ici identifié deux vecteurs de la désirabilité des { API } web : leur construction sur le modèle du *peep show* ; les différentes médiations qui instituent la confiance dans le dispositif. Il nous reste désormais à vérifier notre hypothèse que la { documentation } des { API } a pour fonction d'exciter le désir des { développeurs } en exhibant la { polychrésie } des données. Il s'agit de montrer ostensiblement « tout ce qu'on peut faire » avec ce à quoi donne accès l'{ API }, en vue de maximiser les affects désirants. Cette exhibition suppose des médiations sémiotiques. Nous avons donc porté une attention particulière, dans notre corpus, aux formes et aux discours de cette exhibition. Nous en avons relevé trois : le portail (IV. 2. C. a) qui montre toutes les applications possibles d'une { API } ; la liste (IV. 2. C. b) qui montre les différents niveaux de granularité des { objets } auxquels donne accès une { API } ; le guide enfin (IV. 2. C. c), qui textualise les processus architextuels en jeu et facilite ainsi l'utilisation d'une { API }. L'étude de ces trois formes permettra d'invalider notre hypothèse de départ. Si ces dernières exhibent certes une { polychrésie }, elles le font de manière indirecte : ce n'est pas l'écriture des { développeurs } qui est sollicitée comme désirable, mais le clic des internautes. La question se déplace alors vers la construction de la désirabilité du clic, qui fait l'objet de la partie suivante.

a Le portail comme porte d'entrée

La première forme, c'est donc celle du **portail**. Imaginaire puissant des formes éditoriales de réseau[673], le portail donne une image spatialisée de la lecture et de son orientation, l'image d'une lecture faite de passages et de seuils. Dans notre cas, les pages qui remplissent cette fonction de portail sont les pages d'accueil de la { documentation } : developers.facebook.com/ et dev.twitter.com/web/overview. Pourquoi peut-on les qualifier de « portails » ? En vertu de deux éléments : d'abord parce que ce sont des pages qui ont pour principale fonction de renvoyer à d'autres pages. Ce sont des points d'entrée, des passages. Et dans cette organisation des passages, ces pages donnent à lire une pluralité. Elles

673 Candel, Étienne. L'imaginaire du « portail » : le cas de Rezo.net. *Communication & langages*. 2005, nº 146, p. 19-34.

articulent différentes façons de faire : différents moyens d'accès aux données, différents { systèmes d'exploitation }, etc. En cela, elles reposent sur la mise en visibilité d'une forme de totalité, ou en tout cas d'exhaustivité.

API, application, développeur, discrétisation, documentation, objet, *plugin*, polychrésie, système d'exploitation, *timeline*

Voir glossaire tome II, p. 3-25

Le cas le plus clair est la page developers.facebook.com[674]. Les quatre rubriques principales (*app monetization*, *app invites*, *Social plugins*, *Messenger*) renvoient aux quatre grandes parties de la { documentation }. Ces quatre rubriques proposent d'ores et déjà une vision de ce à quoi sert l'{ API } : à générer de la richesse (« *monétiser votre application* »), à construire une forme de popularité (« *invitation à votre application* ») et une forme de sociabilité (les « *plugins* sociaux » et la « messagerie »). La seconde catégorie (« Découvrez plus de produits », « *Discover more products* ») mélange encore plus les genres. On trouve des liens vers les façons de gagner de l'argent avec une { application }, de construire un jeu pour Facebook, d'intégrer à sa page des { *plugins* } sociaux ou de faire la publicité pour son { application }. De plus, tous ces liens peuvent être classés selon des visées techniques (par { système d'exploitation } : iOS, Android) ou selon des visées éditoriales (selon que l'on cherche à coder un jeu ou que l'on soit une entreprise médiatique – « *Media publishers* »). L'ensemble dessine un accès vers tout ce qu'il est possible de réaliser avec l'{ API } de Facebook.

La forme du portail sert donc dans ce cas à l'exhibition de toutes les possibilités de l'outil d'écriture, par un mélange de tous ses aspects : économiques, techniques et éditoriaux.

674 Voir annexe nº 25.

b La liste comme exhibition des possibles

La seconde forme, c'est celle de la **liste** des propriétés de chaque { objet } à laquelle donne accès l'{ API }. Par exemple, la page dev.twitter.com/embedded-timelines/parameters.com, qui décrit tout les paramètres d'une { *timeline* } ancrée[675]. En quoi ces listes participent-elles à créer du désir ? Parce qu'elles sont des formes d'exhibition de ce qu'on peut faire avec une { API }. Nous avons montré que { discrétisation } va de pair avec manipulabilité et recombinaison. Plus un { objet } est discrétisé, plus nous avons accès à un degré fin de précision, plus ses usages peuvent être multiples car adaptables selon le contexte. La liste des paramètres d'un { objet } informatique est l'ensemble des leviers sur lequels un { développeur } peut agir. À cet égard, l'exhaustivité de la liste, le niveau de détail donné par la { documentation } de l'{ API } participe à construire le désir de la donnée. Elle donne à voir tout ce qui peut être fait, toutes les manipulations auxquelles peuvent se prêter les données. En cela, elle est une exhibition de leur { polychrésie } et participe à en construire la désirabilité.

III 3 C b
discrétisation va de pair avec maniabilité et recombinaison
Voir p. 323

c Le guide et la mise en abyme du processus architextuel

Une troisième forme enfin, c'est celle du **guide**, qui montre comment réaliser telle ou telle opération, en détaillant chacune des étapes[676]. C'est une forme récurrente dans notre corpus : chaque { objet } informatique est accompagné d'un guide. Là encore, il faut relier cette forme à la délégation de l'écriture. Un guide a une visée didactique : il montre une façon de faire pour qu'ensuite celui qui l'ai lu soit autonome. Un guide prend par la main, afin de mieux la donner. Le guide est donc une forme communicationnelle nécessaire dans ce contexte : il faut faire l'effort de montrer le plus clairement possible comment fonctionne techniquement l'{ API } afin que les { développeurs } externes soient ensuite le plus autonomes possible.

675 Voir annexe nº 26.
676 Voir annexe nº 27.

API, appel

Voir glossaire tome II, p. 3-25

La particularité de notre corpus réside dans le fait que les guides en question indiquent la marche à suivre exhibant les différentes opérations réalisées lors d'un { appel } à l'{ API } :

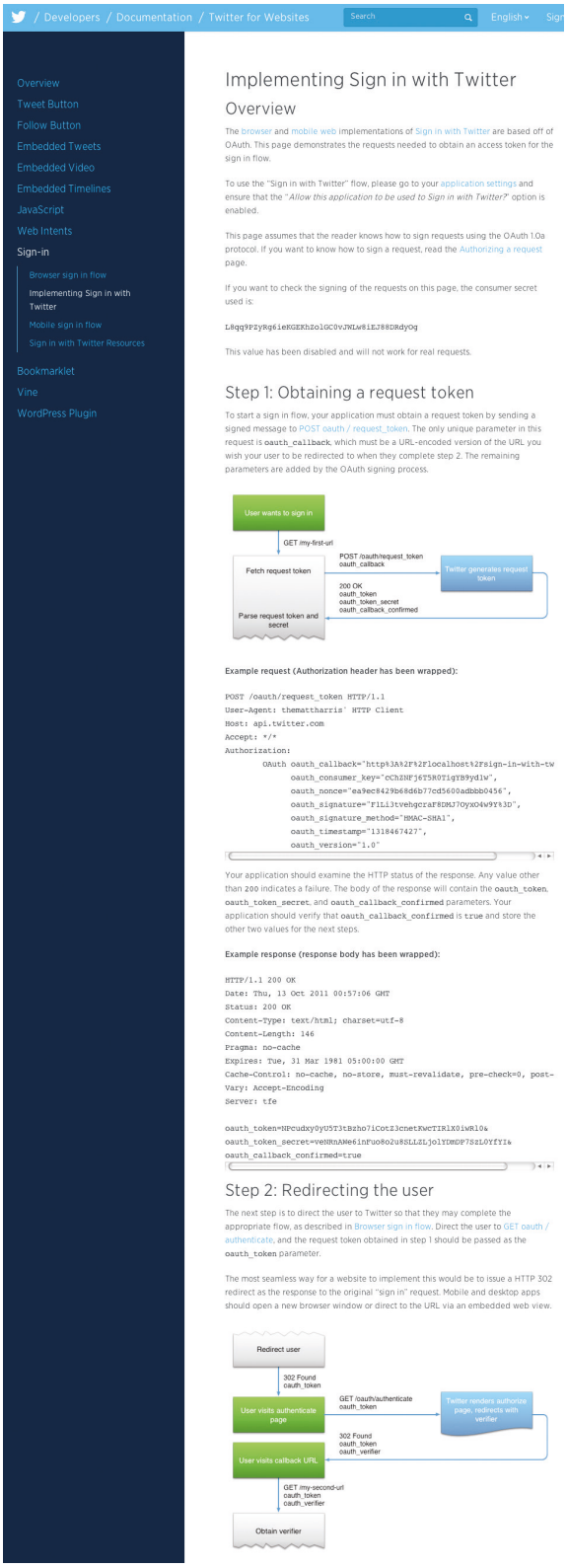


Fig. 42. La forme du « guide » dans la documentation des API (1). Capture d'écran du site dev.twitter.com-web-sign-in-implementing (détail). Réalisée le 14 juillet 2015.

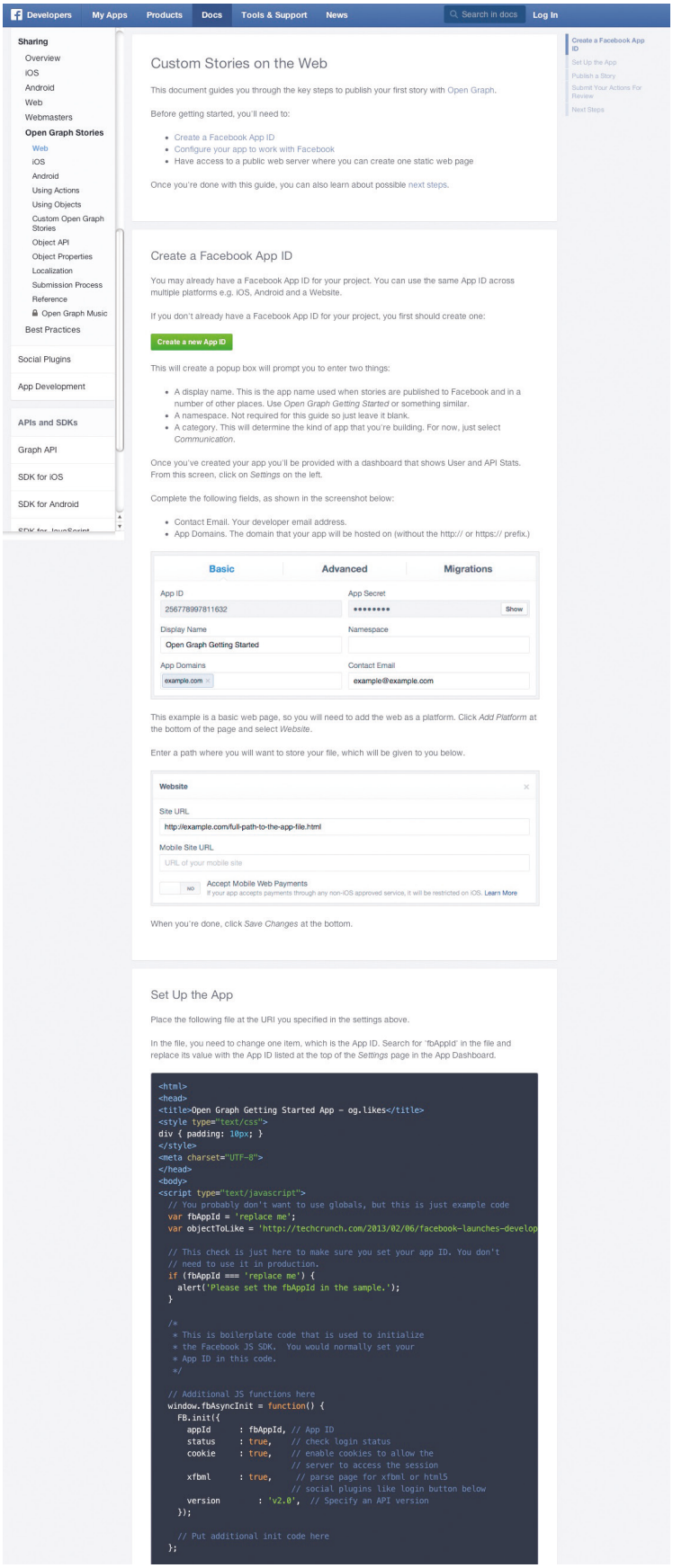


Fig. 43. La forme du « guide » dans la documentation des API (2). Capture d'écran du site developers.facebook.com-docs-opengraph-getting-started. Réalisée le 28 juillet 2015.

Les deux figures ci-dessus sont deux exemples de guide. La figure 42 détaille les procédures d'authentification (*sign-in*) sur Twitter ; la figure 43 explique comment publier une « histoire » (*story*) grâce à l'{ API } de Facebook. Sur chaque page, particulièrement sur celle de Twitter, la { documentation } de l'{ API } montre non seulement les étapes à respecter, mais aussi les opérations qu'effectue le serveur de Twitter ou de Facebook, le système d'{ appel } et de réponse, etc. En d'autres termes, la { documentation } textualise les processus computationnels en jeu[677].

API,
appel,
architexte,
développeur,
documentation,
polychrésie,
programme,
requête,

Voir glossaire tome II, p. 3-25

Pourquoi cette textualisation et en quoi sert-elle la désirabilité des données ? La réponse à ces questions se trouve dans la notion d'{ architexte }. L'{ API }, en tant qu'{ architexte }, est un outil d'écriture informatique. Pour qu'il puisse être utilisé, il est nécessaire d'en fournir le mode d'emploi. Dans le cas d'autres { architextes }, l'utilisation de métaphores particulièrement ancrées dans la culture visuelle permet de se passer d'un mode d'emploi détaillant par le menu le rôle de chaque icône. Ici, nous avons affaire à des processus complexes et propres aux services concernés (Facebook, Twitter…), qui reposent sur des métaphores demandant une culture informatique plus spécialisée : « { requête } », « authentification », « propriétés »… Mais on ne peut pas se reposer sur cette seule culture pour savoir comment fonctionne l'{ API }. C'est pourquoi la textualisation de ses processus est essentielle. Elle joue un rôle didactique (« Comment ça marche ? ») qui contribue à rendre désirable ce à quoi donne accès une { API }, précisément parce qu'on peut peut en imaginer des usages possibles. C'est une condition nécessaire, mais pas suffisante, à cette désirabilité.

677 C'est par ailleurs le propre d'un architexte : donner forme intelligible et visible aux calculs de la machine, afin d'en permettre l'utilisation. Là où cet exemple s'avère particulièrement complexe, c'est que ce qui est représenté, c'est justement un processus architextuel. La page de la documentation de Facebook, montre comment sont générées automatiquement des formes de narration d'une activité et comment créer ces formes. Il s'agit donc d'un architexte (génération automatique d'une « histoire ») représenté dans un *autre* architexte (les requêtes API qui permettent d'engager le processus de création d'une « histoire »).

Notre hypothèse de départ sur l'exhibition de la { polychrésie } est-elle finalement confirmée ? Il semblerait que non. Si une { API } construit effectivement un désir du trivial par un dispositif similaire à un *peep show* dans lequel on peut avoir confiance, difficile de parler de « maximisation des affects désirants » des { développeurs }. Et ce pour la simple raison que nous n'avons pas relevé d'éléments de discours qui valorise explicitement « tout ce qu'on peut faire » avec l'{ API }, en tout cas pas directement auprès des { développeurs }. Il y est montré l'ensemble des opérations possibles, comment les réaliser, mais pas de phrases explicites comme « réalisez toutes vos envies » ou « construisez des applications originales ».

En revanche, il est dit dans la { documentation } que ce sont les *internautes* qui vont « vouloir cliquer » ou que les formes que créent l'{ API } « […] affichent votre contenu d'une manière plus engageante[678] […] ». Cette problématique du désir est donc à double détente : on exhibe certes la { polychrésie } des données, mais en vue de maximiser le désir des internautes, ceux qui, *in fine*, vont lire les formes générées par les { API }. Ce sont eux qu'ils faut inciter à lire, à « partager », à « s'engager ». C'est donc ce clic qui est recherché et qui est rendu désirable[679]. Pourquoi ce déplacement ? Peut être parce que le clic – une action technique – est devenu activité sociale : il permet de « partager », de « joindre une conversation »… Ce qui serait équipé, ce ne serait pas tant l'écriture de { programmes } ou de pages web que la sociabilité même. Il y aurait donc une instrumentalisation des { API } web : un outil se voit attribué des pouvoirs bien supérieurs à ses prérogatives initiales. C'est l'hypothèse que nous allons désormais étudier.

678 « […] *displaying your content in a more engaging way* […] ». dev.twitter.com/cards-types/summary, capture réalisée le 14 juillet 2015.

679 Ce qui concorderait d'ailleurs avec les analyses du *digital labor* comme d'un capitalisme constitué autour de micro-opérations comme des clics sur des boutons « J'aime » ou « Partager ». Voir Cardon, Dominique et Casilli, Antonio A. *Op. cit.*

L'analyse de l'équipement de la circulation et du régime de désir associé a donc déplacé nos questions. Partant de l'hypothèse de la construction de la désirabilité des données par l'exhibition de leurs réutilisations possibles, nous avons constaté que la { documentation } des { API } construisait surtout la désirabilité du clic, en soulignant que cliquer sur un { tweet } ou une publication ancrée, c'est améliorer son « expérience de lecture », c'est « partager »… Ces formes se chargent d'un pouvoir : celui de « socialiser » les pages sur lesquelles ils s'ancrent. Cliquer, c'est partager.

API, documentaion, industrialisation, tweet

Voir glossaire tome II, p. 3-25

Nous voilà en face de ce qu'on pourrait nommer un nœud techno-sémiotique : aux formes sémiotiques qui permettent une action technique sont attribuées des prérogatives sociales. Pour démêler ce nœud, nous proposons d'y voir le résultat d'une « instrumentalisation » des { API }, troisième processus de l'{ industrialisation }.

API,
code,
forme-texte,
interface,
interopérabilité,
petites formes,
programme,
Web,
widget

Voir glossaire tome II, p. 3-25

Nous faisons l'hypothèse que c'est parce que les { API } servent initialement à assurer l'{ interopérabilité } entre composants informatiques (IV. 3. A) – c'est-à-dire le bon échange de données – qu'elles furent propices à une instrumentalisation. Leur fonction de communication, au sens informatique du terme, a été réappropriée comme fonction de communication entre les humains. Pour vérifier cette hypothèse, nous retraçons quelques étapes d'un parcours d'instrumentalisation des { API }, en identifiant à chaque étape un acteur la façon dont les prérogatives des { API } sont requalifiées.

La première de ces étapes passe par le droit états-unien à partir des années 1960, notamment par le *Copyright Act* de 1978. La question est de savoir comment protéger légalement le { code informatique } et ce afin de ne pas enrayer l'innovation du marché naissant de l'industrie informatique. Dans ce contexte, les { interfaces } permettant l'{ interopérabilité } puis les { API } vont être exclues de la législation sur le droit d'auteur afin de favoriser l'innovation (IV. 3. B). Après cette première instrumentalisation ({ API } = { interopérabilité } = innovation), nous en étudions une deuxième plus récente : le projet « French Tech » et plus particulièrement un rapport écrit par l'entrepreneur français Tariq Krim à l'attention de la Ministre Fleur Pellerin. Dans ce rapport, les les { API } sont présentées comme des leviers économiques pour favoriser la croissance et la transparence de l'action publique. Les { API } sont redéfinies comme vecteurs d'« ouverture » (IV. 3. C). Enfin, nous analysons le discours que tiennent Facebook et Twitter sur les formes écrites par l'{ API }. Nous montrons que ce discours est fondé sur les liens historiques entre { Web } et { interopérabilité } et qu'il promeut l'idée que les { formes-textes } sont des moyens faciles de rendre sociables des pages web (IV. 3. D).

3 L'INSTRUMENTALISATION DES API WEB : DE L'INTEROPÉRABILITÉ AU PARTAGE, OU COMMENT DÉMÊLER LE NŒUD TECHNO-SÉMIOTIQUE ?

Les { API } web équipent la circulation des textes et construisent la désirabilité du clic. Il faut donner envie aux internautes de cliquer, car cliquer sur un bouton, sur un { *widget* }, c'est « partager », « aimer », « participer à une conversation »… Autant de termes qui suggèrent qu'un geste technique – le clic – s'est chargé, dans le contexte des { API } web de Facebook et de Twitter, d'un pouvoir symbolique : celui de créer des liens de sociabilité. Pourquoi cette promotion axiologique du clic ? Comment, d'un outil facilitant l'écriture de { programmes } informatiques, les { API } sont-elles devenues des outils permettant de produire des { petites formes } sociabilisantes ? Dans ce nœud techno-sémiotique qui consiste à prendre pour argent comptant l'idée selon laquelle cliquer sur un bouton c'est « partager », quel rôle jouent les { API } web ?

Opposer la « visée initiale » des { API } à une utilisation contemporaine « dévoyée » serait une impasse pour répondre à ces questions. Nous proposons de penser plutôt en termes de circulation : les { API } web, en tant qu'objets circulants, sont aux prises avec de multiples réappropriations et requalifications. Parmi ces réappropriations, il y a celle qu'Yves Jeanneret, reprennant les travaux de Pierre Mœglin, nomme l'« instrumentalisation sous le critère du pouvoir[680] ». Il s'agit d'une façon de requalifier les buts d'un dispositif communicationnel afin d'en renforcer la portée et l'opérativité symbolique. Yves Jeanneret prend l'exemple du système documentaire de Paul Otlet et Henri Lafontaine au début du xx^e^ siècle qui, non content d'être un système de classement efficace, prétendait réaliser la paix universelle dans le contexte troublé de l'époque[681]. Il s'agit d'une « instrumentalisation » en ceci qu'un dispositif se voit attribué de nouveaux mérites et, incidemment, un pouvoir accru par rapport à sa tâche initiale.

680 Jeanneret, Yves. *Op. cit.*, 2014, p. 157 ; Mœglin, Pierre. *Outils et médias éducatifs : une approche communicationnelle*. Grenoble : Presses Universitaires de Grenoble, 2005, p. 126.

681 Jeanneret, Yves. *Op. cit.*, 2014, p. 155.

API, application, documentation, interopérabilité, logiciel, *middleware*, Web

Voir glossaire tome II, p. 3-25

Notre hypothèse est que les { API } web ont été instrumentalisées sous le critère du pouvoir autour de la notion d'{ interopérabilité }. Un rapide examen des différentes définitions montrera que l'objectif initial des { API } est de permettre l'{ interopérabilité } entre composants informatiques et que cette { interopérabilité } est un certain avatar – technique – de la communication (IV. 3. A) : il s'agit de permettre l'échange entre données informatiques. Ainsi, l'instrumentalisation des { API } serait le résultat d'une extension de ces prérogatives communicationnelles. Parce que ce sont des outils de communication – au sens technique du terme –, elles se prêtent à une valorisation comme outils de communication – au sens social du terme.

Pour vérifier cette hypothèse, nous dessiner une trajectoire d'instrumentalisation. Il s'agit de prendre différents moments de la vie sociale des { API }, d'en montrer la richesse en tant qu'« êtres culturels[682] » et la manière dont leur rôle est requalifié au fil de leur circulation. Nous montrons qu'à chacun de ces moments, la question de l'{ interopérabilité } permise par les { API } ressurgit pour être réorientée vers d'autres fonctions. Nous isolons trois de ces moments, avec à chaque fois un acteur qui redéfinit le rôle des { API } :

i) Les { API } comme garante de l'innovation, par le droit états-unien (IV. 3. B) ;

ii) Les { API } comme facteur d'ouverture, par un rapport gouvernemental rédigé par Tariq Krim, entrepreneur dans le numérique (IV. 3. C) ;

iii) Les { API }, dans le contexte spécifique du { Web } et de ses imaginaires, comme permettant le partage, tel que présenté par la { documentation } de Facebook et Twitter (IV. 3. D).

682 Jeanneret, Yves. *Op. cit.,* 2008, p. 16.

3 A L'interopérabilité : une certaine conception de la communication au cœur des API

L'instrumentalisation des { API } se joue autour de la question de l'{ interopérabilité }, comprise comme une certaine conception de la communication. Pour prouver cette idée, nous avons repris la définition des { API } qu'on trouve dans des encyclopédies d'informatique. Cela permet de montrer que les { API } sont présentées comme des outils de « communication », parce qu'elles permettent l'échange de données entre composants informatiques (IV. 3. A. a). Ce qui nous permet ensuite de souligner le rôle que jouent les { API } – par cette conception de la communication – dans l'{ interopérabilité } en informatique (IV. 3. A. b). Nous aurons ainsi établi le lien entre { API } et { interopérabilité }, lien qui fera l'objet principal des requalifications successives étudiées dans les parties à venir.

a De l'interopérabilité comme avatar de la communication

La notion de « communication » y est très présente dans la dizaine d'encyclopédies que nous avons consulté. La définition la plus courante est qu'une { API } est une « spécification de la communication[683] » ou encore une manière d'établir « [...] la communication entre des programmes applicatifs[684] ». Avec la mise en réseaux des ordinateurs, les { API } deviennent un élément central du « { *middleware*[685] } », soit ces { logiciels } qui connectent différentes { applications } entre elles afin de garantir leur fonctionnement mutuel. La métaphore utilisée par les auteurs est celle de la colle : les { *middleware* }, donc les { API }, sont des couches d'écriture qui « collent ensemble » (« *glue together* ») différentes couches logicielles d'un système informatique.

683 Illingworth, Valerie. (dir.). *Dictionnaire d'informatique*. Paris : Hermann – Technique et Documentation Lavoisier, 1991, p. 19.

684 Volonino, Linda et Dalal, Pragati. Middleware. Dans : Bidgoli, Hossein (dir.), *op. cit.*, p. 43.

685 Volonino, Linda et Dalal, Pragati. *Idem*, p. 33-44.

API, application, format, interopérabilité, langage de programmation, logiciel, matériel, *middleware*, programme, système d'exploitation

Voir glossaire tome II, p. 3-25

La conception de la communication qui est présentée dans ces définitions est d'abord une définition technique, informatique au sens où elle ressort de la théorie de l'information[686] : une { API } est un outil de communication en ceci qu'elle définit un protocole précis d'échange de données entre deux éléments informatiques. Ainsi, les données peuvent être utilisées par différents { logiciels } ou composants { matériels }.

Cette définition de la communication amène au problème de l'{ interopérabilité } et de son utilité dans l'économie et l'histoire du { logiciel }. Qu'est-ce que l'{ interopérabilité } ? C'est la capacité d'un { logiciel }, d'un document, d'un { programme }, à pouvoir être exécuté par différents environnements informatiques. Idéalement, l'exécution est la même peu importe la machine, le { système d'exploitation }, les autres { logiciels } installés… Par exemple, le { format } de fichier PDF se distingue par sa grande { interopérabilité } : un fichier PDF peut s'ouvrir sous MacOS X, sous Windows, avec de multiples { applications }, l'aspect du document restera – à quelques erreurs fortuites près[687] – le même.

L'{ interopérabilité } devient particulièrement importante – et problématique – dans les années 1980. Avec la généralisation de la micro-informatique et la multiplication des éditeurs de { logiciels } et des fabricants de { matériel }, le paysage informatique se complexifie. Le nombre de machines différentes augmente, ainsi que les différentes versions d'un même { système d'exploitation }. Le modèle économique dominant de vente de { logiciel } devient la vente de masse, ce qui pose des problèmes de compatibilité. Comment assurer que la dernière version d'un { logiciel }, vendu à un moment t, soit compatible avec les différentes machines et { systèmes d'exploitation } présents sur le marché ? La réponse se trouve, en partie, dans l'{ interopérabilité }. En mettant en place des moyens de comprendre comment telle machine, tel système, tel { langage } informatique fonctionne, on peut plus facilement adapter une nouvelle machine ou { logiciel } aux environnements existants.

I | 1 | D | b
le modèle économique dominant…
Voir p. 98

686 Illingworth, Valerie (dir.). *Op. cit.*, p. 98.

687 Comme nous le proposerons dans le chapitre V, ces erreurs ne sont en rien fortuites si on apprend à ne pas les regarder comme des erreurs.

b API et interopérabilité

C'est dans ce contexte que les { API } et plus globalement le { *middleware* } deviennent un enjeu crucial pour assurer cette intéropérabilité. Une { API } permet en effet de donner accès au fonctionnement d'un composant informatique, permettant donc d'adapter et de rendre intéropérable d'autres composants. Les { API } permettent de « [...] développer la portabilité des applications entre machines et aussi à développer le bon fonctionnement des applications avec différents outils[688] [...] ». Si les fabricants de ces outils, ou ceux du { logiciel }, publient une { API }, il est possible de rendre ces différents éléments compatibles entre eux, puisque l'{ API } permet de comprendre comme ces éléments fonctionnent et comment interagir avec eux.

Cette explication est encore d'ordre technique, ou plutôt techno-économique : elle concerne les { API } en tant qu'outils permettant l'{ interopérabilité }, enjeu qui devient important dans le cadre d'une certaine structuration du marché de l'industrie informatique. Pourquoi cette explication reste d'ordre technique ? Parce que si nous avons montré que les { API } témoignent d'une pensée de la communication, il s'agit d'une communication entre machines. La problématique de l'{ interopérabilité }, à cet égard, ne concerne que la compatibilité des machines et { logiciels } entre eux. Comment passe-t-on de la communication entre les machines à la communication entre les êtres humains ? En d'autres termes, comment l'{ interopérabilité } en vient à être un problème communicationnel, au sens des sciences humaines et sociales ?

688 « [...] *to support application portability across computing systems and also support operability of applications with different graphic devices* [...] ». Carson, Georges S. Computer Graphics: standards. Dans : Reilly, Edwin D., Ralston, Anthony et Hemmendinger, David (dir.), *Encyclopedia of Computer Science*. Londres : Nature Publishing Group, 2000 [1976], p. 379.

3 B L'interopérabilité comme garante de l'innovation : le droit états-unien autour du cas *Oracle v. Google*

API,
code,
code source,
interface,
interopérabilité,
logiciel,
programme

Voir glossaire tome II, p. 3-25

La première phase de la trajectoire d'instrumentalisation des { API } a lieu aux États-Unis, à partir des années 1960. Elle concerne la législation américaine sur le droit d'auteur, dans le cadre d'un marché naissant : celui de l'informatique grand public.

Pourquoi commencer par le droit ? Pour des raisons chronologiques tout d'abord : c'est la première phase de la trajectoire que nous voulons dessiner, puisque celle-ci commence en 1960. Pour des raisons culturelles et historiques ensuite. Comme le fait remarquer Milad Doueihi, les débats sur le numérique et ses effets, ont longtemps été l'objet de juristes et de technologues[689]. Que ce soit pour comprendre les effets du { code }, dont la loi et son fonctionnement fournissent une utile métaphore[690], ou pour entériner légalement les mutations engendrées par le numérique. Dans tout les cas, le droit a joué historiquement un rôle prépondérant dans la vie sociale du { code } : quelle position prend le législateur vis-à-vis des { logiciels } et de leur écriture ? Quel est le statut légal du { code informatique } ? L'analyse des décisions états-unienne en la matière[691] permettra de montrer comment une possibilité technique (l'{ interopérabilité }) des { interfaces } et des { API } en particulier fut instrumentalisée comme garantissant l'innovation du marché. Par cette instrumentalisation, l'{ interopérabilité } ne devient plus seulement une propriété technique. Elle devient le vecteur de croissance économique. Par l'entremise de la loi états-unienne, les { API } en viennent donc à être liées à des enjeux industriels et économiques.

Cette instrumentalisation peut être résumée en trois phases. La première commence dans les années 1960, où s'élaborent les bases juridiques autour du { code } et de la propriété intellectuelle. Le droit états-unien reconnaîtra dans le { code informatique } deux composantes : ce qui relève de la création propre à son auteur ; ce qui relève des commandes machiniques et ne peut donc être protégé par la propriété intellectuelle, n'étant pas

689 Doueihi, Milad. *Op. cit.*, 2008, p. 12.

690 Voir le fameux « *Code is law* » dans Lessig, Lawrence. *Op. cit.*

691 Cette focale sur les États-Unis s'explique de deux manières. Historiquement, c'est de là que viennent les principaux acteurs de l'industrie du logiciel. D'un point de vue législatif et académique ensuite, le statut législatif du code a été discuté dès les années 1960 et les principaux travaux sur le sujet sont l'œuvre d'auteurs états-uniens.

considéré comme une œuvre de l'esprit (IV. 3. B. a). Les procès ultérieurs montreront une extension de cette seconde catégorie aux { interfaces } en général (*Computer Associates International v. Altai*, 1988-1992, IV 3. B. b) puis aux { API } en particulier (*Oracle v. Google*, 2010-2016, IV. 3. B. c) et ce au nom de l'{ interopérabilité } et de l'innovation. Par ces multiples décisions, les { API } en viennent ainsi à être mises au service de l'innovation et du développement du marché de l'informatique, parce ce qu'elles permettent l'{ interopérabilité } (IV. 3. B. d).

a Les bases juridiques autour du code informatique : entre œuvre de l'esprit et structure de commande

La question générale que se posent les législateurs états-uniens émerge à la fin des années 1960. Elle peut se formuler ainsi : quelle est l'attitude à adopter vis-à-vis du { code informatique } ? Est-ce que le modèle de la propriété intellectuelle (copyright) peut servir de cadre de référence ? Si ce n'est pas le cas, quel est le modèle de protection légale qui puisse permettre de résoudre les problèmes de plagiat éventuel, sans pour autant freiner le développement économique d'un marché alors en pleine explosion ? Plus précisément encore : qu'est-ce qui, dans le { code source } d'un { programme }, doit être protégé ? Question subsidiaire à celles-ci : est-ce que les { interfaces } qui permettent l'{ interopérabilité } informatique doivent être fortement protégées ou doit-on laisser l'industrie élaborer *de facto* un standard de compatibilité ?

Trois modèles de protection sont possibles, qui sont autant de façons d'envisager le { code informatique[692] } : celui de la propriété intellectuelle (copyright) ; celui du brevet (*patent*) ; celui du secret commercial (*trade secret*). Dans le premier cas, le { code } est considéré comme « œuvre de l'esprit » au même titre qu'un livre ou qu'une chanson. Dans le second, le { code } est considéré comme une invention, au même titre qu'une ampoule ou qu'un aspirateur. Dans le troisième cas, le { code } tombe sous le coup des lois appliquées par exemple à un médicament. Une quatrième solution, hybride, est moins interventionniste : elle consiste à laisser le marché développer de lui-même (*sui generis*) un standard et la loi jugera au coup par coup et selon les circonstances le niveau de protection à adopter. C'est ce dernier modèle qui fut préconisé par le juriste Breyer dans les

692 Abramson, Bruce. Promoting Innovation in the Software Industry: A First Principles Approach to Intellectual Property Reform. *Boston University Journal of Science and Technology Law*. 2002, vol. 81, p. 76.

années 1960 et qui donnera la base législative aux juridictions futures. C'est aussi le travail de Breyer qui aboutira au *Copyright Act* de 1976 dont l'une des clauses (l'article 102b) est centrale dans notre argumentation.

autothéticité, code, développeur, fonction, interopérabilité, JavaScript, programmation, programme

Voir glossaire tome II, p. 3-25

Breyer établit deux points essentiels quant au statut légal du { code } et des interfaces de programmation. Le premier est la distinction entre ce qui, dans un { programme } informatique, tient du « texte écrit » (*written text*) et ce qui tient des processus fonctionnels (*functionnal processes*[693]). Le « texte écrit » en est sa part créative, propre au { développeur } qui a écrit le { programme }. Cette part est reconnue par Breyer comme « œuvre de l'esprit » et est protégée par le *Copyright Act*. En revanche, un { programme } est aussi contraint par une syntaxe, des noms de commande, etc. Par exemple les mots « `function ()` » en { JavaScript } permettent de déclarer une { fonction }, soit une suite d'actions effectuées sous conditions. Ce n'est pas ce que le législateur considère comme un acte créatif, au sens où le { développeur } ne peut pas faire autrement que d'écrire ce mot bien précis, avec cette syntaxe précise. Tous ces paramètres, qui déterminent en grande partie ce qu'est programmer, vont être classés dans les « processus fonctionnels ». En d'autres termes, le statut législatif du { code informatique } est fondé sur la dualité du { code } comme écriture[694]. D'un côté, c'est une pratique d'écriture humaine qui demande une forme de créativité ; de l'autre, c'est une écriture destinée aux machines et qui repose sur un certain nombre de « commandes » purement conventionnelles[695].

693 Samuelson, Pamela. The Uneasy Case for Software Copyrights Revisited. *Op. cit.*, p. 1748.

694 Bachimont, Bruno. Intelligence artificielle et écriture dynamique : de la raison graphique à la raison computationnelle. Dans : Fabbri, Paolo et Petitot, Jean (dir.), *Op. cit.* ; Knuth, Donald E. Literate Programming. *Op. cit.*, p. 97-111.

695 On retrouve la dualité de l'écriture informatique relevée par Bruno Bachimont : par son aspect orthothétique, le code est une œuvre de l'esprit ; par son aspect autothétique, le code est une « structure de commande ». Voir Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 198-199.

Le second point établi par Breyer est que tout ce qui tient de la commande conventionnelle est à considérer comme la « structure, la séquence ou l'organisation » (« *structure, sequence and organization* », ou « SSO ») du { programme } et qu'en tant que tel elle ne doit pas être protégée par la propriété intellectuelle. C'est ce qui se trouve dans la section 102b du *Copyright Act*, qui précise ceci :

> « […] *en aucun cas le copyright pour une œuvre originale ne s'étend à toute idée, procédure, processus, système, méthode, concept, principe ou découverte, et sans égard à la forme par laquelle* [cette idée, procédure, etc.] *est décrite, expliquée, illustrée ou incarnée par l'œuvre*[696] ».

Le copyright protège seulement une œuvre, pas les idées qu'elle contient, les méthodes de production de cette œuvre ou encore les méthodes de travail de celui qui l'a produit. Pour prendre un exemple simple, *Don Quichotte* – en tant qu'œuvre originale – est protégée. Le roman picaresque, la routine d'écriture de Cervantes, le fait que le livre soit découpé en chapitre… Tout cela tient des « procédures, systèmes, méthodes… » d'écriture et ne peuvent pas être protégés par le *Copyright Act*, en vertu du la section 102b[697]. De la même façon, ce qui est susceptible d'être protégé dans le { code informatique }, c'est sa part créative, le « texte écrit ». En revanche, les « processus fonctionnels » comme les noms des commandes, les diverses contraintes de syntaxe… Tout cela est considéré comme relevant de la « structure, séquence ou organisation » (SSO) du { programme } et tombe donc sous le coup de l'exception prévue par la section 102b de la loi états-unienne sur le copyright.

Voilà posées les bases de la législation états-unienne sur le { code informatique } à la jonction entre une pratique d'écriture (la { programmation }), sa part technique ({ autothéticité } du signe, { interopérabilité }), sa part créative et l'industrie dans laquelle cette pratique se situe. Les décisions de justice qui sont rendues articulent ces quatre dimensions et tournent autour de deux questions : quel est le modèle de protection à adopter ? Qu'est-ce qui ressort des SSO et qui ne doit donc pas être protégé par le copyright ? La tendance globale est celle d'un modèle *sui generis* souple et

696 « *In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.* » United States Copyright Office. *Copyright Law of the United States, title 17*, §102(b), [en ligne]. Décembre 2016. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.copyright.gov/title17/92chap1.html#102. Cité par Samuelson, Pamela. Oracle v. Google: Are APIs Copyrightable? *Op. cit.*, p. 26. Nous soulignons.

697 Cette analogie a pour seule fonction la pédagogie et ne tient pas compte de l'existence d'un domaine public, qui est un autre problème.

d'une extension des prérogatives des SSO aux { interfaces } permettant la compatibilité entre machines et / { logiciels } – et donc incidemment aux { API }. Ce faisant, la justice américaine instrumentalise l'{ interopérabilité } : elle favorise cette dernière **au nom** de l'innovation et du développement industriel. L'{ interopérabilité } devient le terreau favorable à l'innovation et non plus seulement le moyen pour deux composants informatiques d'échanger des données. Nous montrerons cette instrumentalisation avec l'étude de deux décisions : *Computer Associates International v. Altai* (1988-1992) et surtout *Oracle v. Google* (2010-2016).

API, application, code source, interface, interopérabilité, langage de programmation, logiciel, programme, système d'exploitation

Voir glossaire tome II, p. 3-25

b Le cas *Computer Associates v. Altai* : des interfaces considérées comme « fonctionnalités »

Les années 1980 marquent une évolution significative dans les rapports entre législation et industrie informatique. Elles sont marquées par la massification du marché et par une scission de plus en plus nette entre non seulement le *hardware* et le *software*, mais aussi entre les fabricants de { systèmes d'exploitation } et les fabricants d'{ applications }.

Le modèle *sui generis* trouve alors ses limites. Le plus important, car il a fait jurisprudence, est le procès *Computer Associates International v. Altai*, entamé en 1988 et jugé en 1992. L'enjeu de ce procès est de déterminer quels sont les éléments d'un { programme } qui peuvent faire l'objet d'un copyright. *Computer Associates International* (CA International) était l'éditeur de CA-Scheduler, un { logiciel } de planification de tâches. Grâce à un élément du { programme } appelé « adaptateur » (Adapter), ce { logiciel } était compatible avec les trois différents { systèmes d'exploitation } de l'IBM 370 : VSE, MVS, CMS. En 1982, la société Altai développe son { logiciel } de planification appelé Zeke. Zeke fonctionne sur IBM 370, mais sous un seul { système d'exploitation } : VSE. Altai voulait rendre son { logiciel } compatible avec les deux autres { systèmes d'exploitation } de la machine. Pour cela, l'entreprise recrute un employé de CA International, qui entre dans l'équipe développant Zeke et met au point Oscar 3.4, l'équivalent pour Zeke de l'Adapter de CA-Scheduler. Le { code source } d'Oscar reprend explicitement la structure du { code source } d'Adapter. Voyant cela, CA International intente un procès à Altai pour atteinte au droit d'auteur et à la protection du secret industriel[698].

698 Pour un résumé factuel du procès, voir (en anglais) : Computer Associates International, Inc. v. Altai, Inc., 2017. *Wikipédia*. [En ligne] [mis en ligne le 28 septembre 2009] [dernière modification le 11 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Computer_Associates_International,_Inc._v._Altai,_Inc.

Peut-on condamner un fabricant pour avoir integré, dans le { code source } de son { logiciel }, non pas mot pour mot le { code source } d'un { logiciel } concurrent mais son organisation pour assurer l'{ interopérabilité } de son { logiciel[699] } ? Dans le procès *CA International v. Altai*, la Cour juge que les { interfaces } entre deux { logiciels } – ici entre une { application } (CA-Scheduler) et un { système d'exploitation } (ceux de l'IBM 307) – fait partie des « fonctionnalités » du { logiciel } et donc ne peuvent pas être protégées par le droit d'auteur. Plus largement, *CA International v. Altai* fait jurisprudence sur l'idée que les « [...] interfaces entre programmes sont des éléments du programme qui ne sont pas protégés par la loi sur le droit d'auteur[700] ».

Computers Associates International v. Altaï marque une étape importante dans l'instrumentalisation de l'{ interopérabilité }. C'est la première fois que les { interfaces } permettant l'{ interopérabilité } entre { logiciels } sont explicitement exclues de la loi sur le copyright, en tant que méthodes de création.

c *Oracle v. Google :* des API comme « fonctionnalités »

Le cas *Oracle v. Google* reprend les attendus de *Altai* et en étend les conclusions (les { interfaces } font partie des SSO et ne sont donc pas protégées par le *Copyright Act*) aux interfaces de programmation.

Ce procès porte sur le statut de propriété intellectuelle d'une { API }. En 2005, Google cherche à développer un { système d'exploitation } pour ses téléphones mobiles, projet qui deviendra le système Android. Pour ce faire, des premières négociations ont lieu avec Sun, l'éditeur du { langage de programmation } Java, pour acheter une licence d'utilisation du { langage } et de ses fonctionnalités afin de développer leur propre système. Devant le refus de Java, les ingénieurs de Google utilisent tout de même ce { langage } comme base de travail et copient trente-sept des cent soixante-dix-sept {API } de Java. Oracle attaque Google en justice en mai 2010 pour atteinte au droit d'auteur (*copyright infringement*) en ce qui concerne la « structure, l'ordre et l'organisation » (*structure, sequence and organization*, ou « SSO ») de ses { API }. Après un premier jugement

699 Samuelson, Pamela. The Uneasy Case for Software Copyrights Revisited. *Op. cit.*, p. 1767-1770.

700 « *Programs interfaces were elements of programs that copyright law did not protect* ». Samuelson, Pamela. *Idem*, p. 1772.

rendu en mai 2012 par le juge William H. Aslup en faveur de Google, un appel dénié par la Cour Suprême puis un second jugement, la Cour Fédérale du district de Californie a définitivement statué en mai 2016 en faveur de Google, estimant que ces derniers avait eu un « usage raisonnable » (*fair use*) de l'{ API } Java et qu'il n'y avait donc pas de préjudice porté à Oracle[701].

API, code, développeur, interface, interopérabilité, logiciel, *subroutine*, variable

Voir glossaire tome II, p. 3-25

I 2 *subroutine*

Voir p. 111

La question centrale est de savoir si une { API } – en l'occurrence celle de Java – peut être protégée par la loi sur le copyright, tant dans sa structure que dans le nom des composantes de l'{ API }. Oracle s'appuie principalement sur deux arguments : l'affaire *Whelan v. Jaslow* de 1983 qui portait sur la copie de { *subroutines* } d'un { logiciel } et pour laquelle la Cour avait jugé que les { *subroutines* } pouvait être considérée comme des « [...] œuvres littéraires conventionnelles[702] » ; la part créative de l'écriture d'une { API }, qui peut donc faire entrer l'{ API } sous le régime de la propriété intellectuelle en tant qu'œuvre de l'esprit. À ces arguments, Google oppose principalement les deux arguments suivants : ce n'est pas *Whelan v. Jaslow* qui a fait jurisprudence, mais bien *Computer Associates International v. Altai*, où il a été décidé que les { interfaces } entre machines et / ou { logiciels } ne tombent pas sous le coup du *Copyright Act* de 1976, car elles font partie de la « structure, séquence et organisation » de ce travail. De plus, l'interface de programmation Java fait partie d'une « contrainte externe » à laquelle ont du s'astreindre les { développeurs } de Google pour favoriser l'{ interopérabilité } et à ce titre, cette contrainte ne peut être mise sous copyright au nom du paragraphe 102b de la *Copyright Law*. L'{ API } est une « [...] nécessité fonctionnelle pour permettre la compatibilité entre programmes informatiques[703] ».

701 Pour un résumé factuel du procès, voir (en anglais) : Oracle America, Inc. v. Google, Inc., 2017. *Wikipédia*. [En ligne] [mis en ligne le 10 mai 2014] [dernière modification le 24 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Oracle_America,_Inc._v._Google,_Inc. Pour une analyse de la première décision de la Cour Fédérale, voir Samuelson, Pamela. Oracle v. Google: Are APIs Copyrightable? *Op. cit.*, p. 25.

702 « [...] *conventional literary works* ». Samuelson, Pamela. *Idem*, p. 26.

703 « [...] *functionnal requirements for achieving compatibility with other programs* ». Samuelson, Pamela. *Idem*, p. 27.

Tout l'enjeu est donc de déterminer si une { API } est plutôt de l'ordre de l'œuvre d'esprit, ou de l'ordre des méthodes de production d'autres écritures. Si c'est le premier cas, une { API } est protégée par le droit d'auteur. Dans l'autre cas, elle ne l'est pas. Après un appel, la justice états-unienne finira par donner raison à Google, en établissant que :

a) les { API } ne sont pas une œuvre de l'esprit mais font partie de la structure d'un { logiciel }, les fameuses « SSO » (« structure, séquence ou organisation ») qui étaient déjà le nœud du procès *Computer Associates International v. Altai*. La jurisprudence *Altai* se voit donc étendue aux { API } ;

b) Les noms des éléments d'une { API }, par exemple le nom des { variables }, des méthodes d'accès aux données, ne peuvent pas non plus être protégés par le *Copyright Act*, car ce sont « [...] plus que de simple noms – ce sont des symboles dans une structure de commande[704] ». Cette décision est par ailleurs cohérente avec le statut du { code } dans la législation états-unienne, considéré à la fois comme commande destinée à une machine et comme texte lu par des humains résultant d'un acte de création intellectuelle.

704 « [*the names are*] *more than just names – they are symbols in a command structure* ». *Ibidem*.

d Des interfaces de programmation mises au service de l'innovation

API, interopérabilité, logiciel, matériel

Voir glossaire tome II, p. 3-25

Que nous apprend ce passage par la législation états-unienne sur le copyright et les { API } ? Qu'il en va de la défense de l'{ interopérabilité } et *in fine* du développement économique de l'industrie informatique. La plupart des combats juridiques tournent autour de cette question, pour finalement aboutir à une disqualification du modèle du copyright, au profit de l'{ interopérabilité } des composants informatiques, tant { matériels } que { logiciels[705] }. Comme le résume Pamela Samuelson, « [...] l'approche de la loi états-unienne a produit l'idée aujourd'hui partagée que les interfaces étaient essentielles pour l'interopérabilité et qu'elles ne peuvent pas être protégées par la loi sur le copyright (bien qu'elles puissent être parfois brevetées)[706] ». Toujours selon Samuelson, dans la continuation des recommandations de Breyer, toutes ces décisions favorables à l'{ interopérabilité } ont largement contribué au développement économique de l'industrie informatique et notamment celle du { logiciel[707] }.

Ce faisant, elles instrumentalisent l'{ interopérabilité } et les interfaces de programmation, au moins depuis la juridiction *Oracle v. Google*. En effet, ces décisions tendent à ériger l'{ interopérabilité } comme une valeur supérieure à la propriété intellectuelle, ce parce qu'elle favorise le développement économique. Économie et technique sont alors intrinsèquement liées : être interopérable n'est pas seulement une propriété technique, c'est un moteur de croissance économique. Les { API }, en tant qu'{ interfaces } (cf. *Oracle v. Google*), favorisent l'{ interopérabilité } et donc l'innovation. Elles se chargent d'une valeur qui est tout autant technique qu'économique. Leur fonction initiale de communication entre composants informatiques se trouve instrumentalisée, au bénéfice d'une valeur économique (l'innovation) et selon la structuration d'un secteur industriel (l'informatique).

705 Samuelson, Pamela. The Past, Present and Future of Software Copyright Interoperability Rules in the European Union and United States. *European Intellectual Property Review*. 2012, vol. 34, nº 4, p. 230. Orientation états-unienne qui rejoint d'ailleurs celle de la législation européenne sur la question. En 1991, la *Software Directive* (« Directive sur le logiciel »), pousée par le lobby ECIS (*European Committee for Interoperable Systems* , ou « Comité Européen pour les Systèmes Interopérables »), composé d'industriels de l'informatique comme Sun ou Olivetti cherche à faire reconnaître que les copies de code et / ou d'interfaces ne doivent pas être considérées comme du plagiat.

706 « [...] *the US common law approach produced the now prevailing consensus that interfaces essential to interoperability are unprotectable by copyright law (although they are sometimes patented)* ». Samuelson, Pamela. *Idem*, p. 235.

707 Samuelson, Pamela. *Idem*, p. 234.

3 C Des API web à une « feuille de route technologique » pour le gouvernement français : la valorisation symbolique des API comme vecteurs d'ouverture

La première étape de cette trajectoire d'instrumentalisation nous a fait passer par la loi états-unienne et a montré comment les { API } sont devenues vecteur d'innovation. Dans cette seconde étape, nous allons montrer, à travers l'analyse du rapport gouvernemental « Les développeurs, un atout pour la France » (IV. 3. C. a), que les { API } deviennent des outils d'« ouverture » dans un contexte politique et institutionnel érigeant cette notion comme valeur de l'action publique (IV. 3. C. b).

a « Les développeurs, un atout pour la France » : un rapport entre économie et politique

« Les développeurs, un atout pour la France[708] » est un rapport remis en mars 2014 au ministère des petites et moyennes entreprises, de l'innovation et de l'économie numérique par Tariq Krim, entrepreneur français alors « Vice-Président écosystème et innovation » du Conseil National du Numérique (CNN). Ce rapport a été commandé à Tariq Krim par la Ministre Fleur Pellerin, dans le cadre plus global du projet « French Tech » lancé en novembre 2013[709]. Il s'agit d'un projet politique de développement des start-up françaises spécialisées dans le numérique, en sélectionnant des zones urbaines particulièrement propices.

Dans ce rapport, Tariq Krim mentionne deux fois les { API }. La première fois, c'est comme moyen d'« ouvrir des opportunités d'affaires » aux start-up françaises. Krim soutient que le modèle du financement direct est insuffisant et qu'il faut surtout permettre aux entreprises françaises du numérique d'« [...] expérimenter et déployer leurs solutions "dans le monde réel"[710] ». Pour ce faire, il propose de généraliser « [...] l'accès à des jeux de données et des interfaces de programmation (API) autour des-

708 Krim, Tariq. *Les développeurs, un atout pour la France* Rapport adressé au Ministère de l'Économie, du Redressement productif et du Numérique, 6 mars 2014.

709 French Tech est un projet de « labellisation de Métropoles French Tech » qui a pour ambition de « structurer l'écosystème français sous la forme d'un réseau rassemblant Paris et 13 autres écosystèmes remarquables qui sont les têtes de pont de la French Tech en régions ». Voir Gouvernement français. La French Tech : une ambition collective pour les start-up françaises. [En ligne], 2017. [Dernière modification le 15 mai 2107] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.gouvernement.fr/action/la-french-tech-une-ambition-collective-pour-les-start-up-francaises.

710 Krim, Tariq. *Op. cit.*

quelles des start-up pourraient développer de nouveaux services[711] [...] ». L'argument est celui qu'en permettant l'accès à plus de données, on encourage l'innovation puisqu'on donne plus de « matériaux » aux jeunes entreprises qui n'ont pas les moyens économiques ou technologiques de disposer de ces données. On encourage ainsi la réappropriation des données et leur circulation.

API, interopérabilité

Voir glossaire tome II, p. 3-25

La seconde occurrence du terme { API } apparaît dans la « feuille de route technologique pour l'État, les ministères et les opérateurs publics ». L'argument de Tariq Krim est le suivant : les récentes innovations technologiques (Internet mobile, objets connectés, HTML5, « données massives ») ne sont pas une évolution mais un « [...] nouveau cycle technologique[712] » dont le gouvernement n'a pas pris la pleine mesure, notamment dans le domaine de la santé, de l'éducation et de l'énergie. L'auteur propose alors six « axes technologiques », dont l'« [...] utilisation et ouverture d'accès aux données grâce à des interfaces de programmation (API) qui permettent notamment d'ouvrir facilement l'accès à des applications mobiles[713] [...] ».

711 Krim, Tariq. *Idem*, p. 11.

712 Krim, Tariq. *Idem*, p. 14.

713 *Ibidem*

b Les API comme outils d'une « ouverture »… et d'une « transparence » ?

Comment interpréter ce discours d'escorte des { API } ? Ces dernières sont tout d'abord réinsérées dans un discours devenu courant sur la technique comme « révolution » auxquelles les politiques publiques devraient s'adapter. Surtout, ce discours tend à rapprocher les { API } d'une fonction d'« ouverture » : ouverture aux données, ouverture de l'accès… L'ouverture est vue ici comme une valeur positive, euphorique, au contraire de données qui seraient fermées, propriétaires et inaccessibles, représentation particulièrement puissante dans le champ de l'action publique[714]. Les données informatiques en premier lieu, gagnent à être « ouvertes[715] », c'est-à-dire publiées, rendues disponibles, lisibles, réappropriables. Les { API } sont un des moyens de réaliser cette « ouverture » et cette « transparence ».

Par cet ensemble de discours, dont le rapport de Tariq Krim est un exemple, on comprend comment les { API }, d'un outil utile pour assurer l'{ interopérabilité } logicielle et / ou machine, en sont devenues à être des gages « d'ouverture » dans un contexte plus global où cette notion est devenue une valeur euphorique ambiante. Il faut « ouvrir » ce qui est fermé, rendre visible ce qui est invisible, participer à la « transparence » de la vie publique. On peut supposer que la technique informatique, parce qu'elle charrie un imaginaire de la « boîte noire » et du secret[716], est une cible fantasmatique de choix. Il faut « ouvrir » des machines qui manipulent des données phénoménologiquement inaccessibles à l'humain. Parmi ces solutions d'ouverture, on trouve les { API }, qui sont alors redéfinies symboliquement : elles participent à l'{ interopérabilité } (plan technique), à l'innovation (plan économique) et à l'ouverture (plan symbolique).

714 Qu'on pense à l'initiative récente de l'« Open Governement », lancée en 2011, à laquelle participe la France et qui donne lieu à de nombreuses initiatives centrées sur « l'ouverture des données publiques », dont la dernière en date fut un « *data camp* » à l'Assemblé Nationale le 5 décembre 2016. Voir Chevalier, Paul-Antoine. Retour sur le 1er data camp de l'Assemblée Nationale. Dans : *Le blog d'Etalab* [en ligne]. Billet du 5 décembre 2016. [Mis en ligne le 5 décembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.etalab.gouv.fr/retour-sur-le-1er-data-camp-de-lassemblee-nationale ; Mabi, Clément. *Le débat CNDP et ses publics à l'épreuve du numérique : entre espoirs d'inclusion et contournement de la critique sociale*. Thèse de doctorat. Compiègne : Université de Technologie de Compiègne, 2014.

715 Goëta, Samuel. *Op. cit*.

716 Souchier, Emmanuël. L'écrit d'écran, pratiques d'écriture & informatique. *Op. cit.*, p. 118.

3 D Les API web et l'interopérabilité comme « partage »

API, format, HTML, interface, interopérabilité, langage de programmation, standardisation, Web, *widget*

Voir glossaire tome II, p. 3-25

Jusqu'ici, nous avons parlé des { API } en général et non pas spécifiquement des { API } web, qui sont l'objet propre de notre travail. En tant qu'{ interfaces }, les { API } participent à l'{ interopérabilité } informatique. En cela, elles participent à structurer un marché économique fondé sur l'innovation, innovation protégée par la loi. La dernière étape de notre parcours fait l'hypothèse que les { API } sont devenues des outils de partage. Pourquoi ? Nous étudions deux pistes. La première, c'est que le développement du { Web } repose déjà sur la concaténation entre transmission d'informations entre machines – autre façon de parler de l'{ interopérabilité } – et partage entre humains. Garantir l'{ interopérabilité }, c'est garantir le partage (IV. 3. D. a). Les { API } web s'insèrent dans ce contexte, mais pas uniquement. Notre seconde piste porte sur le discours tenu par Facebook et Twitter sur les { *widgets* }. Comment ces formes sont-elles qualifiées de « sociales » (IV. 3. D. b) et de quel social parlons-nous (IV. 3. D. c) ? Ces deux pistes explorées, nous pourrons alors conclure cette troisième sous-partie.

a Web et interopérabilité

Il faut commencer par rappeler que l'{ interopérabilité } est un enjeu particulièrement sensible sur le { Web }. Cela se voit dans le choix de { formats } ouverts promu par le *World Wide Web Consortium* (W3C), l'organisme de { standardisation } du { Web[717] }. Le W3C est ouvert à tout membre désirant rejoindre l'organisation – moyennant finances – mais regroupe majoritairement des industriels du { Web }. Fondé par Tim Berners-Lee en octobre 1994, la tâche principale du W3C est de « [...] standardiser le HTML dans un format gratuit qui allait permettre à n'importe quel navigateur conforme aux standards d'accéder aux pages web[718] ». En concurrence avec l'*Internet Engineering Task Force* (IETF), le W3C finira par s'imposer comme l'organisme principal de { standardisation } du { Web }, notamment grâce à l'imposition du { format } { HTML }.

717 Halpin, Harry. La souveraineté numérique. *Multitudes*. 2009, nº 35, p. 201-213 ; Sire, Guillaume. *Concevoir le Web et ses conflits* [Article en pré-publication]. À paraître.

718 Halpin, Harry. La souveraineté numérique. *Op. cit.*, p. 210.

Les travaux sur le W3C et la { standardisation } du { Web[719] } montrent que l'{ interopérabilité } est une des valeurs cardinales du réseau[720]. Si tous les standards du { Web } ne sont pas nécessairement compatibles avec d'autres { langages }, il est certain que l'{ interopérabilité } agit comme un imaginaire fondateur du réseau, un horizon qu'il convient de réaliser à travers la normalisation du { Web }. Chaque document du W3C spécifie ainsi en page de garde que « ce document œuvre à l'intéropérabilité du Web ». Concrètement, le choix de l'{ HTML } comme { langage } de base du { Web } est représentatif : il est un { format } ouvert – c'est-à-dire non propriétaire ; il n'appartient pas à une entreprise en particulier et ne peut donc faire l'objet d'un brevet ou d'un copyright – bien documenté ce qui permet à tout navigateur de lire et d'afficher une page web.

L'essor des { API } web dans les années 2000 s'appuie sur cet imaginaire d'un réseau devant être « ouvert » et interopérable[721], imaginaire qui se réalise dans la négociation de certains { formats } et { langages }. Les { API } participent *a priori* à l'{ interopérabilité } du { Web }, en ceci qu'elles permettent la circulation des textes et des données, en donnant à lire la structure informatique de ces données et en en permettant l'accès. Leur essor ne peut donc se comprendre que si on les replace en amont dans une quête d'{ interopérabilité } qui est celle du { Web } et de sa { standardisation }.

Ce que montre Robert Bodle, c'est qu'avec les { API } web de Google, Amazon ou encore Facebook, l'{ interopérabilité } endosse deux fonctions. La première est techno-sémiotique : l'{ interopérabilité } devient une « [...] nouvelle façon pour les gens d'interagir entre eux[722] ». La seconde est économique : l'{ interopérabilité } est « [...] utilisée stratégiquement [par les entreprises] afin d'établir leur domination du marché[723] ». Ces deux fonctions sont simultanées : l'{ interopérabilité } a des enjeux à la fois techno-sémiotiques et économiques.

719 Halpin, Harry. La souveraineté numérique. *Op. cit.*, p. 201-213 ; Flanagin, Andrew J., Farinola, Wendy Jo Maynard et Metzger, Miriam J. The technical code of the internet/world wide web. *Critical Studies in Media Communication*. 2000, vol. 17, nº 4, p. 409-428.

720 Bodle, Robert. Regimes of Sharing. *Op. cit.*, p. 324.

721 Bodle, Robert. *Idem*, p. 325.

722 « [...] *interoperability, enabled by Open API's, provides new way for people to interact with one another* ». Bodle, Robert. *Idem*, p. 321.

723 « [...] [*interoperability*] *is used strategically* [...] *in order to establish market dominance* ». Bodle, Robert. *Idem*, p. 328.

API, développeur, documentation, interface, interopérabilité, petites formes, *plugin*, programmation, signe passeur, Web, *widget*

Voir glossaire tome II, p. 3-25

Les { API } web sont un moyen pour les grandes plateformes du { Web } d'assurer une position dominante sur le marché. Sur quel marché ? Le marché des données utilisateurs et de leur monétisation pour des annonceurs publicitaires. Sans entrer dans les détails de ce modèle économique que certains nomment « *digital labor*[724] », l'idée est la suivante : les activités scripturaires réalisées sur les réseaux dits sociaux – comme cliquer sur un bouton *Like*, évaluer une traduction automatique, partager un statut – font partie d'une chaîne de production de valeur au profit des plateformes qui organisent ces activités. Dans ce contexte, la diffusion de formes textes outillant ces activités (bouton *Like*, « Partager » et plus largement toutes les { petites formes } de notre corpus) permet de recueillir plus de données et d'élargir la portée de ce travail quotidien des internautes. De ce point de vue, les { API } web servent effectivement à établir une domination du marché du *digital labor*.

La fonction techno-sémiotique nous intéresse plus fondamentalement. L'{ interopérabilité } est une « façon pour les gens d'interagir entre eux ». Voilà une formule qui prend acte – tout en l'éludant – d'un saut entre une activité de lecture (le clic) et une activité sociale (le partage). Ce passage est construit, notamment par une { interface } qui se veut « sociabilisante » (au sens où elle met en rapport des « profils » qui donnent une image de l'identité de l'internaute) mais également par une infrastructure technique qui permet la circulation et l'échange des fragments identitaires construits par l'{ interface }. Il est aussi construit par un discours que portent les plateformes (Facebook et Twitter) sur les formes que leurs { API } permettent d'écrire.

b Les *widgets* : simplifier la propagation des formes… et du social ?

Comment Facebook et Twitter qualifient leurs { *widgets* } ? Le discours de la { documentation } peut se résumer en deux points : simplicité d'utilisation ; équipement de la sociabilité. La différence entre les deux firmes va résider dans le statut de cette sociabilité : soit elle est préexistante et doit simplement trouver une expression sur les réseaux socionumériques (Facebook), soit elle doit être provoquée. Les { petites formes } servent alors à exhorter à la sociabilité, notamment par l'appel à un sentiment esthétique (Twitter).

724 Cardon, Dominique et Casilli, Antonio A. *Op. cit.*

La simplicité d'utilisation est une constante de notre corpus. Facebook promet grâce à ses boutons J'aime ou Partager de « rendre simple le social[725] » et ce « en un clic[726] ». Les *posts* ancrés sont un « [...] moyen simple de mettre des publications sur votre site web[727] ». Twitter prétend de son côté « rendre le partage facile[728] » en intégrant un bouton « Tweet » dans une page web. Mieux que ça : ce bouton permet de créer une « [...] expérience prête-à-partager[729] ». Cette simplicité est ambivalente : elle porte à la fois sur la { programmation } – soit le travail des { développeurs } – et le « partage » – soit l'activité des internautes. L'{ API } est à la fois un moyen simple d'écrire des pages web, mais également un moyen simple de produire des textes sur le réseau.

D'ores et déjà, on voit que ces formes se chargent d'un pouvoir : celui d'être synonyme d'une activité sociale ou plus précisément d'équiper la sociabilité des internautes. C'est le second point commun entre le discours de Facebook et Twitter. Facebook par exemple affirme que l'{ API } « [...] donne plus d'opportunités de partager[730] » ou encore « [...] perme[t] aux gens de facilement partager[731] ». Les { petites formes }, rassemblées sous l'appellation de « *plugins* sociaux », sont « [...] le moyen le plus facile de rendre votre application ou votre site social[732] ». Voilà résumé en une phrase l'argument principal de ces plateformes : les formes que permet de produire l'{ API } (boutons, { *plugin* } page, cartes Twitter, publications ancrées, etc.) « socialisent » les pages web sur lesquelles ces formes apparaissent. Rien d'étonnant par ailleurs quand on comprend les { API } comme des outils de développement économique et symbolique d'industries définies comme des « réseaux sociaux ». De ce point de vue, les { API } de Facebook et Twitter participent à établir ces plateformes comme sociales, en promouvant l'idée qu'un clic – sur un { signe passeur } comme un bouton « J'aime » – est une activité sociale.

725 « *Social made simple* ». developers.facebook.com/products/social-plugins (capture réalisée le 28 juillet 2015).

726 facebook.com/docs/plugins?local=FR (capture réalisée le 29 juillet 2015).

727 « *Embedded posts are a simple way to put public posts* [...] *into the content of your web site* [...] ». developers.facebook.com/docs/plugins/embedded-posts (capture réalisée le 29 juillet 2015).

728 « *make sharing easy* ». dev.twitter.com/web/overview/best-practices (capture réalisée le 14 juillet 2015).

729 « [...] *a ready-to-share experience* », dev.twitter.com/web/overview/best-practices (capture réalisée le 14 juillet 2015).

730 « [...] *giving users more opportunities to share* ». developers.facebook.com/products/sharing (Capture réalisée le 29 juillet 2015).

731 « *Make it easy for people to share* ». developers.facebook.com/products/sharing (Capture réalisée le 29 juillet 2015).

732 « *Social plugins : the easiest way to make your app or your website social* ». developers.facebook.com/products/social-plugins (Capture réalisée le 28 juillet 2015).

c Deux API, deux visions différentes de la sociabilité

API, application, développeur, graphe, graphe social, petites formes, tweet

Voir glossaire tome II, p. 3-25

La différence entre Facebook et Twitter réside dans la nature de cette sociabilité. D'un côté, on trouve dans l'{ API } de Facebook l'idée qu'il y aurait des activités qui seraient sociales (aimer un livre, publier une vidéo, aller au restaurant) et que ces activités peuvent trouver une traduction sur le réseau, notamment parce qu'elles peuvent être représentées sous forme de { graphe }. C'est l'idée centrale de ce que Facebook appelle des « histoires » (*stories*). Une « histoire » est par exemple le fait qu'une personne X est allée au restaurant Y, a écouté telle chanson, a vu tel film…[733] Utiliser l'{ API } de Facebook, parce qu'elle est structurée en amont par le { graphe social }, permet d'écrire ces « histoires ». Mais ces histoires, ou plutôt ces activités, préexistent à Facebook, qui ne fait que – ou ne prétend que – « traduire » ces activités en les donnant à lire d'une certaine manière : comme des histoires que l'on peut intégrer à un { graphe social }. La sociabilité selon Facebook est une histoire à cartographier.

III 3 A a
graphe social
Voir p. 311

Twitter en revanche promeut la vision d'un social comme processus, comme ce qui doit être créé par des actions telles que le partage ou la publication. Les { petites formes } jouent dans ce schéma le rôle de catalyseur, de déclencheur de la sociabilité. Ainsi, on trouve de multiples occurrences de verbes d'action, directement liées aux { petites formes } : les cartes Twitter « […] créent de l'engagement pour vos { tweets[734] } » ou « […] déclenchent le téléchargement[735] » d'une { application }. Les boutons « Tweeter » et la façon dont ils préparent le partage d'un lien sont « une façon puissante de donner la possibilité à vos internautes de partager votre contenu[736] », le bouton « Mentionner » « […] encourage un nouveau { tweet[737] } ». Plus largement, les { petites formes } Twitter « […] encourage[nt] votre public sur Twitter à partager votre contenu et à s'abonner à votre profil[738] ». Nous sommes en plein dans cette logique de délégation : l'{ API } permet de produire des formes qui donnent du pouvoir aux internautes. Mais elles ne font pas que donner du pouvoir :

IV 1
logique de délégation
Voir p. 343

733 developers.facebook.com/docs/sharing/opengraph. Capture réalisée le 28 juillet 2015.

734 « […] *drive engagement from your tweets* ». dev.twitter.com/cards/overview. Capture réalisée le 14 juillet 2015.

735 « […] *drive downloads for your app* ». dev.twitter.com-cards-types-summary. Capture réalisée le 14 juillet 2015.

736 « *a powerful way to enable your users to share your content* ». dev.twitter.com/web/intents. Capture réalisée le 14 juillet 2015.

737 « […] *to encourage a new tweet* ». dev.twitter.com/web/tweet-button/mention-button. Capture réalisée le 14 juillet 2015.

738 « […] *to encourage your twitter audience to share your content and subscribe to your Twitter account updates* ». dev.twitter.com/web/overview. Capture réalisée le 14 juillet 2015.

elles sont présentées par Twitter comme exhortant au clic. Elles « déclenchent », elles « créent », elles « encouragent »… Toutes ces formes sont qualifiées d'agissantes, provoquant certaines activités (des clics), lesquelles sont ensuite enregistrées et sémiotisées par la plateforme comme « lecture », comme « engagement » ou comme « consommation ». Ces formes n'encodent donc pas un social préexistant mais sont faites pour déclencher le processus de sociabilité : publication, partage, lecture. La sociabilité pour Twitter sont des émotions à déclencher.

Comment sont-elles déclenchées ? Twitter insiste sur la dimension esthétique des formes. Il est ainsi dit que les cartes Twitter permettent une « […] magnifique expérience de consommation[739] », elles sont un moyen d'« enrichir » le texte par des vidéos, des photos, par exemple pour engendrer du trafic sur un site web[740]. Plus généralement, les ressources mises à disposition des { développeurs } par Twitter *via* son { API } « […] aident à créer une expérience charmante et sécurisée pour nos utilisateurs en commun[741] ». Ici, Twitter s'adresse directement aux { développeurs }, en explicitant le double public des { API } : les { développeurs } et les internautes. Ces derniers, en tant qu'ils utilisent Twitter mais aussi en tant qu'ils cliquent sur des boutons ou téléchargent des { applications } qui utilisent leurs données, ont affaire tant à Twitter qu'aux { développeurs } externes. Il s'agit donc de préserver les utilisateurs et de faire de leur clic et de leurs lectures non pas seulement une activité « sociale », mais une activité « charmante », agréable à l'œil. Cette dimension esthétique participe selon nous de cet « appel à l'écriture » que représentent les { petites formes } des écrits de réseau.

III **2** **B** **b**
appel à l'écriture
Voir p. 300

739 « […] *beautiful consumption experience* ». dev.twitter.com/cards/overview. Capture réalisée le 14 juillet 2015.

740 « *You can attach rich photos, videos and media experience to your tweets that drive traffic to your site* ». dev.twitter.com/cards/overview. Capture réalisée le 14 juillet 2015.

741 « *The following ressources will help you to create a safe and delightful experience to our mutual users* ». dev.twitter.com/overview/general. Capture réalisée le 14 juillet 2015.

API, documentation, interface, interopérabilité, petites formes, Web

Voir glossaire tome II, p. 3-25

Twitter, autant que Facebook, développent donc *via* la { documentation } de son { API } un large discours autour des { petites formes } qui en vient à les parer de vertus économiques et surtout socialisantes : les boutons sont appelés des « modules sociaux » qui « simplifient le social ». Il est clair que cette construction n'est pas l'apanage de n'importe quelle { API }, mais que la délimitation de notre corpus à des services de réseaux « sociaux » explique que nous nous trouvions en présence de cette socialisation des formes. À partir du moment où les { petites formes } générées par l'{ API } mettent en circulation des fragments textuels qui sont autant de fragments d'une identité polyphonique, il est compréhensible que ces formes soient qualifiées de « sociales ».

Nous voici rendu au cœur de la relation techno-sémiotique que nous voulions analyser : les { API } web de Facebook prétendent « équiper le social » et le « rendre simple », précisément parce qu'elles reposent sur une instrumentalisation de l'{ interopérabilité } comme condition du partage, ce partage étant entendu comme une activité de communication. Considérer qu'un bouton Partager est un « outil du social » est donc acter cette instrumentalisation de l'{ interopérabilité }, la considérer comme allant de soi alors qu'elle résulte d'une longue élaboration, comme nous venons de le montrer. Élaboration qui construit un complexe techno-sémiotique (cliquer = partager ; interopérable = social) difficile à démêler et dont les { API } bénéficient symboliquement : elles sont des outils liés aux valeurs euphoriques du partage.

Conclusion

En ayant étudié la trajectoire des { API } comme « êtres culturels[742] », nous pouvons désormais répondre à notre question initiale. Comment les { API } web en sont venues à incarner des valeurs de sociabilité comme le « partage » ? Comment ont-elles été instrumentalisées comme outil de cette sociabilité ? Notre hypothèse était que c'est la fonction de communication – au cœur des { API } par la question de l'{ interopérabilité } – qui expliquait cette instrumentalisation. Ce sont en leur fonctionnement technique des outils de communication, au sens informatique du terme. C'est ce fonctionnement technique qui aurait fait l'objet d'une instrumentalisation.

Force est de constater que la situation est un peu plus complexe. Il est certain qu'une { API } – comme toute { interface } – a pour fonction première de permettre l'{ interopérabilité } informatique et que le sort des { API } est intrinsèquement lié à cette notion d'{ interopérabilité }. En revanche, si les { API } font l'objet d'une instrumentalisation par la loi états-unienne, c'est surtout en tant qu'outils d'écriture. Une écriture située dans une industrie, industrie dont le développement est assurée en partie par l'{ interopérabilité }. Dans ce contexte, les { API } ne sont pas mises au service de la communication entre les humains, mais plutôt au service de l'économie et de l'innovation. Les différentes décisions judiciaires prises dans les années 1980-90 (*Computer Associates International v. Altai*) et 2000 (*Oracle v. Google*) vont dans ce sens. Le couple { API } / { interopérabilité } prend une autre direction lorsqu'on s'intéresse plus particulièrement aux { API } web. Le réseau amène en effet son lot d'imaginaires techniques, dont celui d'un échange d'information facilité qui passe par la mise au point de standards. Mais il existe un flou quant à savoir si cet échange concerne les machines ou les êtres humains. C'est ce flou qui entretient la possibilité d'une instrumentalisation de l'{ interopérabilité }. De plus, le développement du { Web } autour de grandes plateformes aux prétentions sociabilisantes (Facebook, Google, etc.) conduit les { API } web à être les outils d'une « sociabilité rendue facile », en plus de fournir un avantage compétitif à ces plateformes engagées dans une bataille autour des données et de leur monétisation. Enfin, l'{ API } en tant que modalité d'accès à des données informatiques trouve une autre utilisation comme levier « d'ouverture » des données, notamment des données publiques dans un

742 Jeanneret, Yves. *Op. cit.*, 2008, p. 13.

API, forme-texte, interopérabilité, petites formes, standardisation

Voir glossaire tome II, p. 3-25

contexte politique favorisant ce désir « d'ouverture ». L'instrumentalisation se joue donc à trois niveaux : les { API } au service de l'innovation ; les { API } au service de l'ouverture ; les { API } au service du partage. On ne peut pas pour autant dire que ces trois niveaux découlent de la fonction initiale de communication d'une { API }. En revanche, on peut dire que l'intéropérabilité joue un rôle clé dans toutes ces instrumentalisations.

On comprend dès lors comment ces instrumentalisations participent à une « économie des passages » où la valeur se crée par la circulation. { interopérabilité }, innovation, partage, ouverture… Autant d'avatars de la circulation selon qu'elle concerne des données, des idées ou des textes. Par ces réappropriations, les { API } prennent donc une place singulière dans le déploiement de cette économie. Elles en sont non seulement des rouages essentiels de dissémination, notamment en ce qui concerne les { formes-textes }, mais elles participent à instaurer ces formes textes comme sociables. L'instrumentalisation des { API } joue donc un rôle dans la valorisation tant symbolique qu'économique des textes qu'elles permettent de produire et de faire circuler.

Cette circulation ne va pas sans poser problème. Si les { API } web tiennent de l'ingénierie technique, symbolique et économique de la circulation, cela pose en creux le fantôme de l'altération. L'altération est une autre figure de la circulation des textes et des données, cette fois du côté dysphorique : perte de données, vieillissement du support d'inscription… Accroître la circulation des textes, de ce point de vue, ce n'est pas seulement accroître le partage et rendre le monde plus « ouvert ». C'est aussi user les textes, augmenter leur faculté à se métamorphoser, à s'altérer, à devenir illisibles, aussi bien pour les machines (corruption de données) que pour les humains (perte de sens). Or, c'est cette lisibilité qui fonde la valeur des « { petites formes } » produites par les { API } web. Comment alors préserver, malgré une circulation accrue, l'intégrité des textes ? Et quels sont les bénéfices que Facebook ou Twitter tirent de cette préservation et de cette organisation ?

Au terme de ces instrumentalisations par lesquelles les { API } sont érigées comme outil de la sociabilité, reste un paradoxe. Le but des industries médiatisantes comme Facebook et Twitter est d'équiper et de maximiser la circulation des textes, cela maximise en même temps leur altération, menaçant donc leur lisibilité. Or, comme nous l'avons mentionné dans l'introduction de cette partie, le propre des pratiques lettrées est de garantir un certain seuil de conservation des textes, malgré leur diffusion et leur nécessaire altération. C'est pourquoi nous sommes face au paradoxe suivant : minimiser l'impact des différentes instanciations des textes (leurs ancrages dans différents contextes) demande un très gros travail éditorial de { standardisation }. D'où l'hypothèse que la { standardisation } visuelle des { formes-textes } est un antidote à l'altération induite par la circulation.

API, industrialisation, interopérabilité, standardisation, *widget*

Voir glossaire tome II, p. 3-25

Comment s'organise cette { standardisation } ? D'abord par des recommandations de mise en page, logos officiels, éléments graphiques des boutons… Si le texte s'ancre dans la page, il doit s'acquitter de certaines taxes, de « frais de mouillage » pour pouvoir apparaître (IV. 4. A). Deuxièmement, l'analyse du cas des { *widgets* } montrera que cette { standardisation } visuelle s'applique particulièrement à la polyphonie propre à tout texte et la valorise (IV. 4. B), cette dernière étant, dans le contexte d'une industrie des passages, un enjeu symbolique. Cette polyphonie s'organise toutefois au détriment de certaines entités : les machines qui permettent l'affichage et la reproduction du texte. Ce qui nous permettra d'analyser l'idéologie du texte sous-jacente à la notion d'{ interopérabilité } (IV. 4. C). Parce qu'elle repose sur une abstraction des conditions matérielles de production du texte, elle entraîne *in fine* un partage idéologique entre les acteurs ayant voix au chapitre (les humains) et ceux qui ne l'ont pas (les machines). Au-delà de l'{ interopérabilité } permise par les { API } et de la place centrale qu'elle occupe dans l'économie des passages se profile donc une question à la fois politique et sémiotique.

4 STANDARDISATION DES FORMES, FRAIS DE MOUILLAGE ET GESTION DE LA POLYPHONIE. LES MACHINES OUBLIÉES ?

D'expédient technique pour assurer l'{ interopérabilité } à outillage du « partage » entre êtres humains, les { API } web ont donc connu une promotion axiologique non négligeable. Mais cette promotion pose un problème majeur. Avec une circulation accrue s'accroissent en effet les phénomènes de transformation, de déperdition, de corruption. Cette altération[743] est un processus consubstantiel à la communication : toute mise en circulation entraîne reprise et transformation. Les textes, en tant qu'objets communicationnels, circulent, donc s'altèrent.

L'hypothèse faite par Yves Jeanneret d'une { industrialisation } de la trivialité revient donc à faire l'hypothèse d'une { industrialisation } de l'altération[744], ce qui est paradoxal. Plus des textes circulent, plus de la valeur se crée pour les industries médiatisantes organisant cette circulation *via* des outils comme les { API }. Or, plus les textes circulent, plus ils s'altèrent. Plus leur lisibilité est donc menacée ainsi que la place prépondérante qu'occupent Facebook et Twitter dans cette circulation. Car si nous sommes effectivement dans une industrie des passages, alors cette altération représente une perte de capital – économique tout autant que symbolique – pour les industries qui organisent la circulation. Comment donc concilier maximisation de la circulation et maîtrise de l'altération ? Comment préserver la lisibilité et la valeur des textes de réseau malgré leur circulation ?

743 Jeanneret, Yves. *Op. cit.*, 2008, p. 87.

744 Attention, la maîtrise de l'altération a existé avant l'émergence des industries médiatisantes contemporaines comme Twitter ou Facebook. Une bibliothèque, par exemple, est une institution dont la fonction est en partie de contrôler la circulation des textes et leur altération, notamment physique – d'où les problématiques de restauration et de conservation des œuvres, qu'elles soient imprimées ou manuscrites. En revanche, c'est la dimension industrielle de ces pratiques de maîtrise de l'altération qui est inédite.

Notre hypothèse est que les { API } web seraient un dispositif de { standardisation[745] }, définie comme « [...] l'intention et le développement matériel d'un processus de reproduction des formes du texte sur un média, que celle-ci soit artisanale, industrielle ou automatisée[746] ». Cette { standardisation } intervient comme antidote à l'altération. En standardisant les formes du texte, on peut en contrôler la circulation et l'altération. Quel rôle jouent donc les { API } dans la { standardisation } des { formes-textes } qu'elles permettent de faire circuler ?

API, *bug*, développeur, documentation, forme-texte, interopérabilité, machine computationnelle, petites formes, standardisation

Voir glossaire tome II, p. 3-25

Un premier élément de réponse tient dans les recommandations, voire Les injonctions, que l'on trouve dans la { documentation } des { API } de notre corpus. On y découvre que les { API } web sont le lieu d'une « pratique lettrée[747] » qui a pour fonction d'organiser et de standardiser la circulation des textes en contrôlant la reproduction fidèle de l'image du texte. Jeux d'icônes à utiliser, principes de design à respecter... La { documentation } met en place une véritable marche éditoriale que les { développeurs } doivent respecter sous peine de se voir couper l'accès aux données. Si la page est un port où s'ancre le texte au gré de son odyssée, les plateformes imposent donc certains « frais de mouillage » desquels doit s'acquitter le texte pour pouvoir s'incarner dans une page donnée (IV.4. A).

Ce contrôle éditorial a deux effets principaux. D'un côté, les { petites formes } sont des moyens de gérer une forte polyphonie énonciative, constitutive de tout texte, mais aujourd'hui rendue visible et valorisée. Le discours lisible dans la { documentation } des { API } valorise Facebook et Twitter comme entités qui organisent cette polyphonie (IV. 4. B). D'un autre côté, cette valorisation de la polyphonie se fait au détriment de certains acteurs, en l'occurrence les { machines computationnelles }. L'une des constantes de notre corpus, c'est en effet la minimisation, voire l'exclusion, de l'action des machines dans l'énonciation en cours dans les { petites formes }. L'outillage technique doit, malgré sa complexité et les éventuels { *bugs* } d'affichage, garantir la reproduction à l'identique du texte tel que voulu par des humains (utilisateurs et { développeurs }). Comme si, dans un dernier retournement du problème, le « partage » entre humains était devenu tellement central qu'il fallait réduire à la portion congrue les problèmes d'{ interopérabilité } et par là évacuer ce qui, dans l'image du texte, revient aux procédures computationnelles (IV. 4. C).

745 Mœglin, Pierre. À la recherche de l'industrialisation du tutorat à distance. *Op. cit.*, p. 251.

746 Jeanneret, Yves. *Op. cit.*, 2014, p. 10.

747 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 11-40.

4 A. Les frais de mouillage des petites formes

Comment une { API } web participe-t-elle à la { standardisation } des formes des textes de réseau ? À cette question, nous répondons par le concept de « pratique lettrée » emprunté à l'historien et anthropologue Christian Jacob (IV. 4. A. a) : tout texte fait l'objet d'un contrôle tant sur son aspect que sur ses contenus, en fonction d'attendus historiques et culturels propres à chaque contexte d'écriture. Dans notre cas, nous montrons qu'une { API } web est le lieu d'une pratique lettrée en ceci que s'y exerce – à travers la { documentation } – un contrôle sur la seule image du texte (IV. 4. A. b). Ce contrôle vise à garantir, malgré l'intense circulation des formes propre à l'industrie des passages contemporaine, le maintien de cette image qui est aussi une image de marque (IV. 4. A. c).

a L'API comme lieu d'une « pratique lettrée »

En posant la question de la { standardisation } des formes du texte, nous nous situons dans le champ des « pratiques lettrées » telles que définies par Christian Jacob : les opérations qui instituent l'existence matérielle d'un texte, sa lisibilité et ses modalités de circulation grâce à un double contrôle

> « […] *contrôle sur la forme et la lettre, visant à construire la lisibilité du texte, c'est-à-dire son intelligibilité, dans un horizon de réception défini par des usages éditoriaux, les règles du langage et du style, une bibliothèque, des savoirs et des traditions. Contrôle sur les contenus, en référence à des normes culturelles, telles que la clarté et la logique, l'orthodoxie, le droit et la loi, les valeurs partagées dans un milieu social, une idéologie dominante, la grille sémantique générale qui donne sens à l'expérience du monde pour une culture donnée*[748]. »

Ce double contrôle a également pour conséquence de fixer les critères de répétabilité d'un texte : « L'essentiel réside dans le point de vue culturel qui définit le champ de l'acceptable, le ratio de la perte et du gain inhérent à la transmission, entendue dans son double sens de mise en forme d'un héritage à transmettre et d'appropriation de ce qui est hérité du passé[749] ».

748 Jacob, Christian. *Idem*, p. 12.

749 Jacob, Christian. *Idem*, p. 19.

À quel moment, dans une société donnée, décide-t-on que c'est le **même** texte qui circule, malgré ses altérations et transformations ? Au nom de quels attendus culturels, de quelles représentations canoniques du texte ? Quelle est la part de l'« altération acceptable » dans la circulation des textes ?

API, développeur, documentation, petites formes, tweet

Voir glossaire tome II, p. 3-25

Dans le cadre des { API } d'industries médiatisantes comme Facebook et Twitter, cette notion de pratique lettrée est particulièrement importante en ceci qu'elle nous permet de poser deux questions. Si nous prenons comme hypothèse de travail que l'{ API } est le lieu d'une pratique lettrée, alors on peut supposer que c'est le premier type de contrôle décrit par Christian Jacob qui sera exercé : le contrôle sur « [...] la forme et la lettre, visant à construire la lisibilité du texte [...] ». Il en va en effet des industries médiatisantes de se désintéresser du « contenu » des textes, tant que ces derniers circulent[750]. Le contrôle que ces industries viseraient au travers de leurs { API } serait donc essentiellement celui sur la forme du texte, son « image[751] ». C'est ce que nous avons appelé les « frais de mouillage » des formes des textes de réseau.

Mais si ces formes sont faites pour circuler et donc amenées à se transformer, il est également fort probable que les { API } web fixent en partie ces critères de répétabilité qui nous mettent en présence du même ou de l'autre : ce { tweet } ou ce statut que nous voyons inséré sur une page web est bien le **même** que nous retrouverons sur Facebook ou Twitter et il reste le **même** si nous nous trouvons sur un autre site. L'{ API } serait donc le lieu d'une pratique lettrée en ceci qu'elle fixe – par sa { documentation } – ces bornes du répétable. C'est ce que nous traitons dans la partie IV. 4. A. c. de cette sous-partie.

b Les recommandations, ou frais de mouillage

Lorsqu'un navire veut s'amarrer en sécurité, il doit choisir un point sûr, à l'abri du vent, assez profond pour ne pas risquer de s'enliser… Ce lieu est appelé un « mouillage ». Par extension, on désigne en marine par « mouillage » la manœuvre d'amarrage dans ce lieu propice. Certains ports ou lieux de mouillage font payer des taxes pour pouvoir s'abriter : c'est ce qu'on appelle des « frais de mouillage ». Si, comme le suggère Illich[752], la page est devenue un « port » dans lequel s'ancre le texte, on peut

III 1 A
la page est devenue un port
Voir p. 252

750 Jeanneret, Yves. *Op. cit.*, 2014, p. 10.

751 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

752 Illich, Ivan. *Op. cit.*, p. 118.

alors filer la métaphore et penser que cet ancrage n'est pas gratuit. Il y a des « frais de mouillage » du texte, certaines procédures à respecter, certaines conditions posées, certaines taxes desquelles on doit s'acquitter… Sans quoi le texte ne pourrait s'amarrer et donc apparaître aux yeux du lecteur.

Ces frais de mouillage sont définis par Twitter ou Facebook dans certaines « recommandations » ou *guidelines*. Ainsi dans les règles d'utilisation de Facebook destinées aux { développeurs }, il précisé section 4 (« Encouragez des utilisations correctes », « *Encourage proper use* ») règle 2 : « Respectez l'aspect et les fonctionnalités de Facebook[753] ». Respecter l'aspect de Facebook, c'est notamment utiliser les icônes officielles, ne pas en changer la couleur, l'orientation, ne pas rajouter du texte sur ces icônes[754]… Autant de recommandations qui cadrent d'un point de vue graphique la production des { petites formes } et plus largement l'utilisation de l'{ API }.

Mais c'est Twitter qui est peut être le plus précis sur ces questions, en fournissant les « règles d'affichage » (*display requirements*) de son « capital de marque » (*brand assets*).

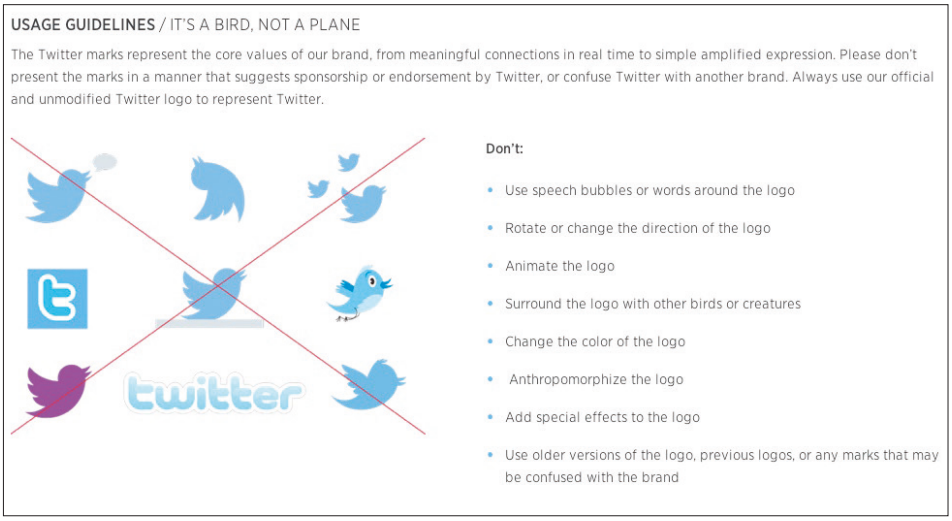


Fig. 44. Recommandations graphiques de Twitter (1). Capture d'écran de about.twitter.com/company/brand-assets. Capture réalisée le 14 juillet 2015 (détail).

753 « *Respect the way Facebook looks and functions.* ». developers.facebook.com-policy, capture réalisée le 29 juillet 2015.

754 Pour l'ensemble de ces recommandations, voir Facebook. Centre de ressources marketing [en ligne]. 2016. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.facebookbrand.com/.

Fig. 45. . Recommandations graphiques de Twitter (2). Capture d'écran de about.twitter.com/company/brand-assets. Capture réalisée le 14 juillet 2015 (détail).

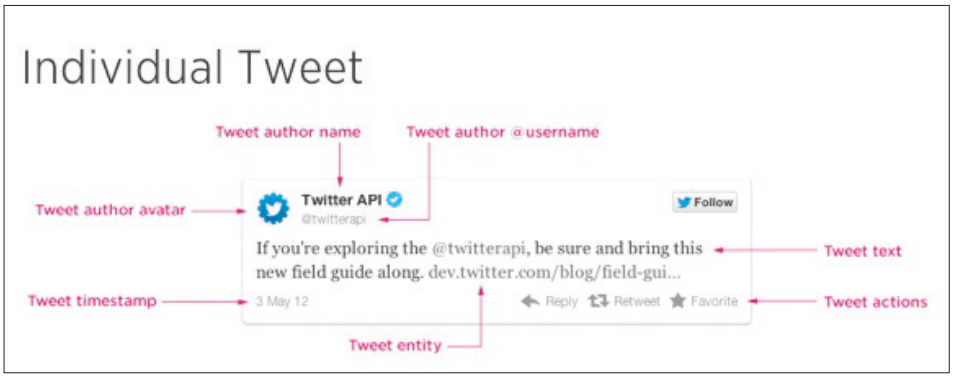


Fig. 46. Recommandations graphiques de Twitter (3). Capture d'écran de dev.twitter.com/overview/terms/display-requirements. Capture réalisée le 14 juillet 2015 (détail).

Les illustrations ci-dessus présentent quelques recommandations faites par Twitter. Il est demandé de : « […] toujours, utiliser le logo officiel et non-modifié de Twitter pour représenter Twitter[755] […] », de respecter la taille du logo, de « [t]oujours mettre en capitales les mots "Tweet" et "Retweeet"[756] […] ». Il est aussi précisé que « [p]eu importe où elles apparaissent, les images de marque officielles de Twitter doivent toujours fonctionner de la même façon que dans nos applications[757] ». Ces recommandations portent essentiellement sur la manière dont le texte apparaît

755 « […] *always use our official and unmodified Twitter logo to represent Twitter* […] ». about.twitter.com/company/brand-assets. Capture réalisée le 14 juillet 2015.

756 « [*Tweet and Retweet*] *must always be capitalized* […] » about.twitter.com/company/brand-assets. Capture réalisée le 14 juillet 2015.

757 « *No matter where they appear, official Twitter assets should always function the way they do in our apps.* ». about.twitter.com/company/brand-assets. Capture réalisée le 14 juillet 2015.

et sur la manière dont il est efficace techniquement (les { signes passeurs } sur Twitter doivent rester { signes passeurs } en dehors du site). On parle donc bien de certaines conditions d'ancrage du texte à respecter Et le mot condition est faible : ce sont plutôt des impératifs à respecter. En effet, si ces règles ne sont pas observées, Twitter se réserve le droit de couper l'accès à l'{ API } au { développeur } qui ne les respecterait pas[758]. Il s'agit donc bien de « frais de mouillage », au sens fort du terme : un tribut à payer pour pouvoir utiliser l'{ API } et certaines formes textuelles associées.

API, développeur, signe passeur

Voir glossaire tome II, p. 3-25

II | 1 | D | b
couper l'accès à l'API…
Voir p. 181

c L'enjeu : garantir l'image du texte

Pourquoi imposer ces conditions ? Quels en sont les enjeux ? Les exemples ci dessus nous ont mis sur une piste : le but de ces « frais de mouillage » est de garantir le maintien de l'image du texte malgré la circulation de ce dernier. L'enjeu de ce maintien est de ne pas mettre en danger l'image de marque des plateformes. Loin d'être un point négligé, la manifestation du texte, son ancrage, est quelque chose de particulièrement contrôlé, précisément parce qu'il y a là un enjeu symbolique et économique que les plateformes – agissant alors comme des marques défendant une image – intègrent dans leurs { API }.

Là encore, c'est Twitter qui se montre le plus clair sur le sujet, sur la page dédiée about.twitter.com/company/brand-assets. Il y est par exemple décrit comment leurs logos servent à « […] identifier la source ou l'origine de tous les produits Twitter[759] ». Un peu plus loin, il est dit du petit oiseau bleu, emblème de Twitter, qu'il « […] représente les valeurs fondamentales de notre marque » et qu'il ne faut donc pas en modifier l'aspect. Cela va jusqu'à l'espacement entre le logo et les mots qui suivent, espacement devant être de 0,5 fois le diamètre du cercle dessiné par l'oiseau bleu : « Afin de maintenir les exigences de la marque Twitter, l'espacement entre le logo et les *hasthags*, les mentions, les noms d'utilisateurs ou tout texte doit être précis[760]. »

758 Avec le principe d'authentification expliqué en II. 1. D. b., il est très simple d'identifier un développeur utilisant l'API, chacun ayant un numéro d'identification unique.

759 « [*the Twitter logo*] *identifies the source of origin of any Twitter's products* ». about.twitter.com/company/brand-assets, capture réalisée le 14 juillet 2015.

760 « *To maintain the standard of the Twitter brand, spacing between the logo and hashtags, usernames or texts need to be precise.* » about.twitter.com/company/brand-assets, capture réalisée le 14 juillet 2015.

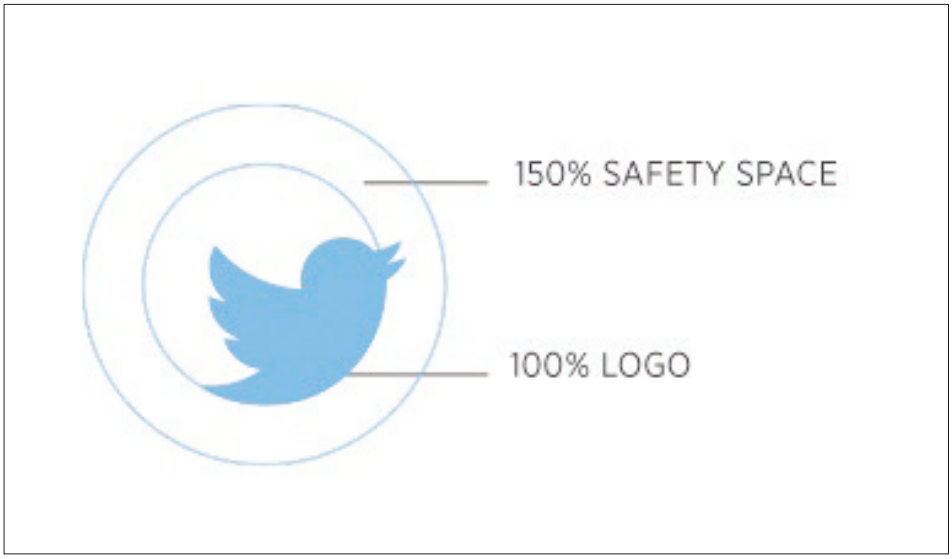


Fig. 47. La documentation de l'API : un contrôle de l'image du texte jusqu'aux moindres détails. Capture d'écran de about.twitter.com/company-brand-assets, réalisée le 14 juillet 2015 (détail, profil personnel).

Garantir l'intégrité formelle du texte est donc corrélé à la question de l'image de marque des industries médiatisantes organisant la circulation de ce texte. Cette image de marque passant par les logos et autres icônes, qui incarnent les « valeurs fondamentales » de Twitter et fait partie de « l'expérience » qu'elles produisent, il faut que ces logos eux aussi subissent le minimum d'altérations[761]. D'où cet ensemble de recommandations et de précautions, qui pose toutefois une nouvelle question. Qu'est-ce qui est mis en avant dans cette pratique lettrée ? Quelle vision du texte est promue ? En somme, quels sont les bénéfices symboliques de ce contrôle éditorial ?

761 Nous faisons la suggestion que ce contrôle sur l'image du texte a partie liée avec la délégation de l'écriture étudiée par ailleurs (IV. 1) propre aux API. Twitter et Facebook mettent entre les mains de développeurs externes la circulation des données et des textes produits sur ces plateformes. Étant des industries médiatisantes qui tirent profit de cette circulation, elles externalisent ce qui fait leur valeur : la circulation et apparaître comme ce qui organise cette circulation. Elles doivent donc protéger, par une pratique lettrée, cette circulation et préserver ainsi leur image de marque.

4 B Maîtriser la polyphonie et s'exhiber maîtrisant

L'enjeu sémiotique du contrôle éditorial de Facebook et Twitter est désormais clair : garantir l'intégrité de l'image du texte des { petites formes } malgré leur circulation et prévenir ainsi les altérations inhérentes à toute circulation. Pour des industries médiatisantes, cet enjeu sémiotique est un enjeu économique et symbolique : c'est l'image de marque de ces industries qui circule et c'est leur savoir-faire (organiser la circulation) qui est en jeu. On peut remonter encore d'un cran et se demander ce que gagnent ces plateformes à montrer comment elles permettent à des textes de circuler. Que disent les { formes-textes } de réseau du rôle que se donnent Facebook et Twitter dans la circulation de ces formes ?

API, forme-texte, industrialisation, petites formes, standardisation

Voir glossaire tome II, p. 3-25

Nous commencerons par reprendre certains attendus de la théorie de l'énonciation éditoriale, ce qui nous permettra de reformuler nos questions initiales. Si tout texte est polyphonique, que cette polyphonie inclut toutes les entités qui participent à l'existence matérielle du texte, que toute circulation est transformation – notamment matérielle, alors on peut penser que l'{ industrialisation } de la circulation est aussi une { industrialisation } de la polyphonie (IV. 4. B. a). Dans ce contexte, les { API } de Facebook et Twitter ne font pas qu'organiser la polyphonie constitutive de tout texte : c'est la polyphonie même qui fait l'objet d'une { standardisation } (IV. 4. B. b). L'étude de cette { standardisation } nous amènera au constat de la production de formes extrêmement polyphoniques et donc à l'hypothèse d'une d'une valorisation symbolique en sus de la { standardisation } visuelle : Facebook et Twitter s'exhibent en tant que chefs d'orchestre des textes de réseau, occupant une place privilégiée dans cette polyphonie (IV. 4. B. c).

a Organiser et maîtriser les voix du texte : de l'énonciation éditoriale…

petites formes

Voir glossaire tome II, p. 3-25

Comment les { petites formes } organisent-elles la polyphonie des textes de réseau ? Pour comprendre d'où vient cette question, il faut rappeler ce qu'est la polyphonie d'un texte, comment elle est consubstantielle à tout texte et comment elle se donne à lire en une « énonciation éditoriale ». Ce n'est qu'une fois ces bases établies que nous pourrons répondre à notre question initiale.

La notion de polyphonie est particulièrement utilisée en linguistique, en théorie littéraire et en sémiotique. Elle désigne de manière générale le fait que dans un texte, compris à la fois comme discours et comme objet matériellement attesté, on trouve plusieurs « voix » (*poly-*, multiple ; *-phonos*, voix[762]). La polyphonie dépasse le seul cas de la citation ou de l'intervention signée dans un texte. Imaginons par exemple la préface d'un livre ou une mention « note du traducteur » : ici, la cohabitation de plusieurs voix est assez claire, car assumée et explicitée par des marqueurs discursifs : « NdT » ; signature… Mais la polyphonie comprend plus généralement tous les « […] horizons verbaux et idéologiques[763] », tous les autres textes qui constituent la culture dans laquelle un texte particulier s'inscrit, même si ces textes ne sont pas explicites. C'est ce que Kristeva a appelé « l'intertextualité », autre avatar de cette polyphonie[764]. Pour reprendre l'exemple de Don Quichotte, on pourrait dire que ce roman, parce qu'il se base sur le modèle du roman de formation, entretient une relation intertextuelle avec *Perceval ou le Roman du Graal* de Chrétien de Troyes. Il y a en cela « polyphonie » car ce n'est pas que Cervantes qui « parle » à travers son œuvre, c'est aussi toute une culture littéraire et historique. On peut plus largement considérer que tout acte d'énonciation est polyphonique[765], en ceci qu'il prend place dans une culture et qu'il articule plusieurs voix.

IV 3 B a
l'exemple de Don Quichotte
Voir p. 383

762 Todorov, Tzvetan. *Mikhaïl Bakhtine : le principe dialogique*. Paris : Éditions du Seuil, 1981.

763 Todorov, Tzvetan. *Idem*, p. 89.

764 De Biasi, Pierre-Marc. Théorie de l'intertextualité. Dans : *Encyclopædia Universalis* [en ligne]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.universalis.fr/encyclopedie/theorie-de-l-intertextualite/.

765 Ducrot, Oswald. Esquisse d'une théorie polyphonique de l'énonciation. Dans Ducrot, Oswald : *Le dire et le dit*. Paris : Les Éditions de Minuit, 1984, p. 171-233.

La polyphonie fut élargie à tous les acteurs qui permettent l'existence matérielle du texte[766]. Selon Emmanuël Souchier, l'énonciation peut passer par des circuits autres qu'un énonciateur articulant une suite de mots dans un langage donné. Il y a une « énonciation éditoriale » qui est « […] produite ou proférée par toute instance susceptible d'intervenir dans la conception, la réalisation ou la production du livre, et plus généralement de l'écrit[767] ». Selon Emmanuël Souchier, pour rester sur l'exemple du livre, il n'y a pas que l'auteur identifié qui énonce (celui dont le nom est habituellement en première de couverture). Il y a également le typographe, l'éditeur, la collection, le metteur en page…).

Cette théorie se base sur plusieurs attendus. Le premier est qu' « […] il n'y a pas de texte qui, pour advenir aux yeux du lecteur, puisse se départir de sa livrée graphique[768] ». Un texte, avant d'être lu, doit être vu. Il est donc avant tout une certaine organisation matérielle de signes. La prise en compte de la visibilité d'un texte – avant même la question de sa lisibilité et de son « sens » – est donc une prise en compte de son existence en tant qu'objet matériel. Second point : cette visibilité est construite, aménagée. Mise en page, typographie, chapitrage, choix du grammage du papier : autant de savoirs et de pratiques qui permettent l'existence matérielle du texte. C'est le travail proprement « éditorial » et ce travail est une énonciation, au sens où il y à dans ces opérations quelque chose qui est « dit » du texte donné à lire, quelque chose qui en construit les sens possibles. Troisième attendu : tout texte porte la marque de ces opérations éditoriales. On peut donc faire une analyse sémiotique – et non seulement linguistique – de l'énonciation : une analyse qui porte sur le texte non pas en tant que discours, mais avant tout en tant qu'image et objet.

Si la théorie de l'énonciation éditoriale naît dans un contexte livresque, elle peut s'étendre à tout écrit et notamment aux écrits de réseau. Dans notre cas, elle permet de formuler une hypothèse intéressante. Si en effet tout texte porte les marques de sa polyphonie, si on peut intégrer dans cette polyphonie toutes les entités qui interviennent sur ce texte pour le faire exister matériellement, alors on peut penser que plus un texte circule – plus il est altéré – plus il porte les marques de cette altération. La théorie de la trivialité suppose en effet que la circulation ne se limite pas à la reproduction, mais à la modification de ce qui circule. Chaque

766 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

767 Souchier, Emmanuël. *Idem*, p. 141.

768 Souchier, Emmanuël. *Idem*, p. 138.

API, industrialisation, *plugin*, standardisation, tweet

Voir glossaire tome II, p. 3-25

nouvelle circulation est une nouvelle éditorialisation et donc une nouvelle énonciation. Un { tweet } personnel qui, par exemple, serait repris dans un journal ou dans un livre édité n'est plus le même texte : il se déploie dans un autre contexte, ses potentialités de signification changent. Or, si nous envisageons cette trivialité dans sa part sémiotique, c'est-à-dire en ce qu'elle concerne des textes, alors cette circulation marque ces textes, elle les impacte visuellement.

À l'{ industrialisation } de la trivialité des textes répond donc une { industrialisation } de la polyphonie. Les plateformes web contemporaines outillent la polyphonie (instrumentation) et en élaborent les moyens de la reproduction matérielle ({ standardisation }).

b … aux petites formes comme standardisation de la polyphonie

Nous souhaitons désormais montrer que les { API } web participent à cette { standardisation } de la polyphonie énonciative. Pour cela, nous avons sélectionné quelques formes représentatives de cette tendance. La première, la plus évidente car c'est sa fonction sémiotique principale, est le { *plugin* } « commentaires » de Facebook. Ce { *plugin* } permet d'afficher sur une page web un espace où les internautes, à condition d'avoir un compte Facebook, peuvent publier un texte directement sur la page en question.

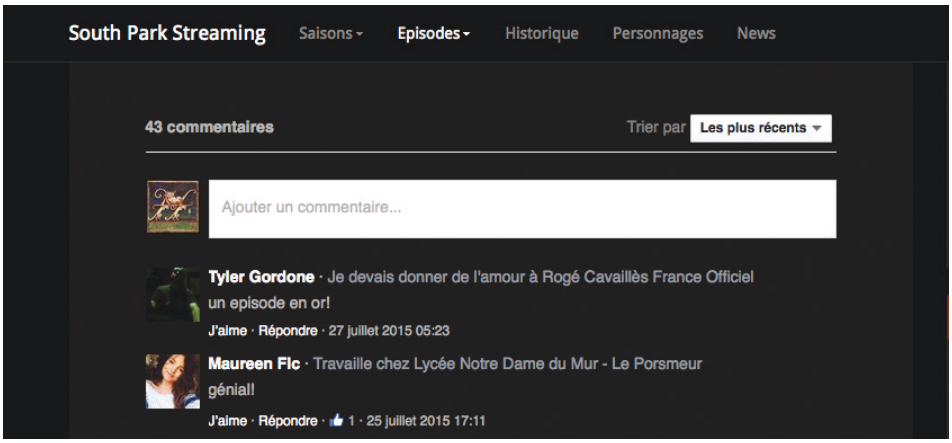


Fig. 48. Le *plugin* « Commentaires » comme gestion de la polyphonie (1). Capture d'écran du site south-park-streaming/saison-5/episode-4/scott-tenorman-doit-mourir.
Réalisée le 7 juillet 2015 (détail).

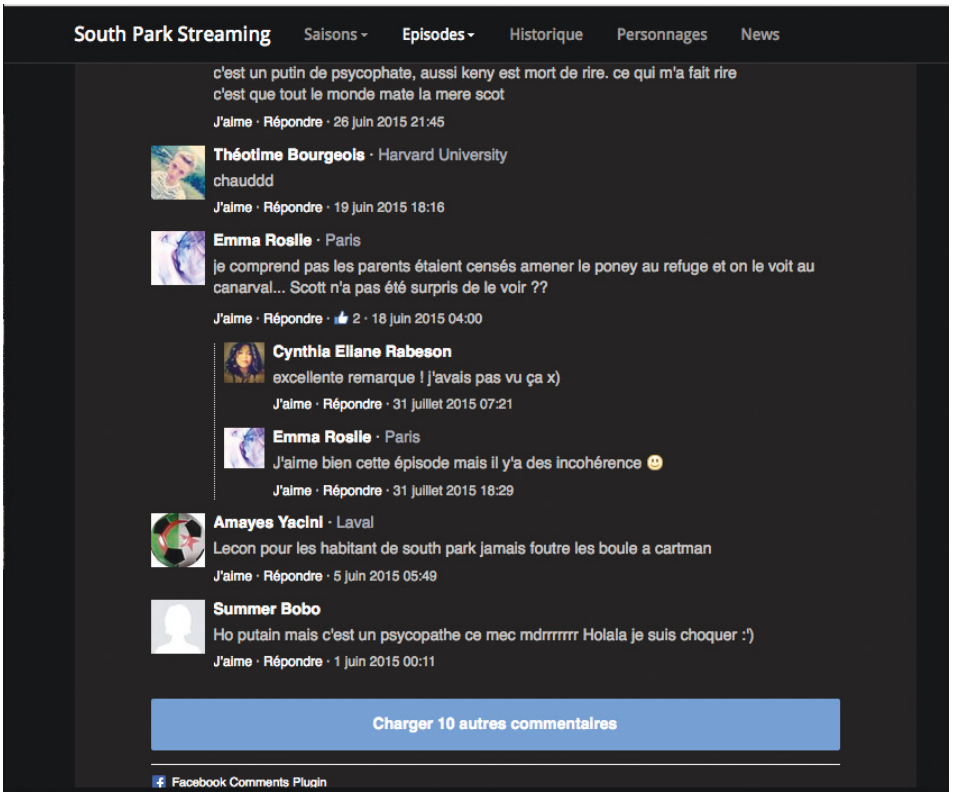


Fig. 49. Le *plugin* « Commentaires » comme gestion de la polyphonie (2). Capture d'écran du site south-park-streaming/saison-5/episode-4/scott-tenorman-doit-mourir.
Réalisée le 7 juillet 2015 (détail).

Comment s'organise la polyphonie sur cette section de la page du point de vue de l'énonciation éditoriale ? Au premier coup d'œil, l'empreinte graphique de Facebook est assez faible : il n'y a pas de cadre délimitant un espace spécifique d'énonciation, la couleur noire en fond de page est la même que sur tout le site… On trouve un seul indice linguistique (« *Facebook comments plugin* », sous l'indication « Charger 10 autres commentaires ») assorti de deux indices icôniques (le logo Facebook à côté de « *Facebook comments plugin* »).

Comment savoir alors que nous avons affaire à un espace d'énonciation qui est permis par Facebook ? Principalement par la reprise de certains codes graphiques typiques de la plateforme : la photo de profil, assortie du nom, qui compose la signature ; le texte qui se place légèrement en retrait de cette signature, une indentation stricte pour tous les niveaux d'énonciation, où une « réponse » à un « commentaire » se situe légèrement plus en avant que la publication principale, mais toujours en répliquant la signature composée de la photo et du nom de l'utilisateur[769]… Cette mise en page régulière, cette répétition des formes à différents niveaux d'inden-

769 Goyet, Samuel. *Op. cit.*, 2011, p. 57-65.

III 3 B b
un moule qui organise la polyphonie
Voir p. 319

tation est en quelque sorte un moule qui organise la polyphonie de l'énonciation dans cet espace. À chaque écrivant est attribuée une place sur la page, place qui est en fait une voix, en vertu de la théorie de l'énonciation éditoriale. Cette place est immuable, composée d'éléments { discrets } assemblés selon une mémoire des formes propre à la culture d'une société (le couple photo + nom rappelant le principe de la photo d'identité, entre autres) et une politique éditoriale propre à Facebook (indentation, taille de l'image, police de caractères). Par le { *plugin* } commentaires, Facebook ne fait donc pas qu'outiller la polyphonie énonciative. Il l'organise et la standardise visuellement : tout nouveau commentaire prendra la même forme que les précédents. Il y a là une forme de « canonicité textuelle », soit la « [...] récurrence de traits phénoménologiques[770] » qui tend à instituer la version standard d'un texte.

API, discret, *plugin*, *timeline*, tweet

Voir glossaire tome II, p. 3-25

Une autre forme qui montre l'organisation de la polyphonie est la { *timeline* } Twitter ancrée. Une { *timeline* } est l'ensemble des { tweets } publiés par un utilisateur. Par le biais de son { API }, Twitter donne la possibilité d'« ancrer » une { *timeline* }, c'est-à-dire de mettre dans une page web l'ensemble actualisé des { tweets } publiés par quelqu'un.

770 Jacob, Christian. Fonder. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 105.

Une { *timeline* } prend typiquement cette forme :

Fig. 50. La *timeline* Twitter et ses multiples énonciateurs. Capture d'écran du site www.timeshighereducation.co.uk/news/academics-meet--declare-support-universities-hit-conflict.
Capture réalisée le 17 juillet 2015 (détail).

III 1 D a
déjà analysé cette forme sous l'angle…
Voir p. 281

Nous avons par ailleurs déjà analysé cette forme sous l'angle de la modularité et du cadrage. Nous aimerions désormais en analyser la polyphonie. Pour ce faire, il faut commencer par faire le relevé de toutes les voix en présence, c'est-à-dire toutes les entités qui sont intervenues dans la production de cette forme, ainsi que des signes par lesquels ces voix sont rendues visibles.

La figure ci-contre présente toutes les instances énonciatives comprises dans la { petite forme }.

petites formes

Voir glossaire tome II, p. 3-25

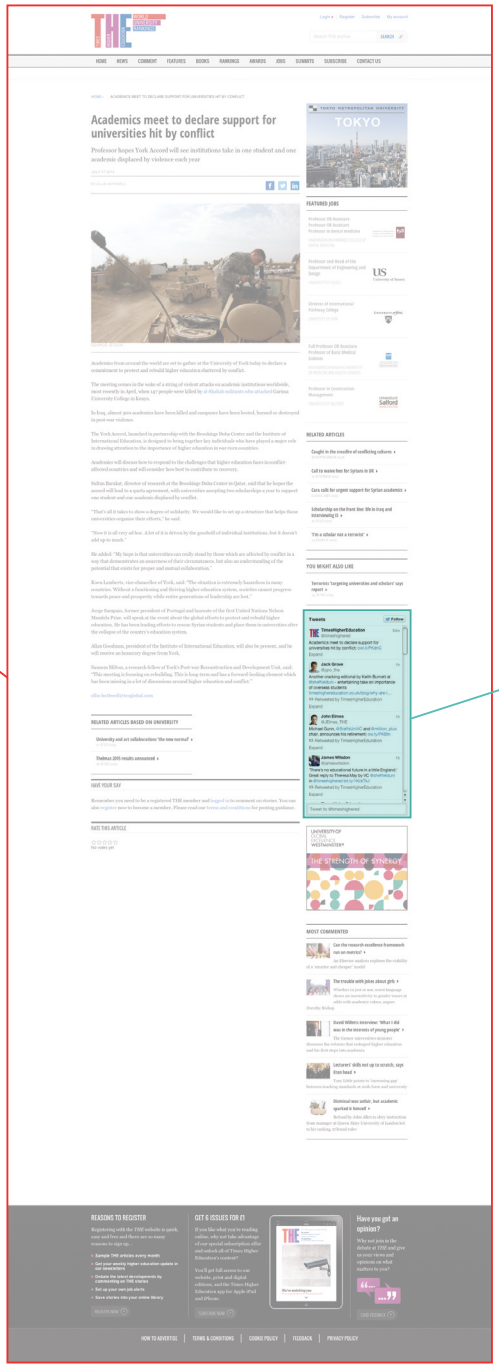
Légende :

Énonciateurs signataires

Énonciateurs potentiels

Énonciateurs machiniques

Énonciateurs cités



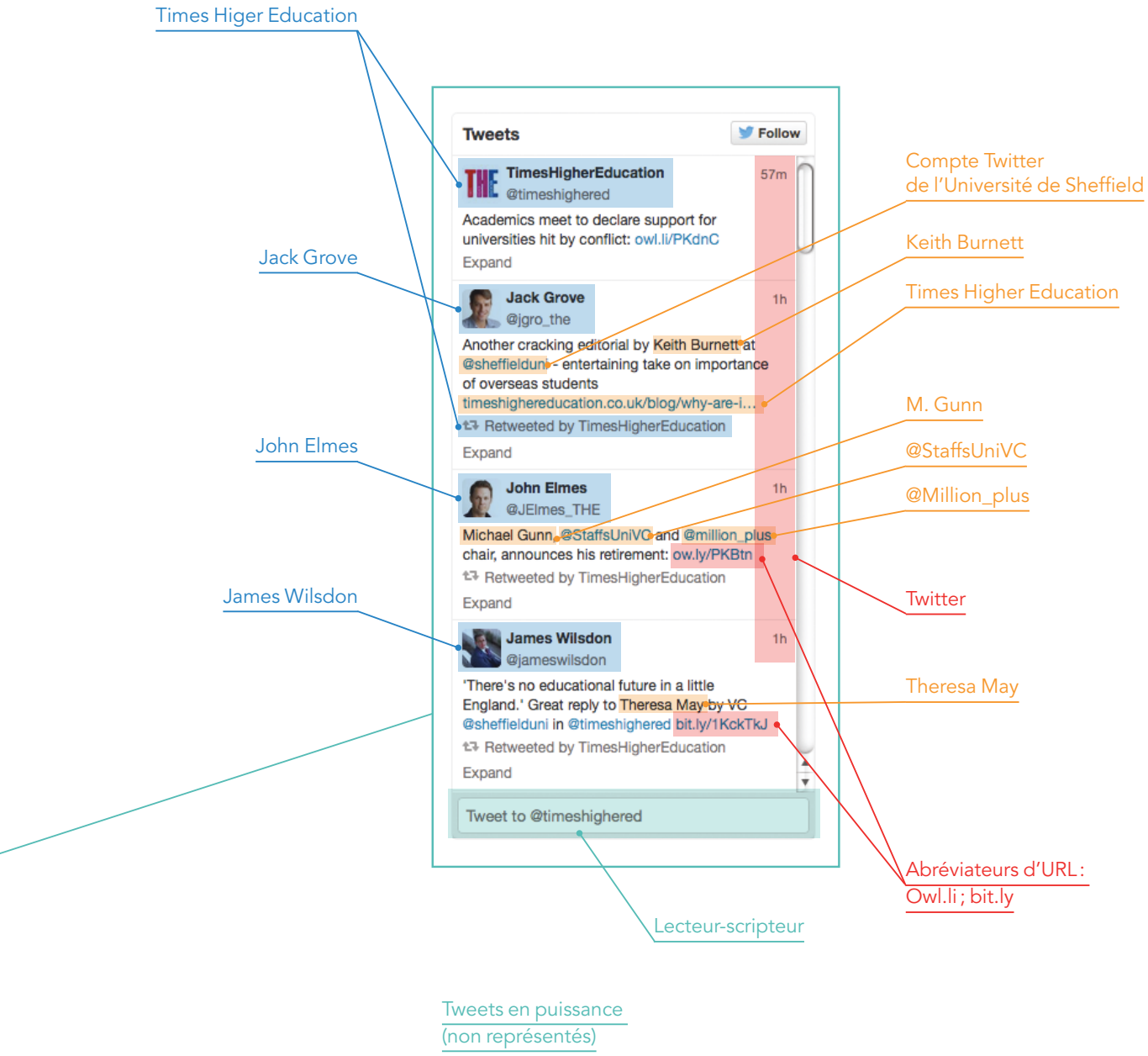


Fig. 51. Tableau récapitulatif des différentes énonciations dans une *timeline* Twitter ancrée. Réalisé à partir d'une capture d'écran du 17 juillet 2015.

Nous avons relevé au moins dix-sept voix différentes, qui se répartissent en quatre ensembles (cf. figure 51). Le premier ensemble est celui des signataires et comprend quatre entités. Le premier, le plus évident, c'est le *Times Higher Education*, qui non seulement accueille la { petite forme } sur la page mais figure dans la { *timeline* } sous trois formes : sa signature (nom de profil + photo d'utilisateur) ; le fait que le compte retweete d'autres { tweets } ; le fait qu'un lien vers son site soit inséré dans le { tweet } d'un autre utilisateur. On trouve ensuite trois autres entités, respectivement Jack Grove, John Elmes et James Wilsdon, qui ont tous publiés un { tweet } republié par *Times Higher Education*. Ces quatres entités énonciatives (THE, J. Grove, J. Helmes et J. Wilsdon) forment un premier ensemble, qu'on pourrait qualifier d'ensemble des « signataires », car leur énonciation est assumée et prise en charge par une signature clairement identifiable.

abréviateur d'URL, API, code, CSS, HTML, JavaScript, petites formes, signe passeur, *timeline*, tweet

Voir glossaire tome II, p. 3-25

Vient ensuite un groupe de sept autres entités énonciatives : Keith Burnett, « @Sheffielduni », timeshighereducation.co.uk, Michael Gunn, « @StaffsUniVC », « @million_plus » et Theresa May[771]. Ce groupe, nous le nommerons le groupe des « cités » : leurs paroles sont rapportées. Ils n'énoncent pas directement mais font partie de la situation d'énonciation et du discours en train de se faire. Ces « cités » sont soit des personnes physiques (M. Gunn, Theresa May), soit des institutions (Université de Sheffield, *Times Higher Education*), mais nous différencions bien les deux car on peut penser que les personnes physiques tirent de leur institution d'appartenance leur légitimité à être citées et qu'en conséquence l'institution joue aussi un rôle dans l'énonciation, rôle distinct des personnes physiques[772].

Un troisième groupe d'entités énonciatives, plus discret cette fois, est ce qu'on pourrait appeler le groupe des énonciateurs machiniques[773]. Ce groupe comprend quatre cas : Twitter, deux { abréviateurs d'URL } et notre navigateur. Twitter « énonce » en effet, au sens de l'énonciation éditoriale, car c'est lui qui organise les { tweets } dans un ordre antéchro-

771 Compte tenu de la proximité des deux énoncés et du contexte, on peut supposer que le « VC @sheffielduni » dans le tweet de James Wilsdon est Keith Burnett, cité par Jack Grove deux tweets plus haut.

772 On pourra rétorquer que J. Wilsdon, J. Grove et J. Elmes sont aussi en quelque sorte « cités » par le *Times Higher Education*. C'est vrai, à condition d'assimiler un retweet à une citation, ce qui est un saut techno-sémiotique périlleux. Deuxièment, la sémiotisation de cette « citation » est très différente pour ces trois derniers cas. Or, cette sémiotisation étant ce qui nous intéresse en premier lieu, nous nous permettons de les distinguer du groupe des « cités » pour les mettre dans celui des « signataires ». On pourra aussi noter que le *Times Higher Education* se retrouve dans deux groupes : il est à la fois « signataire » et « cité ». Rien ne l'empêche, à condition de bien différencier le journal en tant qu'entité de publication (côté « signataires ») et ses articles en tant qu'énoncés mobilisables dans les écrits de réseau sous formes de signes passeurs (côté « cités »).

773 Cette appellation sera largement discutée dans le chapitre suivant. Elle est utilisée ici comme catégorie de description : ce sont des énonciations qui reposent sur des actions réalisées par des machines.

nologique par rapport à leur temps de publication. Le plus récent est ce qui est en haut, organisation typique de la plateforme. En faisant particulièrement attention aux { signes passeurs } sur la page, on remarque aussi que certains liens ne sont pas complets mais prennent la forme d'URL raccourcie (ow.ly / bit.ly). Ces deux { abréviateurs d'URL }, par ailleurs recommandé par Twitter dans son { API[774] }, jouent eux aussi un rôle dans l'énonciation. D'abord, ils brisent l'un des aspects importants du { signe passeur }, qui est d'annoncer graphiquement ce vers quoi il renvoie : difficile ici de savoir d'emblée vers quelle page va nous mener le lien. Ensuite, ils rentrent dans une économie et une métrique du lien, puisqu'ils permettent de calculer le nombre de clics faits sur le lien et d'en proposer des outils d'analyse en recueillant des informations sur l'internaute ayant cliqué. Enfin, dernier acteur à la fois omniprésent et invisible : notre navigateur au moment de la capture, qui a interprété tout un ensemble de textes ({ code } { HTML }, { CSS }, { JavaScript }…) pour produire la forme que nous avons sous les yeux.

Dernier groupe d'énonciateurs prévu par la { petite forme }, c'est celui des énonciateurs potentiels, au nombre de deux. Le lecteur-scripteur tout d'abord, visible à travers le cadre blanc en bas portant la mention « *Tweet to* @timeshighered » et le bouton « *Follow* » en haut à droite. Un clic sur le bouton permet de suivre les publications de *Times Higher Education* sur Twitter ; un clic sur la zone basse de l'écran suffit à placer notre curseur dans un champ de saisie de texte et donc à nous placer en énonciateur potentiel. Nous sommes donc compris dans l'énonciation en cours. Certes comme potentialité, mais il n'en reste pas moins que la forme sémiotique permettant notre intervention est bien là, toujours présente dans une { *timeline* }. De plus, un second énonciateur potentiel est signifié par le fait que la { *timeline* } est une forme actualisée, où peut toujours surgir un nouveau texte. Ainsi, parce qu'elle donne à lire l'état présent de Twitter, la possibilité d'un nouvel énonciateur est maintenue en permanence. Il viendrait alors s'empiler sur le dernier { tweet } publié, en haut de la { petite forme }.

774 dev.twitter.com/overview/t.co. Capture réalisée le 14 juillet 2015.

API,
documentation,
forme-texte,
petites formes,
standardisation,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

Que conclure de cette analyse ? D'abord qu'il est clair qu'une { petite forme } comme une { *timeline* } ancrée organise une polyphonie énonciative à travers un ordre canonique, strictement réglé par la plateforme et son ordre antéchronologique. On retrouve une dynamique similaire à celle de Facebook : Twitter fournit un moule accueillant de multiples énonciateurs. Et ce moule est bien une { standardisation } en ceci que toute les { *timelines* } ancrées suivent ce modèle fourni par l'{ API } de Twitter. Il s'agit donc bien d'une { standardisation } de la polyphonie des textes de réseau qui s'opère à travers ces formes.

Ce qui frappe surtout est le degré de cette polyphonie : seize entités sont ramassées dans un espace finalement assez marginal quand on regarde la page dans son ensemble[775]. Elles sont signifiées par des signes ténus, mais bien présents, à condition d'y prêter une attention particulière. Pour l'instant, nous ne traitons pas du regard à adopter ni des enjeux politiques de cette organisation[776]. Nous soulignons en revanche la virtuosité de l'organisation et la { standardisation } de la polyphonie par les { *widgets* } de Facebook et Twitter. Ce qui nous amène à l'hypothèse suivante : les { API } web ne font pas qu'organiser la polyphonie énonciative propre à tout texte, elles cherchent aussi à la maximiser et à s'exhiber en virtuose de cette intense polyphonie.

775 Voir annexe n° 28.

776 Ces questions seront traitées dans les pages à venir.

c S'exhiber en chef d'orchestre : l'instrumentalisation de la polyphonie

À quoi sert cette virtuosité dans l'organisation de la polyphonie des textes de réseau ? Et pourquoi suggérer à l'internaute d'écrire à son tour dans les { petites formes } qui accueillent cette polyphonie ? Parce que dans le contexte d'une industrie des passages qui est aussi une industrie de la polyphonie – puisque les passages renforcent la polyphonie des textes et les « marquent » – la polyphonie est valorisée, instrumentalisée. Elle devient une richesse à entretenir et à valoriser, comme marque de la circulation des textes. Il ne s'agit pas seulement d'orchestrer les différentes voix constitutives d'un texte, il s'agit de montrer ces voix et d'en inviter de nouvelles à prendre part au concert. Nous préciserons le sens de cette instrumentalisation, en montrant qu'elle se fait, dans la { documentation } des { API }, par deux opérations : un discours sur la « conversation » permise par les { *widgets* } ; le maintien de l'opérativité technique des marqueurs de polyphonie.

III **2** **B** **b**
d'en inviter de nouvelles…
ce qui confirme notre thèse de l'appel à l'écriture,
voir p. 300

La { documentation } de l'{ API } de Twitter constitue le cas le plus évident de la rhétorique de la conversation. Il est ainsi dit à propos des { *timelines* } ancrées qu'elle permettent de « […] rejoindre une conversation mondiale sur un sujet particulier[777] […] ». Twitter est le lieu d'une « conversation ». En écrivant dans cette { *timeline* }, on rejoint la conversation en cours. Il est notable que Twitter réactive ici un imaginaire de l'oralité, alors que paradoxalement ce sont des industries du texte et donc de l'écriture. La médiation sémiotique (la { forme-texte }, le { *widget* }), est niée ou plutôt fait l'objet d'une transformation : elle est le lieu d'une conversation. On retrouve cette idée que le { *widget* } organise la polyphonie, polyphonie cette fois clairement qualifiée par Twitter. C'est une conversation, comme celle que l'on aurait avec des amis, mais à l'échelle mondiale. Même chose pour les { tweets } ancrés : « La signature, les *hashtags*, mentions, et autres composantes clés de l'expérience Twitter aident le public de votre site à se connecter avec la conversation globale qui se passe sur Twitter[778]. » Ce qui se déroule sur Twitter est de l'ordre de la conversation et c'est cette conversation à laquelle les { *widgets* } invitent à participer. Voici donc la polyphonie énonciative des textes de réseau non seulement outillée, mais qualifiée.

777 « […] *tap into the worldwide conversation around a topic* […] ». dev.twitter.com/web/overview. Capture réalisée le 14 juillet 2015.

778 « *Author attribution, hashtags, mentions, and other key components of the Twitter experience helps your site's audience connect with the global conversation happening on Twitter.* » dev.twitter.com/web/embedded-tweets. Capture réalisée le 14 juillet 2015.

API, forme-texte, signe passeur, standardisation, *widget*

Voir glossaire tome II, p. 3-25

Cette dernière citation nous amène à notre second procédé de valorisation de la polyphonie. Une des particularités des { *widgets* }, que ce soit ceux de Twitter ou de Facebook, c'est en effet de préserver l'opérativité technique des { signes passeurs } : cliquer sur un cœur ou un pouce vous fait « aimer » une publication, on peut y « répondre », on peut « citer », « mentionner » un autre utilisateur en étant sur n'importe quelle page web. En d'autres termes, les marques sémiotiques de la polyphonie (mentions, auteur, personnes citées, etc.) sont mises en valeur comme { signes passeurs }, c'est-à-dire comme signes agissants. Agissants au sens où ils permettent d'intervenir dans la « conversation », d'avoir à son tour voix au chapitre. Ce sont bien des « appels à l'écriture », par cette effectuation technique : ils sont marqués comme particulièrement importants, comme des « [...] composants clés de l'expérience[779] [...] » de la plateforme. La transformation des marques de polyphonie en { signes passeurs } participe donc à la valorisation de cette polyphonie : elle est par là incitée à être enrichie, complétée, maximisée par d'autres écritures. Elle doit être entretenue, prolongée et pérennisée.

III 2 B b
appels à l'écriture
Voir p. 300

En somme, instrumentation et instrumentalisation de la polyphonie vont de pair. Les { formes-textes } que permettent de générer les { API } sont, certes, un outillage et une certaine { standardisation } formelle de la polyphonie des textes de réseau. Mais par le discours tenu sur ces formes, par le fait qu'elles soient produites dans le cadre d'industries médiatisantes, par les { signes passeurs } qu'elles mettent en avant, ces formes instrumentalisent la polyphonie. L'enjeu est de produire les textes les plus polyphoniques possibles, pour en nourrir la circulation et son exploitation.

779 « [...] *other key components of the Twitter experience* [...] » dev.twitter.com/web/embedded-tweets. Capture réalisée le 14 juillet 2015.

Par ces formes, Facebook et Twitter organisent et structurent fortement la polyphonie des textes. En même temps, ils s'exhibent organisant cette polyphonie : c'est leur logo qui apparaît, c'est vers leur site que renvoient les { signes passeurs }, les { *widgets* } disséminent un ordre du texte aisément reconnaissable… C'est peut-être là un caractère plus global des industries médiatisantes : maîtriser les cadres de l'écriture, en exploiter un des aspects (la polyphonie énonciative), valoriser cet aspect, tout en laissant libres ces pratiques dans leur strict « contenu ». Voilà la pratique lettrée réalisée par les { API } de Facebook et de Twitter, ce qui confirme et nuance notre hypothèse de départ. Si effectivement les { API } sont bien le lieu d'une pratique lettrée, cette pratique lettrée s'exerce en priorité sur l'image du texte et a comme objet spécifique la polyphonie énonciative. Mais comme nous l'avons montré, ce n'est pas un simple contrôle de la bonne lisibilité de formes polyphoniques. C'est aussi une valorisation de cette polyphonie et la monstration d'une virtuosité dans son organisation visuelle.

4 C Les machines oubliées ? De l'interopérabilité comme idéologie du texte

Il est temps désormais de s'intéresser à la dimension politique de l'énonciation éditoriale et de la gestion de la polyphonie. Tout texte est polyphonique et cette polyphonie s'organise visuellement en une « énonciation éditoriale ». Les { *widgets* } de notre corpus sont des modes d'organisation de la polyphonie des textes de réseau, comme nous venons de le montrer. Mais quels voix sont privilégiées ? Par quels moyens sémiotiques ? Et surtout au profit de qui, de quoi ? Quels sont les jeux de pouvoir qui s'exercent dans l'énonciation éditoriale des { *widgets* } de notre corpus ?

algorithme, API, énonciation computationnelle, forme-texte, HTML, interopérabilité, langage de programmation, petites formes, programme, *timeline*, tweet, *widget*

Voir glossaire tome II, p. 3-25

IV 3 A
une possibilité technique d'échange…
Voir p. 379

IV 3 C
instrumentalisée comme condition du partage…
Voir p. 391

Il s'agit ici de passer de la notion de polyphonie à celle d'« hétérophonie énonciative[780] ». Un texte n'est pas le point de rencontre de plusieurs « voix », mais bien de plusieurs types d'entités, hétérogènes. Il n'y a pas que de l'énonciation au sens humain du terme engagée dans la production d'un texte. Partant de cette idée, nous faisons l'hypothèse que c'est l'action des machines qui est oubliée dans ces formes et ce par le biais d'une conception idéologique de l'{ interopérabilité } qui a de fortes incidences sémiotiques. Jusqu'ici nous avons analysé l'{ interopérabilité } en tant qu'elle est une possibilité technique d'échange entre composants informatiques, progressivement instrumentalisée comme condition du « partage » entre êtres humains. Mais dès lors que ce « partage » passe par des { formes-textes } réplicables – les { *widgets* } –, l'{ interopérabilité } sert à maximiser cette réplication des formes : un { tweet } écrit en { HTML } peut s'ancrer dans n'importe quelle page, être affiché par n'importe quel navigateur, puisque l'{ HTML } est un { langage } standard. Le but est de limiter au maximum les altérations possibles, en tout cas certaines altérations : celles faites par les machines permettant l'affichage du texte. Ce dernier doit pouvoir être reproduit, en dépit des influences éventuelles des conditions matérielles de sa reproduction. En revanche, les altérations humaines, « volontaires », qui résultent de la circulation du texte dans un espace social, sont autorisées. Le « partage » que permet les { petites formes } serait alors aussi un partage entre altérations autorisées et interdites, entre énonciateurs humains et machiniques.

780 Cormerais, Franck et Gilbert, Jacques Athanase. Entretien avec Emmanuël Souchier. *Études digitales : le texte à venir*. 2016, nº 1, p. 189-214.

Notre démonstration se fera en deux temps. Premièrement, nous montrons, exemples de { *widgets* } à l'appui, que dans les formes lisibles à l'écran les actions des { programmes } ou { algorithmes } qui permettent l'existence de l'écrit à l'écran sont soit ramenées à des entités humaines, soit invisibilisées (IV. 4. C. a). Nous soutenons ensuite que cette invisibilisation est en partie le fruit d'une conception idéologique de l'{ interopérabilité }. Alors que l'intense circulation des textes permise par les { API } pourrait donner l'occasion d'une mise en visibilité des incompatibilités éventuelles entre couches techniques, l'idéal d'{ interopérabilité } pousse à nier la médiation technique, à l'invisibiliser. La médiation technique (ici, des entités machiniques et calculatoires) doit se faire oublier et se contenter de répliquer le texte à l'identique (IV. 4. C. b). Ainsi, l'analyse de l'{ interopérabilité } dans ses effets symboliques nous amènera à poser l'hypothèse d'une invisibilisation de l'« { énonciation computationnelle } », hypothèse que nous examinons dans le chapitre suivant.

a Une invisibilisation des machines ?

Quelles voix sont valorisées dans les { *widgets* } et autres { petites formes } ? Les exemples montrés jusqu'ici nous mettent sur la voie d'une première réponse : ce sont les énonciateurs humains qui sont mis en avant. Ce sont eux qui apparaissent « naturellement » quand on regarde une { *timeline* } ou un statut Facebook. Si l'on veut voir autre chose, d'autres entités agissant dans le texte lu, notamment les machines, il faut rompre ce regard « infra-ordinaire » et faire l'hypothèse que le texte, dans son organisation visuelle, propose une vision idéologique des entités à l'œuvre dans la production de ces formes[781].

781 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 144-145.

Fig. 52. La mise en visibilité du calcul (1) : des machines ne faisant que cadrer l'écriture. Capture d'écran du site timeshighereducation.co.uk (détail).
Capture réalisée le 17 juillet 2015.

Fig. 53. La mise en visibilité du calcul (2) : des machines ne faisant que compter les clics. Capture d'écran du site twitter.com (profil personnel, détail). Capture réalisée le 18 juillet 2015.

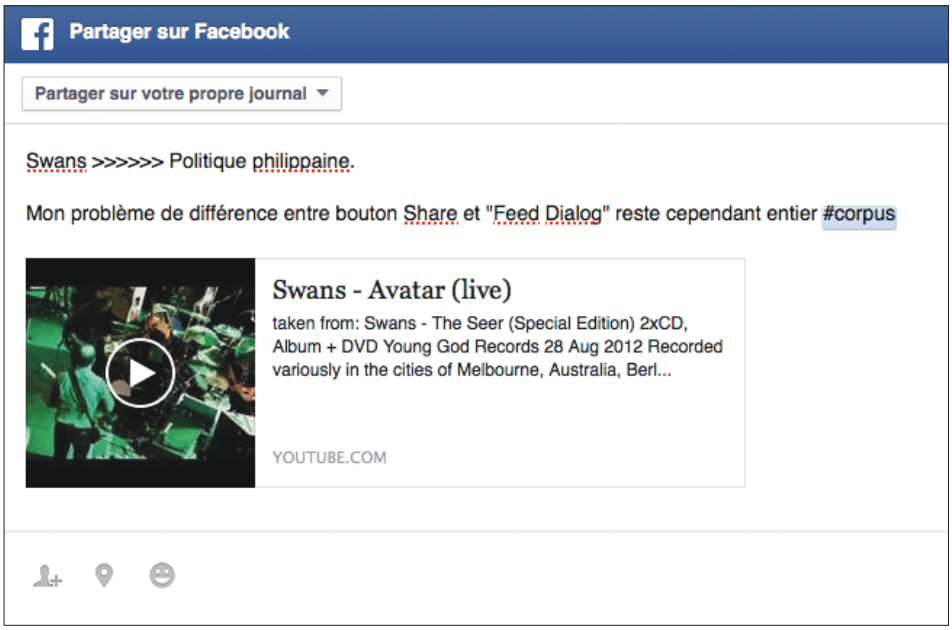


Fig. 54. La mise en visibilité du calcul (3) : des machines dont la spécificité des procédures est niée à l'écran. Capture d'écran du site facebook.com (profil personnel, détail).
Capture réalisée le 12 août 2015.

algorithme, calcul, signe passeur, *widget*

Voir glossaire tome II, p. 3-25

Dans le premier exemple, la figure 52, les mentions et autres { signes passeurs } renvoient exclusivement à des profils personnels, tenus par des utilisateurs de Twitter. Le cadre du bas nous donne la possibilité de s'adresser au *Times Higher Education*, sans que Twitter ou le mode d'action de son { algorithme } soit inclus dans la « conversation » que prétend organiser le { *widget* }. La seule action propre de Twitter dans cet exemple, c'est de permettre l'écriture, de dater les différentes publications et de les organiser en conséquence. C'est une tâche considérable, mais aucun des signes traduisant ce travail éditorial n'est activable. Les { signes passeurs } en présence nous amènent plutôt à communiquer avec des profils d'utilisateurs.

De la même façon dans le second exemple – la figure 53 –, l'action de la plateforme est rendue visible par le nombre de « retweets » et de « favoris ». C'est bien un { algorithme } qui a compté tout cela et le résultat de ce { calcul } est mis en valeur par la plateforme. Mais un clic sur ces chiffres nous renvoie à une galerie de visages (visible à droite des chiffres), soit la galerie des utilisateurs qui ont republié ou ont déclaré « l'aimer » *via* un clic. Dans ces deux cas, il s'agit d'une vision anthropocentrée au sens où la « conversation » que veut instaurer Twitter a lieu entre humains. Les machines sont réduites à des domestiques au service de cette conversation, en témoigne ce qui est donné à voir de leurs actions : elles comptent, cadrent, organisent les écrits.

Le troisième exemple – la figure 54 – propose non pas une relation anthropocentrée aux machines, mais plutôt acte leur invisibilisation. Cette fenêtre apparaît quand on clique sur le bouton« Partager sur Facebook » en dessous d'une vidéo hébergée sur le site YouTube. La forme qui en résulte est celle d'un lecteur vidéo sur Facebook, avec des informations sur la vidéo. C'est exactement la même forme que celle d'une « carte » Twitter ou que celle qui serait produite par un clic sur un bouton « Partager » sur n'importe quel autre site. Le même titre apparaîtrait, la même image de prévisualisation, le même chapeau, etc. Or, YouTube utilise ici le « *Feed Dialog* » de l'{ API } et non pas le bouton « Partager ». Ce sont deux fonctionnalités différentes[782], mais qui donnent le même résultat éditorial. C'est la même forme qui est produite, malgré une différence dans la manière dont l'{ API } traite informatiquement cette forme. En d'autres termes, il n'y a aucune différence de traitement sémiotique, alors que la médiation technique n'est pas la même. Ce qui fait le propre de cette médiation (la différence entre le *Feed Dialog* et le bouton Partager) est niée, rendue invisible dans le texte[783].

API,
***bug*,**
code,
format,
industrialisation,
interopérabilité,
petites formes,
programme

Voir glossaire tome II, p. 3-25

b L'interopérabilité comme horizon idéologique

Il y a donc dans les formes produites par les { API } web une mise sous silence de la médiation technique qui permet l'existence de ces formes. Cette médiation a pourtant un statut paradoxal : comment penser que les machines sont « invisibles », alors qu'elles sont omniprésentes dans le bon ancrage du texte, dans son affichage ? Ce paradoxe n'en est pas vraiment un, si l'on remet ces écrits dans la dynamique vue par Emmanuël Souchier autour de la typographie et de son invisibilisation : là aussi, le travail du typographe est mis au service du texte de « l'auteur ». La police de caractères est à la fois visible sémiotiquement – et c'est d'ailleurs là toute sa qualité, elle rend visible et donc lisible le texte – mais négligée symboliquement. Il semble donc que les écrits d'écran reconduisent, avec de nouveaux attendus, une tendance entamée un peu plus tôt dans l'histoire de l'{ industrialisation } du texte. Nous soutenons que l'{ interopérabilité }, envisagée cette fois comme idéologie du texte poussant à une dévalorisation symbolique des machines, est l'un de ces attendus propre aux écrits d'écran.

782 Pour être tout à fait honnête, nous n'avons pas réussi à comprendre la nature de cette différence, mais elle est malgré tout attestée dans l'API.

783 Pour voir qu'il s'agit de deux procédures différentes, il ne faut plus regarder la forme produite, ni même la page, mais l'adresse du navigateur, où la requête est formulée différemment que celle pour un bouton « Partager ».

Sur quelle conception des acteurs techniques du texte repose l'{ interopérabilité } telle qu'elle est donnée à voir dans les { petites formes } générées par une { API } ? L'{ interopérabilité } est ce qui permet à des éléments informatiques d'échanger des données et de fonctionner entre eux. Cette { interopérabilité } fut instrumentalisée au profit de la circulation des textes, posant un problème de gestion de la polyphonie, gestion qui s'accompagne d'une instrumentalisation. Dans le contexte d'industries médiatisantes comme Facebook ou Twitter, l'{ interopérabilité } joue de plus un rôle idéologique vis-à-vis des acteurs du texte. À quoi sert en effet l'{ interopérabilité } ? Elle permet à des { petites formes } de se répliquer, de circuler. Ces { petites formes } ancrant du texte dans la page, il s'agit également de conserver le contenu du texte, en dépit des possibles altérations.

IV 3 A
ce qui permet à des éléments informatiques…
Voir p. 379

IV 3 D
instrumentalisée au profit de la circulation des textes
Voir p. 394

IV 4 B
un problème de gestion de la polyphonie
Voir p. 413

De quel type peuvent être ces altérations dans le cas des écrits d'écran ? « é » qui se transforment en « &eacute », vidéo qui ne se joue pas ou très pixellisée, police de caractères inhabituelle, liens qui ne s'affichent pas dans la couleur prévue… Autant de { *bugs* } d'affichage qui laissent apparaître les multiples actions prévues dans le { code informatique } pour réaliser la version « canonique » du texte, celle documentée dans l'{ API } des plateformes web. Bien souvent, ces { *bugs* } sont dus à des problèmes de conversion, de compatibilité entre les différentes machines et { programmes } engagés dans la production du texte. C'est ce qu'on pourrait appeler « l'aléa machinique » : des facteurs non totalement maîtrisables et qui tiennent à la complexité des processus informatiques engagés dans la production du texte.

À quoi sert l'{ interopérabilité } dans ce contexte ? À limiter cet aléa machinique. Pour afficher une { petite forme }, il faut en passer par de nombreux acteurs non-humains[784], sans compter la circulation des textes qui rend l'aléa machinique encore plus important. En choisissant des { formats } interopérables qui permettent de produire le texte malgré la diversité des { programmes } et des machines impliqués dans cette production, on promeut donc l'idée que le texte doit pouvoir être reproduit sans que les machines comptent. Il s'agit d'utiliser la puissance industrielle de reproduction de l'informatique, sans en accepter les altérations qu'elle peut générer au passage.

784 Nous préciserons dans le chapitre suivant en quoi une machine ou un programme sont « non-humains », en dépit du fait que tous deux sont programmés par un humain.

API, algorithme, énonciation computationnelle, forme-texte, industrialisation, interopérabilité, logiciel, matériel, script, standardisation, système d'exploitation

Voir glossaire tome II, p. 3-25

Cette conception du texte comme « devant se répliquer à l'identique » indépendamment du complexe { matériel } et { logiciel } qui permet de le lire se retrouve dans les { API } web contemporaines. Elle repose sur une abstraction (l'oubli de la matérialité et des procédures techniques) qui exclut de la scène des acteurs du texte tout ce qui n'est pas humain (machines, { algorithmes }, { scripts }, navigateurs, { systèmes d'exploitation }, serveurs…) et qui pourtant structure de bout en bout la production du texte. Cette abstraction entraîne une dévaluation symbolique de l'énonciation propre à ces machines, ce qu'on pourrait appeler « l'{ énonciation computationnelle[785] } ». Il s'agit désormais de comprendre ce qu'est cette « { énonciation computationnelle } » et plus largement comprendre la manière dont l'écriture informatique construit un certain rapport idéologique aux machines.

785 Collomb, Cléo et Goyet, Samuel. *Op. cit.*, 2015.

CONCLUSION

Ce chapitre achevé, qu'en est-il de notre hypothèse de départ ? En partant de l'idée que les { API } participent à une { industrialisation } de la trivialité, nous avons commencé par analyser l'outillage de la circulation, en montrant que les { API } web sont structurées autour d'une logique de délégation de l'écriture. À ce titre, elles équipent la circulation des { formes-textes } des écrits de réseau. Cet équipement ne peut toutefois être efficace que si les { API } sont construites comme des lieux de désir, motivant ainsi l'écriture. Dans un troisième temps, nous avons analysé le statut de l'{ interopérabilité } permise par les { API }, en démontrant que se joue autour de cette notion une instrumentalisation des { API }. Ces dernières deviennent au fil de leur circulation des êtres culturels vecteurs d'innovation, d'ouverture et de partage. Mais l'{ interopérabilité } pose également le problème de l'altération des formes circulantes, ce à quoi répond une { API } en tant que lieu d'une « pratique lettrée ». Il s'y joue une double { standardisation } : { standardisation } de l'image du texte ainsi que de la polyphonie énonciative.

Si notre hypothèse de départ est confirmée, la question de la polyphonie nous pousse à un déplacement. Nous avons jusqu'ici entendu l'économie comme création de valeur dans le cadre d'une industrie des passages. Mais pour comprendre la manière dont les procédures computationnelles interviennent dans les textes de réseau et éventuellement montrer la manière dont elles sont invisibilisées, il faut passer à une conception politique de l'économie, autour de ce que Michel De Certeau nomme « l'économie scripturaire[786] ». L'écriture et le texte ne sont pas seulement la cheville ouvrière d'une industrie contemporaine. Ce qui se joue dans l'écriture, ce sont aussi des rapports de domination et de valorisation symboliques où le texte est un lieu qui organise sémiotiquement ces rapports. En passant ainsi d'une lecture économique à une lecture politique, on peut alors qualifier précisément ce que nous avons appelé « { énonciation computationnelle } », ce qui nous permet par la suite de comprendre le rôle que joue cette énonciation dans les textes de réseau et la manière dont cette énonciation spécifique est donnée à lire ou ignorée. On pourra alors élargir la question de la polyphonie et du rôle des { API } aux relations que l'on peut tisser, à travers le texte, avec des entités non-humaines comme les procédures computationnelles.

786 DE CERTEAU, Michel. *Op. cit.*, p. 195-225.

VERS UNE SÉMIOTIQUE NON ANTHROPOCENTRÉE DES MÉDIAS INFORMATISÉS

API, industrialisation

Voir glossaire tome II, p. 3-25

Notre analyse du rôle des { API } dans l'{ industrialisation } contemporaine de l'écriture nous a amené devant le constat d'une forte polyphonie des écrits d'écran, et notamment une forte présence des acteurs techniques qui permettent l'existence de l'écrit à l'écran. Les { API } font donc partie d'un mouvement plus global. Ce sont des exemples particulièrement représentatifs de ce qu'on pourrait appeler la part machinique des écrits d'écran. L'analyse a montré comment cette part machinique est minorée, au nom d'une conception abstraite du texte qui doit se répliquer à l'identique malgré ses nombreuses rééditorialisations.

Le présent chapitre a pour vocation d'interroger cette part machinique. Comment la prendre en compte ? Quelle relation les outils théoriques mobilisés jusqu'ici construisent-ils avec les médias informatisés ? Ce chapitre constitue donc une rupture et une montée en généralité par rapport à notre objet. Après un chapitre à dominante historique (I), anthropologique (II), sémiotique (III) puis économique (IV), voici un chapitre épistémologique et politique : il s'agit de faire l'examen de la façon dont les notions d'écriture et de texte configurent nos relations aux machines d'écriture que sont les médias informatisés. Notre hypothèse générale est qu'il existe une certaine conception du texte et de l'écriture informa-

tique qui invisibilise la part machinique des écrits d'écran, mais qu'un déplacement vers la question de l'énonciation permet de redonner une place aux machines dans les textes numériques.

Pour pouvoir mettre en lumière cette part machinique, et donc incidemment la définir, il faut commencer par montrer en quoi humains et machines sont différents tant en degrés qu'en nature (V. I). Ce sans quoi on risquerait de confondre, dans un écrit d'écran, ce qui est du ressort de l'humain et ce qui est du ressort de la machine. Loin de penser une supériorité ou une infériorité de l'un sur l'autre, nous montrons une altérité réciproque, dont le point de rencontre privilégié est le { code informatique } comme texte et la { programmation } comme pratique d'écriture (V. 2). Mais là encore il faut examiner ces concepts. Comment un texte organise-t-il cette rencontre ? Et comment plus spécifiquement l'idée de { code source } tend-t-elle à instituer un certain type de relation avec les ordinateurs ? Pour répondre à ces questions, nous définissons le texte comme un lieu de pouvoir, et l'écriture comme une « pratique mythique[787] » qui organise ces jeux de pouvoir. En d'autres termes, quel est « l'économie scripturaire[788] » de la { programmation } ? En repartant de deux caractéristiques de cette écriture – le fait que l'on parle de { code « source » } et que ce soit une écriture d'instructions à une machine – nous montrons que la { programmation } s'organise autour de deux imaginaires : la source (V. 3) et la commande (V. 4). Ces deux imaginaires ont notamment pour effet de produire une invisibilisation des { entités computationnelles } impliquées dans la production des écrits d'écran. Ceci démontré, nous pourrons proposer des pistes pour reconstruire une relation aux { machines computationnelles }, en leur reconnaissant une modalité énonciative par ce qu'elles ont de propre : le { calcul } (V. 5).

calcul,
code,
code source,
entité computationnelle,
machine computationnelle,
programmation

Voir glossaire tome II, p. 3-25

787 De Certeau, Michel. *Op. cit*, p. 199-200.

788 De Certeau, Michel. *Idem*. Chapitre X : « L'économie scripturaire », p. 195-224.

1 DES HUMAINS ET DES MACHINES… RECONSTRUIRE UNE ALTÉRITÉ

Dans quelle mesure, lorsque nous tapons sur le clavier d'un ordinateur ou lorsque nous manipulons une tablette, nous avons affaire à quelque chose d'« autre » que nous-même ? Quelque chose dont le mode de fonctionnement est différent du nôtre ?

Il est indispensable de répondre à ces questions afin de poser les conditions d'une intelligence de ce qui est spécifique aux médias informatisés et de la façon dont ils participent à la polyphonie des écrits d'écran. Le but de ce chapitre est en effet de proposer une approche de la sémiotique des écrits d'écran qui permette de prendre en compte l'action propre des médias informatisés, sans réduire cette dernière à des enjeux humains. L'enjeu est de faire surgir toutes les « voix » des textes numériques. Mais si nous posons d'emblée que ce à quoi nous avons affaire à travers ces écrits ce ne sont qu'à d'autres humains, nous risquons alors de passer à côté de ce qui fait le propre de la médiation technique et de la façon dont elle participe à ce que nous lisons. C'est pourquoi il est nécessaire dans un premier temps de comprendre en quoi les médias informatisés sont des entités non-humaines, des altérités avec lesquelles nous composons.

V 4 D

passer à côté de ce qui fait le propre de la médiation technique

Voir p. 511 pour l'exemple des algorithmes.

Pour souligner cette altérité, nous avions proposé avec Cléo Collomb le terme de « { machines computationnelles } », entendues comme « […] l'ensemble des objets techniques dont le fonctionnement repose sur du calcul binaire intégré dans une machinerie[789] ». Cela nous permet de grouper tout un ensemble de dispositifs (tablettes, *smartphones*, ordinateurs) en mettant l'accent sur leur mode de fonctionnement calculatoire, tout en soulignant que ce sont des objets techniques automatisés (des « machines »). Le terme de « { machines computationnelles } » est une façon de reformuler celui de « médias informatisés[790] ». « Médias informatisés » met l'accent à la fois sur la fonction sociale du dispositif technique – c'est un média soit ce qui « […] permet les échanges signifiant entre les

789 Collomb, Cléo et Goyet, Samuel. *Op. cit.* 2015, p. 1.

790 Davallon, Jean, Després-Lonnet, Marie, Jeanneret, Yves, *et al. Op. cit.* ; Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Op. cit.*, p. 100-105.

calcul, machine computationnelle, programmation

Voir glossaire tome II, p. 3-25

hommes[791] » – et sur un certain aspect de son fonctionnement technique – il traite de l'information au sens logistique du terme[792]. { Machines computationnelles } n'a pas vocation à remplacer le terme de médias informatisés, mais à apporter un éclairage différent sur une même catégorie d'objets concrets[793]. Il s'agit d'éclairer sous un autre angle une réalité composite. Par leurs aspects communicationnels, les ordinateurs sont des machines permettant l'échange signifiant entre les humains. Par leur aspect historique et technique, ce sont des machines à calculer automatiques. Après avoir, dans le premier chapitre, mis l'accent sur l'aspect technique, puis dans les chapitre III et IV sur l'aspect communicationnel, nous revenons maintenant à l'aspect technique. Plutôt que de poser en premier lieu « l'échange entre les hommes », nous considérons que nous avons d'abord affaire à une machine, soit un objet technique au fonctionnement particulier. Ce fonctionnement, nous l'ancrons dans le { calcul } plutôt que dans la théorie de l'information, ce qui permet de faire ressortir une autre histoire, une autre culture scientifique, celle-là même dont nous avons relevé les principaux imaginaires dans le chapitre II. Dans ce chapitre, nous privilégierons donc le syntagme de « { machines computationnelles } », même si celui de « médias informatisés » ou d'« ordinateur » peut ressurgir de temps à autre, pour des raisons stylistiques.

791 Jeanneret, Yves. *Op. cit.*, 2007, p. 91.

792 Jeanneret, Yves. *Idem*, p. 60-63.

793 Davallon, Jean. Objet concret, objet scientifique, objet de recherche. *Op. cit.*, p. 30-37.

Il nous faut donc dans un premier temps reconstruire une altérité, montrer en quoi ces { machines computationnelles } nous sont autres. La notion d'écriture a jusqu'ici servi à lier humains et machines par une même activité, que ce soit du point de vue métaphorique[794] ou historique[795]. Nous souhaitons provisoirement défaire ce lien, en repartant du point de départ : l'écriture. Les { machines computationnelles } sont autres parce qu'elles inaugurent dans l'histoire de l'écriture une rupture sémiotique, tant qualitative (V. 1. A) que quantitative (V. 1. B). Un ordinateur calcule[796] à des échelles et à des volumes qui nous sont phénoménologiquement inaccessibles. Cette altérité n'est pas remise en cause par le fait qu'il soit programmé. Il existe en effet toute une tradition autour de l'argument de la { programmation } (V. 1. C), qui tendrait à prouver que, parce que certaines machines sont programmées, elles pourraient se réduire à des considérations humaines simplement transposées dans des commandes. Sans nier le fait que les ordinateurs sont programmés, et que donc ce qu'ils produisent peut être analysé comme le produit d'êtres humains, nous ferons valoir qu'il existe une part d'imprévu, un aléa machinique, et c'est cet imprévu résiduel qui nous intéresse.

794 Le modèle de Turing, comme nous l'avons montré, se base sur une comparaison entre la façon de calculer d'un humain et d'une machine, en supposant que le calcul peut être considéré comme une écriture.

795 Chez les théoriciens des médias informatisés, considérer ces machines comme des outils d'écriture permet de les placer dans une histoire et une culture qui déjouent les discours promotionnels sur l'innovation technique et la dimension « révolutionnaire » de ces dispositifs. Ainsi, les machines computationnelles sont ramenées dans le giron de l'histoire des médias.

796 Ce calcul étant une forme d'écriture, depuis le modèle de Hilbert repris par Turing, on peut dire en ce sens qu'une machine « écrit », même si cette écriture est fort différente de celle des humains.

1 A La rupture sémiotique

Les { machines computationnelles } sont différentes des humains en nature, voilà la première chose que nous voulons souligner. Cette altérité tient à deux points. Tout d'abord, leurs opérations ne sont pas accessibles à nos sens (V. 1. A. a) et cela en raison de leur matérialité même. Ce que l'on peut voir, c'est le niveau de l'affichage (la sémiotisation du { calcul }) et non le niveau de l'inscription (ce que la machine manipule et mémorise). Cette différence entre inscription et affichage (V. 1. A. b) est une rupture significative dans l'histoire de l'écriture et fait porter l'altérité de la machine à un niveau anthropologique : il en va de notre condition humaine et de notre appareil perceptif propre, différent en nature de la façon dont un ordinateur fonctionne.

architecture matérielle, calcul, carte perforée, formalisme, logique booléenne, machine computationnelle, porte logique

Voir glossaire tome II, p. 3-25

a Une rupture phénoménologique ancrée dans le matériel

La première chose qui différencie les { machines computationnelles } des êtres humains est le fait qu'elles opèrent à des échelles inaccessibles aux sens humains. Un ordinateur fonctionne grâce à un flux électrique conduit dans des circuits et à travers des composants électroniques comme des { portes logiques[797] }. Lorsque le composant laisse passer le flux[798], ce phénomène électronique est encodé avec une valeur positive (1). Lorsque le composant ne laisse pas passer le flux, cela est encodé avec une valeur nulle (0). L'agencement des différents composants – ce qu'on appelle l'« { architecture matérielle } » – permet donc de matérialiser des structures langagières { formelles }, fondées sur la { logique booléenne }. La miniaturisation et l'électronique permettent de produire ces opérations logiques en très grand nombre et très rapidement.

Nous avons montré dans les chapitres I et II en quoi cette { architecture matérielle } est l'expression d'une pensée combinatoire ramenée à ses plus simples atours : deux éléments non-ambivalents (1/0) ; des règles de permutations (par exemple la { logique booléenne }) et une économie maximale de moyens entre les éléments de base et le nombre de combinaisons possibles. Ce qui nous intéresse désormais est d'en montrer l'altérité : nous humains ne sommes pas en mesure de sentir la différence entre zéro

797 Une porte logique est un composant informatique construit sur le modèle de la logique formelle de Boole, et qui matérialise les opérations de cette logique : « ou », « et », « vrai si seulement », etc.

798 Leur sensitivité est reglée de façon à définir des seuils de pertinence autour de cinq volts, mais ce seuil peut changer selon les composants.

et cinq volts. C'est un constat d'ordre phénoménologique : nous n'avons pas l'appareil perceptif adéquat pour appréhender cette différence, tant du point de vue spatial que temporel. Un ordinateur va trop vite pour nous, et à une échelle trop petite[799]. En d'autres termes, la manière dont le média informatisé manipule l'information nous est inaccessible.

Cela vaut aussi pour le stockage de cette information : toucher du doigt le support magnétique d'un disque dur ou voir la mémoire flash d'une clé USB n'est d'aucune aide pour nous. Nos sens sont trop « grossiers », en tout cas inadaptés[800]. Les machines, même si leurs opérations sont extrêmement élémentaires, ont une « finesse de perception[801] » dont l'humain ne saurait se prévaloir. Ce qui ne veut pas dire qu'elles nous sont incompréhensibles, mais cela signifie qu'il faut en passer par des médiations à notre échelle pour comprendre leur action : la page, la { carte perforée }, l'écriture sous forme de 1 et de 0, le tableau d'instructions…. Il faut rendre sensible les opérations de la machine, en les transformant selon les règles de la raison graphique.

Voici donc une première différence significative entre les humains et les machines, et elle est ancrée dans l'organisation matérielle de ces dernières. La seconde concerne leur mode de fonctionnement : l'écriture.

799 Un transistor moderne exécute des instructions en quelques dix-milliardièmes de seconde et sur un espace de moins de 100 mm2. Voir Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Op. cit.*, p. 1.

800 On peut considérer que les technologies d'inscription magnétique ou phonographique (disques à microsillons) sont eux aussi inaccessibles aux sens humains sans dispositifs de conversion. On ne peut pas écouter un disque vinyl ou regarder une cassette VHS en les touchant de la main. L'information ne nous est pas accessible directement. Pour ces raisons, certains auteurs font de ces médias le véritable « fond culturel » des médias informatisés, en ceci qu'ils instaurent une relation spécifique au signe, caractérisée par le secret et la nécessité d'une conversion sensible. C'est vrai du point de vue des imaginaires du média, mais d'un point de vue technique, le fonctionnement est tout autre. Les vinyls ou VHS sont des enregistrements analogiques, ce qui veut dire que l'état de la matière (le microsillon, l'oxyde de fer de la bande magnétique) est analogue au signal que la machine amplifie. Le numérique introduit la notion d'encodage : tout contenu est ramené à une suite de 1 et de 0 qui n'a pas de rapport direct avec la matérialité de l'information encodée. Il y a donc une rupture en termes d'abstraction. De ce point de vue, l'informatique fait appel à une toute autre histoire des médias et des sciences. Sur ce sujet, voir Kittler, Friedrich A. *Gramophone, film, typewriter*. Stanford : Stanford University Press, 1999 [1986] ; Manovich, Lev. *The Language of New Media*. Cambridge : MIT Press, 2000.

801 Nous avons conscience de l'anthropomorphisme dont nous faisons ici preuve pour décrire la machine. Nous ne savons pas si un ordinateur « perçoit », nous ne souhaitons pas traiter cette question ici. La métaphore de l'organe pour décrire une machine – qui est introduite par Von Neumann en 1945 dans le *First Draft* […] – est une commodité rhétorique plutôt qu'une catégorie descriptive. L'enjeu est de rendre notre propos le plus clair possible.

b La scission entre l'inscription et l'affichage

I 3 A
machine « à écrire »
Voir p. 138

Dès lors que ce sont des machines « à écrire », la rupture est en effet sémiotique et elle est radicale. Comme le relève Emmanuël Souchier, c'est la première fois dans l'histoire de l'écriture que le support et le signe sont décorrélés[802]. Lorsque nous écrivons à l'aide de l'une de ces machines, ce que nous écrivons est stocké sous une forme inaccessible à nos sens. Nous voyons le signe qui s'affiche mais nous ne sommes pas en mesure de percevoir la manière dont il est stocké. Il y a une rupture entre la trace et le signe, c'est-à-dire entre le geste d'inscription – amené à être mémorisé – et le signe visible – amené à être lu[803]. Un signe est visible à l'écran mais il n'est pas inscrit et mémorisé sous la même forme, ni à la même échelle. Sans médiation architextuelle, nous n'y avons pas accès.

algorithme, calcul, littératie

Voir glossaire tome II, p. 3-25

La rupture est donc d'ordre anthropologique, et non une question de { littératie } ou de culture informatique : nous écrivons avec des machines qui prennent en charge une partie de cette écriture (l'inscription et le stockage) sans que **phénoménologiquement** nous puissions y avoir accès. À chaque fois que nous écrivons avec un média informatisé, nous composons avec quelque chose qui nous est étranger dans son fonctionnement. L'écriture avec ces machines est donc toujours une pratique d'hybridation entre deux modes d'existence, entre deux façons d' « écrire ».

I 3 C d
deux façons d'« écrire »
Voir les notions d'écriture-calcul et d'écriture-texte p. 153

Voilà une première irréductibilité des machines, et c'est une différence de nature. La seconde est une différence de degrés. Elle a trait à la complexité des couches scripturaires engagées dans la production d'un écrit d'écran.

802 « Trace et support ne vieillissent plus ensemble [...] ». Souchier, Emmanuël. *Op. cit.*, p. 108.

803 *Ibidem*.

1 B Couches & strates… Un mille-feuilles d'écritures

Quand bien même nous avons sur l'écran un texte visible et lisible, sommes-nous en mesure de retracer toutes les opérations et médiations techniques qui ont permis l'existence de ce texte ? A-t on les moyens de représenter – et donc de se donner les moyens de comprendre – tous les processus calculatoires engagés par une machine ? Cette question est d'importance : elle consiste à déterminer si l'altérité de ces médias est aussi une différence de degrés. Ce n'est plus ici la nature physique du { calcul } qui importe, c'est le volume des opérations réalisées et leurs imbrications. Toute la difficulté tient au fait que les écrits d'écran sont des « écrits en strates[804] » où la notion de « couches » joue un rôle central dans leur organisation logique[805]. Nous soutenons que cette stratification des écritures entraîne une différence non seulement de nature mais de degrés entre humains et machines.

Pour démontrer cette hypothèse, nous commencons par analyser comment s'organise cette écriture en couches, en montrant une scission entre unité sémiotique (la page web par exemple) et unité technique (l'une des procédures informatiques engagées dans la production de la page) (V. 1. B. a). C'est cette scission, et la nécessité d'unifier les différentes couches techniques engagées en une unité intelligible pour l'humain, qui amène à créer des procédures de conversion entre les différentes couches, ce qui multiplie le nombre de micro-opérations réalisées par la machine. En prenant l'exemple d'un { algorithme } de tracé de segment (IV. 1. B. b), nous montrons que ces micro-opérations sont à la fois nécessaires – pour rendre l'écrit d'écran lisible – mais tellement nombreuses qu'elles ne peuvent être représentées une à une. Tant et si bien que nous préférons parler de « mille-feuilles » d'écriture plutôt que de strates (V. 1. B. c) : chaque couche engagée dans la production d'un écrit d'écran est dépendante d'une autre, et elles sont si nombreuses qu'il est impossible de pouvoir se les représenter dans leur totalité, pour des raisons à la fois intellectuelles et matérielles. En cela, un ordinateur est différent d'un humain en degrés et non seulement en nature.

804 Cotte, Dominique. Écrits de réseaux, écrits en strates. Sens, technique, logique. *Hermès*. 2004, nº 39, p. 109-115.

805 Bachimont, Bruno. *Op. cit.*, 2010, p. 169.

a La dissociation entre unité technique et unité sémiotique : l'exemple de la page web.

API,
application,
balise,
boucle,
code source,
CSS,
DOM,
HTML,
JavaScript,
langage de programmation,
logiciel,
***middleware*,**
objet,
programmation,
programme,
script,
système d'exploitation,
widget

Voir glossaire tome II, p. 3-25

Toute page web, tout document sur un traitement de texte, toute { application } nécessite le fonctionnement conjoint de plusieurs { programmes[806] }. Ainsi, à l'unité sémiotique ou pratique (une { application }, une page…) ne correspond pas nécessairement une unité technique (une multiplicité de { programmes }).

Pour faire comprendre cette dissociation entre unité sémiotique et unité technique, prenons l'exemple d'une page web. Pour qu'elle puisse s'afficher en tant que « page », c'est-à-dire comme unité sémiotique et cognitive puisant dans une mémoire des formes, il faut qu'au moins quatre types de textes interagissent : le document écrit en { HTML }, le { CSS }, les { scripts } et le { DOM }.

L'{ HTML }, pour « *HyperText Markup Langage* » ({ langage } hypertextuel à { balise }) est un { langage } de **description** de la structure et du contenu d'une page web. À l'aide d'un système de { balises } délimitées par des chevrons simples (<>), on peut indiquer à une machine ce qui doit être affiché et comment l'organiser dans la page : niveaux de titre, mise en page, insertion d'un lien, d'une image, etc. On appelle le document écrit en { HTML } le { code source[807] } d'une page.

Il n'est pas encore question de la **forme** du texte. C'est le { CSS } (*Cascade Style Sheet*, feuilles de styles en cascade) qui définit l'aspect du texte, en se basant sur les niveaux de titres définis dans le document { HTML }. Par exemple, si dans le { CSS } on définit que tout titre de niveau 1 sera composé en Garamond et tout paragraphe avec une indentation de 1,25 cm, le navigateur lira le { CSS } puis le { code source } de la page pour déterminer à quels blocs de texte appliquer cette mise en forme.

806 Il ne faut pas confondre ici le « programme » et une « application ». Nous entendons par application un logiciel qui permet une utilisation spécifique d'un ordinateur. Nous entendons par programme un ensemble d'actions automatisées et écrites sous la forme d'un langage de programmation.

807 Nous reviendrons sur cette appellation problématique.

En l'état, la page web est statique. Ce sont les { **scripts** } qui permettent d'inclure dans la page des évènements déclenchés sous condition : un clic révélant une zone de texte, un décompte, des textes qui s'organisent selon leur date de publication… Les { scripts }, au contraire du { CSS } et de l'{ HTML }, sont des documents qui ne sont pas descriptifs mais utilisent des structures algorithmiques : un { langage } de { script } comme { JavaScript } se base par exemple sur beaucoup de { boucles } de conditions (« si ceci, alors fais cela, à condition que, sauf si », etc.).

Enfin, le { **DOM** } (*Document Object Model*, ou « Modèle à Objets d'un Document ») est une autre façon de représenter la page web. Cette dernière est représentée non pas sous formes de { balises } comme dans l'{ HTML } mais sous forme d'{ objets }, concepts issus d'un modèle de { programmation }. Les { scripts } peuvent faire appel au { DOM } pour interagir avec les éléments de la page, la représentation en { objets } étant plus adéquate à la structure de la plupart des { langages } de { scripts }.

I **1** **E** d
modèle de programmation
Voir p. 106

Il y a donc au moins quatre documents engagés dans la production d'une page web, quatre documents qui assument des fonctions bien différenciées mais qui sont en constante interaction, au sens informatique du terme : chaque document utilise certaines informations présentes dans un autre document pour fonctionner correctement. Et nous ne parlons pas ici de l'infrastructure réseau qui permet le transfert de données, ni du { logiciel } qui interprète ces documents (le navigateur), ni du { système d'exploitation } qui gère le navigateur et tous les autres { programmes } en jeu dans l'affichage de cette page web.

Où se situent les { API } dans ces couches ? Elles sont une partie des différents { programmes } impliqués dans l'existence de la page. Elles se situent « entre » différents documents (ce sont donc bien des { *middleware* }, des { logiciels } « entre deux[808] »). Twitter et Facebook utilisent le protocole HTPP pour transférer les données de leurs serveurs vers le navigateur de l'internaute, les { *widgets* } sont écrits en { HTML } et font appel à des { scripts } pour récupérer certaines données confidentielles (clés d'identification par exemple) auprès des plateformes publiant l'{ API }.

L'unité de la page est donc tout sauf une unité d'un point de vue technique, et pour qu'apparaisse cette unité sémiotique, il faut que de nombreux acteurs techniques fonctionnent en concorde, dont les { API[809] }.

808 Volonino, Linda et Dalal, Pragati. Middleware. Dans : Bidgoli, Hossein (dir.), *Op. cit.*, p. 33-44.

809 Rappelons ici la dimension idéologique de l'interopérabilité : il faut harmoniser les différentes couches d'écriture informatique pour pouvoir produire une représentation unifiée aux yeux du lecteur. Voir IV. 4. C. b.

b Une multitude de micro-opérations : l'exemple de l'algorithme de Bresenham

À cette scission entre unité technique et unité sémiotique s'ajoute le nombre considérable d'opérations de conversion entre chaque couche. Pour produire un texte numérique, il faut faire fonctionner ensemble de nombreux { langages }, protocoles, composants { matériels }, { programmes }, { interfaces }… et assurer les conversions[810], transformations qui assurent le passage entre le technique et le symbolique, entre le { calcul } de la machine et le signe lisible par l'humain.

algorithme,
***bug*,**
calcul,
discret,
HTML,
interface,
langage de programmation,
matériel,
pixel,
programme

Voir glossaire tome II, p. 3-25

Nous prendrons un exemple simple de ces conversions : l'{ algorithme } de Bresenham, développé par Jack Bresenham en 1965. Alors ingénieur chez IBM, il cherche un moyen de recréer un tracé sur un écran d 'ordinateur[811]. Le problème est simple, et articule très précisément un support (l'écran) et les formes sémiotiques qu'il permet (la ligne). Un écran informatique, par son organisation en { pixels }, rend difficile le tracé d'un segment et encore plus d'une courbe. Comment tracer un segment ou une courbe sur un espace – l'écran – composé de { pixels }, c'est-à-dire de points { discrets } répartis sur un espace orthonormé ? Comment tracer un arc de cercle ou un cercle complet avec un ensemble de carrés ? Cette quadrature du cercle est résolue par le { calcul } : par une suite finie d'instructions réalisables par une machine (l'{ algorithme } de Bresenham), on peut arriver à tracer un segment ou un cercle sur un écran, en dépit de la structure matérielle du support. Le résultat est une approximation, par accumulation de { pixels }, d'un tracé tel qu'on peut en trouver sur une feuille de papier.

810 Doueihi, Milad. *Op. cit.*, 2008.

811 Algorithme de tracé de segment de Bresenham, 2017. *Wikipedia*. [En ligne] [mis en ligne le 26 mai 2006] [dernière modification le 7 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Algorithme_de_trac%C3%A9_de_segment_de_Bresenham.

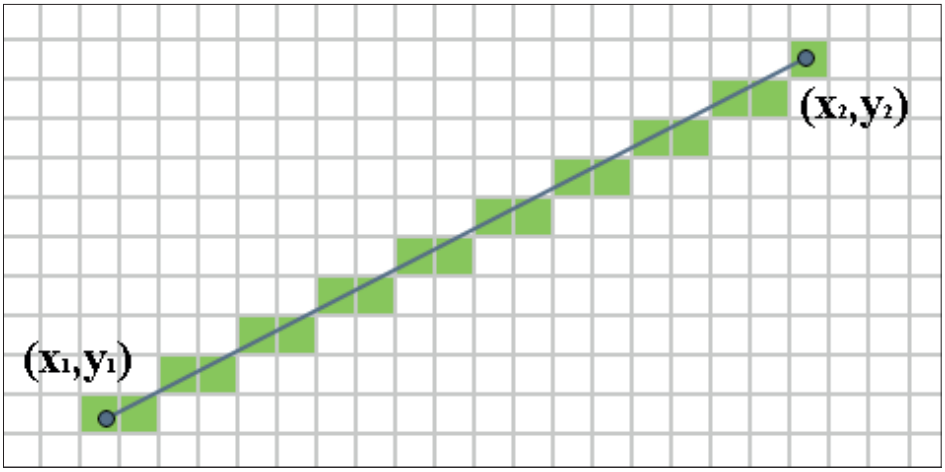


Fig. 55. Tracer une courbe avec des carrés : l'exemple de l'algorithme de tracé de Bresenham. Croquis réalisé par Zelda@fr.wikipedia — Travail personnel, Réalisé avec Photoshop, domaine public. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://commons.wikimedia.org/w/index.php?curid=17510903.

Cet exemple est assez daté, et il est fort probable que d'autres { algorithmes }, plus performants, ont remplacé la solution proposée par Bresenham. Il n'en reste pas moins qu'il nous fait prendre conscience de toutes les micro-actions qui entrent dans la composition de ce que nous lisons à l'écran : qu'on s'imagine combien de conversions de ce type sont effectuées pour afficher n'importe quelle page à l'écran ! Les { algorithmes } comme ceux de Bresenham ont une fonction de mise en forme du texte. Ils participent à l'énonciation éditoriale des écrits d'écran et reposent sur des { calculs } et des conversions : le tracé continu est converti en unités { discrètes }. Dans tout écrit d'écran, on trouverait une multitude de ces micro-actions faites par la machine et qui rendent possible le texte.

c Un mille-feuilles irreprésentable d'entités computationnelles au sein d'une même machine

Plus que de « strates », on pourrait donc parler de fonctionnement en « mille-feuilles ». Nous n'avons pas quelques strates empilées et indépendantes, mais une multitude de couches interdépendantes, et c'est précisément la multitude de ces couches qui fait la saveur de ce que l'on peut voir à l'écran. Si l'on isole une couche, on peut en analyser la cohérence interne (corriger les { *bugs* } d'un document { HTML } par exemple), mais on ne peut pas complètement présumer de l'aspect final de la page. Pour cela, il faut réinsérer cette couche dans l'ensemble des autres couches qui font l'existence de l'écrit d'écran. Tout comme le goût

d'un mille-feuilles ne se résume pas à l'addition des différentes couches qui le composent, la consistance sémiotechnique d'un écrit d'écran (mise en page, aspect du texte, efficacité des { signes passeurs }…) ne se résume pas à l'empilement de différentes écritures. Elle résulte d'une mise en coprésence de toutes ces écritures afin de permettre des échanges, des transferts et des actions réciproques. En ce sens, on peut dire que les écritures informatiques sont à resituer dans un contexte : dans un milieu composé d'autres textes qui vont permettre à chacun des textes individuels de fonctionner, tant sémiotiquement que techniquement.

API,
architexte,
binaire,
calcul,
discret,
entité computationnelle,
HTML,
interface graphique,
JavaScript,
langage de programmation,
langage machine,
machine computationnelle,
signe passeur,
système d'exploitation,
TCP/IP,
texte computationnel,
tweet

Voir glossaire tome II, p. 3-25

Cette analogie permet enfin de répondre à notre question initiale. Peut-on comprendre l'ensemble des processus calculatoires de la machine ? C'est-à-dire : peut-on comprendre toutes les couches d'écriture qui composent un écrit d'écran donné ? C'est théoriquement possible, mais très difficile dans la pratique. On peut comprendre les éléments de base ou une couche, mais il est très difficile de tout maîtriser, voire impossible. Maîtriser l'{ HTML } ou le { JavaScript } ne veut pas dire que l'on connaîtra le fonctionnement du protocole { TCP/IP } d'Internet ou du navigateur utilisé. De même, savoir lire du { langage machine } { binaire } ne veut pas dire qu'on saura écrire un { système d'exploitation } ou utiliser une { API } web. Ce sont là plusieurs { langages }, plusieurs savoirs, plusieurs domaines de compétences autour d'un même objet. Le premier problème tient donc aux savoirs nécessaires pour comprendre toutes les couches d'écriture impliquées dans la production d'un écrit d'écran.

Le second problème est d'ordre matériel. Peut-on rendre compte de toutes les micro-opérations de la machine ? Il faudrait pour cela représenter une par une **chaque** opération réalisée, et donc rendre lisibles **toutes** les conversions invisibles réalisées par la machine. Cela prendrait un temps considérable, demanderait une très grande connaissance de tous les processus engagés, et surtout cela nécessiterait matériellement de la place ! Cela est dû au phénomène d'inflation scripturaire relevé à propos des { tweets }. Changer de strate de représentation de l'écrit d'écran, « remonter » dans les couches de l'{ architexte } fait enfler la masse des écritures. Le même objet (un { tweet }, une courbe…) représenté sous sa forme calculatoire prend matériellement plus de place, tout simplement parce qu'il faut alors détailler chacune des opérations { discrètes } réalisées par la machine, et remonter à des niveaux de plus en plus élémentaires.

III 3 C c
inflation scripturaire
Voir p. 325

Ce qui amène à un paradoxe : nous ne pouvons pas connaître tout ce qui se passe à l'intérieur d'une { machine computationnelle }, même si cette machine est un objet matériellement attesté. Pour déjouer ce paradoxe, et préciser notre pensée, il faut distinguer trois niveaux :

a) La { **machine computationnelle** } : l'objet fait de composants électroniques qui permet la matérialisation du { calcul } et son exécution matérielle. Cette machine est finie, on peut la désigner par un singulier : un ordinateur, une tablette, un téléphone…

b) Les { **entités computationnelles** } : l'ensemble des opérations de { calcul } au sein de la { machine computationnelle }, réalisées par les différentes couches d'écriture dont nous parlons plus haut. Ce sont ces opérations qui sont en nombre indéfini. Elles représentent l'indéfini dans le fini – la machine. Ici, le pluriel s'impose car il n'y a jamais qu'une opération à la fois. La plupart de ces entités sont invisibles, certaines se donnent à voir dans…

c) les { **textes computationnels** } : ce qui s'affiche à l'écran, et qui est produit par l'ensemble des { entités computationnelles }. C'est le niveau de l'unité sémiotique de la page, de l'{ interface graphique }. Toutes les { entités computationnelles } n'apparaissent pas : certaines sont montrées, exhibées, d'autres sont masquées, ces jeux de visibilité établissant des rapports de pouvoir.

V | 2 | B | a
jeux de visibilité établissant des rapports de pouvoir
Voir p. 467

Notre argument est qu'entre ces trois niveaux, il faut se résigner à n'avoir accès qu'à certaines parties. Nous pouvons saisir la { machine computationnelle } dans son ensemble, mais il nous est impossible de nous représenter l'ensemble des { entités computationnelles }. Les { textes computationnels } nous sont intelligibles, mais là encore nous n'avons accès qu'à une partie des entités computationnelles à l'œuvre. Il y a toujours plus « à l'intérieur », ou en tout cas il y a toujours plus en jeu que ce que nous pouvons voir à l'écran.

1 C L'argument de la programmation

algorithme, calcul, énonciation machinique, programme

Voir glossaire tome II, p. 3-25

La différence entre l'humain et les médias informatisés est donc tant de degrés (V. I. B) que de nature (V. I. A). Il y a trop de { calculs } en jeu, et ces { calculs } sont inaccessibles à nos sens. Pourtant, une machine est programmée : en un sens elle obéit à l'être humain. On ne peut donc pas dire que la machine soit complètement « autre », puisque derrière chacune d'entre elles il y aurait un ou des humains qui auraient simplement transposé leur volonté dans des { programmes } ou des { algorithmes }. Par conséquent, ce à quoi nous avons affaire, ce sont bien en définitive d'autres êtres humains. Pourquoi alors prendre la peine de théoriser une « { énonciation machinique } », puisqu'il n'y a de toute façon que des humains qui énoncent dans un texte numérique ?

Cet argument selon lequel une machine informatique ne fait que ce qu'on lui commande de faire est pour nous central car il apparaît dès les premières machines programmables et structure tout un lot d'attitudes vis-à-vis des ordinateurs et de leur production (V. 1. C. a). Nous ne disons bien sûr pas que les ordinateurs ne sont pas programmés, ou que cela n'a aucune importance. Nous voulons avant tout montrer que cette question a tendance à effacer ce que nous appelons l'aléa machinique (V. 1. C. b) : le fait qu'on ne peut pas complètement déterminer l'issue d'un { calcul } à partir de ses conditions de départ. Cet aléa est pour nous inhérent à toute pratique d'écriture et tient à la nature scripturaire de nos relations avec la machine (V. 1. C. c). Ainsi, l'altérité machinique n'est pas remise en cause par le fait que ces machines soient programmées. Au contraire : la programmmation est le lieu par lequel on peut comprendre cette altérité, à condition de prendre en compte l'aléa machinique.

a La programmation contre la part machinique : une (pas si) vieille histoire

L'argument selon lequel, parce qu'une machine est programmable, elle n'est que le substitut technique d'une volonté humaine remonte au XIX^e^ siècle et est corrélé à tout un ensemble d'autres questions : est-ce qu'une machine peut être qualifiée d' « intelligente » ? Peut-elle « créer » ? Est-elle « autonome » au sens où elle pourrait faire montre d'une pensée ou d'une action propre ? Ces questions ont cours depuis les débuts de l'informatique elle-même – en fait depuis les premières mécanisations du raisonnement et du { calcul }. On en doit les premières formulations à Ada

Lovelace (1815-1852), connue pour avoir écrit le premier { algorithme } entièrement exécutable par une machine, la machine analytique de Babbage[812]. En 1842, Lovelace traduit et commente un rapport sur cette machine à calculer, dans lequel elle mentionne que parce que la machine calcule automatiquement, cela peut être pris pour « une forme d'intelligence », expression sur laquelle elle renvoie immédiatement à une note. Dans cette note, Lovelace va au devant des « […] idées exagérées qui pourraient émerger en ce qui concerne le pouvoir de la Machine Analytique [de Babbage][813] », idées exagérées qui peuvent tant être une surdétermination ou une sous-détermination de la technique et de ses effets. Dans le paragraphe suivant, Lovelace précise sa pensée :

> « *La Machine Analytique n'a aucune prétention à* ***créer*** *quoi que ce soit. Elle peut faire tout* ***ce que nous sommes capables de lui formuler comme ordre***[814]*. »*

Une machine programmable ne peut « créer » (*originate*) quoi que ce soit, c'est-à-dire qu'elle ne peut faire preuve d'intelligence (entendue comme faculté de production de savoir lié au contexte) ni de créativité : elle ne peut être, seule, à l'origine de quelque chose. Avec une machine programmable, rien de nouveau ou de surprenant ne peut arriver : elle ne peut traiter que ce que nous, humains, sommes capables de lui formuler en ses termes, ce qui suppose que nous sachions déjà ce que nous soumettons à la machine. En ce sens, une machine à calculer automatique est un gain de temps et d'énergie, mais n'amène pas *directement* à la production de nouvelles connaissances[815]. L'argument de Lovelace peut finalement être résumé assez simplement : on ne retrouve dans une machine que ce qu'on y a mis. L'*input* est équivalent à l'*output*, ou plutôt l'*output* peut être déduit de l'*input*. Ce qu'on met en entrée se retrouve à la sortie. La machine permet simplement (et c'est déjà considérable) de gagner du temps de { calcul }. Mais il n'y a aucun résultat qui ne soit pas déjà présent en germe dans ce qui est programmé en amont, et donc dans ce qu'un humain a écrit.

812 La machine analytique de Babbage est un calculateur mécanique théorisé et partiellement mis au point par l'ingénieur anglais Charles Babbage (1791-1871). Babbage fut le premier à penser une machine qui puisse calculer automatiquement par le biais de cartes perforées où l'on pourrait représenter les opérations à réaliser. En cela il est un des précurseurs de l'informatique et de la programmation.

813 « *It is desirable to guard against the possibility of exaggerated ideas that might arise as to the powers of the Analytical Engine.* » Lovelace, Ada. Sketch of the analytical engine invented by *Charles Babbage*, Esq. by *L-F. Menabrea*, of Turin, officer of the Military Engineers. Dans Taylor, Richard (dir.), *Scientific memoirs, selected from the transactions of foreign academies of science and learned societies and from foreign journals.* Volume III. London : Taylor and Francis, 1843, p. 666-731.

814 « *The Analytical Engine has no pretensions whatever to* ***originate*** *anything. It can do whatever we* ***know how to order it*** *to perform.* » Lovelace, Ada. *Idem*, note G, p. 722. C'est l'auteur qui souligne.

815 Lovelace, dans la même note, envisage toutefois le fait que de telles machines peuvent participer *indirectement* à la production de nouvelles connaissances, parce qu'elles permettent des manipulations inédites du savoir existant.

b Donner sa chance à l'aléa machinique

V | 4 | D | b
travaux en sciences sociales sur les algorithmes
Voir p. 513

Cet argument se retrouve sous plusieurs formes, dans différents contextes[816], notamment aujourd'hui dans les travaux en sciences sociales sur les { algorithmes }. Nous ne contestons pas l'idée que la machine ne « crée » pas ou ne peut pas faire preuve « d'initiative » au sens où elle pourrait nous proposer quelque chose de complètement inattendu et non-prévisible compte tenu de ses conditions de départ -de sa { programmation }. Il ne s'agit pas de donner des prérogatives humaines (l'intelligence, la créativité) à la machine[817]. Nous signalons simplement que, comme le dit Turing, « les machines nous surprennent souvent[818] » et qu'il faut expliquer et prendre au sérieux cette « surprise ». Tout le problème des conceptions de la { programmation } dans la lignée d'Ada Lovelace, c'est qu'elles proposent une vision instrumentale[819] de la machine : la médiation technique est le simple décalque d'une volonté. Nous restons dans l'ordre du même : nous restons entre humains. Cela nous empêche de penser les éventuels décalages entre ce qui est programmé, préécrit, et ce que nous retrouvons à l'issue du { calcul }. Or, ces décalages existent : il y a toujours des { *bugs* }, des résultats inexpliqués, surprenants… Il y a des « aléas machiniques » et nous sommes plus intéressé par des théories nous permettant de construire une relation avec les machines qui laissent apparaître cet aléa, plutôt que de seulement considérer les humains engagés dans la situation. Nous voulons faire apparaître la pluralité et la polyphonie des écrits d'écran, y compris les voix non-humaines.

algorithme, ***bug***, **calcul, développeur, programmation, programme**

Voir glossaire tome II, p. 3-25

Pour cela, nous proposons l'idée que dans tout acte de { programmation }, on ne retrouve pas **toujours** ce qu'on y a mis, et on ne le retrouve pas **à l'identique**. Il y a des décalages qui nous surprennent, et ces décalages sont le produit de la médiation technique même, indépendamment des humains qui organisent cette médiation. Cette idée est soutenue empiriquement par la constante des { *bugs* } et par la complexité de l'informatique contemporaine : nous ne sommes pas en mesure de dire tout ce qui intervient dans un { programme } ou dans un { algorithme }, ce qui laisse ouverte la possibilité de l'aléa machinique.

816 Turing le reprend dans son article de 1950 sur « l'intelligence artificielle ». Voir Turing, Alan M. Computing Machinery and Intelligence. *Op. cit.*, p. 447.

817 Ou en tous cas pas sans redéfinir précisément ce que peut être l'intelligence ou la créativité selon des termes qui peuvent être favorables à la machine, c'est-à-dire qui laissent la possibilité d'une forme d'intelligence ou d'une créativité machinique.

818 « *Machines take me by surprise with great frequency.* » Turing, Alan M. Computing Machinery and Intelligence. *Op. cit.*, p. 450.

819 Collomb, Cléo. *Op. cit.*, 2016, p. 180 et *passim*.

c Un aléa propre à toute pratique d'écriture

Cet aléa est dû en outre à la nature scripturaire de la { programmation }. Que provoque l'écriture ? Une surprise similaire : la surprise de voir que ce qu'on a écrit n'est pas exactement ce qu'on a pensé. Entre la pensée et sa forme médiée (l'écriture), il y a un décalage. Décalage induit par la confrontation entre un individu et une culture, une langue, un support… tout cela configurant l'acte d'écriture et faisant du texte ainsi écrit un « composite[820] ». Le geste de l'inscription, la confrontation avec un support crée des variations par rapport à ce qui est attendu. Tout comme l'écriture n'est pas le strict décalque de la pensée ou de la parole[821], la { programmation } n'est pas le décalque de la volonté de certains { développeurs } ou des entreprises les employant. La { programmation } est le produit d'une négociation entre des humains et des non-humains, sans qu'on puisse rigoureusement rabattre l'un sur l'autre.

Pour poser les bases épistémologiques d'une relation avec les machines qui prenne en compte leur altérité, la première chose à faire est donc de reconnaître cette altérité et de renoncer à la prétention de vouloir tout comprendre à ce que l'on voit. Lorsque nous faisons face à un écran, nous sommes bien sûr mis en relation avec des humains, des institutions, une culture, et ce par la médiation d'un texte. Mais nous sommes aussi mis en relation avec des machines qui ne fonctionnent pas selon le même mode que nous, et dont nous ne pouvons pas comprendre tous les tenants et les aboutissants. Cela ne veut pas dire que ces objets nous soient complètement étrangers, et qu'on devrait pour cette raison renoncer à toute intelligibilité. Il s'agit de se donner les conditions d'une relation qui puisse permettre de trier ce qui, dans un écrit d'écran, ressort de logiques communicationnelles humaines et ce qui ressort de logiques computationnelles machiniques.

820 Le Marec, Joëlle. *Op. cit.*

821 Derrida, Jacques. *Op. cit.*

Tout le problème étant que ces dernières logiques ne peuvent apparaître qu'à condition de se doter des outils conceptuels appropriés. En quoi nos outils (la sémiotique, les catégories d'écriture et de texte) posent-ils problème ? C'est ce que nous allons désormais montrer, en revenant à une question simple : par où est-on mis en relation avec les { machines computationnelles } ? Depuis le paradigme logique de Von Neumann, cette relation passe par des textes : une { interface graphique }, le { code source } d'un { logiciel } ou d'une page web, une ligne de commande… Autant de formes matérielles signifiantes qui construisent un espace de relation entre des machines et des humains. Comment donc l'écriture configure-t-elle cette relation ? Comment nos appellations ordinaires (« { code source } », « { interface } ») construisent-t-elles des rapports problématiques aux machines et à leur spécificité ?

I 2 A
paradigme logique de Von Neumann
Voir p. 114

code source, interface, interface graphique, logiciel, machine computationnelle, programmation

Voir glossaire tome II, p. 3-25

Humains et { machines computationnelles } ont donc deux modes de fonctionnement bien distincts, tant en degrés qu'en nature. Ceci montré, où se rencontrent humains et machines ? Par quoi sont-ils mis en relation, et comment cela structure-t-il cette relation ? Nous partons de l'hypothèse que leur point de rencontre privilégié, c'est le texte, et plus particulièrement ce qu'on appelle le { code source }. En quoi est-il un point de rencontre privilégié ? Tout d'abord parce que le { code source } est un texte hybride (V. 2. A) : il est lisible à la fois par un humain et une machine et fait agir celle-ci selon les instructions de celui-là.

Mais cet espace de mise en relation n'est pas neutre : par l'organisation des signes, des relations de pouvoir s'instaurent. En optant pour une conception politique du texte (V. 2. B.) dans la lignée de De Certeau, nous montrerons que la { programmation } prolonge une conception mythique de l'écriture où l'humain ordonne le monde autour de lui selon sa volonté propre.

Plus particulièrement, nous formulerons l'hypothèse selon laquelle la { programmation } prolonge cette conception autour de ce que Wendy Chun a nommé une « fétichisation[822] » du { code } (V. 2. C). Le { code informatique } est vu comme une écriture parfaitement performative : le mot fait agir une machine selon les volontés d'un humain. Le { code } est fétichisé comme source de ce qu'on voit à l'écran, tout en étant réduit à un rôle vicariant[823] : il ne fait que traduire des enjeux humains. Cette fétichisation s'organise selon nous autour de deux imaginaires primordiaux dans l'économie scripturaire de la { programmation } : la source (V. 3) et la commande (V. 4).

application,
code,
code source,
interface,
interface graphique,
langage de programmation,
machine computationnelle,
programmation

Voir glossaire tome II, p. 3-25

822 Chun, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Configurations*. 2008, vol. 16, nº 3, p. 299-324.

823 Est vicariant ce qui tient lieu de, qui remplace, qui assure la tâche de quelqu'un ou de quelque chose d'autre. Terme originellement religieux (dérivé de « vicaire »), il est habituellement utilisé en médecine ou en écologie pour désigner par exemple un organe qui supplée à un autre.

2 « ICI, ON NE COMPREND QUE CE QUI EST ÉCRIT[824] ». LE CODE COMME LIEU DE RENCONTRE

Nous avons jusqu'ici montré l'altérité des { machines computationnelles }. Une fois cette altérité établie, quels sont les lieux où humains et machines sont mis en relation ? Quelles sont les médiations qui instaurent ces lieux, et comment ces médiations configurent-elles nos relations aux machines ? Ce que l'humain et la machine ont en commun, c'est le texte et l'écriture. Nous pouvons voir les résultats des opérations de la machine grâce à une { interface graphique } (page web, { interface } d'une { application }…), nous pouvons écrire un { code source } qui donnera un certain nombre d'instructions pour la machine. Tant le { code source[825] } que l'{ interface graphique[826] } peuvent être considérés comme des textes[827]. Tous deux sont une organisation signifiante de signes qui crée un pont entre les machines et les humains. De la même façon, la machine « écrit » d'une certaine manière, les humains écrivent d'une autre manière mais tous deux partagent cette activité en commun.

I | 3 | C
les humains écrivent d'une autre manière
Voir p. 150

C'est la question du texte qui nous occupe pour l'instant. La spécificité de notre corpus est qu'il rassemble ces deux types de textes : des captures d'écran tirées de pages web tel que générées par notre navigateur ; le { code source } que le navigateur traite pour produire ces pages. Si les pages web sont aujourd'hui, en sémiotique, un objet d'analyse bien balisé, prendre le { code source } comme texte est une décision qui demande à être justifiée. Premièrement, si nous nous intéressons aux points de rencontre entre humains et machines, le { code source } – entendu comme l'ensemble des documents rédigés dans un { langage de programmation }

824 De Certeau, Michel. *Op. cit.*, p. 199.

825 Souchier, Emmanuël et Pomian, Joanna. Informatique et pratiques écrivantes. *Op. cit.*, p. 121-130 ; Knuth, Donald E. Literate Programming. *Op. cit.*, p. 97-111.

826 Candel, Étienne. *Textualiser les interfaces. Opérativité et épistémologie d'une requalification*. Habilitation à diriger des recherches. Paris : Paris-Sorbonne, 2015.

827 Nous adoptons pour l'instant une définition très large et imprécise d'une « interface graphique ». Nous entendons par ce terme tout écrit d'écran qui permet l'utilisation d'une machine computationnelle par le biais de signes intelligibles pour l'humain. Ces signes puisent dans une culture sémiotique plus ou moins spécialisée, et reposent sur des métaphores plus ou moins compréhensibles selon la littératie du scripteur. Par exemple, les « *Graphical User Interface* » (GUI) apparues dans les années 1990 (avec un « bureau », des « dossiers », des « documents ») généralisent la métaphore de la bureautique alors que la ligne de commande, autre interface graphique, a un fonctionnement sémiotique différent et nécessite des savoirs plus spécialisés. Même si tous deux – GUI et ligne de commande – sont des « interfaces graphiques » au sens où elles textualisent de façon intelligible pour l'humain les opérations de la machine.

architecture de Von Neumann, code, code source, écriture-calcul, langage de programmation, machine computationnelle, programme

Voir glossaire tome II, p. 3-25

qui permettent l'exécution d'un { programme } ou d'une page web – est un point de rencontre privilégié : il est un ensemble d'instructions que l'on donne à un ordinateur, rédigé dans un { langage } et selon une syntaxe nécessitant de se plier à une certaine logique technique. Si le { code source } est donc un texte qui rassemble humains et machines, il permet et construit cette relation tout à la fois. Il faut donc analyser en quoi le { code source } est un cadre instituant de nos relations aux machines, en faisant l'hypothèse qu'il existe une certaine conception du { code } qui invisibilise l'action des machines.

Pour vérifier cette hypothèse, nous montrons d'abord que la notion de { code } comme texte hybride dérive de l'{ architecture de Von Neumann }, au sens où c'est avec ce choix technique que la { programmation } devient une pratique d'écriture et le { code } un texte (V. 2. A). Si le { code } est un texte, comment organise-t-il les relations entre humains et machines ? Quelle place est donnée à chaque acteur ? Pour répondre à ces questions, il faut réorienter notre définition du texte. Plutôt que de s'interroger sur l'organisation et les modèles culturels mobilisés dans cette organisation, nous considérons avec Michel De Certeau les effets politiques du texte (V. 2. B). Tout texte s'inscrit dans une économie scripturaire, qui organise les relations entre les entités impliquées dans sa production. En cela, le texte est un lieu de pouvoir. Nous pourrons alors étendre les thèses de Michel de Certeau à la { programmation }, attendu que la { programmation } est une pratique d'écriture. Cela nous amènera à mettre en évidence une dynamique de « fétichisation[828] » du { code informatique } (V. 2. C), cette fétichisation étant la conception du { code } qui participe à invisibiliser les machines.

Ainsi sera cernée l'une des spécificités de « l'économie scripturaire » de la { programmation }, ce qui nous permettra dans la partie suivante d'étudier deux imaginaires structurants de la { programmation } : la « source » (V. 3) et la « commande » (V. 4).

828 Chun, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Op. cit.*, p. 299-324.

2 A Du texte comme lieu de rencontre : l'hybridité du code source

Qu'est-ce qui nous permet de dire que c'est le texte et l'écriture qui sont le point de rencontre entre humains et machines ? En ce qui concerne l'écriture, nous avons montré comment Turing utilise cette analogie pour décrire les opérations de sa machine : une { machine computationnelle } « écrit » au sens où elle inscrit des symboles univoques sur un espace discrétisé et monodimensionnel. Cette « { écriture-calcul automatique } » est motivée par un texte préalable : les instructions données à la machine, ce qu'on appelle le { code source }. Mais qu'est-ce qui fait qu'on peut programmer, donner des instructions à une machine, par le biais d'un texte ? À partir de quand a-t-on donné des instructions à un calculateur électronique par le biais d'une écriture ? Quel est le cadre intellectuel et technique dans lequel cette opération liant texte et instructions machiniques est possible ? Et surtout, comment ce cadre institue-t-il une certaine relation aux machines programmées ?

I 3 B
Turing utilise cette analogie
Voir p. 143

I 3 C e
écriture-calcul automatique
Voir p. 154

Cette conception de la { programmation } comme texte est selon nous une conséquence du paradigme logique de Von Neumann (V. 2. A. a) adopté à la fin des années 1940. C'est avec ce choix technologique que la { programmation } devient une pratique d'écriture et le { code source } un texte hybride (V. 2. A. b), assurant le lien entre humains et machines. La prise en compte de cette hybridité permet de différencier trois niveaux d'analyse du { code source } (V. 2. A. c), selon le pôle (humain / machine) choisi comme point d'entrée, et ainsi de préciser à quel niveau nous nous situons.

I 2 A
paradigme logique de Von Neumann
Voir p. 114

a Retour au paradigme logique de Von Neumann : la programmation comme production d'un texte

architecture de Von Neumann, calcul, carte graphique, carte perforée, codage, code, code source, langage de programmation, machine computationnelle, matériel, programmation, programme, technologie de l'intellect

Voir glossaire tome II, p. 3-25

Le cadre intellectuel qui préside à la transformation de la { programmation } comme écriture, nous l'avons déjà croisé : c'est l'adoption à la fin des années 1940 de l'architecture dite « de Von Neumann ». C'est un modèle d'organisation matérielle et logique de la machine où les instructions sont stockées et traitées sous la même forme que les données. Pour effectuer un { calcul }, il faut en effet un certain nombre de « données », éléments sur lesquels vont porter le { calcul } et un certain nombre d'instructions, soit ce que la machine doit faire avec les données. Aujourd'hui, l'{ architecture de Von Neumann } est l'organisation de base de tout ordinateur. Données et instructions peuvent être représentées sous la même forme : une ligne de { code } où vont être indiqués à la fois quoi calculer et comment le calculer.

Mais il n'en a pas toujours été ainsi et au moment où Von Neumann et ses collègues pensent cette nouvelle façon de programmer, le modèle alors dominant est celui de l'ENIAC (*Electronic Numerical Integrator And Computer*). L'ENIAC est une machine composée de différentes unités, chacune consacrée à une opération spécifique : additionner, multiplier, contrôler les instructions, stocker des informations, etc. Les données sont rentrées sous la forme de { cartes perforées }, mais les instructions sont transmises en cablânt les différentes unités entre elles[829]. Dans ce contexte, programmer n'est pas une écriture mais un agencement physique de câbles reliant les différentes parties de la machine entre elles. Chaque reprogrammation est un réagencement de ces câbles, ce qui prend un temps considérable. C'est pour pallier à ces désagréments (reprogrammation lente et fastidieuse) que l'{ architecture de Von Neumann } est adoptée, avec le but escompté : si données et instructions prennent la même forme, par exemple des { cartes perforées } ou un texte intelligible pour une machine, il est bien plus simple et bien plus rapide de reprogrammer cette machine[830].

I 2 A b
chaque reprogrammation est un réagencement
Voir p. 115

829 De Mol, Liesbeth, Carlé, Martin et Bullynck, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Op. cit.*, p. 1011-1046.

830 Rappelons ici l'importance du caractère « général » d'un ordinateur : ce sont des machines créées pour pouvoir résoudre rapidement un grand nombre de problèmes, tant que ces problèmes sont formulables en termes calculables. La machine est générale, les problèmes sont particuliers : c'est une certaine configuration de la machine. Dans ces conditions, une machine générale mais dont la reprogrammation pour chaque problème est lente et complexe présente un intérêt moindre qu'une machine plus rapide à reprogrammer.

I 2 A c
paradigme logique de la programmation
Voir p. 116

L'{ architecture de Von Neumann } entraîne deux autres conséquences. La première a été étudiée dans ce travail : c'est l'avènement d'un paradigme logique de la { programmation }, soit l'idée qu'un { programme } ne dépend plus principalement de l'agencement { matériel } d'une machine mais peut être représenté de façon abstraite sur une feuille de papier. La seconde conséquence, en lien avec la première, c'est que la { programmation } devient une pratique d'écriture, avec tous les avantages de cette { technologie de l'intellect } : on peut représenter un { programme } informatique sur un espace restreint (une page, un livre), annoter et modifier plus facilement, imaginer le déroulement du { programme } par sa seule organisation sur la page… La { programmation } devient production d'un texte. À partir de ce moment, un ordinateur devient une machine commandée par l'écriture. En reprenant l'expression de De Certeau, on peut dire qu'« [elle] ne comprend que ce qui est écrit[831] […] ».

b Le code source comme texte hybride

Le « { code } » acquiert par conséquent un nouveau statut : il doit être intelligible autant pour une machine que pour un être humain. Il devient un texte hybride où se rencontrent ces deux façons de faire, ces deux façons d'écrire. Il devient ce qu'on appelle aujourd'hui un « { code source } » : une liste d'instructions pour une machine, rédigée par un humain et lisible par ces deux entités. Il est donc temps de passer un peu de temps sur cette notion de « { code source } », puisqu'il est un des lieux privilégiés de nos relations avec les { machines computationnelles }.

Le { code source } désigne un ensemble de textes ou ressources permettant l'exécution d'un { programme } ou d'une page web. Chaque texte contient un certain nombre d'instructions données à la machine, rédigées selon la syntaxe d'un { langage de programmation[832] }. Initialement, le { code source } est lié à la machine sur laquelle le { programme } s'exécute. En effet, pour que les instructions soient intelligibles pour une machine, il faut que ces instructions soient rédigées en respectant la syntaxe propre à cette machine, à commencer par son { encodage } initial. Imaginons par exemple que nous voulions écrire un { programme } qui utilise telle

831 « "Ici, on ne comprend que ce qui est écrit". Tel est la loi interne de ce qui s'est constitué comme "occidental" ». De Certeau, Michel. *Op. cit.*, p. 199.

832 D'un point de vue technique, il existe toutefois une différence importante entre un programme et une page web, notamment en ce qui concerne le code source. Lorsque qu'une application est écrite, par exemple en langage C++, le ou les documents produits doivent être compilés, c'est-à-dire traduits en langage machine. Lorsque l'application est lancée, les fichiers compilés sont « exécutés », sont retraduits sous forme intelligible par les humains, la plupart du temps l'interface graphique de l'application. Le code source n'est donc pas mobilisé directement et systématiquement lors de l'utilisation de l'application. Au contraire, un fichier HTML ou JavaScript est dit écrit en langage interprété, au sens où le document n'est pas compilé au préalable mais traité directement par le navigateur web.

ressource de la { carte graphique } ou qui nécessite d'interagir d'une certaine manière avec la mémoire de la machine. Nous avons besoin de savoir quelles sont les instructions qui vont déclencher ces actions, et ces instructions sont encodées (telle série de caractères { binaires } entraîne telle action) selon les composants { matériels } d'une machine donnée. Initialement donc, le « { code source } » désigne un texte certes écrit par un humain mais déterminé par la matérialité de la machine à qui s'adresse aussi ce texte.

Avec le développement et la démocratisation de la { programmation }, cette écriture va progressivement s'abstraire de la matérialité et donc de la spécificité des machines. Dans la lignée du mouvement d'abstraction de l'écriture décrit par ailleurs, il n'y a plus besoin aujourd'hui de connaître le { langage machine } pour pouvoir écrire un { code source }. Mais cette abstraction est le résultat d'une dynamique historique et communicationnelle, qui tend à faire du { code source } un texte adressé à des humains plutôt qu'à des machines.

I | 2 | B
décrit par ailleurs
Voir p. 122

c Des trois niveaux du code.

Il faut donc étudier le { code source } comme un gradient entre ces deux pôles : l'humain et la machine. Un { code source } est un texte, comme son nom l'indique, codé, c'est-à-dire qu'il traduit en { langage } informatique ce que le { développeur } ordonne à la machine de faire afin qu'elle produise une { interface } utilisable par un utilisateur. C'est un texte fondamentalement hybride, situé entre la machine et l'être humain. En revanche, « l'être humain » peut désigner soit le { développeur } (un professionnel du { code }) soit l'utilisateur, qui est engagé avec le { code } d'une page web *via* une { interface graphique }. On peut donc considérer que trois entités sont engagées : les { entités computationnelles } ; le { développeur } ; l'utilisateur. Chacun entretient un rapport spécifique au { code }. Les { entités computationnelles } l'exécutent ; le { développeur } l'écrit et le lit ; l'internaute voit ses actions définies par le { code }. Bien sûr, ces trois rapports sont archétypaux et à nuancer : un utilisateur particulièrement versé en { programmation } n'aura pas le même regard qu'un néophyte. Mais sur le gradient des attitudes possibles vis-à-vis du { code informatique }, nous avons choisi, par clarté d'exposition, d'en isoler trois. Il s'agit moins de simplification falsificatrice que de volonté de rétablir une pluralité d'approche du { code } sans être condamné à l'exposition de cas particuliers.

binaire,
***bug*,**
carte graphique,
code,
code source,
commentaire,
développeur,
entité computationnelle,
interface,
interface graphique,
langage de programmation,
langage machine,
matériel,
programmation

Voir glossaire tome II, p. 3-25

Ces trois attitudes sont analysables sous l'angle de la lecture. Le { code source } d'un site web est d'abord lu par des { entités computationnelles[833] }. C'est ce que nous pourrons appeler le niveau technique, qui tire son efficacité du respect de la syntaxe et des règles de composition. Sa qualité esthétique repose en premier lieu sur la correction de son écriture (le { code } ne doit pas comporter de { *bugs* }) mais aussi sur l'économie de l'écriture par rapport au problème posé : le { code } est « élégant » ou « beau » quand il accomplit son rôle avec le minimum d'instructions, ou par une solution peu attendue, ce qui permet d'utiliser le moins de mémoire possible. C'est à ce niveau que s'exerce ce que Bruno Bachimont appelle la « raison computationnelle[834] » ou ce qu'Adrian MacKenzie nomme la « performance[835] » du { code }.

Il sera lu, en tant que document rédigé par un { développeur }, par d'autres { développeurs }. C'est ce que nous pourrons appeler le niveau socio-sémiotique, soit le { code } en tant que document dont l'organisation visuelle fait appel à une certaine culture professionnelle. Dans ce cas, le { code } est envisagé non pas du point de vue de l'{ interface } qu'il permet d'afficher mais comme document de travail. La valeur esthétique du { code } ne tient plus à son efficacité machinique mais à sa lisibilité pour les humains (indentation, utilisation d'un vocabulaire clair, { commentaires } fournis). Cette lisibilité a un enjeu également politique et économique, car un { code } suffisamment commenté pourra être manipulé, repris par d'autres { développeurs }… il pourra en somme circuler dans un espace socio-professionnel plus large, même s'il reste identique d'un point de vue technique. C'est ce dernier niveau que MacKenzie appelle la « performativité[836] » du { code }.

833 Je ne discuterai pas ici de la nature métaphorique ou non du terme de « lecture » appliqué aux médias informatisés.

834 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 195-225.

835 Mackenzie, Adrian. The Performativity of Code : Software and Cultures of Circulation. *Op. cit.*, p. 71-92.

836 *Ibidem*.

Enfin, il prépare et structure une { interface } à même d'être lue et interprétée par un internaute. C'est le niveau techno-sémiotique, soit le { code } en tant qu'il fixe les conditions de possibilités d'écriture de l'utilisateur. C'est l'analyse du { code source } comme { architexte }, où se pose un problème éminemment politique de contrôle et délégation de l'écriture. La « beauté » du { code } ou sa valeur esthétique ou politique tient alors surtout à la marge de manœuvre qu'il laisse à l'utilisateur[837].

architexte,
interface,
code,
code source,
programmation

Voir glossaire tome II, p. 3-25

Ces trois niveaux permettent de poser les bases théoriques d'une compréhension plus fine de ce qu'est un « { code source } », en permettant de circuler sur le gradient constitué par les deux pôles que ce texte met en relation : les machines et les humains. Reste le présupposé principal de notre démonstration : le texte est un espace de mise en relation. Ici, ces relations concernent les humains et les machines, mais ce n'est qu'un exemple d'une propriété plus générale de tout texte. Le { code }, étant un texte, est un espace de mise en relation. Cette proposition demande d'expliciter la conception du texte que nous mobilisons.

837 Masure, Anthony. *Le design des programmes : des façons de faire du numérique*. Thèse de doctorat. Paris : Paris 1, 2014.

2 B Une conception politique du texte

Pour comprendre comment cette scripturisation de la { programmation } produit des effets politiques, il nous faut changer notre définition du texte, autour du travail de Michel De Certeau sur l'« économie scripturaire » moderne et sur l'écriture comme « mythe » . Nous entendons par texte un espace de mise en relation de différentes entités, cette mise en relation se faisant par le biais de signes et sous l'égide du pouvoir : à certaines entités sont assignés certains rôles plus ou moins valorisés. En ce sens, le texte est un lieu de pouvoir[838] (V. 2. B. a), un lieu d'où s'exerce un pouvoir (celui d'écrire) ; un lieu où se lisent des rapports de pouvoir (par l'image du texte). Dans ce contexte, l'écriture est un « mythe » au sens où elle a pour but d'organiser les relations de pouvoir à l'œuvre dans le texte (V. 2. B. b). Écrire, c'est articuler différentes entités au sein d'un texte et organiser des relations de pouvoir.

Une fois réorientées ces définitions du texte et de l'écriture, nous pourrons voir comment le { code informatique } prolonge cette économie scripturaire, notamment en fétichisant le { code source } comme décalque d'une volonté humaine (V. 3)

a L'écriture espace du propre, le texte comme lieu de pouvoir

Michel De Certeau élabore sa théorie de l'écriture dans le cadre plus global d'une interrogation sur l'extension de la « modernité » occidentale (qu'il situe entre le XVII^e^ et le XVIII^e^ siècle), la manière dont cette modernité s'est articulée autour de l'écriture et dont elle a progressivement évacué l'oralité[839]. D'emblée, De Certeau se préoccupe des enjeux de pouvoir : il voit l'écriture telle qu'elle s'est développée en Occident comme une mise au pas de certaines pratiques, de certains savoirs qui auraient résistés au développement de l'écriture. « L'économie scripturaire » (le titre du chapitre X de *L'Invention du quotidien* où se trouve l'essentiel des citations que nous allons mobiliser), c'est cette expansion de l'écriture en Occident comme activité noble, créant du savoir, de la valeur, et organisant plus largement la société.

838 Collomb, Cléo et Goyet, Samuel. *Op. cit*. 2015, p. 2.

839 « Je cherche à entendre ces fragiles effets de corps dans la langue, voix multiples, mises à distance par la triomphale *conquista* de l'économie qui, depuis la " modernité [...] s'est titularisée sous le nom d'écriture. Mon sujet, c'est l'oralité, mais changée par trois ou quatre siècles de travail occidental. » De Certeau, Michel. *Op. cit.*, p. 195.

Qu'est-ce qu' « écrire » pour De Certeau ?

> *« Je désigne par écriture l'activité concrète qui consiste sur un espace propre, la page, à construire un texte qui a pouvoir sur l'extériorité dont il a d'abord été isolé*[840]*. »*

Trois termes sont essentiels pour comprendre cette définition. « *L'espace propre* » tout d'abord. Écrire, c'est inscrire des signes sur une surface, que ce soit du papier, de la pierre ou un écran[841]. Cet espace est construit, découpé, séparé du « cosmos traditionnel » et dans cette séparation « [...] désensorcelé des ambiguïtés du monde[842] ». La page est une abstraction : elle est extraite du monde – un monde complexe, multiple – et institue « une surface autonome » qu'un sujet écrivant va pouvoir modeler à sa guise[843]. C'est en ce sens que c'est un espace « propre » : propre au sens où elle appartient spécifiquement à l'écrivant qui peut y déployer son écriture ; « propre » au sens où elle est en quelque sorte « nettoyée » de toute interférence extérieure, de toute la complexité du monde. La page, dans cette découpe et ce retrait, est un « geste cartésien [...] instaurant, avec un *lieu* d'écriture, la maîtrise (et l'isolement), d'un sujet devant un *objet*[844] ». De Certeau rejoint Illich sur ce point[845] : l'écriture, et plus particulièrement la page comme unité intellectuelle où vient s'ancrer le texte, a participé à construire cette figure occidentale du « sujet pensant », seul maître de ses désirs et de ses pensées. À cette pensée « en propre » correspond la page où va se déployer cette pensée. De ce point de vue, écrire, c'est se produire en tant que sujet abstrait et isolé, dans un espace qui est le sien.

Le « texte » est défini par De Certeau comme « des fragments ou matériaux linguistiques [traités] dans cet espace [...] de manière à produire un ordre[846] ». Sur l'espace de la page, un sujet assemble des éléments disparates, crée une architecture. Cette architecture est une mise en ordre : le sujet organise par l'écriture ces fragments selon ce qu'il juge important, accessoire, etc. C'est cette dimension architecturale qui fait du texte un lieu de pouvoir : l'organisation de la page se fait en fonction d'une axiologie,

840 De Certeau, Michel. *Idem*, p. 199.

841 Nous supposons que De Certeau entend par « page » une unité plus intellectuelle que matérielle, dans la lignée d'Illich. Ce n'est pas parce que nous écrivons sur un écran que nous n'avons pas le regard construit par la « page » comme mètre-étalon.

842 De Certeau, Michel. *Op. cit.*, p. 199.

843 Ces réflexions rejoignent en partie celle d'Anne-Marie Christin sur la « pensée de l'écran » comme opération fondamentale de l'écriture. Voir Christin, Anne-Marie. *Op. cit.*, p. 6.

844 De Certeau, Michel. *Op. cit.*, p. 199.

845 Illich, Ivan. *Op. cit.*, p. 105-106.

846 De Certeau, Michel. *Op. cit.*, p. 199-200.

d'un système de valeurs, reléguant le négligeable dans les marges, mettant l'important au centre ou de manière visible. La définition certalienne du texte rejoint ici la théorie de l'énonciation éditoriale[847], cette dernière fournissant les outils d'analyse de ce lieu de pouvoir qu'est le texte.

Ce texte – cet agencement de fragments – n'est pas vain mais « […] a pour "sens" de renvoyer à la réalité dont il a été distingué en vue de la changer. Il vise une efficacité sociale[848]. » C'est le troisième aspect de l'écriture : le texte a « […] pouvoir sur une extériorité dont il a d'abord été isolé ». Écrire, c'est s'abstraire du monde et se retrancher dans son « espace propre », choisir les éléments du monde pertinents pour exercer notre « vouloir propre » et ensuite disposer ces éléments sur la page. Mais en retour, cette disposition a pour enjeu de transformer le monde, cette « extériorité ». Une fois filtré, nettoyé par la production d'un texte, le monde est transformé par ce texte. L'écriture, dans ce sens, n'est pas seulement la production d'un sujet, mais aussi la production du social, sa modification par filtration préalable. Écrire, c'est se produire comme sujet et participer à produire un monde à soi. De ce fait, le monde n'est plus « reçu mais fabriqué[849] ». L'écriture est ce témoignage de l'exercice d'une volonté souveraine, dans son espace propre, qui recrée artificiellement un monde à l'image de sa volonté.

De Certeau prend l'exemple de *Robinson Crusoé*, ou de l'idée de « révolution » comme fruits de cette modernité scripturaire. On peut prendre l'exemple, plus simple, d'un architecte qui dessinerait un bâtiment. Il commence par prendre une feuille de papier ou ouvrir un { logiciel } dédié. C'est son « espace propre » où il va pouvoir dessiner. Il organise ensuite à la surface de l'écran ou de la page un bâtiment en puisant dans plusieurs champs disparates : sa culture visuelle, les recommandations du client, les obligations légales (issues de secours, accès handicapés…). Il précise les différents matériaux à utiliser, la forme des pièces, la taille du bâtiment, son insertion dans le paysage… Il agence des fragments épars du monde extérieur, pour les rassembler par son geste d'écriture. Cet agencement crée une nouvelle réalité (ici un bâtiment) qui va se réinsérer dans le monde ainsi mis en ordre : le texte trouve son « efficacité sociale ». Au mieux, le bâtiment sera construit. Au pire, le dessin restera à l'état de projet. Mais même comme projet, il est un élément du monde que l'architecte pourra remobiliser lors de ses prochaines écritures. Le projet aura donc modifié le monde existant.

logiciel

Voir glossaire tome II, p. 3-25

847 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

848 De Certeau, Michel. *Op. cit.*, p. 200.

849 *Ibidem*.

b L'écriture comme mythe

API, code source, logiciel, programmation

Voir glossaire tome II, p. 3-25

Évidemment, cet exemple montre à quel point cette définition de l'écriture est en apparence fausse. L'architecte compose selon la culture (esthétique, technique) dans laquelle il évolue ; son { logiciel } le contraint d'une certaine manière, ainsi que l'ensemble des lois régissant la construction d'un bâtiment. Sans compter les impératifs économiques de sa production, qui varient considérablement selon la situation. Il n'y a donc pas de « propre » dans l'écriture, puisque nous sommes toujours en négociation avec des autres : notre culture, le langage, le cadre juridique ou économique, le support, les lecteurs supposés, imaginés, attendus… Comment comprendre alors cette définition De Certeau ?

L'évacuer comme fausse serait négliger le fait que cette définition de l'écriture est un « mythe », défini comme « […] un discours fragmenté qui s'articule sur les pratiques hétérogènes d'une société et qui les articule symboliquement[850] ». En Occident, nous articulons symboliquement des pratiques hétérogènes par l'écriture. L'écriture est un rassemblement et une unification, ou plutôt : elle a vocation à l'être. C'est un « mythe » en ceci que De Certeau ne se situe pas au niveau des pratiques concrètes – marquées par la pluralité – mais au niveau des imaginaires, des horizons idéologiques, des systèmes de valeurs. Écrire, cela **doit** être rassembler, se constituer en sujet maître de lui-même. C'est « […] l'utopie fondamentale et généralisée de l'Occident moderne[851] ». Une utopie qui a des effets concrets sur la manière dont la société s'organise : on apprend à écrire au plus jeune âge, on apprend à oublier la matérialité du papier, à négliger certains acteurs du texte, à ne considérer que l'auteur… Par ces médiations, l'écriture comme mythe gagne en opérativité symbolique.

850 De Certeau, Michel. *Idem*, p. 198.

851 De Certeau, Michel. *Idem*, p. 200.

Fort de ces conceptions du texte et de l'écriture, l'hypothèse que nous faisons désormais est que la { programmation } et les { API } sont un développement particulièrement poussé de cette « économie scripturaire », de ce mythe occidental de l'écriture, et notamment par le biais du { code source } et des imaginaires qu'il convoque. En quoi la { programmation } fait-elle partie de cette conception mythique de l'écriture ? Comment prolonge-t-elle et radicalise-t-elle la dynamique mythique que De Certeau a vu poindre dans la modernité occidentale ? En somme, quelle est l'économie scripturaire du { code source } ?

2 C La « fétichisation » du code, ou le prolongement informatique de l'économie scripturaire moderne

code, code source, interface, langage de programmation, logiciel, programmation

Voir glossaire tome II, p. 3-25

Si la { programmation } est une pratique d'écriture[852] et que l'écriture peut être comprise comme « mythe » au sens de Michel De Certeau, quel peut être l'économie scripturaire de la { programmation } ? Notre hypothèse est que la { programmation } confirme et radicalise la tendance relevée par De Certeau : l'écriture est une pratique d'abstraction, de subjectivation, qui a pour visée la transformation du monde extérieur. La { programmation } renforce cette dynamique notamment parce qu'elle est une écriture de commande : le { code } fait agir la machine. Cet aspect de commande nourrit le fantasme de ce que la théoricienne des médias Wendy Chun a appelé la « fétichisation[853] » du { code } (V. 2. C. a). Cette fétichisation repose sur deux dynamiques : l'ensourcellement et la sourcification (V. 2. C. b) : comment un { code } devient-il source de quoi que ce soit ? Par quelles médiations techniques et sémiotiques est-il institué comme ayant du pouvoir ? Et de quel pouvoir parle-t-on exactement ? Est-ce que tout ce qu'on voit à l'écran est encodé au préalable, et si non pourquoi est-il important de penser, malgré tout, qu'un { code } est à la source de ce qu'on lit ?

Une fois ces questions traitées, nous pourrons alors étudier les imaginaires respectifs de ces deux dynamiques : celui du { code source } (V. 3) et celui de la commande (V. 4).

a Fétichisation & sourcellerie

La « fétichisation » est une hypothèse de Wendy Chun. Selon elle, le { code source } est « fétichisé » et nous, chercheurs et utilisateurs, sommes victimes d'une « sourcellerie » (*sourcery*). Cette sourcellerie est en fait selon Chun la conséquence d'un aveuglement induit par le { logiciel }. Le { logiciel }, au sens d'un ensemble technique permettant la manipulation d'un ordinateur, est constitué d'une { interface } et d'instructions écrites dans un { langage } informatique. Un { logiciel } « [...] joint le mot à un résultat, le verbe à l'action[854] » : il efface, dans son { interface }, les

852 Souchier, Emmanuël et Pomian, Joanna. Informatique et pratiques écrivantes. *Op. cit.*, p. 121-130.

853 Chun, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Op. cit.*, p. 299-324.

854 « [...] *conflates word with result, logos with action* ». Chun, Wendy Hui Kyong. *Idem*, p. 303.

éventuelles dissociations entre le { code } et ce qui est affiché[855]. De telle sorte qu'en tant qu'utilisateur, on peut vite être conduit à penser que tout ce qui est affiché dépend d'un { code } préécrit. Le { code } est vu comme seul agent de ce que l'on voit : il est fétichisé comme source et posé, par exemple par Lev Manovich et les tenants des *software studies*, comme « l'essence même » des médias informatisés, essence dont la connaissance libère l'être humain, « l'illumine[856] ».

Cette fétichisation repose sur le fonctionnement scripturaire des médias informatisés. En effet le { code informatique } comme jonction (idéale) du verbe et de l'action est le résultat de l'efficacité machinique de l'écriture informatique, de son pouvoir de performativité, c'est-à-dire de faire fonctionner la machine. Si nous écrivons une instruction avec une syntaxe appropriée, la machine va l'exécuter. Elle va modifier son état interne en fonction des signes inscrits. Or, un { code } n'est jamais initialement performant : il ne l'est qu'une fois placé dans un environnement technique l'investissant de ce pouvoir. Il n'y a donc pas de { code source } pour Chun, mais bien plutôt des re-sources, des opérations par lesquelles un { code } est « sourcé ». Tout se passe comme si, parce que le { code } (sous certaines conditions) fait agir la machine, il déterminait selon les mêmes modalités la page que l'on lit. C'est en cela que l'ensourcellement est fondamentalement un fétichisme technique, car c'est considérer que le niveau technique du { code } (ensemble d'instructions données à la machine) se décalque tel quel au niveau sémiotique (celui des { interfaces }). Or, entre ce qui est vu et ce qui est programmé, il y a un écart significatif : déplacements, conversions, processus algorithmiques, recomposition de données… Toutes ces opérations sont la plupart du temps masquées ou ignorées, ce qui participe à la fétichisation du { code }.

b La sourcellerie comme processus, autour de deux imaginaires : la source et la commande

Nous aimerions désormais prolonger cette idée de la fétichisation du { code source }, ou sourcellerie, en approfondissant un point noté par Chun mais peu exploité : la sourcellerie est un processus. Il y a donc des étapes, des opérations par lesquelles un { code } est ensourcellé. Selon nous, certaines de ces opérations se traduisent à l'écran, c'est-à-dire qu'elles comportent un pan sémiotique.

855 Le *bug* ou le *glitch* sont de l'ordre de cette désynchronisation entre code et interface.

856 Manovich, Lev. *Op. cit.*, 2000, p. 48.

La sourcellerie envisagée comme processus comporte deux mouvements complémentaires mais distincts : la sourcification et l'ensourcellement.

La sourcification, c'est l'ensemble des opérations par lesquelles un document rédigé en { langage de programmation } devient un ensemble d'instructions données à la machine, c'est-à-dire se charge d'un pouvoir technique. Le texte devient archi-texte. La sourcification répond donc à la question : d'où vient l'*archè* de l'{ architexte } ?

architexte,
calcul,
code,
code source,
entité computationnelle,
formalisme,
HTML,
interface,
langage machine
langage de programmation,
logiciel,
machine computationnelle,
programme

Voir glossaire tome II, p. 3-25

L'ensourcellement, c'est l'ensemble des opérations par lesquelles le { code informatique } est construit comme unique source d'une page ou d'un { logiciel }. Il se charge donc d'un pouvoir essentiellement symbolique. L'ensourcellement répond à la question : pourquoi avons-nous tendance à considérer un document { HTML } comme l'unique source d'une page web, et plus largement le { code } d'un { programme } comme source de ce que nous lisons ?

Étant attaché à l'analyse des imaginaires de l'écriture informatique et de leurs effets politiques et idéologiques, nous nous concentrerons sur l'ensourcellement. La sourcification est le résultat de procédures techniques qui dépassent le cadre de notre thèse. En revanche, l'ensourcellement est un processus qui engage les représentations que nous nous faisons des { machines computationnelles } et de notre façon d'écrire pour elles.

Après avoir montré en quoi l'écriture est un mythe, et en quoi l'écriture informatique prolonge cette tendance, nous allons désormais approfondir cette idée de sourcellerie en formulant l'hypothèse que l'ensourcellement s'appuie principalement sur deux imaginaires du { code } : la source et la commande. Pour vérifier cette hypothèse, nous analyserons successivement la notion de « source » puis celle de « commande ». Nous chercherons par là à mettre en évidence ce qu'est l'économie scripturaire spécifique du { code informatique }, c'est-à-dire la façon dont ce type d'écriture, compte tenu de ses spécificités techniques et symboliques, organise nos rapports aux machines et aux { entités computationnelles }.

Si le { code } est le texte où peuvent se rencontrer humains et machines, ce texte n'est pas neutre en ceci qu'il structure les modalités de cette rencontre. Deux imaginaires – la source et la commande – participent de l'établissement d'une économie scripturaire qui fait de la machine la simple exécutante de la volonté d'un être humain, ne laissant aucune place à son activité propre, là où elle pourrait nous surprendre.

Afin de recréer une relation aux machines dans ce qu'elles ont de propre, et d'échapper à l'ensourcellement, il faut donc commencer par un examen attentif de l'imaginaire de la source. Cet imaginaire est tout d'abord le résultat d'une concaténation entre l'origine et le savoir (V. 3. A). Savoir d'où vient ce que nous voyons à l'écran, ce serait mieux comprendre les médias informatisés. Une telle perspective amène bien souvent à remonter les couches d'écriture, en pensant mettre à jour les rapports de pouvoir à l'œuvre.

Nous montrons que cette conception pose d'importants problèmes épistémologiques, et qu'il est alors tentant de faire valoir l'équivalence { formelle } des différentes couches d'écriture (V. 3. B). Mais là encore, c'est une impasse car impossible alors de comprendre les différences entre un { code source }, une { interface } grand public ou encore du { langage machine }, car tous ces textes représentent un { calcul } équivalent.

C'est finalement par la théorie de l'énonciation éditoriale[857] que nous pourrons déjouer l'imaginaire de la source. Le { code informatique } organise selon son image du texte les rapports entre humains et machines. Cette représentation n'apporte pas plus de connaissances qu'une page web par exemple, mais donne à lire différemment la place des machines dans l'énonciation en cours (V. 3. C).

857 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

L'origine est, en tout,

imaginaire

Valéry, Paul. *Cahiers. Tome 23, 1940***. Paris : Centre National de la Recherche Scientifique, 1960, p. 592.**

3 L'ÉCONOMIE SCRIPTURAIRE DE LA PROGRAMMATION (1/2) : L'IMAGINAIRE DE LA SOURCE

code, code source, programmation

Voir glossaire tome II, p. 3-25

Une fois montré que le { code informatique } est un texte organisant les rapports entre humains et machines – la { programmation } étant la pratique d'écriture produisant ces textes particuliers – il faut interroger l'économie scripturaire de la { programmation } : les imaginaires que cette pratique mobilise et la manière dont ils tissent nos relations aux machines. Pour cela, nous repartons de deux aspects fondamentaux de l'écriture informatique : c'est une écriture organisée en strates ; c'est une écriture faite d'instructions : ce qu'écrit un humain fait agir la machine. Ces deux propriétés nous renvoient à deux éléments structurants en { programmation } : le fait qu'on parle de « { code source } » et l'idée de « commande ». Dans cette sous-partie, nous nous concentrons sur la notion de { code source }. Quel est l'imaginaire de la source ?

V | 1 | B
organisée en strates
Voir p. 445

Notre hypothèse pour cette sous-partie est que la notion de source convoque un imaginaire puissant du pouvoir, et que cet imaginaire constitue par certains aspects un obstacle quand il s'agit d'analyser du { code source }. Comme le rappelle Almuth Grésillon quand elle présente l'intérêt d'une démarche génétique, gnose et génèse ont une même racine indo-européenne : « [...] la connaissance est toujours liée de près aux cheminements des origines[858] ». Savoir, c'est remonter vers l'origine. La connaissance des origines procurerait un savoir supplémentaire, et donc une forme de libération ou de maîtrise accrue. C'est cela que nous appelons l'imaginaire de la source, et nous en montrons les effets lorsqu'il qualifie du { code informatique }.

L'imaginaire de la source s'appuie sur le fait que les médias informatisés sont organisés en nombreuses strates d'écriture. La métaphore de la source permet d'organiser cette complexité par des rapports d'antériorité logique : le texte situé « avant » ou « en amont » est la condition de possibilité de celui affiché « en aval ». La source est donc une façon de lier une topologie (les couches successives) à une axiologie, c'est-à-dire un système de valeurs : la couche d'écriture la plus basse serait la plus puis-

858 Grésillon, Almuth. *Eléments de critique génétique : lire les manuscrits modernes*. Paris : Presses Universitaires de France, 1994, p. 3.

sante, la plus décisive pour comprendre ce qui se passe en haut, à l'écran. Ainsi, on pense pouvoir mieux expliquer les écrits d'écran en analysant le { code source }. Ce faisant, on prépare le terrain à un ensourcellement : on érige la source comme le lieu véritable du pouvoir. Aller plus « bas » donnerait accès à plus de pouvoir ou à une meilleure compréhension des médiations en cours. C'est ce que nous appelons la « tentation de l'origine » (V. 3. A.) et nous montrons en quoi elle conduit à un ensourcellement. Pour déjouer cette tentation, deux solutions s'offrent à nous. On peut considérer le { code } comme une représentation possible d'un { calcul } (V. 3. B), envisagé du point de vue de son idéalité computationnelle[859]. Cette approche a le mérite de déjouer la question de la source (qui n'est pas plus importante que d'autres représentations du { calcul }), mais pose une équivalence généralisée où toutes les représentations se valent. Si l'on veut comprendre le { code source } en son contexte, sans pour autant prétendre avoir accès à un savoir plus déterminant qu'en analysant d'autres couches d'écriture, il faut en dernière analyse réaffirmer, avec la théorie de l'énonciation éditoriale, qu'un { code informatique } est un texte hybride qui construit cette hybridité par son image du texte (V. 3. C). En faisant varier la mise en page d'un { code } { HTML }, on comprend que ce qu'on appelle habituellement le { code source } d'une page web n'est qu'une façon de représenter cette hybridité, qui puise dans certaines cultures professionnelles de l'écriture. Ainsi, la question n'est pas tant de savoir de quoi le { code } est-il la source, mais à quelle culture du texte il fait appel, et comment il organise les relations entre humains et machines. Un { code source } n'est pas un texte qui permet de « mieux » comprendre les pages web, il est simplement une façon de les réprésenter qui fait appel à des savoirs différents. Ce faisant, nous aurons donné un modèle d'analyse du { code informatique } qui désamorce les questions sous-jacentes du pouvoir bien souvent liées au terme de { code source }.

API,
architexte,
calcul,
code,
code source,
HTML,
interface,
JavaScript,
langage
de bas niveau
langage
de haut niveau
langage machine,
***middleware*,**
programmation

Voir glossaire
tome II, p. 3-25

859 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit.*

3 A La tentation de l'origine

En quoi l'expression de « { code source } » peut-elle poser un problème épistémologique ? Dans quelle mesure conduit-elle à un ensourcellement et en quoi cet ensourcellement est-il préjudiciable pour qui veut prendre le { code source } comme objet d'analyse en sémiotique ?

Le terme de { code source } est une façon d'organiser les différentes strates des écritures numériques, mais en présupposant que la couche la plus basse est la plus décisive, là où se jouent les enjeux de pouvoir (V. 3. A. a). Une première conséquence problématique de cette conception du { code } est la tentation de remonter toujours plus loin dans les différentes couches d'écriture (V. 3. A. b), tentation que l'on voit poindre par exemple dans la notion d'{ architexte } qui lie en un même geste intellectuel commande et origine. Mais alors où s'arrêter ? Quelle est la couche la plus décisive ? Cette remontée vers les origines peut amener à considérer que les { interfaces } ne sont que des leurres, et que seule importe la matérialité calculatoire de la machine (V. 3. A. c), représentée par le { langage machine } composé de suites de 0 et de 1, la couche ultime d'écriture. Une telle approche, prônée notamment par l'archéologue des médias Friedrich Kittler, nous empêche finalement de comprendre le poids culturel des textes numériques, en survalorisant leur ancrage dans la matérialité du { calcul }. Ainsi, nous aurons à la fois montré en quoi la notion de { code source } est – potentiellement – un facteur d'ensourcellement et quelles sont les conséquences épistémologiques de cet ensourcellement.

a « Haut » et « bas » niveau : de la topologie à l'axiologie des écritures

L'ensourcellement a rapport avec la métaphore de l'espace qui structure la { programmation } et permet d'organiser mentalement les multiples couches d'écriture. On parle de { langages de « haut niveau } », opposé à des { langages de « bas niveau } », les { API } sont des { *middleware*[860] } faisant le lien entre deux niveaux, etc.

Que veulent dire ces termes ? Les langages dits de « { haut niveau } » ({ JavaScript }, { HTML }, C++) sont des langages dont la syntaxe ressemble au langage humain. Ils sont donc dits être plus « près » de l'humain. Les { langages de « bas niveau } » sont des langages dont la syntaxe est

860 Volonino, Linda et Dalal, Pragati. Middleware. Dans : Bidgoli, Hossein (dir.), *op. cit.*, p. 33-44.

application,
architexte,
assembleur,
code,
code source,
interface graphique,
langage de haut niveau,
langage de programmation,
langage machine,
programme

Voir glossaire tome II, p. 3-25

proche du { langage machine } (les suites de 0 et de 1, instructions propres à chaque modèle). Un { langage de haut niveau } traduit les opérations « basses » en langage intelligible pour la majorité des humains. Ces termes sont métaphoriques car matériellement, un { programme } écrit en { assembleur } n'est pas exécuté dans une partie de la machine plus « proche » du processeur[861] qu'un { programme } écrit en Java par exemple. Cette métaphore sert surtout à penser les rapports entre l'humain et l'objet technique qu'il manie, en prenant en compte la nature écrite de ce dernier. L'ensemble des couches d'écriture se répartit dans une topologie qui va de la machine, le plus « bas », à l'{ interface graphique } (au plus « haut »), c'est-à-dire ce qui fait le lien visuel et cognitif entre l'utilisateur et la machine. Cette topologie prend acte du gradient relevé par ailleurs : un { code informatique } organise les relations entre des humains et des machines. Le choix du { langage } et de son niveau est une première façon de se situer entre ces deux pôles.

V 2 A c
gradient relevé par ailleurs
Voir p. 464

Or cette topologie est tout sauf neutre si l'on considère que le { code } est la « source » d'une page web, ou qu'un document en { assembleur } est la « source » d'une { application }. Ici la spatialisation se double de rapports de pouvoir et de valorisation tant symbolique qu'économique. L'espace est axiologisé : le « bas » est ce qui serait le plus décisif, là où le pouvoir se décide, là où les décisions de l'utilisateur, du « haut », sont contraintes, ordonnées, préécrites. Parler de { code source }, c'est prolonger cette version des écrits d'écran où c'est le « bas » qui commande, et que pour analyser les rapports de force à l'œuvre, il faudrait « descendre » dans et vers la machine. Cette axiologisation de l'espace participe du phénomène d'ensourcellement puisqu'elle attribue au { code } dit source une grande puissance tant technique que symbolique. Cela entraîne un double risque : celui d'une remontée à l'infini dans les couches d'écriture ; celui de négliger la couche sémiotique des textes numériques (l'{ interface graphique }) au profit du { langage } le plus « machinique » et par conséquent le plus « pur ».

861 Si l'on considère que le processeur est le « cœur » de la machine, ce qui là aussi est discutable.

b L'*archè* dans l'architexte : le risque d'une remontée à l'infini

Il nous semble que la grande majorité des travaux autour de la théorie des écrits d'écran repose sur cette identification entre topologie et axiologie, et ce en raison de la définition même du concept d'{ architexte } :

> « *Nous nommons architextes (de archè,* ***origine et commandement****), les outils qui permettent l'existence de l'écrit à l'écran et qui, non contents de représenter la structure du texte, en commandent l'exécution et la réalisation. Autrement dit, le texte naît de l'architexte qui en balise l'écriture.*[862] »

La définition *d'archè* donnée par Souchier et Jeanneret montre bien l'attendu structurant de cette théorie : remonter dans la chaîne de l'écrit permettrait de mieux cerner les enjeux politiques et idéologiques de la fabrique du texte. En cela, c'est une définition « ensourcellée » (le pouvoir se tient en amont) qui ne tient cependant qu'à deux conditions : qu'il existe des textes « en amont » ou « plus haut » ou « à la source de… » ; que ce qui soit « plus haut » soit également plus « puissant » ou « agissant » que ce qui est en « bas ».

Le risque de cette conception de l'organisation des écritures informatiques est qu'elle conduit à une remontée à l'infini : on peut toujours remonter d'un cran dans la chaîne des écritures, chaque { architexte } étant lui-même le produit d'un autre { architexte }. Nous sommes vite conduit à une tâche inépuisable, et qui pose la question du texte que nous considérons comme source. Y'a t-il un point d'arrêt dans la remontée vers l'{ architexte } ? Et pourquoi ce niveau là et pas un autre ? Doit-on remonter jusqu'au { langage machine } ?

862 Jeanneret, Yves et Souchier, Emmanuël. Pour une poétique de l'écrit d'écran. *Op. cit.*, p. 103. Nous soulignons.

c « Le logiciel n'existe pas ». Un matérialisme radical, exemple d'ensourcellement

assembleur, binaire, BIOS, code, formalisme, industrialisation, interface, interface graphique, langage de haut niveau, langage de programmation, langage machine, logiciel, système d'exploitation

Voir glossaire tome II, p. 3-25

Cette fascination pour la « source » que l'on espèrerait trouver dans le { code informatique } a néanmoins l'avantage de faire porter le doute sur l'{ interface } et ce à quoi elle nous donne accès. Si le pouvoir se décide en amont, pour analyser les enjeux de pouvoir autour du { logiciel }, il faudrait remonter à la source. C'est une rhétorique particulièrement puissante dans le cas d'une lecture critique des « écrits d'écran » et qui peut s'élargir à l'industrie du { logiciel }, comme le fait Friedrich Kittler (1943-2011). Quand il propose la théorie radicale que « le logiciel n'existe pas », il le fait au nom d'une critique radicale des { interfaces graphiques } ou GUI (*Graphic User Interface*) qui introduisent selon lui une rupture dans l'histoire de l'informatique puisqu'elles masquent le fonctionnement de la machine et trompent l'utilisateur[863]. Contre l'{ interface }, Kittler remonte jusqu'au { système d'exploitation } et même jusqu'au { BIOS }, instructions élémentaires de la machine (démarrer, initier le { système d'exploitation }) gravées à même le silicone. C'est pour cette raison que « le logiciel n'existe pas », puisque toutes les opérations de l'utilisateur peuvent être ramenées à de la manipulation { formelle } et combinatoire engravée dans des puces de silicone. C'est ce niveau qui est véritablement décisif, le { logiciel } n'étant plus alors qu'une « [...] pure chaîne infinie d'autosimilarités[864] » : l'icône est traduite en { langage de haut niveau }, lui même traduit en { assembleur }, lui-même traduit en { langage machine }, lui-même traduit dans des transistors guidant un courant électrique. Si tout { logiciel } peut être ramené à du *hardware*, et si le { logiciel } finit par masquer les opérations concrètes de la machine, alors le { logiciel } n'est plus qu'une baudruche, un « [...] commerce de milliards de dollars[865] » reposant sur une certaine configuration de la matière.

Kittler prend en quelque sorte acte de l'équivalence { formelle } des opérations de la machine, équivalence dont nous avons vu qu'elle remonte aux travaux de Hilbert en logique { formelle }. Ce qui est fait au niveau de l'{ interface } peut être ramené à une expression en { langage binaire } et cette expression est bien **équivalente** à celle que nous donne l'{ inter-

II 3 B c
travaux de Hilbert en logique formelle
Voir p. 231

863 Kittler fait partie de ces auteurs, avec Neal Stephenson ou Brenda Laurel, qui écrivent à la fin des années 1980 et début des années 1990 au moment où les interfaces graphiques se généralisent. Progressivement, l'icône remplace la ligne de commande, on ne tape plus les instructions au clavier mais on double-clique sur une image. Leurs conceptions de l'écriture, de la programmation et du rapport à la machine sont profondément marquées par ce passage à un régime généralisé de la métaphore imposé par l'industrie informatique.

864 Kittler, Friedrich. *Op. cit.*, 2015, p. 33.

865 Kittler, Friedrich. *Idem*, p. 42.

face }. Ce qui déplace le problème vers la question de la **représentation :** comment représenter une opération de la meilleure manière ? Kittler part du principe que la représentation donnée par un { logiciel } est un leurre. Pourquoi une telle disqualification ? Au nom d'une méfiance très forte vis-à-vis des développements industriels du { logiciel }, ainsi que d'une valorisation de la matérialité de la machine, et donc des { langages } qui en seraient les plus proches, ultimement le { langage binaire } et peut-être même au-delà[866]. C'est en ce sens qu'on peut dire que Kittler est ensourcelé : sa critique de l'{ interface graphique } se fait au nom du fonctionnement technique du médium, sans prendre en compte l'efficacité sociale et communicationnelle de la métaphore et du masquage des opérations machiniques. Pour le dire très simplement : Kittler croit que coder ou lire en { langage machine }, c'est « mieux » comprendre les médias informatisés et c'est se libérer des illusions de l'industrie du { logiciel }. C'est vrai si l'on veut comprendre comment le courant circule dans la machine et comment ce courant est transformé en instructions logiques et en données stockées sur des supports magnétiques. C'est faux si l'on veut comprendre comment les médias informatisés équipent et assistent un mouvement d'{ industrialisation } des écritures. Ou plus précisément : dire que les opérations que nous faisons sur un traitement de texte comme Word peuvent être faites au moyen de lignes de commande et non d'une { interface graphique } n'explique pas le succès du { logiciel }, ni son emprise sur le marché du traitement de texte. La compréhension de l'opérativité technique des médias informatisés se fait aux dépens de la compréhension de leur opérativité symbolique.

866 Dans son article « Mode protégé », Kittler va encore plus loin et pense qu'il est possible de se débarrasser complètement des métaphores sur médias informatisés pour arriver à un niveau a-représentatif, de « pure » manipulation matérielle. On mesure ici l'écart avec la théorie des écrits d'écran qui part d'un constat similaire mais, si critique qu'elle soit à l'égard des interfaces graphiques, n'en nie pas l'efficacité sociale. Elle essaye plutôt d'en prendre acte et d'en évaluer les effets. Voir Kittler, Friedrich. *Idem*, p. 53-69.

3 B. Déjouer l'axiologie par l'équivalence formelle ?

Comment alors maintenir une position critique sur les médias informatisés sans tomber dans une fétichisation de la source ? Comment déjouer l'axiologie de l'espace inhérente à la métaphore de la source ? La solution se trouve peut être dans l'équivalence { formelle } des { calculs }, cette propriété spécifique de l'informatique, qui permet de :

> « [...] *manipuler les symboles mathématiques de manière purement formelle, c'est-à-dire uniquement en fonction de leur forme syntaxique et indépendamment de leur signification sémantique*[867] ».

architexte,
calcul,
code,
code source,
commentaire,
discret,
formalisme,
HTML,
interface graphique,
langage machine,
objet,
programme

Voir glossaire tome II, p. 3-25

Le { formalisme } permet de manipuler des énoncés d'un point de vue { formel }, en ne considérant plus leur sens mais leur structure. Cette structure peut être ramenée à un { calcul } { discret }, c'est-à-dire ouvert à la manipulation. Partant, peu importe la réalisation concrète d'un { objet } informatique : ce qui compte est sa structure { formelle }. C'est ce que Bruno Bachimont nomme « l'idéalité computationnelle[868] » du numérique, soit l'indépendance de l'information par rapport à son substrat matériel. Bien évidemment, la réalisation concrète importe, car elle limite les possibilités de { calcul }, mais le numérique comme technique d'inscription est porteur d'un idéal d'équivalence { formelle } généralisée de tous les contenus calculables, c'est-à-dire discrétisables et manipulables par le { calcul[869] }.

II | 3 | C | b
idéal d'équivalence formelle généralisée
Voir p. 236

La force de cette théorie est, à l'instar de celle de Kittler, de prendre en compte de très près la structure technique de ces médias. En posant l'équivalence { formelle } des contenus sous la forme de l'information, on peut déjouer la valorisation du « bas » comme plus décisif. Du point de vue{ formel }, un profil Facebook représenté comme { interface graphique }, base de données ou { langage machine } est le même objet.

867 Bachimont, Bruno. *Arts et sciences du numérique. Ingénierie des connaissances et critique de la raison computationnelle*. Habilitation à diriger des recherches. Compiègne : Université de Technologie de Compiègne, 2004, p. 95.

868 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit*.

869 Pour les liens entre cette propriété du calcul informatique et l'idéal scientifique d'une méthode universelle, voir II. 3. C. b.

C'est là toute la différence avec Friedrich Kittler : tous deux proclament l'équivalence { formelle } des objets numériques. En revanche, il n'y a pas chez Bachimont de supériorité de l'un des modes de représentation sur l'autre, au contraire de Kittler. L'{ interface graphique } ne masque pas les opérations de la machine, mais les reformule en des termes équivalents[870].

Sommes-nous pour autant libéré de l'ensourcellement ? Il devient tentant, une fois postulé cette équivalence idéelle, d'affirmer que ce qui est en bas est comme ce qui est en haut et vice-versa. Le { code } { HTML } ne serait alors que la « version informatique » d'une page web. C'est vrai, si l'on s'en tient au niveau { formel }. Mais c'est faux si l'on envisage le { programme } du point de vue de son image du texte et de sa vie sociale. Une page web, affichée sous sa forme { HTML }, fait appel à d'autres cultures du texte et à d'autres savoirs professionnels. Ils ne peuvent donc être posés comme équivalents. C'est d'autant moins possible que cela reviendrait à évacuer tout le travail éditorial propre à l'{ architexte } en tant qu'acteur technique de mise en visibilité du texte. Qu'est-ce qui, dans le { code source }, est affiché à l'écran ? Qu'est-ce qui n'est pas retenu ? Qu'est-ce qui, dans l'exécution du { code }, est caché ou montré[871] ?

Nous tombons dans l'excès inverse de la théorie des écrits d'écran : les médias informatisés ne sont plus saisis en leur contexte spécifique, en situation. Le { code }, ou un { programme }, est envisagé au regard de l'idéalité computationnelle, comme l'une de ses incarnations, sans que ne soit prise en compte sa dimension communicationnelle, sa circulation et sa vie sociale : le fait qu'il soit écrit dans tel ou tel { langage }, qu'il comporte des { commentaires }, qu'il s'insère dans un système économique précis, etc. En d'autres termes, nous ne nous donnons pas les moyens de comprendre « [...] les différences entre les différentes instanciations du code[872] ». Les différentes représentations d'un { calcul } sont ramenées à des équivalents, alors qu'elles sont différentes, ne serait-ce que par leur image du texte.

870 On retrouve dans la définition du code d'Alexander Galloway une position similaire : les contenus calculés sont sur un certain plan équivalent, et le langage machine n'est pas plus « pur » qu'un langage de haut niveau. Galloway, Alexander R. *Protocol: How Control Exists after Decentralization*. Cambridge : MIT Press, 2004, p. 167.

871 Sur ce point, la théorie de Kittler est essentielle, au même titre que l'analyse critique des architextes et de leur pouvoir.

872 « [...] *does not capture the difference between the different instanciations of code* ». Chun, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Op. cit.*, p. 306. Nous reprenons une critique qu'elle adresse à Alexander Galloway.

Nous n'avons finalement pas réussi à échapper à cette axiologisation de l'espace, qui fait du « bas » un niveau plus décisif que le « haut ». L'archéologue des médias qu'est Kittler semble chercher, tel le docteur Livingstone, la source de nos { interfaces } dans un âge d'or de l'informatique d'avant la représentation graphique, au plus près des puces de silicone. La théorie du support permet d'éviter cet écueil, mais ne permet pas de penser la spécificité de chaque couche d'écriture, annulant potentiellement la spécificité de chacun de ces textes et le rôle qu'y jouent les { entités computationnelles }. Comment, dans ces conditions, analyser du { code informatique } en comprenant ce texte en son contexte, autant technique (comprendre les actions qu'il déclenche) que culturel (la façon dont il fait sens pour des êtres humains) ? Comment échapper, autant que possible, à l'ensourcellement sans postuler l'équivalence généralisée des { codes informatiques } au nom d'une conception abstraite du { calcul } ? Une solution serait de ne plus se préoccuper de la question de l'efficacité technique de ces textes (de quoi sont-il la source ?) et de mettre en suspens la notion de { calcul } pour revenir à leur statut de textes. Ce sont des écrits qui ont vocation à être lus par des humains et des machines, et leur organisation visuelle traduit cette fonction hybride. Chaque { langage } est une façon différente d'organiser ces relations, et cette façon peut se lire par une image du texte, et donc par des méthodes sémiotiques. En considérant « l'énonciation éditoriale[873] » du { code informatique }, on pourrait donc déjouer les imaginaires de la source et du pouvoir, pour pouvoir comprendre ces textes en leur contexte sans sacrifier pour autant leur efficacité technique.

bug, calcul, code, code source, entité computationnelle interface, langage de programmation

Voir glossaire tome II, p. 3-25

873 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

3 C L'énonciation éditoriale, contre les imaginaires de la source

En quoi l'énonciation éditoriale peut-elle nous permettre d'échapper à l'ensourcellement sans pour autant postuler une équivalence formelle qui ne nous permet pas de saisir les { codes sources } en contexte ? La théorie de l'énonciation éditoriale repose sur l'idée que tout texte doit être vu pour être lu. Par cette « livrée graphique[874] », le texte est perçu et lu par un groupe social partageant la même culture visuelle, c'est-à-dire les mêmes repères quant à la distribution spatiale du texte. Cette livrée graphique participe du sens du texte, nous entendons par là sa circulation dans l'espace social. Par l'image du texte, on construit des horizons de réception, on organise des lectures possibles. Il y a donc une énonciation, en ceci que quelque chose est dit du texte par sa seule image.

Si tout texte construit par son image des horizons de lecture, alors il en va de même du { code source }. Or, un { code source } est un texte hybride. Il organise les rapports entre des machines et des humains. Il l'organise par sa forme : sa mise en page, la manière dont les actions des uns et des autres sont exprimées… En d'autres termes, loin d'être équivalentes, les différentes représentations d'une page web pourraient être autant de façons d'organiser les relations entre machines et humains, en faisant varier le curseur d'un côté ou de l'autre. En ce sens, ce seraient non pas des « équivalents » mais des textes différents, qui ne font pas appel à la même culture technique, ni à la même culture de lecture. En étudiant les différentes mises en page d'un bout de { code informatique }, on peut donc étudier comment la pluralité des attitudes vis-à-vis du { code } se traduit sémiotiquement. Nous montrons notamment que la mise en page, à travers un traitement de texte adapté, permet d'annuler temporairement l'efficacité machinique du { code } pour, paradoxalement, mieux la préparer : l'organisation spatiale du { code } permet de repérer clairement les étapes logiques, les { *bugs* }, etc (V. 3. C. a). L'analyse de l'indentation du { code } permet quant à elle de montrer comment l'organisation de l'espace a une fonction de spatialisation du { calcul } (V. 3. C. b). Cette organisation est donc une façon de rendre lisible pour les humains des opérations qui ont lieu à des échelles qui ne nous sont pas accessibles.

V | 2 | A | b
texte hybride
Voir p. 463

V | 2 | A | c
pluralité des attitudes
Voir p. 464

874 Souchier, Emmanuël. *Idem*, p. 138.

code,
code source,
script

Voir glossaire tome II, p. 3-25

Cela nous amène à mettre à distance la notion de { code source }, en montrant comment la mise en page prépare la sourcification du { code } et qu'un { code } n'est donc jamais source en soi, mais qu'il est institué comme tel (V. 3. C. c).

a De la mise en page du code comme préparation à son efficacité technique

Pour étudier la mise en page du { code informatique }, nous avons choisi le { script[875] } « almond.js », dont une partie est exécutée par le navigateur quand on consulte le site http://twitter.com. Ce { script } a pour utilité globale d'accélérer le chargement de la page d'accueil de Twitter. Pour ce faire, il automatise le chargement asynchrone[876] de certaines parties de la page, appelées « modules ». Almond.js dépend par ailleurs d'un autre { script }, « requireJS », optimisant lui aussi la gestion des modules de la page. On note d'emblée que cette couche d'écriture se situe dans une perspective de performance technique (améliorer la vitesse de chargement de http://twitter.com) tendant vers l'optimisation d'une activité sémiotique : lire confortablement la page d'accueil de Twitter sans devoir attendre. Cette performance ne peut se réaliser que dans le cadre d'une autre écriture (requireJS) et repose sur une conception modulaire de la page analysée par ailleurs.

III **1**
une conception modulaire de la page
Voir p. 250

875 Un script, dans le contexte du Web, est un fichier permettant de programmer et d'automatiser certaines actions, rendant la page « dynamique », c'est-à-dire que certains contenus peuvent être affichés selon le comportement des internautes.

876 Le chargement asynchrone est une méthode pour améliorer le temps de chargement d'une page Internet. Plutôt que d'attendre que tous les éléments de la page soient téléchargés pour être éxécutés par le navigateur, la page est affichée pendant que certaines de ses parties, par exemple des scripts, sont en cours de téléchargement. Il y a donc deux tâches (éxécution du code HTML; téléchargement d'un script) exécutées en parallèle.

Le { script }, tel que récupéré via l'extension Firebug[877] et ouvert dans le navigateur Firefox, s'affiche ainsi :

Fig. 56. Un exemple de code « au kilomètre » : illisible mais effectif. Capture du Script almond.js, version 0.2.5, © The Dojo Foundation (détails). Capture réalisée le 22 avril 2015.

En l'état, sans mise en page, il est techniquement effectif. Le navigateur le lit et l'exécute[878]. Il fonctionne donc au niveau technique et techno-sémiotique, puisqu'il informe notre lecture du site twitter.com. En revanche, il est très difficilement lisible par un humain. Pour qu'il le soit, il faut appliquer quelques règles de mise en forme, certaines automatisées, d'autres pas.

V 2 A c
au niveau technique et techno-sémiotique
Voir p. 464

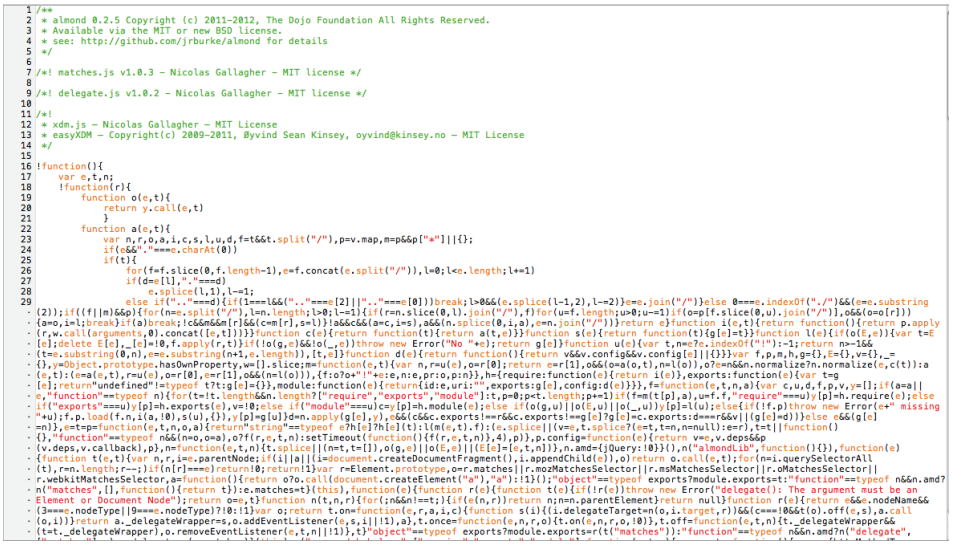


Fig. 57. La mise en abyme de l'architexte : une « désourcification ». Script almond.js, version 0.2.5, © The Dojo Foundation (détails). Capture réalisée le 22 avril 2015. Les premières indentations sont de notre fait.

877 Firebug est une extension pour le navigateur Firefox qui permet d'afficher dans la même fenêtre le code source d'une page web, non pas d'un seul bloc mais en se concentrant sur un élément de la page. On peut ainsi comparer le code source de la page et ce qui s'affiche dans le navigateur lorsqu'on charge la page.

878 Ce n'est pas vrai pour tout les langages informatiques. Le PASCAL par exemple n'est ni compilé, ni exécuté si une certaine mise en page n'est pas respectée.

architexte, carte perforée, code, commentaire, développeur, fonction, format, JavaScript, logiciel, programme, script, variable

Voir glossaire tome II, p. 3-25

La figure 57 est le résultat d'un copier-coller du { script } dans le { logiciel } Fraise, un éditeur de texte destiné à l'écriture de { code informatique }. Nous avons seulement précisé le { format } du document (.js pour { JavaScript }), ce qui a entrainé cette coloration du texte. Opération de conversion qui entraîne déjà une conséquence forte : nous avons un { architexte }, le { script }, représenté dans un autre { architexte }, l'éditeur de texte. Par cette manipulation, almond.js est coupé de son efficacité technique[879]. Il n'est plus un { architexte } mais un texte, comme n'importe quel document Word. L'{ architexte }, ici, c'est le { logiciel } Fraise, et ce { logiciel } automatise certaines mises en forme, avant toute intervention de l'utilisateur :

– ajout du décompte des lignes sur le côté gauche ;

– coloration du texte en vert, orange et rouge[880].

Cette mise en forme et en espace (décompte de la ligne) est faite en fonction d'attendus socioprofessionnels. La ligne, depuis l'utilisation des { cartes perforées[881] }, est l'unité de base de mesure du { code informatique } et a des fonctions importantes. Lorsque le { code } est « débuggué » (l'éditeur de texte vérifie la validité informatique du { code } et signale les erreurs de syntaxe au { développeur }), les erreurs sont indiquées par leur numéro de ligne. Les *Source Lines of Code* (SLOC), ou « lignes de code source », sont une métrique utilisée pour évaluer le travail à fournir lors du développement d'un { programme }, voire pour rémunérer les programmeurs. La ligne est donc dans l'écriture du { code } un mètre étalon à la fois statistique, économique et cognitif, puisqu'elle est un principe d'orientation dans le texte. Elle est un héritage historique, une organisation visuelle de la page et un outil de gestion et de régulation de cette pratique d'écriture en tant qu'elle est liée à un système économique.

879 On pourrait dire que nous avons « désourcifié » le code source. En changeant de logiciel, l'efficacité technique du code est annulée, même si nous n'en avons pas modifié une ligne. Nous l'avons simplement porté dans un autre contexte technique que celui de son exécution.

880 En postulant que la couleur « naturelle » du texte est le noir bien évidemment.

881 Williams, Michael R. *Op. cit.*, p. 252.

La coloration du texte permet quant à elle de visualiser d'un seul coup d'œil ce qui est de l'ordre du { commentaire } (en vert), ce qui est de l'ordre des éléments logiques du texte (en orange) et ce qui est de l'ordre de la chaîne de caractères (*strings*, en rouge[882]). Elle permet d'organiser le texte par ses lecteurs ({ commentaires } pour les humains ; unités logiques et { format } des { variables } pour la machine) tout en préparant sa rédaction. Car si les liens logiques (`if ; else` ; { fonctions }, etc.) sont mis en valeur visuellement, c'est bien parce qu'ils sont les articulations techniques du texte : ils indiquent à la machine quoi faire. Cette mise en page oriente donc le { développeur } vers l'efficacité machinique du texte.

L'analyse ne suffit toutefois pas. Car en l'état le { script } n'a fait l'objet que de modifications automatisées et reste difficilement lisible à l'œil humain. Pour qu'il le soit, il faut l'organiser spatialement à travers une indentation propre à l'écriture du { code }.

882 En informatique, une variable (unité logique mémorisée par la machine) est de type *string* quand sa valeur, autrement dit son contenu, est textuel. Une variable booléenne aura pour valeur *true* ou *false*, suivant les règles de la logique booléenne.

b L'indentation : une spatialisation du calcul

```
 */

/*! matches.js v1.0.3 - Nicolas Gallagher - MIT license */

/*! delegate.js v1.0.2 - Nicolas Gallagher - MIT license */

/*!
 * xdm.js - Nicolas Gallagher - MIT License
 * easyXDM - Copyright(c) 2009-2011, Øyvind Sean Kinsey, oyvind@kinsey.no - MIT License
 */

!function(){
    var e,t,n;

    !function(r){
        function o(e,t){
            return y.call(e,t)
        }

        function a(e,t){
            var n,r,o,a,i,c,s,l,u,d,f=t&&t.split("/"),p=v.map,m=p&&p["*"]||{};
            if(e&&"."===e.charAt(0))
                if(t){
                    for(f=f.slice(0,f.length-1),e=f.concat(e.split("/")),l=0;l<e.length;l+=1)
                        if(d=e[l],"."===d)e.splice(l,1),l-=1;
                        else if(".."===d){
                            if(1===l&&(".."===e[2]||".."===e[0]))
                            break;
                            l>0&&(e.splice(l-1,2),l-=2)
                        }
                    e=e.join("/")
                    }

            else 0===e.indexOf("./")&&(e=e.substring(2));

            if((f||m)&&p){
                for(n=e.split("/"),l=n.length;l>0;l-=1){
                    if(r=n.slice(0,l).join("/"),f)
                        for(u=f.length;u>0;u-=1)
                        if(o=p[f.slice(0,u).join("/")],o&&(o=o[r])){
                        a=o,i=l;
                        break;
                        }
                    if(a)
                    break;
                    !c&&m&&m[r]&&(c=m[r],s=l)
                }
            !a&&c&&(a=c,i=s),a&&(n.splice(0,i,a),e=n.join("/"))
            }

            return e
        }
```

Fig. 58. L'indentation du code comme spatialisation du calcul. Script almond.js, version 0.2.5, © The Dojo Foundation (Détails). Capture réalisée le 22 avril 2015. Les indentations sont de notre fait.

code, fonction, JavaScript, langage de programmation, programme

Voir glossaire tome II, p. 3-25

C'est le même texte qui est présenté ici. Nous l'avons seulement réorganisé en distinguant les blocs de { fonctions }. Une { fonction } est une procédure algorithmique à l'intérieur d'un { langage de programmation }, procédure qui se déclare par un nom (`function`) puis entre parenthèses les paramètres utilisés par la { fonction }. Ensuite entre accolades (`{ }`) on indique en quoi consiste la procédure à suivre, d'où la présence des marqueurs logiques `if`, `else` ou `break`.

Les { fonctions } sont des éléments centraux des { codes informatiques }, car c'est en quelque sorte la manière dont se déroule le { programme }. Par exemple :

```
function test (){
      for (var temoin=0; temoin<4; temoin++){
            if (temoin===0){
                  console.log("Ceci");
            }
            if (temoin===1){
                  console.log('est');
            }
            if (temoin===2){
                  console.log(«un»);
            }
            if (temoin===3){
                  console.log("test");
            }
            else (console.log(" "));
      }
}

test();
```

donne une fois exécuté sur une console { JavaScript } du site codeacademy.com :

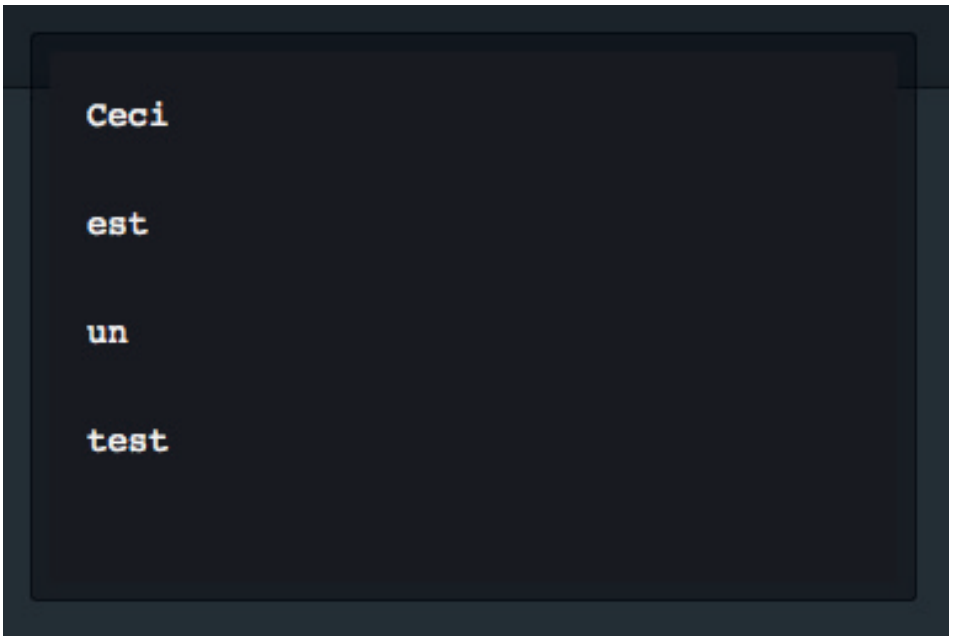


Fig. 59. Exemple d'une fonction écrite en JavaScript. Capture de la console de programmation du site codeacademy.com (détails). Capture réalisée le 28 avril 2015.

L'exemple ci-dessus est très simple d'un point de vue programmatique mais permet de tirer quelques enseignements sur l'importance de la structuration graphique du { code }. Nous avons commencé par déclarer l'existence d'une { fonction } appelée « test », ensuite décrit son organisation interne (premières accolades) et enfin demandé à l'ordinateur d'exécuter[883] la { fonction } (`test ();`).

boucle,
calcul,
code,
code source,
discret,
fonction,
machine computationnelle,
programme,
script,
variable

Voir glossaire tome II, p. 3-25

Cette { fonction } contient une { boucle } `for()`. À l'intérieur de cette { boucle } sont implémentés quatre opérations conditionnelles `if` et une autre `else`. La { boucle } fonctionne ainsi : entre parenthèses, nous avons une condition de départ, une condition d'arrêt et une action à mener pour chaque action qui permet d'arriver à la situation d'arrêt. Ici la situation de départ est une { variable } témoin, ainsi nommée car elle va nous servir de témoin, vaut 0. La { fonction }, en langage dit naturel, se lit donc comme un simple décompte :

Langage naturel	**JavaScript**
pour (`for`) un témoin valant 0…	`for(var temoin=0;`
…augmente la valeur du témoin de 1…	`temoin++)`
…jusqu'à tant que témoin vaille moins que 4	`temoin<4;`
À chaque fois que tu incrémentes témoin de 1, fais les opérations suivantes (second niveau d'accolades)	`{`
si (`if`) témoin est égal à 0…	`if (temoin===0){`
…affiche dans la console…	`console.log`
….le texte indiqué entre parenthèses	`("Ceci");`
Sinon…	`else`
…affiche dans la console un espace vide	`console.log ("");`

La même opération recommence pour témoin valant 1, 2 et 3, avec des textes différents.

883 En informatique, il faut déclarer une fonction pour que la machine mémorise les instructions que contient la fonction. Mais pour que cette suite d'instructions soient effectives, c'est-à-dire calculées, il faut dire à la machine d'exécuter la fonction.

À quoi sert l'indentation, principale modification éditoriale que nous avons faite sur le { script } almond.js ? Premièrement, elle facilite la lisibilité du { code } pour l'être humain. Lorsque commence une nouvelle { fonction }, juste après l'accolade ouvrante « { », il est d'usage de sauter une ligne et d'indenter le texte. L'accolade fermante sera placée au même niveau d'indentation que la ligne ou a été déclarée la { fonction }. Cela permet de repérer d'un seul coup d'œil les différents blocs du texte et la manière dont ils sont articulés[884]. Rendre lisible le { code } pour un être humain, c'est donc aussi l'organiser spatialement, cette organisation dans l'espace reflétant son organisation logique.

Deuxièmement, l'indentation permet la représentation de l'action calculatoire et processuelle de la machine, c'est-à-dire de sa temporalité[885]. Une { machine computationnelle } calcule en suivant une liste d'instructions, ligne à ligne, opération par opération. Elle saute d'états { discrets } en état { discrets }, mais à une vitesse et à une échelle que nos sens ne peuvent percevoir. Son { calcul } est essentiellement un déroulé temporel, plus que spatial, ce que confirme d'ailleurs Turing quand il dit que le { calcul } n'a idéalement pas besoin de la bi-dimensionnalité du papier[886]. Un { calcul }, pour une machine, c'est un pur déroulé temporel : une suite d'actions { discrètes }, avec une situation de départ et une situation d'arrivée. L'indentation permet de redonner à cette pure temporalité une spatialité compréhensible par l'être humain. En indentant, on distribue les différentes étapes du { calcul } dans l'espace d'une page, on le rend lisible. De ce point de vue, un { programme } sous la forme de son { code source } est une spatialisation du { calcul[887] }. C'est-à-dire une représentation spatiale, adaptée au mode de pensée humain, d'un processus temporel calculatoire propre à la machine.

884 En effet, il arrive souvent qu'une fonction soit insérée à l'intérieur d'une même fonction. Cette structure de récursion complexifie énormément la lecture du code si les indentations ne sont pas proprement éxécutées.

885 Bachimont, Bruno. *Op. cit.*, 2010, p. 144. Sa reprise des schèmes kantiens de la sensibilité l'amène à distinguer le calcul de l'écriture, en posant l'écriture comme technologie de l'espace, dans les traces de Jack Goody, alors que le calcul est une technologie du temps, d'où sa distinction entre raison graphique et raison computationnelle. L'indentation est donc dans ce cadre d'analyse une représentation spatiale, adaptée au mode de pensée humain, d'un processus temporel calculatoire propre à la machine.

886 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 245. Voir aussi partie I. 3. C. d.

887 Bachimont, Bruno. *Op. cit.*, 2010, p. 144.

c Conclusion : la sourcification du code

Décompte des lignes, coloration des articulations du texte et indentation sont trois opérations, certaines automatisées d'autres pas, qui construisent la lisibilité humaine du { code informatique }. La ligne permet de repérer facilement les erreurs de syntaxe ; la coloration permet de distinguer le { commentaire } (donc ignoré par la machine) de la { variable } ou de la { fonction } ; l'indentation permet de mieux comprendre le fonctionnement du { script }. Par ces opérations, Fraise et le { développeur } préparent le changement de statut du texte produit. La mise en page prépare le texte à devenir { architexte }, prépare le { code } à devenir « source » de quelque chose, en l'occurrence d'une page web.

architexte, calcul, code, commentaire, développeur, fonction, formalisme, machine computationnelle, programme, script, variable

Voir glossaire tome II, p. 3-25

Ce qui confirme donc que :

a) le { code } n'est jamais « source » en soi mais est institué comme source par différentes opérations ;

b) que les différentes mises en pages d'un { code } ne sont pas différentes représentations d'un même { calcul }, mais que ce sont des textes différents : ils n'organisent pas de la même manière la relation entre humains et machines.

Devant la tentation de l'origine, métaphore bien pratique pour organiser les différentes couches d'écritures des { machines computationnelles }, l'énonciation éditoriale permet d'éviter le piège de la source, en faisant considérer la différence radicale des différents { codes informatiques }. Nous avons ainsi mis à distance le vocabulaire de la source et des imaginaires afférents du pouvoir, sans pour autant s'en remettre à l'équivalence { formelle } du { calcul }. Cela nous a amené à voir comment la mise en page du { code } est effectivement une manière de construire une relation aux machines, en préparant l'efficacité du { code } et en rendant visible aux yeux humains des opérations qui sont menées à une autre échelle d'appréhension. Reste maintenant à examiner l'imaginaire de la commande, autre façon potentiellement problématique d'aborder nos relations aux machines.

L'imaginaire de la source analysé, nous passons désormais à celui de la commande. L'écriture informatique a en effet ceci de particulier qu'elle permet de faire agir une machine. À partir de cette spécificité technique, nous faisons l'hypothèse que l'économie scripturaire des médias informatisés se constitue autour d'une idéologie de la commande qui laisse penser que ce qu'on lit dans un { programme } informatique, ce n'est que la consignation de la volonté d'un être humain.

Nous montrons d'abord que cette conception de l'écriture comme consignation d'une volonté (V. 4. A) remonte à la naissance de la lecture scolastique, ce tournant intellectuel du XII^e siècle décrit par Illich[888], puis se prolonge avec la modernité comme le montre De Certeau. La { programmation } hérite de ces neuf cents ans de culture livresque, notamment avec le paradigme logique induit par l'{ architecture de Von Neumann }. Selon ce paradigme, programmer c'est écrire une suite logique d'opérations idéalement indépendantes d'une machine en particulier. Cette abstraction renforce la conception instrumentale des machines et le primat de la volonté du sujet humain écrivant.

I | 2 | A
architecture de Von Neumann
Voir p. 114

algorithme, architecture de Von Neumann, code, code source, programmation, programme, thélémacentrisme

Voir glossaire tome II, p. 3-25

Conception d'autant plus renforcée que l'écriture informatique est performative, et fait agir une machine, ce qui peut engendrer un ensourcellement et une pensée quasi-magique du signe informatique (V. 4. B). Compte tenu de cette spécificité, nous proposons alors la notion de « { thélémacentrisme } »(V. 4. C) : dans le cas de la { programmation }, l'écriture ne serait plus le décalque de la parole mais de la volonté. La part machinique du { programme } en est par là minimisée, voire annulée, ce que nous montrons à travers l'exemple des { algorithmes } et de la façon dont ils sont pris comme objets d'étude (V. 4. D).

888 Illich, Ivan. *Op. cit.*, chapitre 6 « *From Recorded Speech to the Record of Thought* », p. 93-114.

4 L'ÉCONOMIE SCRIPTURAIRE DE LA PROGRAMMATION (2/2) : L'IMAGINAIRE DE LA COMMANDE

Nous venons de montrer en quoi le terme de { code source } charrie un imaginaire de la source qui peut amener à une fétichisation de cette dernière. Cette fétichisation, nous l'avons appelé avec Wendy Chun l'ensourcellement : la croyance selon laquelle en allant voir le { code source } d'un { programme } ou d'une page web on comprend et on maîtrise mieux les médias informatisés. Contre cette croyance, l'analyse du { code source } d'une page web sous l'angle de l'énonciation éditoriale a permis de montrer que ce qu'on appelle le { code source } est un document différent de la page dont il serait la source. En ce sens, lire le { code source } d'une page web ne permet pas nécessairement de lire plus de choses que sur la page web grand public. Nous lisons simplement un texte différent, qui fait appel à une autre culture technique et professionnelle, qui permet d'avoir un autre regard sur une page web mais qui masque certains choses tout autant qu'il en révèle d'autres.

Ayant traité de la source, nous allons désormais analyser l'imaginaire de la commande. Le { code } a ceci de particulier qu'il est une écriture agissante, une écriture qui fait fonctionner une machine, qui en modifie l'état interne. Dans quelle position cela met l'écrivant humain vis-à-vis de la machine pour laquelle il écrit ? Nous faisons l'hypothèse qu'écrire du { code }, c'est potentiellement – du point de vue imaginaire – commander une machine et la faire obéir à notre volonté. Ce lien entre écriture et volonté n'est pas propre à l'informatique : c'est une dynamique ancienne de l'écriture occidentale (V. 4. A). En revanche, il trouve dans la { programmation } telle qu'elle s'est construite historiquement une nouvelle configuration (V. 4. B). Pour prendre en compte cette dynamique propre à la { programmation }, nous proposerons la notion de « { thélémacentrisme } » (V. 4. C) entendue comme l'idéal d'un assujettissement de l'écriture non plus au discours (*logos*) mais à la volonté (*thelema*) de l'humain. Nous tirons enfin les conséquences de ce { thélémacentrisme }, en prenant l'exemple du traitement des { algorithmes } dans les écrits d'écran (V. 4. D).

4 A. L'écriture comme décalque d'une volonté humaine : une dynamique déjà bien ancrée

La première étape de notre démonstration consiste à montrer le lien qui unit écriture et volonté. En quoi écrire, d'un point de vue imaginaire, revient à consigner sa volonté sur une feuille de papier ou tout autre support ? Il ne s'agit pas de renier le fait que la volonté du scripteur joue une part importante dans toute pratique d'écriture, mais de ne pas limiter l'écriture à cette dimension. Sans quoi on passerait sous silence toute une part du processus d'écriture qui ne tient pas à la volonté du scripteur, tout ce qui n'est pas de la même nature. C'est cette emprise idéologique de la volonté humaine dans l'écriture que nous analysons, pour mieux comprendre ce que cette idéologie invisibilise et comment le remettre en lumière.

Nous montrons d'abord que l'écriture, en Occident, est une pratique qui s'est bâtie sur un idéal de la production d'un texte comme consignation de la pensée et de la volonté propre à un sujet maître de lui-même. Illich avait déjà noté cette tendance au XIIe siècle, autour du passage du texte monastique au texte scolastique (V. 4. A. a), où l'écriture comme consignation de la pensée devient le modèle dominant. Nous faisons alors l'hypothèse que ce modèle acquiert une nouvelle dimension avec le paradigme logique de l'{ architecture de Von Neumann } (V. 4. A. b) qui met l'accent sur la structure logique d'un { programme } et non sur la machine qui l'éxécute. Le vieil idéal d'une écriture abstraite qui n'est que le reflet de l'esprit humain trouve alors en la { programmation } un nouvel avatar et nourrit un rapport instrumental aux { machines computationnelles } : elles ne font qu'exécuter la volonté consignée dans un { programme }.

architecture de Von Neumann, machine computationnelle, programmation, programme

Voir glossaire tome II, p. 3-25

a Retour à la lecture scolastique : l'écriture comme consignation de la pensée

Ce lien entre écriture et volonté, nous l'avons déjà rencontré dans les théories d'Ivan Illich autour du passage entre la lecture monastique et la lecture scolastique au XII^e siècle. La lecture monastique est une pratique où la page est un jardin dans lequel le lecteur déambule. Le texte n'est pas marqué par son abstraction mais au contraire par une forte incarnation dans l'objet (le livre) dans lequel il est présenté et mis en image. Ce qu'on y lit, ce sont des voix : l'écriture est avant tout « consignation du discours[889] ». La lecture scolastique se caractérise au contraire par une forte abstraction du texte, indépendant de ses différents ancrages dans des livres. Ce qu'on y lit ne sont plus des voix mais des pensées. L'écriture devient « consignation de la pensée » et la mise en page traduit l'organisation logique de cette pensée. La page est alors « […] un écran où se projette l'ordre voulu par l'esprit[890] ». Dans cette optique, l'écriture est cette pratique par laquelle nous produisons un texte qui est le reflet de notre esprit. L'organisation du texte est l'image de l'organisation de nos pensées. Cette thèse posée par Illich semble évidente aujourd'hui, précisément parce que nous vivons dans cet univers culturel du texte livresque : abstrait de la page, incarnation d'une *cogitatio*, dans lequel nous pensons trouver les différents moments d'un discours sciemment voulu et organisé par son auteur.

III 1 A
passage entre la lecture monastique et la lecture scolastique
Voir p. 252

De Certeau souligne lui aussi combien l'écriture en Occident est marquée par cette question de la volonté :

> *« Devant sa page blanche, chaque enfant est déjà mis dans la position de l'industriel, de l'urbaniste, ou du philosophe cartésien, – celle d'avoir à gérer l'espace, propre et distinct, où mettre en œuvre son vouloir propre.*[891] »

Écrire, en tant que mythe, est l'exercice d'un « vouloir propre » par lequel l'écrivant se constitue en sujet. Se constituer en sujet, ce n'est pas négocier avec une matérialité parfois réticente, ce n'est pas recueillir les voix consignées sur la page. C'est user de sa seule volonté pour élaborer sa pensée et la disposer sur l'espace d'une page, cette page qui se plie – depuis le XII^e siècle – à ce que l'écrivant veut y inscrire. Dans cette économie, le support est réduit à peau de chagrin, en tout cas dans sa dimension

889 Illich, Ivan. *Op. cit.*, chapitre 6 « *From Recorded Speech to the Record of Thought* », p. 93-114.

890 « […] *the page becomes a screen for the order willed by the mind* ». Illich, Ivan. *Idem*, p. 105.

891 De Certeau, Michel. *Op. cit.*, p. 199.

constitutive voire agissante : il n'est là que pour obéir à la volonté de celui qui écrit. L'acte d'écrire est une mise sous silence de la participation du support et de l'outil à l'écriture.

b Paradigme logique et exercice de la volonté

Nous soutenons que la { programmation } hérite de ces neuf-cent ans de culture du texte, et qu'en conséquence cela participe à invisibiliser ce que peuvent faire les machines. Particulièrement la { programmation } telle qu'elle est entendue depuis la fin des années 1940 et l'adoption de l'{ architecture de Von Neumann }, où données et instructions sont représentées sous la même forme. Nous avons qualifié cette conception de la { programmation } de paradigme logique, car un { programme } peut alors être écrit sur une feuille de papier, et ses différentes phases analysées indépendamment de son exécution par une machine. C'est un passage de la logistique à la logique qui prolonge le mouvement décrit par Illich puis De Certeau : programmer, c'est coucher sur une feuille de papier (un « espace propre ») le déroulé d'un { calcul } qu'une machine va exécuter automatiquement.

I 2 A d
passage de la logistique à la logique
Voir p. 119

Bien sûr, il faut tenir compte des spécificités de cette machine, de ses limitation de mémoire. Il faut formuler les instructions de manière à ce qu'elle les comprenne, c'est-à-dire de manière non ambiguë et selon son { encodage } propre : à tel { code } correspond telle commande, etc. Mais dans ce paradigme, tout cela est secondaire : ce qui importe est le déroulé logique des opérations, non leur réalisation effective par une machine, et ce déroulé logique – ce qu'on appelle le { programme } – est le fruit d'une écriture : la { programmation }. Alors qu'avec une machine comme l'ENIAC, qui n'est pas fondée sur l'{ architecture de Von Neumann }, on ne peut pas « programmer » sans brancher des câbles et organiser matériellement la machine, le paradigme logique qui s'installe vers 1948 consacre l'idée qu'un { programme } peut être correct en dehors de toute implémentation (c'est-à-dire de toute réalisation par une machine), si sa logique interne est correcte. Les erreurs – ou { *bugs* } – peuvent être détectées en amont de cette réalisation, et elles sont la plupart du temps[892] le fruit des humains ayant écrit. Elles résultent d'une volonté mal formulée, c'est-à-dire d'une erreur logique ou d'un humain n'ayant pas respecté la syntaxe du { langage } utilisé.

architecture de Von Neumann, *bug*, calcul, codage , code, développeur, fonction, langage de haut niveau, langage de programmation, langage machine, machine computationnelle, programmation, programme

Voir glossaire tome II, p. 3-25

892 Encore aujourd'hui, dans la plupart des programmes, il est très peu courant que ce soit le *hardware* qui soit à incriminer lorsqu'un *bug* survient. Bien souvent, il s'agit d'une erreur dans l'écriture du programme : une erreur de syntaxe ou de logique.

Le paradigme de Von Neumann introduit donc un recentrage sur les humains, aux dépens des machines : on peut programmer quasi-indépendamment d'une machine spécifique. Programmer nécessite certes d'apprendre un { langage } et sa syntaxe, mais il n'y a pas besoin de connaître le fonctionnement d'une machine particulière. Par exemple, une { fonction } qui servirait à classer des nombres premiers selon leur ordre de grandeur peut être écrite en plusieurs { langages } différents. Le { développeur } devra choisir un { langage } dont il connaît la syntaxe, mais en théorie à aucun moment il devra toucher à l'organisation matérielle de la machine. Il peut écrire son { programme } en s'abstrayant de cette matérialité, parce qu'il existe des procédures qui permettent de convertir les instructions d'un { langage de haut niveau } en { langage machine } spécifique à tel processeur. Le { développeur } peut alors se concentrer sur ce qu'il veut écrire, la structure logique de son texte, en ne se préoccupant presque pas de la machine pour laquelle il écrit. En cela, le paradigme logique de Von Neumann renforce une conception instrumentale des { machines computationnelles } : elles passent au second plan et sont les auxiliaires idéalement parfaites d'une volonté humaine.

4 B. La commande : une spécificité de l'écriture informatique au fort risque d'ensourcellement

application, code, code source, développeur, interface, JavaScript, langage de programmation, logiciel, pixel, programmation, programme

Voir glossaire tome II, p. 3-25

Nous ne disons pas pour autant que les machines sont ou doivent être « autonomes ». Elles ne peuvent pas l'être, leurs actions sont le résultat d'une commande préalable. Car le { code informatique } est, rappelons-le, une écriture de commande, une écriture qui fait agir. Quand nous écrivons dans un éditeur { JavaScript } `alert(«Hello World»)` et que nous cliquons sur « Exécuter », une fenêtre avec affiché « *Hello World* » s'affiche dans notre navigateur. Nous avons, par l'usage de certains signes dotés d'une performance technique, transformé l'état de la matière de notre écran et de notre machine : des { pixels } apparaissent ou disparaissent, des flux électroniques circulent dans des circuits, une aiguille modifie un support magnétique selon des différences de potentiel électrique, etc. Il y a donc, comme le rappelle Galloway, une performativité du signe informatique, performativité à entendre au sens de la linguistique pragmatique : « le code est le seul langage qui soit exécutable[893]. » Cette performativité de l'écriture est spécifique à l'informatique.

Comment cette spécificité de l'écriture informatique renforce-t-elle la conception du texte comme miroir de la volonté humaine ? Quels effets produit-elle sur nos relations à la { programmation } – la rédaction de ces commandes – et aux machines qui exécutent ces commandes ? Nous soutenons que l'imaginaire de la commande recèle un potentiel double risque « d'ensourcellement » dans la concaténation entre l'{ interface } du { logiciel }, son { code } et la volonté du { développeur } ayant écrit le { logiciel } (V. 4. B. a). Ce qu'on voit à l'écran est entièrement déterminé par un { programme } et ce { programme } est le décalque de la volonté d'un { développeur }. Nous donnons ensuite un exemple de cet ensourcellement, à travers un texte de Friedrich Kittler sur cette écriture de commande. Nous montrons comment son analyse est sous-tendue par un rapport magique à l'écriture informatique (V. 4. B. b).

893 « *Code is the only language that is executable.* » Galloway, Alexander R. *Op. cit.*, p. 165.

a Un double danger

En quoi penser le { code } comme écriture de commande est-il doublement problématique et peut-il participer à un ensourcellement ? Du côté des rapports entre { interface } et { code } tout d'abord, la négation de la distance entre ces deux niveaux peut amener à nier « l'inertie machinique[894] » : ce que l'on voit et manipule à l'écran n'est pas le décalque parfait des procédures calculées par la machine. Il y a entre le { code } et la page web ou l'{ application } une distance (et non une causalité directe), distance dans laquelle opère la machine.

Du côté des rapports entre { code } et { développeur } existe le risque de considérer l'instruction comme le décalque d'une volonté humaine. En effet, si ces commandes sont écrites par des humains et exprimées dans un { langage } lisible par des humains, alors on comprend que derrière la causalité établie entre { code source } et { interface } se cache une causalité encore plus problématique entre { code } et volonté humaine : « [...] utiliser des langages alphanumériques de haut-niveau, c'est déjà anthropomorphiser la machine, et réduire toutes ses actions à la commande supposée les provoquer[895] ». Or, il y a dans cette anthropologisation une opération idéologique fondamentale : la neutralisation de l'agir machinique du { programme }. La machine n'a aucune latitude pour agir, elle ne fait qu'exécuter – mais une exécution sans processus, automatique, parfaite. Rêve humain de toute puissance où le verbe se fait acte – à travers une machine réduite au rang d'outil si parfait qu'il en devient invisible[896].

L'ensourcellement est donc double, et repose sur la croyance que :

a) tout ce que je vois à l'écran a son « double » dans le { code source } ;
b) la machine est la simple servante de volontés humaines, que ce soient celles du { développeur } ou de son commanditaire. L'existence de la page lue est ramenée à des entités humaines, sans considérer le média comme constitutif de cette existence, ou alors comme simple condition technique.

894 Bouchardon, Serge. *La valeur heuristique de la littérature numérique*. Paris : Hermann, 2014. L'auteur utilise la notion d'inertie machinique pour qualifier le fait que lorsque qu'on utilise un média informatisé, la machine « résiste », il y a une action de sa part avec laquelle nous devons composer. L'inertie machinique est une manière pour Bouchardon de penser l'interactivité entre l'homme et la machine, et *in fine* d'ouvrir la porte à une polyphonie de l'action. Car dans toute production numérique, ous devons composer avec une action qui n'est pas la nôtre, quelque chose résiste : c'est l'inertie machinique.

895 « [...] *to use higher-level alphanumeric languages is already to anthropomoprphize the machine, and to reduce all machinic actions to the commands that supposedly drive them* ». Chun, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Op. cit.*, p. 309.

896 Ces réflexions sont le fruit d'un travail mené en commun avec Cléo Collomb, alors doctorante au COSTECH. Voir Collomb, Cléo et Goyet, Samuel. *Op. cit*. 2015.

b F. Kittler et les acronymes de *WordPerfect :* un rapport magique à l'écriture informatique

Un bon exemple de cet ensourcellement fondé sur l'écriture-commande nous est, encore une fois, donné par Friedrich Kittler. Dans *Le Logiciel n'existe pas*, il aborde cette question en décrivant l'ouverture du traitement de texte « Word Perfect », populaire à la fin des années 80, supplanté depuis par Microsoft Word[897].

> *« Pour produire un texte avec un traitement de texte […], il faut d'abord s'acheter une suite logicielle commerciale. Il faut ensuite que certains des fichiers de cette suite portent les extensions de nom .EXE ou .COM sans lesquelles aucun traitement de texte ne peut démarrer sous DOS[898]. »*

bug, code, logiciel

Voir glossaire tome II, p. 3-25

Kittler rappelle les conditions matérielles et économiques de l'écriture d'un texte sur média informatisé : il faut acheter des { logiciels }. Ces { logiciels } sont reconnaissables à leur extension (EXE ou COM), indiquant un fichier exécutable en raison des propriétés techno-sémiotiques particulières de ces acronymes :

> *« Les fichiers exécutables et seulement eux entretiennent en effet un rapport particulier à leur nom propre. Ils portent, d'un côté, de grandiloquents noms autoréférentiels comme par exemple WordPerfect, de l'autre, un acronyme plus ou moins crypté car dépourvu de voyelles comme par exemple WP[899]. »*

V 2 A
l'hybridité humain / machine du code informatique
Voir p. 461

Nous retrouvons ici l'hybridité humain / machine du { code informatique }. Le { logiciel }, ou plus précisément les fichiers exécutables, existent sous deux formes : la forme intelligible par le plus grand nombre (« WordPerfect ») et la forme « cryptée » : « WP ». La forme cryptée du { logiciel } est conditionnée dans son existence par les contraintes de mémoire de l'ordinateur[900]. La forme populaire, elle, est découplée du fonctionnement de la machine au profit des seules logiques marchandes et publicitaires[901].

897 Rappellons que cet article est écrit en 1992, et que l'industrie du logiciel était alors bien différente. Les systèmes opérant sous MS-DOS ne disposaient pas d'interface graphique au sens moderne du terme, et il fallait taper la commande voulue pour lancer un logiciel. Cette importance de la commande est centrale dans la pensée de Kittler, qui écrit précisément au moment du passage historique entre la manipulation de la machine par ligne de commande et celle par interface graphique d'ordre icônico-métaphorique (on clique sur des images de dossiers sur un simili « bureau » pour ouvrir un document représenté sous forme d'icône.

898 Kittler, Friedrich. *Op. cit.*, 2015, p. 33.

899 *Ibidem*.

900 *Ibidem*.

901 Ensourcellement dont nous avons relevé plus haut les limites : pour expliquer la puissance et l'adoption massive de l'informatique, Kittler ne tient pas compte de l'importance d'en donner une forme socialement intelligible.

Kittler continue alors sur cette présence curieuse des acronymes (WP, EXE, COM…) :

> « [Les acronymes] *semblent insuffler à l'alphabet, pour la première fois depuis l'invention de celui-ci, des* ***forces magiques****. Le sigle WP* ***fait en effet ce qu'il dit****. Contrairement non seulement au mot "WordPerfect" mais aussi aux mots vides de la vieille Europe tels que "Esprit" ou "Mot", les fichiers informatiques exécutables incluent tous les sous-programmes et les fichiers nécessaires à leur réalisation. L'acte d'écriture consistant à frapper les touches W et P puis "Entrée" d'une console AT ne rend certes pas le mot parfait, mais lance l'exécution de WordPerfect. Tels sont les triomphes des logiciels*[902]*. »*

L'ensourcellement est désormais complet : l'écriture informatique a ceci de « magique » que le signe même (« WP ») fait advenir à l'écran la réalité qu'il désigne (le { logiciel } WordPerfect), puisque le signe informatique est performatif et autothétique[903] : il comporte en lui-même les conditions de sa réalisation (les données et méthodes), au contraire du signe linguistique. Sa réalisation est bien ici à entendre au sens technique : « WP » lance le { logiciel } WordPerfect, mais ne rend pas le mot parfait, contrairement à ce que les discours des industriels voudraient, selon Kittler, nous faire croire. Cette analyse de Kittler montre bien que sa connaissance fine des procédures informatiques se double malgré tout d'un émerveillement devant l'opérativité du signe. De ce fait, les processus en jeu ne sont pas véritablement évoqués ni analysés, et le signe est vu comme opérant « parfaitement », sans { *bugs* } ni écart entre ce que l'utilisateur demande à la machine *via* une commande (« WP ») et l'action que la machine effectue (lancer le { logiciel } WordPerfect).

902 Kittler, Friedrich. *Idem*, p. 33-34. Nous soulignons.

903 Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 204-206.

4 C La tentation thélémacentrique

code, code source, développeur, entité computationnelle, langage de programmation, thélémacentrisme

Voir glossaire tome II, p. 3-25

Cette idée que le { code source } est un texte par le biais duquel un { développeur } commande à une machine transformée en simple exécutante est en première analyse un dérivé du logocentrisme dénoncé notamment par Jacques Derrida. On assujettit une technique d'inscription (l'écriture, le { code informatique }) à une rationalité humaine, sans penser la constitutivité du support. De ce fait, on raffermit la croyance en un hiatus entre la pensée et ses supports, entre l'homme et la technique : la technologie ne joue aucun rôle intellectuel, elle ne fait que consigner ce qu'un humain pense ou veut.

Nous proposons de réexaminer ce rapport d'assujetissement, compte tenu des propriétés scripturaires propres aux médias informatisés. Si le { code } est une écriture de commande, alors ce qui est en jeu n'est plus la consignation du discours mais de la volonté (V. 4. C. a). Nous proposons d'appeler { thélémacentrisme } ce passage du discours à la volonté, et montrons comment il nourrit une vision ensourcellée du { code source } (V. 4. C. b).

a Du discours à la volonté…

Le signe informatique fait agir la machine, conduisant à penser les rapports entre homme et machine sur le mode de la « commande » : l'humain, quand il écrit pour la machine, donne des « instructions », se sert de la « ligne de commande », instructions que la machine « exécute ». En d'autres termes, la technique d'inscription (le { code informatique }) n'est plus assujettie au discours (*logos*) mais à la volonté (*thelema*[904]) de l'humain. Ce que j'écris n'est plus ce que je veux dire mais ce que je veux faire faire à la machine. On pourrait arguer que volonté et discours ne sont que deux caractéristiques d'une « rationalité » humaine. Mais la différence technique nous semble suffisamment importante pour que l'on doive parler, plutôt que d'un logocentrisme, d'un { thélémacen-

904 *Logos* est un terme particulièrement polysémique, circulant, et donc potentiellement dangereux à mobiliser dans une thèse en information-communication. Nous reprenons *logos* au sens utilisé en histoire de l'écriture par Anne-Marie Christin, entendre : discours articulé, parole intelligible. Une critique logocentrique de l'écriture veut dire dans ce sens : l'écriture n'est pas la consignation du discours oral.

trisme[905] }. Le { code informatique } est assujetti à une volonté humaine, ce qui a pour effet symbolique d'empêcher de penser un agir machinique indépendant et de renforcer l'idée que la machine est simple exécutante.

b Faire de l'humain le seul maître à bord : l'idéologie de la commande

Bien sûr, n'importe qui ayant programmé sent bien que la machine n'obéit pas, qu'il y a une forme de résistance, qui tient à la complexité des couches scripturaires engagées et / ou à la complexité des { langages de programmation[906] }. Mais il n'empêche que cette conception servile de la machine reste une représentation puissante, particulièrement ancrée. C'est pourquoi nous parlerons de tentation thélémacentrique, pour ne pas figer les rapports complexes et mouvants entre l'humain et les { entités computationnelles }.

V | 1 | C | a
particulièrement ancrée

Voir p. 452 pour l'analyse de l'argument d'Ada Lovelace sur la créativité supposée des machines.

Pourquoi le terme de tentation thélémacentrique et non pas tendance ? En vertu du rôle politique de la conception de la machine qui y est présupposée. Lorsque De Certeau parle de l'écriture comme « pratique mythique moderne[907] », il interroge cette pratique au regard de sa valeur dans notre société. Écrire, selon lui, est un mythe en ceci que l'on a appris à postuler un sujet en position de maîtrise face à « une extériorité dont il a d'abord été isolé[908] ». Le texte est alors l'intermédiaire permettant la maîtrise du sujet sur le monde. Sur la page, l'écrivant peut exercer une pensée qu'il croit être la sienne propre[909] et ainsi ordonner le monde (cette « extériorité ») à sa guise. En ce sens, écrire est un mythe et une pratique politique, car elle mobilise des représentations organisant les rapports entre l'humain, ses congénères, la nature, les objets techniques, etc.

C'est pourquoi nous parlons de tentation thélémacentrique, en ceci que le { thélémacentrisme } conforte l'être humain dans son rapport avec la machine. Nous trouvons un intérêt à considérer que la machine nous obéit : c'est celui de ne pas remettre en cause le privilège décisionnaire

905 De θέλημα (*thelema*), mot grec signifiant « volonté ». Il ne nous échappe pas que *logos*, tout comme *théléma*, ont un fort héritage religieux. Tout comme le terme de *logos* fut, dans l'Évangile selon Jean, utilisé pour désigner le « Verbe » de Dieu, Matthieu (6:10) parle de la « volonté » (*Thelema*) de Dieu : « Que ta volonté soit faite ». Le terme de « thélémacentrisme », dérivé de logocentrisme, a l'avantage de ne pas se couper de cet héritage religieux. Ironie épistémologique, quelques uns des premiers travaux en France sur les médias informatisés comme outils d'écriture parlaient « d'écriture télématique », en se basant sur la racine grecque *télé* signifiant « à distance ». Notre optique est toute autre, même ces deux termes sont des quasi-homophones. Voir Blanchard, Gérard. Informations : Rencontres de Lure / en télématique. *Communication & langages*. 1981, nº 49, p. 114.

906 Voir la notion d' « inertie machinique » déjà débattue plus haut.

907 De Certeau, Michel. *Op. cit.*, p. 199.

908 *Ibidem*.

909 Rappelons que De Certeau se place ici au niveau mythologique, non dans la description minutieuse de la polyphonie énonciative de tout acte d'écriture.

algorithme,
code,
code source,
entité computationnelle,
HTML,
interface,
logiciel,
thélémacentrisme

Voir glossaire tome II, p. 3-25

de l'humain. Encore une fois, ce { thélémacentrisme } est une tentation : il n'est pas une constante du rapport homme / machine, il dépend du contexte d'utilisation et des représentations propres aux individus. Mais ce concept permet au moins d'identifier et de grouper un certain nombre de discours et d'attitudes envers la technique, en mettant à jour leurs attendus politiques et idéologiques : réaffirmer le *status quo* de la séparation entre l'homme et l'ordinateur, soumettant celui-ci à celui-là par le biais du { code informatique }.

Définie ainsi, la tentation thélémacentrique est un des opérateurs imaginaires d'ensourcellement. Attendu que :

a) le { code informatique } est une écriture de commande ;

b) que cette commande est le fruit d'une écriture et donc de l'exercice souverain d'une subjectivité humaine ;

alors le { code } devient la source unique de ce qui est vu à l'écran, et est, par transitivité, le décalque codé d'une volonté humaine. La complexité des couches scripturaires engagées dans l'effectivité technique du { code } est niée, l'éventualité d'une action propre à la machine est évacuée. Le rapport au { logiciel } ou à la page web est donc fétichisé : non seulement on attribue le { code source } (par exemple un document { HTML }) un pouvoir de causalité, par la vertu de la commande, mais en plus cette causalité est attribuée à des acteurs humains. Double fétichisation donc : du { code } comme pure causalité ; de la machine comme exécutante d'un humain.

4 D Une domestication de la pensée algorithmique ?

Quels sont les effets de cette pensée thélémacentrique ? Nous faisons l'hypothèse que ces effets sont d'ordre politique : le { thélémacentrisme } contribue à négliger l'action des { entités computationnelles } dans la production des écrits d'écran.

Pour prouver cette hypothèse, nous prenons l'exemple des { algorithmes }, objets particulièrement importants dans un champ naissant des sciences humaines et sociales. Si l'{ algorithme } est historiquement un objet lié aux mathématiques et à l'informatique (V. 4. D. a), il est aujourd'hui un objet d'étude privilégié par les chercheurs intéressés par le numérique, notamment par ses conséquences culturelles et politiques (V. 4. D. b). Sans nier l'intérêt de ces travaux, nous souhaitons simplement souligner qu'ils s'appuient sur une conception thélémacentrée du { code } qui produit ce que nous pourrions appeler une « domestication de la pensée algorithmique » (V. 4. D. c). Nous entendons par là une façon de considérer les écrits d'écran et une de leurs composantes – les { algorithmes }, – comme étant la marque de volontés et de logiques humaines. Ce faisant, le texte n'est plus qu'un espace de mise en relation d'humains entre eux, et la médiation technique dans ce qu'elle peut avoir de « sauvage », de non désiré, est passée sous silence. Cet exemple de l'{ algorithme } amènera dans la partie suivante à une réflexion plus large sur la façon dont les { interfaces } et nos façons de l'analyser construisent une invisibilisation de certaines voix du texte : les { entités computationnelles } et leurs actions.

a Algorithme et informatique

algorithme, calcul, entité computationnelle, format, machine computationnelle

Voir glossaire tome II, p. 3-25

Qu'est-ce qu'un { algorithme } ? C'est une manière de résoudre un problème par une liste finie d'actions. Partant d'une situation de départ (*input*), un { algorithme } permet d'arriver à une situation d'arrivée (*output*). On peut donc avoir plusieurs { algorithmes }, pour un même problème, étant donné qu'il y a potentiellement plusieurs façons de résoudre ce problème. En revanche, il y aura toujours une liste finie d'actions. Par exemple, si nous voulons classer les mammifères par ordre croissant de poids ou déterminer le chemin le plus court d'un point A à un point B, nous pouvons mettre au point un { algorithme } qui part d'une situation initiale (l'ensemble des mammifères et leurs poids respectifs moyens ; un fond de carte avec un point A et un point B) et arrive à une situation finale qui est la résolution du problème posé.

L'algorithmie est un domaine des mathématiques bien plus ancien que l'informatique, et qui a suivi un développement autonome. De la même façon, toutes les procédures informatiques ne sont pas algorithmiques. Les { algorithmes }, ne sont qu'une des multiples { entités computationnelles } à l'œuvre dans un ordinateur. En revanche, l'{ algorithme } est un objet mathématique particulièrement important en informatique pour deux raisons. Tout d'abord, leur aspect fini permet de facilement les automatiser – à condition de se doter d'une syntaxe appropriée. Les { machines computationnelles } sont par nature des machines finies, et qui opèrent sur des suites d'opérations finies. Elles peuvent donc facilement calculer selon des { algorithmes },. Il y a en quelque sorte une parenté de nature entre { algorithmes }, et { machines computationnelles }.

V | 1 | B | c
une des multiples entités computationnelles à l'œuvre
Voir p. 449

I | 3 | C | a
des machines finies
Voir p. 150

Deuxièmement, un { algorithme } permet une économie de { calculs }. Un bon { algorithme }, de ce point de vue, est celui qui arrive en le moins d'opérations possibles au résultat attendu. En informatique, cette économie est précieuse car elle fait gagner du temps et de l'espace mémoire. Le cas le plus exemplaire sont les { algorithmes }, de compression, que ce soit d'images (JPEG) ou de sons (MP3). En appliquant les mêmes règles (approximation de fréquences ou de couleurs, suppression de données redondantes, etc.) à tous les fichiers de même { format }, un seul { algorithme } permet de réduire la taille de tout fichier de même type. Cette opération est extrêmement rapide puisqu'il suffit à la machine de suivre la liste des opérations à mener.

b L'algorithme en sciences sociales : une entité computationnelle domestiquée ?

Plus qu'un moyen technique de gagner du temps, les { algorithmes }, connaissent récemment une forte circulation et semblent s'imposer comme l'un des objets privilégiés dans l'analyse des médias informatisés et de leurs effets, que ce soit en sociologie, en esthétique[910] ou en *media studies*[911]. On tente d'en analyser les « pouvoirs[912] » ou on les considère avant tout comme des acteurs « politiques[913] » qui orientent nos façons de vivre et de se comporter sur Internet. L'approche sociologique notamment permet de comprendre comment un { algorithme }, par les critères retenus comme pertinents pour le { calcul }, sont des outils culturels et politiques. Dominique Cardon, dans son analyse du EdgeRank (l'{ algorithme } de classement de Google) montre comment cet { algorithme } est fondé sur des critères de scientificité et de popularité hérités de la bibliométrie américaine[914]. De la même façon, Antonio Casilli soutient que l'{ algorithme } de recommandation de Facebook s'insère dans une « économie du clic » largement ignorée et aux conséquences sociales problématiques[915]. Olivier Ertzscheid rappelle quant à lui régulièrement le rôle que jouent les { algorithmes }, dans l'éditorialisation des textes de réseau contemporains[916].

Ces travaux mettent à juste titre la focale sur les { algorithmes }, en insistant sur le fait que nous n'avons pas affaire qu'à des procédures techniques, mais à des décisions politiques qui s'incarnent dans ces procédures. Le choix des critères et le but assigné à l'{ algorithme } sont effectivement des décisions prises par des humains en vertu de certaines visées politiques, communicationnelles ou plus largement sociales. Nous ne nions pas ce fait. Ce que nous voulons faire valoir, c'est qu'un { algorithme } et ce qu'il

910 Uricchio, William. The algorithmic turn: Photosynth, augmented reality and the changing implications of the image. *Visual Studies*. 2011, vol. 26, nº 1, p. 25-35.

911 Manovich, Lev. *Software takes command: extending the language of new media*. London : Bloomsbury Publishing, 2013 ; Gillespie, Tarleton. The Relevance of Algorithms. Dans : Boczkowski, Pablo J. et Foot, Kirsten A. (dir.), *op. cit.*, p. 167-194 ; Kraemer, Felicitas, Overveld, Kees et Peterson, Martin. Is there an ethics of algorithms? *Ethics and Information Technology* [en ligne]. 2011, vol. 13, nº 3, p. 251-260.

912 Guillaud, Hubert. Limiter le pouvoir des algorithmes. *InternetActu.net* [en ligne]. 10 octobre 2013. [Mis en ligne le 10 octobre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.internetactu.net/2013/10/10/limiter-le-pouvoir-des-algorithmes/.

913 Cardon, Dominique. *Politique des algorithmes : les métriques du web*. Paris : La Découverte, 2013.

914 Cardon, Dominique. Dans l'esprit du PageRank. Une enquête sur l'algorithme de Google. *Op. cit.*, p. 63-95.

915 Casilli, Antonio A. Qui a fait élire Trump ? Pas les algorithmes, mais des millions de « tâcherons du clic » sous-payés. Dans : *Antonio A. Casilli* [en ligne]. Billet du 17 novembre 2016. [Mis en ligne le 17 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.casilli.fr/2016/11/17/qui-a-fait-elire-trump-pas-les-algorithmes-mais-des-millions-de-tacherons-du-clic-sous-payes/.

916 Ertzscheid, Olivier. Un algorithme est un éditorialiste comme les autres. Dans : *affordance.info* [en ligne]. 15 novembre 2016. [Mis en ligne le 15 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.affordance.info/mon_weblog/2016/11/un-algorithme-est-un-editorialiste-comme-les-autres.html.

algorithme, architexte, calcul, code, thélémacentrisme

Voir glossaire tome II, p. 3-25

produit ne sont pas complètement réductibles à des logiques politiques. À rebours de l'idée que « derrière un { algorithme }, il y a des humains[917] », nous soutenons – et rejoignons Turing sur ce point – que les machines « nous prennent souvent par surprise[918] », notamment parce qu'entre les valeurs en entrée (*input*) et les valeurs en sortie (*output*), il peut y avoir des éléments inattendus. Soit parce que nous ne connaissons pas tous les critères calculés et leur pondération, soit parce que l'{ algorithme } manipule des jeux de données dont nous ne maîtrisons pas tous les tenants et aboutissants, soit parce que certaines instructions ne provoquent pas l'effet escompté. Dans tous les cas, un { algorithme } peut produire – et peut-être plus souvent que nous le pensons – des résultats surprenants, même pour ses concepteurs.

c La domestication de la pensée algorithmique

Notre argument porte donc sur ce qui se passe entre l'entrée et la sortie, en d'autres termes sur l'{ algorithme } comme processus de { calcul }. Penser que ce que produit un { algorithme } (la publication d'un statut Facebook par exemple), est le décalque de la volonté de ses concepteurs, c'est nier ce processus. C'est faire preuve de { thélémacentrisme } en ceci que c'est considérer que le { code } – les instructions de { calcul } – n'est que la prolongation de visées humaines. Ce à quoi nous avons affaire, ce n'est plus un texte composite, fait par des machines et des humains, mais simplement par des humains[919]. Ce faisant, on « domestique » non pas une forme de pensée mais au moins une forme de manipulation symbolique réalisée par une entité non-humaine. On ne lit le résultat de cette manipulation que comme le produit d'une volonté humaine. C'est ce que nous appelons la « domestication de la pensée algorithmique ».

Pourquoi tenons-nous à faire valoir cette composante machinique des textes de réseau ? Quelles seraient les conséquences de la prise en compte de cette composante ? Elles sont d'ordre épistémologique et sémiotique. Prendre en compte la composante machinique – et par exemple algorithmique – des textes, cela signifie que lorsque nous analysons sémiotiquement une page Facebook, ou Twitter, ou n'importe quelle autre, nous

917 Idée qui, comme nous l'avons montré en V. I. C. a, remonte à Ada Lovelace et aux premières machines programmables.

918 « *Machines take me by surprise with great frequency*. » Turing, Alan M. Computing Machinery and Intelligence. *Op. cit.*, p. 450.

919 Collomb, Cléo et Goyet, Samuel. *Op. cit*. 2015.

ne sommes pas en prise avec l'univers de sens construit par Facebook, Twitter ou l'auteur de la page. Ou pas seulement : nous voyons aussi des machines calculer. Une sémiotique des écrits d'écran qui prendrait en compte cette remarque n'analyserait donc pas dans les { architextes } des modèles décidés par des concepteurs, mais des lieux – les textes – où se joue la mise en relation d'entités humaines et non-humaines.

Dans la plupart des cas, l'équilibre des forces est toutefois inégal : ce que nous voyons à l'écran, ce sont des textes largement décidés par des humains. Ceux qui énoncent, ce sont en premier lieu des humains ou les entités qui les représentent (entreprises, organisations politiques, etc.) et nous abordons la plupart des écrits d'écran de cette manière. Mais il y a un écart, une part de l'énonciation qui est machinique. Comment dès lors faire valoir cette énonciation, et ne pas retomber dans une idéologie de la source ou de la commande qui tend à l'invisibiliser ?

Désensourcelé des imaginaires de la source et de la commande, nous pouvons désormais proposer une sémiotique qui prenne en compte l'action propre des { machines computationnelles } sans la minimiser ou la ramener à des enjeux humains. Si les { machines computationnelles } sont des entités d'un autre ordre, comment donc accueillir et qualifier la part machinique des textes de réseau ?

bug, calcul, écriture calcul, énonciation computationnelle, entité computationnelle, formalisme, machine computationnelle, thélémacentrisme

Voir glossaire tome II, p. 3-25

Il faut tout d'abord faire preuve de « tact[920] » (V. 5. A). ce qui ouvre la voie aux machines et à leur énonciation. Il faut ensuite se doter d'outils théoriques à même d'aborder ces textes avec tact. Pour cela, nous étendons la théorie de l'énonciation éditoriale et de l'irréductibilité sémiotique[921] aux { machines computationnelles } (V. 5. B). Mais comment reconnaître cette énonciation parmi toutes les autres présentes dans un texte ? Il faut pour cela qualifier le mode d'action des machines, pour comprendre leur mode d'expression. Nous proposons l'idée que c'est le { calcul }, entendu comme exploration des possibles combinatoires, qui définit ces machines, et que c'est donc le { calcul } qui constitue l'irréductiblité computationnelle. Tout texte numérique porte la marque du { calcul } : il est un des possibles actualisé parmi de multiples autres possibles calculés (V. 5. C). Fort de ce résultat, nous proposerons une sémiotique du { *bug* } comme prolongement de cette prise en compte de l'{ énonciation computationnelle } (V. 5. D). C'est par cet objet particulier que nous pourrons continuer à interroger les cadres idéologiques de nos relations scripturaires aux { machines computationnelles }.

920 Le Marec, Joëlle. Le public, le tact et les savoirs de contact. *Op. cit.*, p. 3-25.

921 Souchier, Emmanuël. Le carnaval typographique de Balzac. Premiers éléments pour une théorie de l'irréductibilité sémiotique. *Op.cit.*, p. 3-22.

5 VERS UNE RECONNAISSANCE DE L'ÉNONCIATION COMPUTATIONNELLE

Après avoir montré l'altérité des { machines computationnelles }, après avoir étudié le texte comme point de rencontre entre les humains et les machines, après avoir montré les imaginaires de la source et de la commande qui organisent dans le texte les rapports entre humains et machines, il nous reste à proposer une méthode qui permette de prendre en compte ce que font les { entités computationnelles }, le rôle qu'elles jouent dans les textes de réseau, sans pour autant les réduire à des outils au service de la volonté humaine. Nous avons en effet vu à quel point le { thélémacentrisme }, couplé à une pensée de la « source », peut être puissant pour invisibiliser l'action des machines dans ce qu'elles ont de propre.

Il s'avère en effet que les relations que nous tissons avec les { machines computationnelles } sont marquées soit par des discours dystopiques de « fin de monde », soit par des utopies technophiles. Nous soutenons et partageons de ce point de vue le projet de Cléo Collomb. Il faut faire valoir d'autres narrations[922], d'autres façons de tisser des relations avec ces machines qui ne soient pas des relations d'adoration ou d'asservissement, mais des relations qui prennent acte de la différence constitutive entre humains et machines et qui laissent ces différences s'exprimer.

Notre position est que l'écriture constitue un endroit où machines et humains peuvent tisser de telles relations, à condition d'en donner une définition qui permette ces relations. C'est en effet une activité que les humains et les machines partagent, sans qu'on puisse pour autant dire qu'ils écrivent de la même façon. Avec Cléo Collomb, nous avions proposé en nous basant sur les articles d'Alan Turing la notion d' « { écriture-calcul[923] } », pour différencier l'écriture de la machine de celle d'un humain. Pourquoi avoir gardé le terme d' « écriture » pour qualifier cette activité, alors que nombre de chercheurs ne s'y risquent pas, Turing lui-même étant très précautionneux sur l'emploi du verbe « *to write* » ? Comment une manipulation { formelle } de symboles vidés de signification peut-elle être qualifiée d'écriture ? Ce genre de question est probléma-

I 3 C d
différencier l'écriture de la machine de celle d'un humain

Voir p. 153 pour le détail de cette écriture-calcul.

922 Collomb, Cléo. *Op. cit.*, 2016, p. 236-277.

923 Collomb, Cléo et Goyet, Samuel. Do Computers Write on Electric Screens? *Op. cit.*

tique, précisément en vertu du statut symbolique de l'écriture dans notre société[924]. Comme le montre De Certeau, écrire, c'est entrer dans l'ordre du symbolique, de ce qui fait notre culture, à nous humains. Écrire, c'est faire partie de ceux qui ont le privilège de produire du sens. Refuser le statut d'écrivant aux machines, ce sont les exclure du groupe de ceux qui ont ce privilège. En ce sens, la qualification de « machine d'écriture » au sens fort du terme – de machines qui écrivent – est un geste politique, en tout cas un geste qui vise à organiser un autre type de relations avec les machines. Dire qu'un ordinateur « écrit », c'est lui redonner un pouvoir agissant, une autonomie dans les relations que nous entretenons avec lui. Ce qui ne veut pas dire que nous lui prêtons une intention, une volonté ou même une intelligence. En tout cas pas au sens où un humain « veut » ou « pense », sans quoi une machine ne pourrait jamais « penser » ou « vouloir » ou même « écrire » : la définition étant liée au mode d'action d'un humain, une machine ne pourrait y prétendre. Un ordinateur n'écrit pas *comme un humain*, mais dire qu'il écrit permet de penser la manière dont il agit, configure une certaine situation de communication, participe à cette situation.

Notre problème est donc d'ordre épistémologique : étant donné que les concepts que nous allons utiliser (écriture, texte, signification, énonciation…) sont potentiellement habités par un anthropocentrisme latent, quel cadre théorique peut-on se donner pour repenser la part machinique des écrits d'écran ? Il serait tentant de se débarrasser de toute approche sémiotique, en disant que les catégories *princeps* de la discipline (énonciation, signe…) empêchent de penser ce que fait la machine, ou encore que tout écrit d'écran est déjà une invisibilisation de ce que fait la machine. Plutôt que de renier ce qui a fait jusqu'ici notre méthode, nous voulons montrer dans ce chapitre qu'il est possible d'élaborer une sémiotique qui prenne en compte les machines : leurs actions propres et la manière dont elles énoncent, dont elles participent au texte et à sa signification. Ce serait une sémiotique « non-anthropocentrée[925] », en tout cas une sémiotique plus accueillante vis-à-vis des machines et des { entités computationnelles } et qui permettrait, à terme, de proposer un autre regard sur les écrits d'écran. Un regard qui prêterait attention aux signes habituellement négligés et qui serait par là en mesure d'introduire une faille dans l'idéologie de nos rapports aux machines tels qu'ils sont construits par les textes numériques.

entité computationnelle, machine computationnelle

Voir glossaire tome II, p. 3-25

924 De Certeau, Michel. *Op. cit.*, p. 195-224.

925 Collomb, Cléo et Goyet, Samuel. *Op. cit.* 2015.

L'enjeu de cette dernière partie est donc de proposer une méthode pour étudier sémiotiquement les textes de réseau, en prenant en considération ce que nous avons démontré dans les parties précédentes : les catégories et théories que nous avons à notre disposition ont tendance à minimiser le rôle des { entités computationnelles } dans la production des écrits numériques. Cela demande dans un premier temps de faire preuve de « tact[926] » (V. 5. A) et de se doter d'outil sémiologiques qui permettent ce tact (V. 5. B). Si, comme nous allons le montrer, le tact consiste à laisser ouverte la possibilité de surgissement d'une altérité, comment peut-on envisager les textes de réseau afin de permettre à l'altérité des { machines computationnelles } de surgir ? Comment peut-on voir et lire dans les textes numériques les différentes entités impliquées dans sa production ?

926 Le Marec, Joëlle. Le public, le tact et les savoirs de contact. *Op. cit.*, p. 3-25.

bug, calcul, énonciation computationnelle, entité computationnelle, industrialisation, logiciel, matériel

Voir glossaire tome II, p. 3-25

Une fois ces questions traitées, il faudra qualifier positivement la façon dont les machines et { entités computationnelles } participent de l'énonciation des textes numériques. Plutôt que de les penser comme seulement « autres », comme contrainte matérielle nécessaire ou comme « boîte noire », il faudra déterminer la nature de leur action et comment cette action relève d'une forme d'énonciation. Repartant des acquis du chapitre I sur ce qu'est le { calcul } en informatique, son fonctionnement automatique et combinatoire, nous mettrons en évidence une { énonciation computationnelle } et montrerons quelle est la part de cette énonciation inhérente à tout texte numérique (V. 5. C). Cela nous amènera à l'hypothèse finale de cette thèse qui en est également un prolongement théorique. Une fois isolée l'énonciation computationelle, il est toutefois difficile de construire un corpus qui en permette l'analyse. Notre hypothèse est que l'{ industrialisation } de l'écriture numérique, analysée dans le chapitre IV, a pour particularité d'invisibiliser l'{ énonciation computationnelle } : les textes de réseau doivent apparaître partout de la même manière, sans que le complexe { matériel } et { logiciel } qui en permet l'apparition joue un rôle autre qu'instrumental. Dans quels corpus analyser cette énonciation particulière ? Il est finalement probable que des phénomènes habituellement négligés, tels que les { *bugs* } d'affichage, permettent de deviner, par contraste, le régime idéologique ordinaire des textes de réseau. C'est pourquoi nous proposons une sémiotique du { *bug* } (V. 5. D), afin de poursuivre les réflexions sur l'{ énonciation computationnelle }.

5 A Faire preuve de tact

Le prérequis à une reconnaissance de l'{ énonciation computationnelle }, c'est d'abord de faire preuve de tact. Nous entendons par là une façon de garder ouvertes les possibilités dans une situation de communication donnée (V. 5. A. a). Il s'agit, de façon globale et pas encore appliquée au cas particulier des écrits d'écran, de se mettre dans une posture à même d'accueillir l'hétérogène d'une situation (V. 5. A. b). Dans le cas de notre corpus, le tact consiste en trois atittudes : un retrait (ne pas tenir l'humain comme point d'aboutissement et seul destinataire de ce que nous lisons) ; un décentrage (il y a des { entités computationnelles } impliquées dans la production de ce texte), ce qui amène à une ouverture : l'attention à des éléments du texte que nous aurions habituellement négligés (V. 5. A. c).

a Garder ouvertes les possibilités

Pour Joëlle Le Marec, le tact est « […] ce qui est attendu d'un interlocuteur lorsqu'il fait le sacrifice de la mise en scène de l'intervention dont il vous fait bénéficier et vous décharge d'un devoir de gratitude que vous êtes d'autant plus enclin à éprouver et exprimer que vous en gardez l'entière initiative[927] ».

Là encore, un recadrage épistémologique est nécessaire : Joëlle Le Marec parle de relations interpersonnelles. Médiées par des textes, des institutions et des techniques, certes, mais le « tact » dont elle parle est difficilement applicable à des relations avec des non-humains. A priori seulement, car Joëlle Le Marec précise que le tact est aussi une qualité sémiotique :

> « […] *au plan sémiotique, le tact correspond à une manière de faire fonctionner empiriquement le registre des potentialités, c'est-à-dire de maintenir des potentialités multiples dans le processus de signification qui est en jeu*[928] »

Faire preuve de tact, c'est poser les conditions d'une relation qui laisse apparaître les potentialités de la situation. Plutôt que de refermer la situation par une interprétation rigide (« Ceci veut dire cela ») ou une méthodologie fortement cadrée (Joëlle Le Marec cite l'enquête comme dispositif de production de sens qui « force » les potentialités d'une situation), le tact est une façon de poser des questions ou de lire un texte qui ouvre aux possibles interprétatifs, sans qu'on ait besoin de statuer sur ce qui est dit et par quoi.

927 Le Marec, Joëlle. *Idem*, p. 16.

928 Le Marec, Joëlle. *Idem*, p. 17.

Vinciane Despret préconise une approche similaire, ce qu'elle appelle le tact ontologique[929], cette fois dans le cadre des relations entre les vivants et les morts : il faut se « laisser instruire », laisser le récit devenir une énigme, c'est-à-dire une question dont le résultat importe moins que sa qualité de question : le fait qu'elle laisse en suspens une réponse définitive. Faire preuve de tact ontologique pour Despret, c'est « [...] prendre soin de ce qui confère à la situation sa puissance d'exister[930] », entendre : d'exister comme énigme. Le tact ontologique est à la fois une position de retrait et d'ouverture : retrait de l'enquêteur et de la tentative d'attribuer une signification au récit, au profit des potentialités qu'offre ce récit ; ouverture donc au champ des possibles qu'offre un récit, et ce pour permettre à des entités non-humaines (les morts, dans le cas de Despret), d'apparaître.

C'est en cela que Despret parle d'un « tact ontologique » : c'est un tact qui permet de construire des relations avec des entités non-humaines, qui n'ont pas le même mode d'existence que nous. Qu'est-ce que cela nous apporte dans notre compréhension des écrits d'écran ? Dans notre cas, quand on observe une page web où apparaît un bouton Facebook ou Google, c'est faire preuve de tact que de ne pas supposer qu'on y lit seulement la politique économique et / ou politique de ces entreprises. Ce serait trop rapidement remettre de l'humain dans ce qui apparaît comme produit par une machine.

b Accueillir l'hétérogène

Dire « une machine » suppose toutefois une unicité de la médiation technique. Il serait plus prudent de supposer qu'une multitude de processus automatisés ont permis l'existence de la forme observable. Ces processus sont écrits par des humains, mais réalisés par des machines, selon des modalités que les humains ne maîtrisent pas complètement. C'est le second aspect du tact : il est un « [...] choix de l'hétérogénéité assumée de ce qui entre en jeu dans une relation[931] [...] ». Nous allons même plus loin : le tact consiste à poser une relation de telle façon que cette hétérogénéité puisse apparaître, face à des dispositifs – et le texte au sens certalien en fait partie – qui ont tendance à unifier l'hétérogène et à masquer cette pluralité. « Le » { code }, « la » machine sont autant de moyens d'écrire la relation aux textes informatisés qui ne font pas preuve de tact. Ils masquent la pluralité constitutive de cette relation.

code, écriture-calcul, machine computationnelle

Voir glossaire tome II, p. 3-25

929 Despret, Vinciane. *Op. cit.*

930 Despret, Vinciane. *Idem*, p. 32.

931 Le Marec, Joëlle. Le public, le tact et les savoirs de contact. *Op. cit.*, p. 18.

c Le tact : un retrait, un décentrage et une ouverture

Vincianne Despret et Joëlle Le Marec parlent respectivement des publics de musée et des morts, et toutes deux s'appuient sur des récits ou des enquêtes. À l'inverse, nous étudions des textes et cherchons à comprendre notre relation aux { machines computationnelles }. Peut-on faire preuve de « tact ontologique » envers les machines ? Comment faire valoir ce tact dans un texte, qui est notre mode de relation privilégié avec celles-ci ? À la première question, nous répondons par l'affirmative. Cela nécessite un retrait, un décentrage et une ouverture : un retrait de la norme selon laquelle un texte numérique est fait pour communiquer entre êtres humains et que les machines n'y tiennent qu'un rôle d'exécutantes de volontés humaines. En conséquence, ce retrait est aussi un décentrage : on fait entrer dans l'ordre de la situation de communication des entités qui ne sont pas humaines. Enfin, ce retrait et ce décentrage provoquent une ouverture : on se permet de voir autre chose à l'écran, de lire le texte différemment. On pose les conditions d'un nouveau regard, et donc d'une relation potentiellement différente aux machines. L'ouverture de « potentialités » dans un texte passe donc par le regard que l'on y porte : ce qu'on choisit d'observer en tant que chercheur.

Poser les conditions de ce nouveau regard nécessite donc deux choses. La première, c'est une théorie sémiotique à même de pouvoir accueillir l'hétérogène. Dans notre cas, cela revient à reconnaître le rôle de la médiation technique sans la ramener à des prérogatives humaines – ce qui serait précisément manquer de tact. Un retour à la théorie de l'énonciation éditoriale[932] et à l'hétérophonie[933] constitutive de tout texte permettra un tel tact (V. 5. B). La seconde, c'est de savoir reconnaître dans le texte ce que font les machines, ce qui suppose donc de savoir ce qu'elles font précisément et leur façon de fonctionner. Cela revient à identifier l' « irréductibilité sémiotique » des textes numériques (V. 5. C) : qu'est-ce qui est propre à ces textes, en tant qu'ils sont le produit de machines fondées sur une { écriture-calcul } ?

932 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145.

933 Cormerais, Franck et Gilbert, Jacques Athanase. Entretien avec Emmanuël Souchier. *Op. cit.*, p. 189-214.

5 B Le tact dans le texte : polyphonie énonciative et irréductibilité sémiotique

Une fois identifié ce en quoi consiste le tact, encore faut-il savoir comment il peut s'appliquer à des textes, avant même de voir quelles sont les particularités de ce tact quand nous avons affaire à des écrits d'écran. Comment donc considérer des textes avec tact ? Comment faire du tact une qualité sémiotique ? La première étape, c'est de rappeler la polyphonie énonciative de tout texte (V. 5. B. a). En élargissant cette polyphonie à toutes les entités permettant la production d'un texte, on peut alors, parmi les multiples voix du texte, isoler ce qui relève du mode d'expression particulier du texte lu (V. 5. B. b). Il faut enfin considérer que ce mode d'expression est d'une part irréductible à d'autres modes d'expression, d'autre part qu'il se voit, et qu'il peut donc s'analyser sémiotiquement. C'est donc la question de l'irréductibilité sémiotique des { textes computationnels } (V. 5. B. c) et de la façon dont cette irréductibilité nous permet d'envisager le rôle de la médiation technique avec tact.

algorithme, architecture matérielle, entité computationnelle, script, texte computationnel

Voir glossaire tome II, p. 3-25

a Tact et polyphonie

L'une des composantes du tact, c'est « [...] le choix de l'hétérogénéité assumée de ce qui entre en jeu dans une relation[934] [...] ». Comment faire sentir cette hétérogénéité dans un texte ? En passant par la question des énonciations : tout texte est polyphonique, en ceci qu'il accueille plusieurs énonciations[935], dont une « énonciation éditoriale[936] ». Cette dernière concerne « [...] toute instance susceptible d'intervenir dans la conception, la réalisation ou la production du livre, et plus généralement de l'écrit[937] ». Par exemple, que l'éditeur ou la typographie « énonce » dans le texte imprimé est un fait que la théorie de l'énonciation éditoriale permet de bien comprendre. Mais peut-on y inclure des { algorithmes }, des { scripts } et plus largement des procédures calculatoires automatisées ?

934 Le Marec, Joëlle. Le public, le tact et les savoirs de contact. *Op. cit.*, p. 18.

935 Ducrot, Oswald. Esquisse d'une théorie polyphonique de l'énonciation. Dans : *Op. cit.*, p. 171-233.

936 Souchier, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Op. cit.*, p. 137-145. Voir aussi IV. 4. B. a

937 Souchier, Emmanuël. *Idem*, p. 141.

Notons que Souchier emploie le terme « d'instance », laissant ouverte la possibilité d'une énonciation non-humaine. Nous proposons donc d'élargir la polyphonie induite par l'énonciation éditoriale à toutes les entités concernées dans la lisibilité de l'écrit[938]. Par exemple, le papier choisi « énonce » : par son grammage, il permet à l'encre de pénétrer plus ou moins bien, il influe sur le poids du livre et donc sur sa portabilité ; le sens de ses fibres peut interdire certains gestes d'écriture, selon l'outil utilisé ; par sa composition, il peut plus ou moins bien réagir à l'humidité, ce qui demande des conditions précises de stockage, etc. Toutes les propriétés spécifiques du papier modèlent le texte, lui confèrent certaines possibilités de circulation ou de réappropriation. Et nous ne parlons pas uniquement de la connotation, comme on pourrait dire d'un papier bible qu'il « connote » l'idée de littérature parce qu'il est traditionnellement associé aux éditions de la Pléiade. Quand nous disons que le papier énonce, c'est en fonction de ses **propriétés techniques** *:* épaisseur, poids, composition, rigidité, résistance à certains outils d'écriture, au temps ou à l'humidité…

De la même façon, un ordinateur « énonce ». Ou plus précisément : l'ensemble des { entités computationnelles } – { architecture matérielle } comprise – qui permettent l'existence de l'écrit à l'écran énoncent. En tant que ces processus participent à l'établissement et à la lisibilité du texte, ils font partie de son énonciation éditoriale. Pas seulement parce qu'ils connoteraient une idée de modernité, mais parce qu'ils ont certaines propriétés techniques spécifiques qui modèlent le texte, participent à ses conditions de circulation et font que le texte que nous lisons est celui-ci, et pas un autre. Le même texte imprimé ne serait *pas* le même texte. Mais comment peut-on s'assurer de cette différence par des moyens sémiotiques ?

b Tact et irréductibilité sémiotique : la question des modes d'expression

Il faut pour cela aller un cran plus loin : le texte porte les marques des propriétés techniques spécifiques du mode d'expression qui en a permis l'existence. C'est ce qu'Emmanuël Souchier nomme « l'irréductibilité sémiotique ». Cette théorie nous est précieuse en ceci qu'elle nous permet de dire qu'au-delà de la question de la polyphonie, on peut trouver dans n'importe quel texte – en restant donc dans l'objet privilégié de la sémiotique – une marque de la « voix des machines », la manière spécifique

938 Il y aurait donc des « énoncés sans énonciateurs », si toutefois on limite la notion d'énonciateurs à des humains. Voir Jahjah, Marc. Des énoncés sans énonciateurs ? Du surlignement à la citation dans le dispositif *Kindle* d'Amazon. *Semen*. 2016, nº 41, p. 107-129.

qu'elles ont d'agir. Leurs actions ne seraient donc jamais complètement invisibles, et on pourrait travailler à faire voir cette « part machinique » des textes numériques.

L'irréductibilité sémiotique est une théorie de l'expression propre à chaque média, ce que Souchier nomme des « modes d'expression ». Elle repose sur une idée simple mais aux conséquences importantes : « [...] une modalité d'expression sémiotique n'est pas réductible à une autre[939] ». Autrement dit, il y a quelque chose dans toute modalité d'expression qui ne peut pas être dit par d'autres moyens. Il y a une **irréductibilité** des modes d'expression. Une peinture, ne peut pas être complètement et exhaustivement remplacée par un morceau de musique ou décrite par oral ou par écrit. Il y aura quelque chose de la peinture et dans la peinture qui ne « passera pas » dans ce nouveau mode d'expression. L'exemple de la peinture est assez simple, mais qu'on pense à la distinction entre oral et écrit : la description de peinture dont nous parlons plus haut ne « dira » pas la même chose, selon qu'elle est un discours parlé ou écrit. Ce sont là deux modes d'expression irréductibles l'un à l'autre.

La question des « modes » ou « modalité d'expression » est néanmoins assez floue. Souchier prend l'exemple de la poésie comme mode d'expression. Mais parle-t-il de la poésie comme genre littéraire ou d'un format de composition en particulier (lai, sonnet...) ? Sa définition de l'écriture comme « mode d'expression composite[940] » semble faire pencher la balance vers une acception large de « mode d'expression » : l'écriture rassemble à la fois une dimension linguistique (le sens du mot selon la langue de rédaction), picturale (l'image du texte), matérielle (le support d'inscription et ses propriétés), gestuelle (la technique corporelle permettant l'inscription)... Toutes ces dimensions sont des modes d'expression qui participent à l'énonciation en train de se produire, « di[sent] quelque chose du texte[941] » en train d'être lu. Prises ensembles, elles définissent ce qui fait la spécificité du texte.

939 Souchier, Emmanuël. Le carnaval typographique de Balzac. Premiers éléments pour une théorie de l'irréductibilité sémiotique. *Op. cit.*, p. 20.

940 *Ibidem*.

941 *Ibidem*.

Dans le souci de faire de la place à un maximum de voix du texte, nous optons nous aussi pour une définition élargie des « modes d'expression » : un mode d'expression est tout ce qui permet à un acte de communication d'exister en tant que tel et dont l'assemblage définit les spécificités de cet acte de communication. On peut voir les modes d'expression comme les paramètres d'un acte de communication. Si nous prenons l'exemple d'un sonnet imprimé : une suite de mots (niveau linguistique) organisée selon des règles de grammaire (niveau du code langagier) et de style (niveau du genre poétique) est imprimé (niveau de la technique d'inscription) sur du papier 80 g/m^2 (niveau du support) avec une certaine mise en page (niveau de l'énonciation éditoriale) et relié dans un livre (niveau du média) édité dans une certaine collection (niveau de l'énonciation éditoriale *bis*). Tout cela compose ce que « dit » le poème en question. Tous ces niveaux sont des voix du texte qui ne sont pas réductibles les unes aux autres : un autre papier, une autre collection, une autre technique d'impression et le texte change : l'énonciation n'est plus exactement la même.

Cette acception élargie de l'irréductibilité sémiotique entraîne une conséquence importante pour nous. Le seuil de répétabilité[942] des formes en est considérablement abaissé. Si tout ce qui permet l'existence d'un acte de communication dit quelque chose de cet acte, alors il est très peu probable que nous ayons affaire à la même chose, malgré la répétition matérielle des énoncés. L'exemple de l'écriture est le plus simple : une liste de courses écrite sur un post-it ou sur une feuille arrachée d'un carnet ne sera pas le même texte : ce sont des entités différentes qui sont impliquées dans ce texte, même si la liste est la même sur le plan linguistique. Si cette liste est écrite au crayon à papier, au stylo bille ou avec un feutre à la mine biseautée : les modes d'expression sont différents. Ces différences sont minimes, probablement[943] négligeables prises dans le contexte pragmatique du texte (les courses seront faites que la liste soit écrite au crayon ou au stylo), mais on ne peut pas rigoureusement parler du même « texte » si l'on veut prendre en compte toutes les entités qui y sont impliquées.

IV 4 A a
seuil de répétabilité
Voir p. 407

942 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 19.

943 Nous maintenons le « probablement » car imaginons que la page arrachée du cahier soit légèrement glacée… Alors le graphite de la mine de crayon à papier ne prendra pas aussi bien que sur un papier ordinaire et selon l'endroit où la liste circule (dans une poche de pantalon, un portefeuille…), le graphite se détachera progressivement, rendant la liste plus difficilement lisible. On voit par ce scénario à quel point tous ces paramètres entrent en jeu dans l'acte de communication.

Dans le cas des écrits d'écran, le même texte visualisé dans un { logiciel } différent ne serait pas **non plus** le même texte. Parce que leurs conditions techniques de production ont changé. L'énonciation n'est donc plus la même. Cela rejoint par ailleurs un exemple que nous avons vu à propos du { code source } : la représentation, dans un éditeur de texte spécialisé, d'un extrait de { code source } en annule l'efficacité technique. Le { code } est « désourcé ». Que ce soit au niveau technique ou sémiotique, il n'est donc plus le même, par le simple fait de l'avoir représenté dans un autre { logiciel } : d'en avoir modifié les conditions de production.

V | 3 | C | a
code source
Voir p. 488

c Tact et irréductibilité sémiotique : le saut vers la part machinique des écrits d'écran

Le deuxième aspect de l'irréductibilité sémiotique, c'est précisément qu'elle est **sémiotique** : les entités impliquées dans la production du texte en marquent l'aspect. Elles sont visibles et lisibles. En d'autres termes, tout texte porte les marques propres de ses modes d'expression. On peut donc, par une analyse sémiotique, faire l'inventaire de ces modes d'expression et incidemment des entités impliquées dans le texte. C'est là, selon nous, un saut que fait Emmanuël Souchier. En effet, que tout texte soit composite est entendable, mais cette polyphonie est-elle nécessairement visible dans le texte ?

code, code source, logiciel

Voir glossaire tome II, p. 3-25

Pour faire sentir ce saut, reprenons un exemple utilisé par Bruno Latour et Adam Lowe[944]. L'original des *Noces de Cana* de Véronèse se trouve au Louvre. Une copie se trouve à la fondation Cini à Venise. Cette copie a ceci de particulier qu'elle reproduit au plus près l'original : chaque partie du tableau a été soigneusement scannée à échelle 1:1 et à la lumière blanche afin de restituer le relief de la peinture, puis réimprimée sur un canevas au plus près de celui utilisé par Véronèse… Tant et si bien qu'à en suivre les auteurs, la copie restitue « l'aura » de l'original. Latour et Lowe prennent cet exemple pour réinterroger ce que nous appelons copie et original dans l'environnement technique actuel.

Transformons cet exemple en expérience de pensée. Et si la copie était parfaite ? Et si elle était si minutieuse qu'elle se confond point par point, pigment par pigment, aplat de couleur par aplat de couleur à l'original ? La copie serait toujours réalisée par le biais de techniques numériques, mais

944 Latour, Bruno et Lowe, Adam. The Migration of the Aura – or How to Explore the Original Through Its Facsimiles. Dans : Bartscherer, Thomas et Coover, Roderick (dir.), *Switching codes: thinking through digital technology in the humanities and the arts*. Chicago : University of Chicago Press, 2011, p. 277.

elle serait le clone parfait de l'original. Les deux tableaux seraient exposés exactement dans les mêmes conditions[945]. Comment pourrions-nous alors différencier la copie de l'original ? Ou plutôt, car la question s'est déplacée : qu'est-ce qui nous permettrait de dire : « ce tableau a été peint, l'autre a été numérisé puis imprimé[946] » ? Si l'on en croit la théorie de l'irréductibilité sémiotique, il y aura **nécessairement** quelque chose du tableau qui « trahit » le fait qu'il a été scanné et imprimé. Le tableau portera l'empreinte de son mode de production spécifique, même si cette empreinte est microscopique, quasi invisible. Nous pourrons – par une analyse sémiotique – différencier la peinture de l'imprimé.

Terminons-en avec cette expérience de pensée et revenons aux textes numériques. Que nous permet de penser l'irréductibilité sémiotique ? Que tout écrit d'écran porte les marques de ce qui est propre aux entités qui permettent l'existence de cet écrit. En somme l'irréductibilité sémiotique permet ce retrait nécessaire pour faire preuve de tact envers les machines. Plutôt que de chercher de prime abord ce que le texte « veut » dire et à quel humain il s'adresse, nous posons d'abord la multiplicité de ce que nous avons sous les yeux, en même temps que son unicité : ce texte est unique en ceci qu'il est le produit d'un assemblage d'humains et de non-humains. Ce faisant, on peut se remettre à voir des choses habituellement négligées : temps de chargement d'une page web, pixellisation plus ou moins forte d'un caractère typographique, léger décalage dans l'alignement d'une image ou d'un cadre, marques diverses de l'activité de la machine[947]. Une foule de choses du texte sont réinstaurées comme signifiantes.

945 Au contraire de l'exemple de Latour et Lowe, qui font du cadre d'exposition un élément crucial pour déterminer pourquoi c'est une « bonne » copie.

946 On remarquera qu'en se plaçant du point de vue de l'irréductibilité sémiotique, la question n'est plus : « Qui de l'humain ou de la machine a fait ce tableau ? » La dualité humain / machine, et plus largement humain / objet technique est escamotée au profit d'une question plus riche à notre sens, qui est celle de la spécificité de la production conjointe des humains et des techniques.

947 Pour un exemple concret de ce que peut produire ce genre de regard sur une page web comme le moteur de recherche Google, voir Collomb, Cléo et Goyet, Samuel. *Op. cit*. 2015.

Le retrait provoque donc en retour une ouverture : on fait venir de nouvelles voix au chapitre dans ce qu'elles ont de propre, et notamment les voix des { entités computationnelles } qui ont permis l'existence de l'écrit à l'écran. On peut donc évaluer la « part machinique » des écrits d'écran.

agir computationnel, calcul, énonciation computationnelle, entité computationnelle, irréductibilité computationnelle, machine computationnelle,

Voir glossaire tome II, p. 3-25

Mais on ne peut estimer cette part machinique qu'en en définissant les aspects principaux. Le cas échéant, nous risquerions de confondre les modes d'expression entre eux, prenant pour du computationnel ce qui tient d'un autre mode d'expression. Quelle est donc l'irréductibilité sémiotique des textes numériques ? Qu'est-ce qui fait qu'un écrit d'écran n'est pas un écrit utilisant un autre mode d'expression ? En somme quel est le « dit » propre aux { machines computationnelles } ?

5 C Qualifier l'irréductibilité sémiotique des textes computationnels

Le terme « { machines computationnelles } » trahit déjà notre hypothèse : c'est le { calcul } (*comput*) qui est le propre de ces machines. Lorsque nous avons proposé ce terme de « { machines computationnelles } » avec Cléo Collomb[948], c'était pour insister à la fois sur la dimension technique non-humaine de ces dispositifs (ce sont des *machines*) tout en essayant de saisir ce qu'elles ont de spécifique : elles calculent (elles sont *computationnelles*). C'est par le { calcul } que les textes informatisés sont produits, et c'est la marque du { calcul } qu'ils portent. En revanche, qu'entend-on par « { calcul } » dans le contexte des { machines computationnelles } ? Et plus spécifiquement encore dans ce chapitre : en quoi le { calcul } est-il un mode d'expression, et en quoi constitue-t-il l'irréductibilité sémiotique des écrits d'écran ?

I | 3 | B
elles calculent
Voir p. 143

Le { calcul } informatique est une puissance d'exploration des possibles d'un ensemble combinatoire (V. 5. C. a). Une { machine computationnelle } parcourt automatiquement toutes les possibilités d'un ensemble fini. C'est ce mouvement automatique qui définit un { agir computationnel } (V. 5. C. b) et une forme d'{ énonciation computationnelle }. Pour définir cette énonciation, il faut comprendre qu'elle s'adosse à une autre énonciation : l'énonciation combinatoire (V. 5. C. c). On peut alors définir l'{ énonciation computationnelle } comme une synergie entre le { calcul } comme puissance d'exploration automatique d'un ensemble combinatoire et l'énonciation propre à tout ensemble combinatoire – indépendamment de la dimension calculatoire (V. 5. C. d). Ce qui nous permet de comprendre l'{ irréductibilité computationnelle[949] } des textes numériques, c'est-à-dire la part de ces textes qui revient à l'{ énonciation computationnelle } (V. 5. C. e)

948 Collomb, Cléo et Goyet, Samuel. *Idem* ; Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Op. cit.*

949 Collomb, Cléo. *Op. cit.*, 2016, p. 297, 330.

a Le calcul comme puissance d'exploration des possibles

I 3 C d
un nombre fini de symboles univoques
Voir p. 153

Par { calcul }, nous entendons une suite finie d'opérations { discrètes } effectuées automatiquement par le biais d'un nombre fini de symboles univoques. Calculer pour une machine, c'est effectuer un certain nombre d'opérations sur des éléments dont le sens ne fait pour elle aucun doute. C'est une définition du { calcul } fondée sur les travaux de Turing[950] et de Bruno Bachimont[951], mais qui ne se limite pas au { langage machine } et aux manipulations élémentaires de 1 et de 0. Quand un { programme } manipule une { variable } (un élément du { programme } dont la valeur est stockée par la machine le temps de l'exécution du { programme }), il la traite comme une unité à la valeur univoque et qui se prête à des manipulations précises.

I 3 C b
machines finies à états discrets
Voir p. 152

Cette définition met en avant le fait que les { machines computationnelles } sont des machines finies à états { discrets }. Ce qui suppose que leurs opérations, leur nombre et leur ordre, soient définis en amont. Programmer, de la part de l'humain, c'est fixer les règles du { calcul } tel qu'il va être effectué par des machines. Ce point permet de comprendre une caractéristique essentielle du { calcul } informatique : il « permet de parcourir systématiquement un espace de possibles[952] ». Un ordinateur « calcule » au sens où, en respectant certaines instructions et avec des données de départ, il explore une à une (c'est une machine { discrète }) toutes les possibilités qui s'offrent à lui en fonction du couple données / instructions. Il épuise l'ensemble des possibles contenus dans la situation de départ.

calcul,
discret,
formalisme,
langage machine,
machine computationnelle,
matériel,
pixel,
programme,
texte computationnel,
variable

Voir glossaire tome II, p. 3-25

C'est à ce moment là que la vitesse et l'abstraction { formelle } entrent en jeu. Explorer tous les possibles d'une situation calculable prendrait un temps indéfini pour un être humain, bien plus qu'une vie. En revanche, le coût en énergie et en temps d'une seule opération est extrêmement faible pour une machine. Elle peut passer d'opération en opération, et par conséquent de possible en possible, très rapidement et sans grande difficulté. À condition toutefois que les symboles manipulés soient univoques, qu'elle sache quand arrêter son exploration (la fin d'un { programme }) et que les règles de combinaison donnant lieu aux différentes possibilités soient

950 Turing, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Op. cit.*, p. 230-265.

951 Bachimont, Bruno. *Op. cit.*, 2010 ; Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Op. cit.*, p. 195-225.

952 Bachimont, Bruno. *Op. cit.*, 2010, p. 167-168.

explicites et univoques (d'où la nécessité de l'abstraction { formelle }). Si ces conditions sont réunies, alors une machine pourra calculer l'ensemble des possibles de façon automatique. Automatique, cela signifie que les mouvements du { calcul } – l'exploration – se font indépendamment de toute intervention humaine.

b Une écriture-calcul automatique : l'agir machinique

L'automatisme des { machines computationnelles } est un point important sur lequel il faut s'arrêter. Il est en effet une des caractéristiques des médias informatisés : ce sont des machines qui font, dont l'agencement { matériel } (le courant dans les transistors, les { pixels } à l'écran…) se modifie sans intervention humaine, du moins pendant le temps de la modification[953]. Pourquoi faire valoir cette dimension agissante des { machines computationnelles } ? Qu'est-ce qui différencie sur ce plan les machines d'autres modes d'expression ? Nous soutenons que la prise en considération de cet aspect automatique permet de construire des relations plus équilibrées avec les machines et les { textes computationnels }.

Nous avons pris l'exemple plus haut de l'énonciation du papier : le sens des fibres, l'outil utilisé, contraint l'écriture et les formes graphiques scriptibles[954]. Pour autant, est-ce que le papier agit ? Oui, si on entend par là qu'il participe à la situation de communication[955]. De la même façon, peut-on dire qu'une { machine computationnelle } agit[956] ? Oui, à condition de se doter d'une définition large de l'action comme nous venons de le faire. Mais plus essentiellement : pourquoi cette question importe-t-elle ? La grande différence entre une { machine computationnelle } et du papier, du point de vue du support, c'est que la machine est automatique. Littéralement : elle se meut toute seule, une fois les instructions données. Elle agit donc en un sens beaucoup plus proche de notre façon d'agir, au contraire du papier.

953 Ces machines sont programmées en amont, c'est là qu'intervient l'humain. Mais pendant la compilation ou l'exécution du programme, seule la machine est en mouvement : elle transforme son état intérieur en calculant.

954 Voir par exemple le cas de la calligraphie à la plume, où la plume ne peut remonter sans accrocher le papier. D'où la nécessité d'un *ductus*, c'est-à-dire d'un tracé régulateur des formes des lettres. Ce tracé conditionne l'aspect des lettres, mais aussi les modalités d'apprentissage de cette technique d'écriture. Le papier et la plume, compte tenu de leurs propriétés respectives, jouent donc un rôle dans l'énonciation et plus largement dans la structuration sociale de l'apprentissage de la calligraphie.

955 Collomb, Cléo et Goyet, Samuel. Do Computers Write on Electric Screens? *Op. cit.*

956 Pour une critique du concept d'architexte au regard de cette question de l'agir machinique, voir Collomb, Cléo. *Op. cit.*, 2016, p. 282.

agir computationnel, calcul, énonciation computationnelle, machine computationnelle

Voir glossaire tome II, p. 3-25

Cette forme d'action nous fait vite penser à l'équivalence entre { agir computationnel } et agir humain. Sur ce point, il est important de distinguer l'automatisme de l'autonomie, ce qui permettra de mieux comprendre la différence entre le processus du { calcul } et les règles de ce { calcul }. Une machine est automatique au sens où elle peut assumer seule le passage d'opérations en opérations. En revanche, elle n'est pas autonome au sens où elle a besoin qu'un être humain lui donne des règles de { calcul }. L'humain fixe les règles, la machine explore tout ce qu'il est possible de faire avec ces règles. Les deux tâches sont complémentaires : l'un fait ce que l'autre ne peut pas faire. Une machine ne peut pas se donner des règles de { calcul }, mais un humain n'a pas le temps matériel de mener l'ensemble du { calcul } à bien. Entendues comme cela, les relations entre humains et machines sont des relations de synergie, et non de domination ou de commande. Pour cela, il faut comprendre précisément ce que font les machines et quelle est la nature de leur { calcul }.

c Le calcul, un mode d'expression adossé à un autre : la combinatoire et son « dit »

II 3 *rôle si prépondérant dans l'informatique* Voir p. 215

II 2 *produit des formes esthétiques et culturelles dont l'informatique hérite* Voir p. 187

I 1 A a *une liste finie d'éléments et de règles de permutation* Voir p. 84

I 1 A b *pas besoin de noter toutes les combinaisons* Voir p. 85

On comprend mieux alors pourquoi la combinatoire tient un rôle si prépondérant dans l'informatique. Pas seulement parce qu'elle produit des formes esthétiques et culturelles dont l'informatique hérite, mais parce que le { calcul } des { machines computationnelles } est proprement combinatoire et qu'en conséquence l'{ énonciation computationnelle } s'appuie sur un autre « dit » , qui est celui de la combinatoire.

Un système combinatoire s'organise autour d'une liste finie d'éléments et de règles de permutation. Sur ces bases, on peut produire par permutations un ensemble fini ou indéfini de combinaisons. Un système est d'autant plus performant que le *ratio* entre le nombre d'éléments du noyau (éléments + règles) et le nombre de combinaisons possibles est élevé. Un autre avantage d'un système combinatoire est qu'il permet de prévoir – par le { calcul } – le nombre de combinaisons possibles selon le nombre d'éléments dont on dispose au départ et les règles de permutation que l'on se donne. Il n'y a pas besoin de noter toutes les combinaisons pour en déterminer le nombre.

Y'a-t-il pour autant une « énonciation combinatoire » ? Les textes produits grâce à des principes combinatoires en portent-ils la marque ? Nous avons souligné que la combinatoire appelait une esthétique de l'épuisement et de l'exhaustivité. C'est une première piste pour penser le style des textes qui mettent fortement l'accent sur la dimension combinatoire de leur production. Mais nous pouvons aller un cran plus loin. Les systèmes combinatoires ont fourni depuis longtemps de véritables « matrices énonciatives » : elles permettent de produire à moindre frais un grand nombre d'énoncés et de propositions en tout genre. Les tentatives de Leibniz pour calculer tout ce qui peut être dit[957] ou du Grand Art lullien sont là pour nous le rappeler. Or, le système excède bien souvent les visées de son utilisateur : des combinaisons possibles sont inutiles selon le contexte d'utilisation du système. Eco prend l'exemple de Lulle : certaines propositions rendues possibles par l'*Ars Magna* sont tautologiques : « La Grandeur est grande » par exemple. Dans ce cas il est nécessaire de restreindre le nombre de combinaisons en passant par des règles extérieures au système (ici, la logique et le principe d'identité).

II | **2** | **C**
esthétique de l'épuisement
Voir p. 203

Mais la possibilité existe, et c'est là l'énonciation propre à un système combinatoire : il y a un nombre fini de possibilités en puissance, dont certaines seulement sont actualisées selon le contexte d'usage du système. Les autres combinaisons restent à l'état de possibles non explorés. C'est pour cela que la combinatoire est une « matrice énonciative » : elle est un réservoir d'énoncés possibles, mais un réservoir dont nous pouvons malgré tout déterminer les limites. L'énonciation combinatoire, c'est ce jeu entre le possible actualisé dans le texte et l'ensemble des possibles en puissance, toujours présents et actualisables en respectant quelques règles simples.

On peut penser l'inclusion de ces deux énonciations (la computationnelle et la combinatoire) comme deux niveaux différents : au sein de la combinatoire et de son énonciation, il y a une { énonciation computationnelle }. Toute l'énonciation combinatoire n'est pas contenue dans l'{ énonciation computationnelle }. Les *Cent Mille Milliards de poèmes*[958] de Queneau sont des exemples d'énonciation combinatoire non computationnelle. En revanche, l'{ énonciation computationnelle } est entièrement combinatoire, car c'est son fonctionnement fondamental. Tout comme à l'intérieur du mode d'expression pictural, on peut avoir la gouache, l'aquarelle ou encore la peinture à l'huile, à l'intérieur du mode d'expression combinatoire on trouve le { calcul } informatique.

957 Leibniz, Gottfried W. *Op. cit.*, p. 126.

958 Queneau, Raymond. *Cent mille millards de poèmes*. Paris : Gallimard, 1961.

d L'énonciation computationnelle

agir computationnel, calcul, discret, énonciation computationnelle, formalisme, irréductibilité computationnelle, machine computationnelle, programme, script, texte computationnel

Voir glossaire tome II, p. 3-25

Comment définir l'{ énonciation computationnelle } et qu'apporte-t-elle par rapport à l' énonciation combinatoire ? Elle est l'exploration de tous les possibles du système par le biais du { calcul } { formel }. Un ordinateur fait tout ce qui lui est possible de faire selon les instructions qu'on lui donne. À partir d'éléments { discrets }, il opère toutes les combinaisons possibles et par élimination – réglée par les règles du { programme } – ne retient qu'une seule de ces combinaisons. Mais avant de discriminer, une { machine computationnelle } doit en passer par tous les possibles. C'est là sa spécificité.

C'est cette exploration exhaustive qui fait le propre de l'{ énonciation computationnelle }, permise par la formalisation des manipulations. Bachimont le rappelle très bien et Eco le fait à sa manière quand il mesure la différence entre Lulle et le « calcul aveugle » de Leibniz. Une { machine computationnelle } ne se préoccupe pas du sens de ce qu'elle manipule. Elle peut formuler tout un ensemble de tautologies, de monstruosités, de chimères[959] sans que cela pose le moindre problème en termes { formels }. Ce sont des « chimères » en vertu du contexte d'utilisation du système combinatoire. C'est-à-dire en vertu d'une règle **extérieure** au système. Ce dont n'a pas à se soucier la machine, et c'est bien là sa force : elle exerce son { calcul } « aveuglément », indépendamment de la signification contextuelle de ce qui est manipulé. Elle sélectionne bien sûr un possible parmi tout ce qu'elle a calculé et c'est ce possible qui est affiché et donné à lire. En cela elle « énonce » : elle participe à la composition du texte. Elle y participe par le { calcul }{ formel }, mais d'une manière bien différente de l'humain. Alors que ce dernier limite les possibilités combinatoires par le recours à des règles contextuelles (éviter les tautologies, les résultats sans queue ni tête), une machine doit faire appel à ses seules règles internes, celles précisées par un être humain lorsqu'il écrit un { programme[960] }. L'{ énonciation computationnelle } peut donc être définie comme le fait que tout écrit d'écran résulte du choix, fait parmi

959 Ce qu'on appelle rapidement des « *bugs* », nous y reviendrons.

960 Un programme est, de ce point de vue, l'ensemble des instructions qui permettent à une machine computationnelle de déterminer, parmi tous les possibles combinatoires qu'elle explore, lequel est considéré comme le « bon » en vertu des visées communicationnelles de ce programme. En d'autres termes, un programme, c'est l'internalisation (par une formalisation) des règles contextuelles auquelles fait appel un humain mais qui sont inaccessibles aux machines. Lorsque les développeurs de Twitter écrivent un algorithme organisant les tweets de façon antéchronologique sur la page, ils le font en vertu de modèles journalistiques (la dépêche, le fil d'actualité…) éminemment contextuels. Le navigateur qui exécute cet algorithme pourrait tout à fait organiser les tweets autrement. Pour qu'il le fasse, il faut lui préciser de manière formelle quelle organisation privilégier parmi toutes celles possibles. Le programme en ce sens est une limitation des possibles combinatoires par la formalisation de logiques culturelles qui président habituellement (entendre « humainement ») à cette limitation.

l'ensemble fini de potentialités d'un système combinatoire. Toutes ces potentialités sont explorées automatiquement par le biais d'un { calcul } { formel } intégré dans des circuits électroniques – ce que nous avons appelé plus haut l'{ agir computationnel }.

e L'irréductibilité computationnelle

Comment cette { énonciation computationnelle } se donne-t-elle à lire ? Comment peut-on la repérer dans des textes numériques ? Quelle est la marque du { calcul } propre aux textes numériques ? En d'autres termes, quelle est leur { irréductibilité computationnelle } ?

L'{ irréductibilité computationnelle } est la marque du { calcul } comme choix d'un possible parmi un nombre fini d'autres possibles, tout en maintenant ces autres possibles à l'état de potentialités actualisables (car parcourues par le { calcul }). En d'autres termes, un texte numérique se caractérise par le fait qu'il pourrait toujours être agencé autrement. Il est en effet réductible à des entités { discrètes } et les différentes entités machiniques qui permettent son affichage peuvent manipuler et combiner ces éléments { discrets } de façon différente, faisant alors varier le texte affiché. Ce que certains ont vu comme la « plasticité[961] » ou la labilité des textes numériques, nous la nommons donc { irréductibilité computationnelle } : c'est le { calcul } qui permet la variabilité des textes selon le possible retenu par les machines.

La marque propre aux { textes computationnels }, ce sont donc les nombreuses – même si parfois microscopiques – différences d'affichage selon les machines : une police de caractères aura « sauté » et sera remplacée par une autre, telle page web mettra plus de temps à s'afficher en raison de { scripts } particulièrement lourds pour le navigateur, une requête Google n'aura pas le même résultat selon l'heure de la journée, le pays d'où est lancé la requête… Tous ces phénomènes relèvent de l'{ irréductibilité computationnelle } en ceci qu'ils résultent d'un { calcul } qui discrimine parmi un lot de textes possibles celui qui sera affiché et c'est cette discrimination qui fait la spécificité de **ce texte précis** qui apparaît à l'écran.

On peut alors préciser ce qu'est « l'aléa machinique » dont nous cherchions à maintenir l'existence. Il n'est en rien un aléa car il n'est pas le résultat du hasard mais d'un { calcul }. C'est un des possibles combinatoires résultant de l'interaction entre plusieurs entités machiniques

V | 1 | C | b
cherchions à maintenir l'existence
Voir p. 454

961 Tardy, Cécile et Jeanneret, Yves (dir.). *Op. cit.*, p. 205-216.

bug,
calcul,
énonciation computationnelle,
entité computationnelle,
formalisme,
industrialisation,
interopérabilité,
irréductibilité computationnelle,
machine computationnelle,
script,
système d'exploitation,
texte computationnel

Voir glossaire tome II, p. 3-25

(navigateur, { scripts }, protocoles réseaux, { système d'exploitation }, serveurs…). C'est la marque de ce qui fait l'irréductibilité sémiotique des { textes computationnels }.

Ce faisant, non seulement nous maintenons l'existence de cet « aléa », mais de plus nous l'expliquons comme le produit d'un certain mode de fonctionnement : le { calcul }. En cela, la théorie de l'irréductibilité sémiotique permet de prendre au sérieux l'activité des { machines computationnelles } : elle permet d'en qualifier la nature et d'instaurer avec cette activité une relation pleine de tact. Plutôt que de condamner l'aléa comme un { *bug* } ou comme un fruit du hasard, on comprend que c'est un possible calculé par une machine. C'est donc une proposition faite par une entité non-humaine, suivant son mode de fonctionnement propre – le { calcul } { formel } et combinatoire.

5 D Industrialisation de l'irréductibilité computationnelle et prolongements sémiopolitiques. Vers une sémiotique du *bug* ?

Une fois cette { irréductibilité computationnelle } théorisée, comment peut-elle nous permettre de porter un nouveau regard sur nos objets, ou mieux encore : de porter notre regard vers de nouveaux objets ?

Si l'existence d'une { irréductibilité computationnelle } a été théoriquement démontrée, il est encore difficile de voir comment elle existe sémiotiquement dans des objets concrets et dans quel corpus on pourrait donc l'étudier… Nous faisons l'hypothèse que cette difficulté est liée à l'{ industrialisation } de l'écriture, étudiée dans le chapitre précédent, et particulièrement au rôle central que joue l'{ interopérabilité } dans cette { industrialisation }. Selon nous, les écrits d'écran contemporains reposent sur une conception tellement abstraite du texte que la plupart des marques d'{ énonciation computationnelle } sont masquées (V. 5. D. a). Si en effet le texte doit s'afficher à l'identique en dépit de ses différentes instanciations matérielles, quelle est la place possible pour l'{ énonciation computationnelle } ? Nous détaillerons dans un premier temps cette hypothèse, avant de proposer une liste d'objets concrets capables de nous donner l'occasion d'analyser l'{ énonciation computationnelle }. Contre cette idéologie du texte numérique comme devant se répliquer à l'identique, il nous semble finalement qu'une sémiotique du { *bug* } (V. 5. D. b) permet de déjouer cette idéologie et de reconsidérer la place qu'occupent, dans les écrits d'écran, les { entités computationnelles }.

IV **4** **C**
rôle central que joue l'interopérabilité
Voir p. 428

a L'industrialisation de l'écriture, au détriment de l'irréductibilité computationnelle ?

IV 4 *standardisation des formes-textes* Voir p. 405

III 2 B *forte discrétisation* Voir p. 298

III 3 C b *maximise les combinaisons possibles* Voir p. 323

En premier lieu, l'{ irréductibilité computationnelle } nous permet de préciser le rôle que jouent les { API } web dans la { standardisation } des { formes-textes }. Quand Twitter ou Facebook déterminent, par les recommandations de leur { API }, une version canonique d'un { tweet } ou d'un statut, ces entreprises canalisent le { calcul } des textes possibles. Nous avons vu en effet que cette { standardisation } s'appuie sur une forte { discrétisation } des éléments constitutifs des { formes-textes }. Cette { discrétisation } maximise les combinaisons possibles, et donc les réutilisations. Plus un texte est discrétisé, plus ses possibles combinatoires sont nombreux. De ce point de vue, la { standardisation } des { formes-textes } est l'imposition d'un possible combinatoire. L'enjeu est de guider l'{ énonciation computationnelle } vers un résultat (la forme canonique) dont ces plateformes tentent de garder la maîtrise. Le phénomène d'{ industrialisation } de l'écriture est donc également une { industrialisation } de notre rapport aux { machines computationnelles }, puisque cette { industrialisation } concerne la manière dont l'{ énonciation computationnelle } est traitée. Nous avons en effet montré comment les textes numériques reposent sur une idée abstraite du texte, qui doit se répliquer à l'identique selon les pages où les { formes-textes } s'ancrent. Or, cette conception du texte est également une certaine conception des outils d'écriture que sont les { entités computationnelles } : ces dernières importent peu, et idéalement doivent être invisibles. Si une forme apparaît différement selon le navigateur ou le { système d'exploitation } utilisé, alors quelque chose ne fonctionne pas. Les résultants déviants d'un standard sont définis comme des erreurs ou des formes hétérodoxes : des formes qui ne respectent pas les recommandations graphiques éditées par les plateformes *via* la { documentation } des { API }, ou qui ne sont pas conformes à ce canon.

API,
***bug*,**
calcul,
discrétisation,
documentation,
énonciation computationnelle,
entité computationnelle,
forme-texte,
industrialisation,
irréductibilité computationnelle,
machine computationnelle,
standardisation,
système d'exploitation,
tweet

Voir glossaire tome II, p. 3-25

b Une sémiotique du *bug,* pour rendre visible l'énonciation computationnelle ?

Notre dernier argument, celui qui dessine une future trajectoire de recherche, est que devant cette domestication de l'{ irréductibilité computationnelle }, nous en serions réduits à une catégorie limitée d'objets : les erreurs d'affichage, ou { *bugs* }.

Le lien entre { industrialisation }, { standardisation } et { énonciation computationnelle } permet de penser que cette { industrialisation } participe à un mouvement plus global de domestication et d'invisibilisation de l'{ énonciation computationnelle } : les possibles calculés par la machine sont fortement restreints, si ce n'est limités à un seul résultat. Les autres possibles sont disqualifiés comme des erreurs, au nom d'une version canonique du texte. Avec Cléo Collomb, nous avons déjà proposé l'hypothèse que nos rapports ordinaires aux { machines computationnelles } sont fortement conditionnés par cette idéologie d'une domestication des possibles[962]. En prenant l'exemple d'erreurs d'affichage sur Google Earth, nous avons montré que ces images ne sont considérées comme des { *bugs* } qu'en vertu d'une conception instrumentale des machines où ces dernières doivent obéir aux ordres que nous leur donnons. En l'occurrence : nous fournir une représentation du monde qui obéit aux lois de la physique. Mais c'est là prendre une loi extérieure au système de la machine (les lois de la physique) pour en faire un critère discriminant ce qui est une « bonne » représentation d'une « mauvaise ».

962 Collomb, Cléo et Goyet, Samuel. *Op. cit*. 2015, p. 8-9.

Fig. 60. Les « *bugs* » de Google Earth : l'actualisation d'un des possibles combinatoires. Capture d'écran de Google Earth publiée sur la page http://www.postcards-from-google-earth.com/bronx1/. Voir Valla, Clément. *Postcards From Google Earth* [En ligne]. 2010 – Présent. [Consulté le 2 septembre 2017]. Disponible à l'adresse : http://www.postcards-from-google-earth.com/..

irréductibilité computationnelle, machine computationnelle

Voir glossaire tome II, p. 3-25

La machine qui produit une image comme celle-ci ne commet pas une « erreur », elle suit l'une des possibilités prévues par son système. Cette image porte la marque de l'{ irréductibilité computationnelle }, mais cette énonciation est bien souvent dévaluée, déconsidérée, car ne respectant pas ce qu'on attend de la machine.

En dernière analyse, la prise en compte de l'{ irréductibilité computationnelle } permet de poser l'hypothèse d'une idéologie ordinaire de l'invisibilisation de l'action des { machines computationnelles } et d'introduire une faille dans cette idéologie. Où la représentation canonique déraille-t-elle ? Quand les machines nous surprennent-elles ? Quand ne font-elles pas ce qu'on attend d'elles ?

Avec ces questions s'ouvre tout un champ d'objets hétérogènes :

Fig. 61. Pistes de corpus pour une « sémiotique du *bug* ». Captures d'écran réalisées entre l'année 2015 et l'année 2017 (détails). Issues des sites www.facebook.com (profile personnel), jsfiddle.net, google.com et http://barthes.ens.fr/reptraqueur.

***bug*, énonciation computationnelle**

Voir glossaire tome II, p. 3-25

Le tableau ci-dessus présente un début de corpus de { *bugs* } d'affichage. Toutes ces images ont en commun d'être le surgissement de possibles calculés mais en général masqués, ou au mieux ignorés. Le { *bug* }, de ce point de vue, permet d'étudier comme un négatif photographique notre rapport ordinaire aux machines dans les textes de réseau : le surgissement d'un imprévu nous renseigne sur nos propres attendus quant à ce qui est prévu. Plutôt que de reléguer le { *bug* } à une erreur, nous pouvons en faire le surgissement d'une { énonciation computationnelle } en dehors d'un cadre restreignant fortement cette énonciation. Une telle sémiotique du { *bug* } serait dès lors nécessairement non-anthropocentrée : il est nécessaire, pour faire du { *bug* } une opportunité épistémologique, de se décentrer et de ne pas considérer nos impératifs communicationnels (nous avons besoin que le texte s'affiche de façon lisible, qu'il s'affiche partout de la même façon, ou qu'il s'affiche presque instantanément) comme les seuls impératifs en cours dans la situation. C'est à cette condition que nous pouvons faire preuve de tact envers ces phénomènes et leur porter un regard différent.

Comme on peut l'imaginer, cette sémiotique est peut être très limitée en objets. Cela ne ferait d'ailleurs que confirmer l'hypothèse que l'{ énonciation computationnelle } est invisibilisée : il est très difficile de trouver des { *bugs* } ordinaires, en dehors de pratiques artistiques qui recherchent ce { *bug* } comme la marque d'une « esthétique computationnelle » qui n'est pas notre objet[963]. Cette « sémiotique du { *bug* } » aurait pour autant des enjeux considérables : elle permettrait d'interroger les cadres idéologiques de nos lectures numériques ordinaires.

963 Voir à ce sujet le projet « haunted by algorithms », dans Guess, Jeff et Wagon, Gwenola. *Media Mediums* [en ligne]. 2013-2016. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.mediamediums.net/fr/projects. Voir aussi le « manifeste médiarchéologique » : Bardini, Thierry, Broye, Lionel, Citton, Yves, *et al. Manifeste médiarchéologiste* [en ligne]. [Mis en ligne le 5 octobre 2016] [dernière modification le 19 avril 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://pamal.org/wiki/Manifeste_M%C3%A9diarch%C3%A9ologiste.

Le pluriel est originaire

De certeau, Michel. *Op. cit.* Paris : Gallimard, 2010 [1990], p. 197

CONCLUSION

La fin d'un parcours

La thèse qui se termine à présent avait pour objectif d'interroger la présence de certaines formes graphiques sur le { Web } contemporain. De ce constat empirique, nous avons isolé et analysé le rôle fondamental d'un acteur technique – mais pas seulement – du texte : les interfaces de programmation, ou { API }. En les considérant comme des outils d'écriture – ou { architextes }, il s'agissait de se demander quelle est la fonction éditoriale des { API }. Notre hypothèse de travail était que les { API } promeuvent une vision combinatoire du texte de réseau. Pour vérifier cette hypothèse, nous avons adopté cinq regards, qui furent autant de chapitres dans la thèse : un regard historique, un regard anthropologique, un regard sémiotique, un regard économique et un regard politique ou écologique.

API, architexte, code, logiciel, matériel, programme, Web

Voir glossaire tome II, p. 3-25

Le premier chapitre fut une approche historique et généalogique de notre objet, autour de l'hypothèse que **les { API } sont l'aboutissement de pratiques d'écriture combinatoire qui traversent toute l'histoire de l'informatique**. Cette hypothèse fut confirmée. En partant d'une conception mécaniste de la combinatoire comme système fini de permutations entre unités élémentaires, nous avons d'abord montré que la combinatoire est liée, dans les { API } web, à une facilitation de l'écriture de { programmes } ou de pages web. Elle permet de répondre à la logique de ces outils envisagés comme « plastigrammes[964] » : fournir des blocs de { code } suffisament standards pour être diffusés en masse, mais suffisament adaptables au contexte d'écriture.

C'est donc un usage de la combinatoire indissociable de la structuration d'un marché – celui de l'informatique – fondé sur la différence entre le { matériel } et le { logiciel }. Dans ce marché, penser un { programme } informatique comme un agencement de blocs modulables permet d'en optimiser l'écriture. Il rend cette écriture plus simple, plus souple, plus adaptable aux demandes des clients. En cela, la part combinatoire des

964 Tardy, Cécile et Jeanneret, Yves (dir.). *Op. cit.*, p. 210-212.

API, architecture de Von Neumann, calcul, discret, écriture-calcul, formalisme, logiciel, matériel, programmation, programmation modulaire, programmation orientée objet, programme, *subroutine*

Voir glossaire tome II, p. 3-25

{ API } n'est pas propre à ces outils d'écriture, mais sont l'héritage d'au moins deux autres façons de programmer : la { programmation orientée objet } et la { programmation modulaire }.

En remontant un cran plus haut dans l'histoire de la { programmation }, nous avons montré que l'usage de la combinatoire dans l'écriture informatique n'est pas qu'une simple adaptation à un environnement économique, mais qu'elle est présente dès les débuts de la { programmation } moderne et qu'elle a pour principale fonction de faciliter l'écriture des { programmes }. C'est ce que nous avons montré autour du concept de { *subroutine* }, soit le fait d'isoler – dans un { calcul } particulier – des portions réutilisables de ce { calcul }. On extrait ainsi d'un { programme } un bloc, un module préécrit que l'on peut remobiliser dans d'autres contextes. C'est selon nous la première apparition en { programmation } d'une façon d'écrire que l'on retrouve dans les { API }. Cela repose toutefois sur deux attendus : que les problèmes soient computables, c'est-à-dire divisibles en séquences que l'on peut réutiliser dans d'autres problèmes ; que la { programmation } soit une forme d'écriture, c'est-à-dire que données et instructions soient fournies à la machine sous la même forme. Il faut donc que programmer soit une pratique d'écriture. C'est ce que nous avons appelé la « scripturisation de la programmation », permise par l'adoption de l'{ architecture de Von Neumann } en 1946. Avec ce modèle de construction, les instructions formulées à un ordinateur sont fournies par le biais d'une écriture et non plus par l'agencement physique de ses composants matériels. La balance entre { matériel } et { logiciel }, entre sciences de l'ingénieur et science { formelle }, consubstantielle à l'informatique[965], penche alors du côté de la logique et du { formalisme } et donc du côté d'une abstraction combinatoire de l'écriture. En d'autres termes, les { API } sont des outils d'écriture combinatoire parce qu'elles héritent de certains modèles de { programmation }, mais aussi parce qu'elles héritent d'un cadre intellectuel où l'abstraction { formelle } du { logiciel } est très importante, dans lequel le { matériel } compte peu, voire pas. C'est cette abstraction qui, selon nous, permet de maximiser le jeu combinatoire de l'écriture.

Mais cette généalogie de la combinatoire ne s'arrête pas là. Si la combinatoire est présente dès les débuts de la { programmation }, nous avons vu que cela tient en partie à la nature des problèmes computables : ils peuvent

965 Tedre, Matti. *Op. cit.*

être divisés en problèmes élémentaires qu'on peut ensuite réassembler une fois résolus. Cela nous a amené à remonter encore un cran plus haut : si la combinatoire tient une place prépondérante en informatique, ce n'est peut être pas seulement parce qu'elle simplifie la { programmation }. C'est peut être également parce que l'informatique est une science qui s'est historiquement construite sur un certain modèle d'écriture combinatoire. Pour prouver cela, il a fallu démêler l'écheveau des liens entre informatique et écriture et repartir de la fondation de l'informatique comme science à la fin des années 1930. L'informatique naît de la logique, plus précisément des travaux d'Hilbert sur l'arithmétisation de la logique : on peut déterminer par le { calcul }, entendu comme écriture { formelle }, la vérité des propositions et énoncés mathématiques. On peut alors construire une machine qui effectue ces calculs, ce qui fut théorisé dans l'article d'Alan M. Turing en 1937 : *On Computable Numbers, with an Application to the Entscheidungsproblem*. Sous condition toutefois de se doter d'une conception { formelle } de l'écriture, ce que nous avons appelé l'« { écriture-calcul[966] automatique } ». Cette { écriture-calcul } possède cinq caractéristiques : elle est { formelle }, monosémique, { discrète }, combinatoire et unidimensionnelle. C'est seulement sous ces conditions qu'une machine peut l'exécuter automatiquement, entendre : sans intervention humaine.

Cette { écriture-calcul automatique }, fondement théorique de l'informatique, est largement tributaire d'une pensée combinatoire : tout ce qui est calculable peut être ramené à une permutation { formelle } d'unités { discrètes }, un ensemble de 0 et de 1 inscrits dans les cases d'un ruban de papier infini. Si les { API } sont des outils d'écriture combinatoire, ce n'est donc pas seulement parce que cela facilite l'écriture de sites web ; pas seulement parce qu'elles héritent de dizaine d'années de modèles de { programmation }. C'est aussi parce que ce sont des outils d'écriture informatique et que l'informatique est construite sur une conception combinatoire et { formelle } de l'écriture.

Les pratiques d'écriture combinatoire traversent donc toutes les couches historiques et sémiotiques de notre objet, depuis les façons d'écrire avec une { API } jusqu'au fonctionnement théorique d'un ordinateur en passant par les modèles de { programmation }. Ce résultat acquis, nous avons en quelque sorte renversé le problème. Plutôt que de prendre le texte

966 Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Op. cit*

de Turing comme un point d'arrivée, nous l'avons pris comme point de départ d'une nouvelle question : à quels imaginaires de l'écriture renvoie l'{ écriture-calcul } ? Nous voulions vérifier l'hypothèse que **la combinatoire n'est pas seulement le mode de fonctionnement des machines informatiques et une solution bien pratique pour écrire des { programmes }, mais aussi un puissant foyer d'imaginaires de l'écriture**.

développeur, écriture-calcul, formalisme, jetons, programme, technologie de l'intellect / intellectuelle

Voir glossaire tome II, p. 3-25

Cette hypothèse s'est **partiellement confirmée** dans le **deuxième chapitre**. Plus précisément, elle a été étendue et reformulée. Étendue car nous ne nous sommes pas contenté de la combinatoire mais sommes reparti de la machine de Turing et des caractéristiques de son écriture : elle est chiffrée, combinatoire et universelle[967].

Notre analyse de l'imaginaire du chiffre a montré qu'il ne se réduit pas à la statistique et à la mesure, mais que le chiffre possède au moins trois fonctions. Premièrement, il est une puissance d'abstraction. D'abord parce qu'il permet la commensurabilité, soit le fait de poser un plan d'équivalence entre plusieurs choses de nature différente ; ensuite parce que bien des systèmes numériques – dont le nôtre – utilise le principe de position qui permet une combinatoire efficace mais nécessite une grande abstraction intellectuelle, ainsi que le zéro comme représentation de l'absence. Deuxièmement, le chiffre ne se limite pas à l'abstraction et à une forme de rationnalité calculatoire. Il est aussi lié à des pratiques mystiques, notamment parce qu'il permet la mesure et la prédiction. Il est le garant de l'harmonie universelle – comme c'est le cas chez les Mésopotamiens – et il permet la maîtrise du temps, comme dans le cas du compte long maya où le chiffre est un fardeau porté par les dieux. Troisièmement, un retour à notre corpus a montré que le chiffre est un vecteur de confiance : par l'encryptage de { jetons }, il tient lieu de substitut à un pacte passé entre { développeurs } et plateformes, ainsi qu'entre internautes et { développeurs }.

Deuxième caractéristique de l'{ écriture-calcul } : c'est une écriture combinatoire. Là encore, nous avons renversé la perspective : plutôt que de considérer la combinatoire comme propre à l'informatique, nous avons considéré l'informatique comme exploitant un principe combinatoire propre à tout système d'écriture. Ce qui nous a amené à l'hypothèse que la combinatoire est une { technologie de l'intellect } : elle produit des formes de pensée spécifiques. La question devient alors : quelles formes de

967 Universalisme du reste explicitement revendiqué par Turing lorsqu'il ne dit s'occuper dans son article que des « machines à calculer universelles ».

pensée la combinatoire a-t-elle produites en Occident ? Peut-on considérer l'informatique comme une de ces formes ? À la première question, nous avons donné deux réponses. Premièrement, la combinatoire amène à une métaphysique de l'épuisement. Parce qu'elle permet, à partir d'un nombre limité d'unités élementaires, de calculer un ensemble considérable de possibles, la combinatoire est une force d'épuisement du possible. Mais dans le même temps, elle pousse à s'interroger sur ce qui échappe à ce système, ce qui tient de l'incombinable. En cela, la combinatoire oscille entre l'épuisement du fini et l'espérance de l'infini. C'est cette oscillation, cette tension qui explique selon nous les liens attestés entre combinatoire et pratiques mystiques. Deuxièmement, la combinatoire fournit, à travers les lettres de l'alphabet, le modèle sensible d'une méthode analytique de résolution des problèmes. C'est la thèse que propose Mario Vegetti en étudiant la pensée de l'écriture chez Platon[968] : l'apprentissage de l'écriture est fondamental dans la construction d'une pensée, si ce n'est scientifique, tout du moins analytique car l'écriture repose sur le découpage du mot en lettres, c'est-à-dire en unités élémentaires ouvertes à la recombinaison. Depuis le modèle platonicien, le caractère combinatoire de l'écriture est donc lié à la quête d'une méthode scientifique.

Ce qui nous conduit à la troisième caractéristique de l'{ écriture-calcul } : c'est une écriture aux prétentions universelles. D'où vient cet universalisme ? Comment est-il lié à la combinatoire comme { technologie de l'intellect } ? Il a fallu pour répondre à ces questions se plonger dans l'histoire des sciences occidentales, habitées par la quête d'une « méthode universelle » au moins depuis les travaux de Raymond Lulle au XIII^e siècle. Cette recherche d'une méthode universelle repose en grande partie sur une sémiotique, autrement dit une théorie du signe : il faut se doter d'une forme d'écriture qui permette d'encoder le réel, de l'unifier sous un même régime de signification et ainsi permettre des manipulations combinatoires pour découvrir l'ensemble des savoirs à partir d'un petit nombre de règles. La quête d'une méthode universelle en Occident est donc directement liée à la propriété combinatoire de l'écriture. Ce point acquis, nous avons alors pu relier le { formalisme } hilbertien à cette quête pluriséculaire : le { calcul } comme écriture est un nouvel avatar de cette méthode universelle, mais aux prérogatives cette fois limitées aux mathématiques. Lorsque Turing qualifie donc sa machine d'« univer-

968 Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans : Détienne, Marcel (dir.), *op. cit.*, p. 387-419.

selle », il prend place dans cette longue histoire, tout en la reformulant à sa manière. L'universel, chez Turing, est une puissance d'imitation : une machine est dite universelle lorsqu'elle peut imiter le comportement d'une autre machine, attendu que ce comportement peut être formulé en des termes calculables. Nous identifions ainsi l'une des caractéristiques de l'universalisme numérique : il est une faculté d'imitation par le { calcul }, c'est-à-dire une écriture combinatoire et { formelle }. Cet universalisme est le produit d'un triple processus : historique, scientifique et technique. Historique parce qu'il est le dernier avatar de la quête d'une méthode universelle grâce à une écriture adaptée ; scientifique car c'est le { formalisme } hilbertien qui va fournir autour du { calcul } les fondements théoriques de cette écriture ; technique enfin car on peut construire une machine qui exécute ce { calcul }.

API,
calcul,
discret,
discrétisation,
documentation,
formalisme,
formes-textes
graphe social,
petites formes,
standardisation,
technologie
de l'intellect /
intellectuelle,
***timeline*,**
tweet,
widget

Voir glossaire tome II, p. 3-25

Parti de l'écriture combinatoire des { API } pour arriver à la combinatoire comme { technologie intellectuelle } ayant produit, en Occident, un universalisme calculatoire, nous terminons ainsi la première partie. Nous avons pu cerner quelques traits caractéristiques de cette « science combinatoire » qu'est l'informatique en tant que technique d'écriture. L'hypothèse que nous avions formulée à partir de ce résultat était que que **cette science s'est hybridée avec des logiques industrielles** : elle est devenue une **industrie du texte**.

Le premier mouvement de vérification de cette hypothèse est notre troisième chapitre, consacré à la question suivante : **comment l'universalisme combinatoire de l'informatique transforme-t-il, à travers les { API }, les textes de réseau ?** Nous avions fait l'hypothèse que **cet universalisme combinatoire renforce la modularité et la manipulabilité des textes de réseau**.

Cette hypothèse fut confirmée par une analyse des { *widgets* } de notre corpus. La métaphore de l'« ancrage », omniprésente dans la { documentation } des { API }, réactive une conception du texte comme abstraction et de la page comme port, conception née au XII^e^ siècle, autour du passage de la lecture monastique à la lecture scolastique[969]. Les propriétés du { calcul } radicalisent cette abstraction mais la dynamique globale est similaire : une page est le contenant temporaire de modules qui viennent s'ancrer de temps à autre. Devant cette abstraction du texte, nous avons rappelé la matérialité de tout ancrage et en avons étudié les modalités

969 ILLICH, Ivan. *Op. cit.*

précises. En rappelant d'abord que le lieu de l'ancrage n'est pas neutre. Il est essentiel à deux titres : il est un lieu de mémoire, où les formes du texte se sédimentent au fil de leurs ancrages, expliquant la { standardisation } visuelle des textes de réseau ; il permet également de spécifier la fonction énonciative du { *widget* }, par le jeu des cadres.

Nous avons ensuite réinterrogé la notion de « { petites formes[970] } ». Si, au premier abord, les { *widgets* } sont les formes élémentaires d'une « grammaire éditoriale », un examen plus attentif montre qu'il n'en est rien. Chaque { petite forme } (un { tweet }, une { *timeline* }, une vidéo ancrée…) peut être divisée en plus petites unités. Ce sont des « macro-formes », elles-même divisibles. Ce qui nous a permis de faire le lien entre la modularité des formes et leur { discrétisation }, l'une des dynamiques fondamentales du numérique comme nous l'avions montré dans le premier chapitre. Quelles sont les conséquences de la { discrétisation } sur les { formes-textes } ? En reprenant les hypothèses de Bruno Bachimont, couplées à ce que soutient également Illich a propos du texte livresque, nous avons montré que les { API } web prolongent la dynamique liant { discrétisation } et manipulabilité du texte. C'est particulièrement visible dans notre corpus avec ce que nous avons appelé un « appel à l'écriture » : les { *widgets* } de Facebook ou de Twitter sont des formes qui invitent à être commentées, republiées, aimées.. Elles portent en elles les possibilités de leurs rééditorialisations.

C'est pourquoi un troisième moment de ce troisième chapitre fut consacré à ce lien entre manipulation et éditorialisation, autour de deux cas d'éditorialisation automatique : les cartes Twitter et la génération automatique d'un profil Sens Critique grâce au « { graphe social } » de Facebook. Dans ces deux cas, la { discrétisation } permise par le { calcul } joue un rôle primordial. Elle permet une plus grande manipulabilité des formes, ce qui nous a aidé à cerner l'une des principales caractéristiques de la fonction éditoriale des { API } : en partant d'une structure { discrète }, les { API } de Facebook et de Twitter recomposent des formes sémiotiques connues (le profil, la carte, le lecteur vidéo, etc.) en puisant dans une mémoire des formes. Leur travail éditorial consiste donc à allier { calcul }, histoire du texte, { discrétisation } et mémoire des formes. C'est pourquoi la { documentation } des { API } propose une très grande { discrétisation } des { petites formes } : en maximisant cette { discrétisation }, on maximise les

970 Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *op. cit.*, p. 165-201.

API,
code source,
développeur,
discrétisation,
formes-textes,
industrialisation,
interopérabilité;
jetons,
polychrésie,
standardisation,
Web,
widget

Voir glossaire tome II, p. 3-25

manipulations possibles des { formes-textes } et donc leur rééditorialisations. Ainsi, l'abstraction des formes se double bien d'une manipulabilité accrue, dynamique que les { API } web exploitent intensément.

Mais à quoi, ou à qui, profite cette { discrétisation } maximale proposée par les { API } web ? Pourquoi favoriser à ce point les possibilités de manipulations et de rééditorialisations du texte ? Il fallait, pour répondre à ces questions, s'intéresser aux **enjeux économiques** des { API } web. C'est ce que nous avons fait dans notre **quatrième chapitre**, en proposant l'hypothèse que **les propriétés techniques des { API } web sont mises au service d'une « industrie des passages[971] ».**

Cette hypothèse fut confirmée en quatre temps, structurés autour de la tétrade instrumentation / désir / instrumentalisation / { standardisation }. L'instrumentation tout d'abord : les { API } web sont des outils de délégation de l'écriture. Elles donnent la main à des { développeurs } et internautes pour utiliser et mettre en circulation des données propriétaires. Mais l'outil ne fait pas que donner la main : il la guide vers une plus grande circulabilité des formes. C'est particulièrement frappant dans le cas des cartes Twitter où nous avons montré que dans le { code source } de ces formes, c'est toute la chaîne de publication qui est anticipée et préécrite pour que la carte se réplique à l'identique, permettant alors son intense circulation sans altération visuelle.

Mais il ne sert à rien d'équiper l'écriture si cette dernière n'est pas rendue désirable. C'est pourquoi nous avons formulé l'hypothèse que les { API } construisent ce que l'économiste Frédéric Lordon nomme une *épithumè* : un régime de désir, une façon de rendre désirable l'utilisation d'un dispositif. Dans le cas des { API }, nous avons avancé que cette *épithumè* se construit par trois processus. Premièrement, une { API } fonctionne, du point de vue du désir, sur le modèle du *peep show* : elle autorise une œillade vers des données qui sont rendus ainsi disponibles –seulement en partie – mais dont l'intégrité n'est jamais remise en cause. À travers une { API }, on lorgne vers des données, mais nous n'en avons ni libre disposition, ni propriété. Deuxièmement, le désir est instauré par la construction de la confiance envers l'{ API }, dont nous avons relevé quatre formes : le { jeton } d'identification ; la « santé » de l'{ API } ; la contractualisation explicite de l'échange de données ; les recommandations formulées aux { développeurs }. Troisièmement, nous avons fait l'hypothèse que le

971 Jeanneret, Yves. *Op. cit.*, 2014, p. 644, 670, 681.

désir de l'{ API } est aussi construit sur l'exhibition de sa { polychrésie }. Notre argument était le suivant : si une { API } doit éveiller le désir, c'est notamment en montrant tout ce qu'on peut faire avec, toutes les possibilités combinatoires qu'elle offre. Cette hypothèse fut en partie invalidée : ce qui est rendu désirable, ce n'est pas tant les possibilités d'écriture offertes par l'{ API } mais plutôt le clic des internautes. Les { *widgets* } ont vocation à faire cliquer les internautes et c'est ce clic qui est, selon la { documentation } des { API } de notre corpus, recherché et désirable car il est une marque de sociabilité, de « partage », d' « expérience » de lecture.

Ce qui nous a conduit à la question suivante : comment les { API } sont-elles devenues des outils d'écriture de la sociabilité ? En d'autres termes : comment ont-elles été instrumentalisées ? En retraçant une « trajectoire d'objets[972] », nous avons mis en lumière que c'est la fonction technique d'{ interopérabilité } des { API } qui a été instrumentalisée, en trois étapes. Tout d'abord par le droit états-unien, qui exclut les { API } de la législation sur le *copyright* au nom de l'innovation ; par certains acteurs économiques qui voient dans l'{ interopérabilité } un levier de croissance économique et un moyen d'assurer l'« ouverture » des données ; par leur évolution conjointe avec le { Web } enfin, contexte dans lequel les { API }, parce qu'elles permettent l'{ interopérabilité }, sont devenues des vecteurs de partage et de sociabilité.

Notre hypothèse de départ était alors confirmée : les { API } participent à une industrie des passages en ceci qu'elles équipent l'écriture de formes circulantes, rendent ces formes désirables et c'est bien une de leur caractéristique technique – l'{ interopérabilité } – qui est mise au service d'une plus grande trivialité des textes de réseau.

Il fallait alors revenir à la question de l'éditorialisation. Car la circulation dont nous parlions jusqu'ici est plutôt un idéal de réplicabilité des formes à l'identique. Or, le spectre de l'altération accompagne ce fantasme de la réplication : plus un texte est amené à circuler, à être republié, manipulé, plus il est amené à se transformer, à s'altérer. Il en va donc de l'image de marque de Facebook et Twitter d'assurer que le texte garde sa lisibilité et sa forme, malgré son intense circulation. Empruntant à Christian Jacob le concept de « pratique lettrée[973] », nous avons montré que les{ API } de notre corpus sont des lieux d'{ industrialisation } d'une pratique lettrée :

972 Casemajor, Nathalie. Matérialisme numérique et trajectoires d'objets : les artefacts numériques en circulation. *Op. cit.*

973 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *Op. cit.*, p. 12.

elles standardisent l'image du texte, en contrôlent à des échelles massives la diffusion et la réplication à l'identique. L'analyse de cette pratique lettrée se déploie autour de trois axes. Premièrement, les plateformes imposent des « frais de mouillage », soit un ensemble de recommandations voire d'obligations pour ancrer des publications afin qu'en soit garantie l'image du texte ; deuxièmement, à travers les { *widgets* }, ces plateformes s'exhibent en chef d'orchestre d'une polyphonie énonciative ; mais elles minimisent, voire excluent, de cette polyphonie les machines et { programmes } qui ont servi à produire ces textes. Ainsi, l'{ interopérabilité } n'est pas seulement une question technique et économique, elle est un enjeu idéologique : vouloir que les formes du textes se répliquent à l'identique malgré leur circulation dans différents environnements éditoriaux provoque une invisibilisation – un oubli ? – des machines qui permettent l'existence du texte à l'écran.

API, calcul, code, code source, écriture-calcul, interopérabilité, matériel, programmation, programme, *widget*

Voir glossaire tome II, p. 3-25

Un **dernier mouvement** était donc nécessaire, pour comprendre comment se jouait, dans l'écriture, ces rapports idéologiques. Notre hypothèse de départ repartait de notre étonnement initial[974] : **les { API } nous mettent en présence d'entités machiniques intervenant dans la production des écrits d'écran**, écrits dans lesquels se jouent des enjeux politiques d'**invisibilisation ou de monstration** de ces entités machiniques.

Cette hypothèse fut à moitié confirmée, ou en tout cas affinée. La question ne fut pas tant de savoir si les machines sont montrées ou masquées. La question fut plutôt de savoir : comment les outils théoriques que nous avons mobilisé configurent une relation aux machines ? Et qu'est-ce qui, dans ce que nous avons appelé jusqu'ici « les machines » est masqué ou exhibé ? Au profit de quel type de relation ? Si effectivement (comme nous l'avions montré à la fin du chapitre IV), certains acteurs techniques sont réduits à la portion congrue dans les écrits d'écran, comment montrer cette présence ? Comment lui donner une nouvelle place, la remettre en lumière ? Il a fallu pour cela repartir des acquis du chapitre I autour de l'{ écriture-calcul } : un ordinateur est radicalement différent d'un humain, tant en degrés qu'en nature. Cela tient à la nature de son écriture et au volume des opérations réalisées. Il constitue donc, pour nous, une altérité. En revanche, il partage avec nous un espace : le texte. En effet, depuis le paradigme logique de Von Neumann, programmer, c'est écrire une suite d'instructions à une machine, ce { code } pouvant être lu tant

974 Voir introduction « Le constat d'une présence ».

par l'humain que par un ordinateur. C'est donc le { code } qui constitue un lieu de rencontre, en tant que texte hybride.

Mais ce lieu de rencontre n'est pas un lieu neutre : il est un lieu de pouvoir, comme nous l'apprend De Certeau. Le texte est un lieu de pouvoir au sens où il est là où s'exerce un pouvoir – celui d'écrire – et un lieu où sont distribuées les places respectives de chacun des acteurs du texte. Ainsi, il fallait comprendre comment la notion de « { code } » configure une certaine relation aux machines, car il est probable que c'est justement dans notre conception du { code } que réside l'invisibilisation des machines.

C'est ce qui fut montré à travers l'analyse de la notion de « { code source } », qui mobilise au moins deux imaginaires fondateurs : la source et la commande. Ces deux imaginaires sont problématiques parce qu'ils participent de ce que nous avons appelé avec Wendy Chun Hui Kyong d'un « ensourcellement ». L'imaginaire de la source tout d'abord, qui repose sur l'idée que le pouvoir se trouve avant, en amont. Cette conception pose deux problèmes. Premièrement, elle tend à masquer la pluralité des textes engagés (il n'y a pas **un** { code } source de tout ce que l'on voit à l'écran). Deuxièmement, elle nourrit la tentation de « remonter » dans les couches d'écritures afin de déterminer où se trouve le pouvoir. Or, il n'y a pas de { code } qui soit plus « puissant » ou plus « source » qu'un autre. Il y a des textes, qui sont des façons de représenter un { calcul } et donc les relations entre humains et machines. Chaque { code } donne à lire une certaine organisation de ces relations.

Second imaginaire : celui de la commande. Il s'appuie sur une propriété technique du { code } : c'est une écriture qui fait agir une machine. Mais là encore, il a fallu examiner avec attention ce que cela dit de nos rapports aux machines, en posant comme hypothèse que l'imaginaire de la commande est lié à un autre imaginaire : le fantasme d'une écriture qui soit le décalque de la volonté d'un sujet pensant. Un retour à Illich nous a appris que cette conception de l'écriture naît au XII^e^ siècle avec le texte livresque. Nous avons ensuite soutenu que la scripturisation de la { programmation }, engagée par le paradigme de Von Neumann, s'appuie sur cette conception scolastique de l'écriture : le { programme } informatique est la consignation de l'*ordinatio* d'une pensée et non la négociation avec du { matériel } informatique (câbles, unités de { calcul }, etc.). Cette conception de la { programmation } se couple avec les propriétés technique de l'écriture informatique pour produire ce que nous avons

algorithme,
***bug*,**
calcul,
code,
écriture-calcul,
énonciation computationnelle,
industrialisation,
irréductibilité computationnelle,
programmation,
thélémacentrisme

Voir glossaire tome II, p. 3-25

appelé le { thélémacentrisme } : le { code informatique } ne serait pas uniquement, en tant qu'écriture, la consignation d'une pensée. Ce serait également la consignation d'une volonté, au sens où cette écriture fait faire : elle déclenche les actions d'un ordinateur. D'où la reformulation du logocentrisme en { thélémacentrisme } : ce qui est en jeu dans l'imaginaire de la commande, c'est l'idée que la machine ne fait qu'exécuter les volontés d'êtres humains. Ce primat de la volonté produit des effets sur les relations que nous tissons aux machines : elles n'ont plus aucune place, plus aucune marge de manœuvre. Ce que nous avons montré à travers l'exemple du traitement des { algorithmes } en sciences humaines. Si, effectivement, tout { algorithme } est écrit par des humains et donc n'est pas neutre mais le relais de conceptions idéologiques situées, il n'en reste pas moins qu'il y a quelque chose dans l'{ algorithme } qui échappe à ses concepteurs. On ne peut donc pas dire qu'il n'y a « que » des humains. Il y en a beaucoup, certes, mais il y a aussi une part proprement machinique dans les { algorithmes } et plus largement dans tout ce qui est produit par une machine fonctionnant sur de l'{ écriture-calcul }.

Il a fallu donc essayer de qualifier ce qu'est cette « part machinique » et comment on peut la prendre en compte dans une approche sémiotique. Partant de l'idée qu'il fallait faire preuve de « tact ontologique », nous avons proposé l'idée que ce tact revient, en sémiotique, à partir de la polyphonie énonciative de tout texte et plus précisément encore de son « hétérophonie[975] ». Un texte est le produit de plusieurs modes d'expression, certains humains, d'autres pas, tous participant à son énonciation. Or, chaque mode d'expression possède une « irréductibilité sémiotique » : il apparaît d'une certaine manière dans le texte lisible.

Voici posée les bases théoriques d'une reconnaissance de « l'{ irréductibilité computationnelle } » : le mode d'expression propre à tout texte numérique en tant qu'il est le produit du { calcul }. Restait à qualifier positivement cette { irréductibilité computationnelle } : qu'est-ce que le { calcul } en tant que mode d'expression ? Premièrement, il est une exploration systématique des possibles d'un ensemble combinatoire. C'est ce qui définit l'{ énonciation computationnelle }. Deuxièmement, cette exploration se fait automatiquement, c'est-à-dire par une machine qui change d'elle-même son état interne. C'est l'agir machinique – un agir automatique mais non pas autonome. Troisièmement, le { calcul } comme

975 Cormerais, Franck et Gilbert, Jacques Athanase. Entretien avec Emmanuël Souchier. *Op. cit.*, p. 189-214.

mode d'expression est la spécification d'une énonciation combinatoire plus large, qui se caractérise par le fait qu'un système combinatoire propose à partir d'une liste réduite d'éléments de base un grand nombre de possibilités. Le { calcul } est une façon d'actualiser l'ensemble de ces possibles par le biais d'une écriture automatique.

Comment cette { énonciation computationnelle } apparaît-elle à l'écran ? En d'autres termes, quelle est l'irréductibilité sémiotique du { calcul } ? Au vu du travail mené jusqu'ici, nous définissons l'{ irréductibilité computationnelle } comme le fait que le texte que nous lisons est l'un des possibles combinatoires retenu par les entités qui président à son { calcul }. Il peut donc toujours être autre, on peut passer d'une combinaison à une autre. Cette ultime question n'a toutefois pas trouvé de réponse définitive, tant il s'avère que les textes numériques que nous lisons ordinairement ne permettent pas de voir cette { énonciation computationnelle }. D'où une dernière hypothèse, qui restera en suspens : les effets conjugués du { thélémacentrisme }, du paradigme logique de la { programmation } et de l'{ industrialisation } contemporaine de l'écriture relèguent les marques du { calcul } dans les marges symboliques de nos lectures ordinaires. { *Bugs* }, erreurs en tout genre… Ces objets négligés et bien souvent voués à disparaître, à être « corrigés », seraient là où apparaîtrait le mieux – entendre dans ce qu'elle a de propre – cette présence que nous interrogions en commençant cette thèse.

Lignes de force

Ce parcours une fois retracé, il en ressort quelques grands thèmes, des « lignes de force » qui traversent et structurent l'ensemble de notre propos. Nous en proposons quatre, cette liste n'étant bien sûr pas exhaustive mais nous semble synthétiser au mieux notre travail.

La première de ces lignes de force, c'est la question de l'abstraction permise par l'écriture. Par abstraction, nous n'entendons pas seulement le fait que l'écriture permet d'isoler dans un flux de parole une séquence de lettres ou de mots. Nous entendons également l'abstraction par rapport à un support matériel, comme c'est le cas pour le « texte livresque[976] » : le texte est abstrait de son support, ce dernier prenant la fonction métaphorique de port d'attache, de lieu provisoire d'apparition. L'abstraction est donc une opération intellectuelle qui permet d'isoler, de manipuler et de réincarner dans un second temps les unités ainsi abstraites. Cette conception abstraite du texte sous-tend la conception modulaire des { API } que nous avons étudiée dans le chapitre III ; elle sous-tend également le développement des { *subroutines* } dès le début de la { programmation } : on isole au sein d'un { programme } informatique des séquences de { calcul } qu'on va ensuite remobiliser et adapter selon les contextes. Elle sous-tend enfin l'invisibilisation des { entités computationnelles } étudiée dans notre dernier chapitre : si le texte est un objet intellectuel qui peut s'abstraire des conditions matérielles de son apparition, le champ est ouvert à une dévaluation symbolique des acteurs qui permettent cette existence matérielle.

API,
architecture de Von Neumann,
architexte,
calcul,
code,
écriture-calcul,
entités computationnelle
formalisme,
industrialisation,
petites formes,
programmation,
programme,
subroutine

Voir glossaire tome II, p. 3-25

Nous attirons toutefois l'attention sur un point important : toute écriture n'est pas abstraite. Il ne s'agit pas de dire, par déterminisme technique, que l'écriture amène nécessairement à élaborer des procédures intellectuelles d'abstraction. Nous disons plutôt que l'abstraction est le propre de certains systèmes d'écriture, ceux que Pierre Déléage nomme les « écritures intégrales détachées » : des écritures qui, par leurs propriétés sémiotiques, permettent de « transcrire n'importe quel discours » et donc de « [...] se passer entièrement du recours à la mémoire et à la transmission orales[977] ». L'écriture peut ainsi se « détacher » – s'abstraire – de son contexte d'énonciation. En ce sens, l'abstraction est l'une des conséquences possibles d'une écriture intégrale.

976 Illich, Ivan. *Op. cit.*

977 Déléage, Pierre. *Op.cit.*, p. 179.

Si Pierre Déléage maintient un lien quasi-indéfectible entre écriture et discours, parce qu'il s'intéresse aux naissances des systèmes d'écriture et à leur contexte institutionnel, nous ne saurions discuter de ce lien ici. Il est en revanche désormais clair avec cette thèse que l'informatique est une forme d'écriture détachée intégrale, à deux titres. Par sa constitution tout d'abord : l'{ écriture-calcul } est une écriture détachée, hautement détachée même puisque c'est une de ses conditions de fonctionnement. Sans abstraction { formelle }, sans « idéalité computationnelle[978] », pas de { calcul } possible. Dans ses usages ensuite : elle hérite de pratiques d'écriture et de conceptions du textes issues d'une autre écriture détachée intégrale : l'alphabet vocalique. Peut-on alors considérer l'informatique comme une cinquième écriture détachée intégrale[979] ? La rupture sémiotique et sémantique induite par le { calcul } plaide en faveur de cette thèse. Mais il serait imprudent de trancher cette question ici. Soulignons plutôt que l'informatique comme technique d'écriture peut être considérée comme la dernière mutation de l'écriture intégrale occidentale une fois cette dernière construite autour de l'alphabet vocalique. Cette évolution se joue autour de la séquence chronologique suivante : la conception platonicienne de l'écriture comme modèle d'une méthode analytico-combinatoire ; Hugues de Saint-Victor et le passage du texte monastique au texte scolastique au XII^e siècle ; le { formalisme } hilbertien et sa conception du { calcul } comme écriture ; l'automatisation de cette écriture par Turing en 1937 ; l'adoption de l'{ architecture de Von Neumann } à la fin des années 1940 qui consacre la { programmation } comme écriture et le { code } comme texte. L'abstraction de – et par – l'écriture est le fil rouge de toutes ces étapes, que nous retraçons ici à grand traits.

Deuxième ligne de force de cette thèse : l'{ industrialisation } des pratiques lettrées. L'approche par les « { petites formes } » et les { architextes } nous avait déjà situé dans un contexte d'{ industrialisation } de l'écriture. Ce contexte est déjà largement étudié par les sémioticiens des écrits d'écran[980]. Notre focale sur l'éditorialisation, soit la construction de la visibilité et de la lisibilité des textes, a déplacé le regard. Ce qui s'industrialise, ce n'est pas seulement l'écriture. C'est aussi les pratiques lettrées, soit toutes les pratiques de contrôle du texte, tant dans son aspect formel que dans

978 Bachimont, Bruno. Pour une critique phénoménologique de la raison computationnelle. *Op. cit.*

979 Selon Pierre Déléage, seulement quatre écritures détachées ont vues le jour : l'écriture sumérienne ; l'écriture maya ; l'écriture hiéroglyphique égyptienne ; l'écriture chinoise.

980 Gomez-Mejia, Gustavo. *Op. cit.* ; Davallon, Jean. *Op. cit.*

son contenu lingustique[981]. Ce contrôle se fait en fonction de normes culturelles, culturelles, politiques, techniques… Dimensions qui ont été remarquées dans les { API } web et étudiées dans cette thèse.

API, architexte, binaire, code, code source développeur, entités computationnelle, HTML, industrialisation, interface graphique, petites formes, programmation

Voir glossaire tome II, p. 3-25

Les { API } sont donc un lieu d'{ industrialisation } de pratiques lettrées. Cela signifie qu'elles standardisent et outillent des façons d'éditer le texte, de le rendre lisible en fonction d'une certaine culture combinatoire de l'écriture. Cela signifie également qu'une écriture savante, habituellement dévolue aux spécialistes – la { programmation } – s'étend par le biais des { API } à des échelles nouvelles. Mais il ne s'agit pas seulement d'une hybridation entre culture lettrée et culture informatique, car cela serait trop vite opposer les deux. Avec les { API }, c'est **l'informatique comme culture du texte qui rentre dans le champ des écritures ordinaires**. Les { API } proposent en effet aux { développeurs }, mais également aux internautes novices, de manipuler du { code informatique } et d'en comprendre l'effet. Une certaine façon – combinatoire et modulaire – de penser le texte se trouve par là industrialisée. C'est un acquis important de cette thèse et un déplacement par rapport aux travaux réalisés jusqu'ici en sémiotique des écrits d'écran. Alors que la plupart notent, à juste titre, que le numérique textualise les pratiques d'écriture en reprenant et en intégrant des cultures savantes plus anciennes de l'écriture (typographie, journalisme, etc.), la donne est un peu différente avec les { API }. C'est la { programmation } – une pratique lettrée[982] parmi d'autres – qui se voit proposée au plus grand nombre. C'est donc une pratique d'écriture consubstantielle à l'informatique qui investit le champ des écritures ordinaires.

Troisième ligne de force : notre thèse est faite de passages incessants entre technique et sémiotique, entre { code } et { interface graphique }, entre raison computationnelle et raison graphique. Cette dynamique est constitutive de notre problématique, entre { code } et formes du texte. Elle est également le fruit de notre méthodologie et de la constitution de notre corpus, fait de captures d'écran de { petites formes } assorties du { code } { HTML } correspondant. Ce mouvement, cette dynamique d'allers-retours traduit selon nous l'hybridation au cœur de nos objets : ce sont des outils de { programmation }, mais aussi des outils d'édition ; ce sont des { architextes } dont la fonction est de faire le pont entre le technique et le symbolique[983]… De tels objets appellent ces allers-retours, par leur fonc-

981 Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *op. cit.*, p. 12.

982 Knuth, Donald E. Literate Programming. *Op.cit.*, p. 97-111.

983 Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Op. cit.*, p. 102-103.

tion, leurs usages et leur histoire. Quels résultats ont produits ces allers-retours ? Premièrement, le texte s'est avéré un concept opérant pour penser ces passages, à condition de faire attention à certaines qualifications pouvant induire des relations problématiques aux { entités computationnelles }, en premier lieu le terme de « { code source } ». Deuxièmement, ces passages ne se sont pas faits sans transformations, sans changements de forme. Nous avons notamment remarqué une « inflation scripturaire »à chaque changement d'échelle : plus le texte est représenté sous une forme informatique, discrétisée et manipulable, plus le volume des écrits augmente. On comprend alors mieux la nécessité technique et sociale des architextes : en prenant en charge la gestion des couches d'écritures les plus complexes – donc les plus volumineuses –, ils facilitent la production des textes numériques. Ce ne sont donc pas seulement des opérateurs de conversion. Ce sont également des opérateurs d'abréviation, de condensation de l'écriture[984].

Quatrième ligne de force enfin : la polyphonie énonciative et plus largement la question des composites[985]. Ce que nous avons cherché à faire sentir tout au long de ces pages, c'est le caractère hybride, pluriel des phénomènes étudiés. C'est pourquoi nous avons fréquemment eu recours à l'histoire tout au long de notre démonstration. En plus de situer les phénomènes étudiés dans une histoire longue du texte et de l'écriture, cela a permis de faire émerger cette pluralité. Les { API } ne sont pas qu'un phénomène technique, mais font appel à des conceptions anciennes du texte ; l'informatique n'est pas seulement le produit d'une logique rationnelle. Ses racines plongent dans des projets scientifiques et religieux, dans une histoire des sciences où des pratiques qu'on jugeraient aujourd'hui magiques ne sont jamais loin[986]. De la même façon, le { binaire } et la combinatoire ne se résument pas à une mécanique implacable mais ouvrent à des imaginaires religieux de l'écriture et à des rapports mystiques aux chiffres. Il nous semblait intéressant de montrer que l'informatique, qu'on pourrait considérer au premier abord comme l'aboutissement d'une certaine rationnalité occidentale, n'est pas si « évidemment » rationnelle. Envisagée du point de vue d'une technique combinatoire aux prétentions universelles, l'informatique rejoint effectivement un courant majeur de la

984 Cette remarque confirme par ailleurs que la « conversion » est avant tout une transformation,un changement de forme.

985 Le Marec, Joëlle. *Op. cit.*

986 C'est tout le mérite de Yates et de Rossi d'avoir montré à quel point la science moderne se construit sur des cadres de pensée qui ont ensuite été exclus du champ de la pensée « scientifique » autorisée. Voir Yates, Frances A. Op. cit., 2014 [1966] ; Yates, Frances A. *Op. cit.*, 1999 ; Rossi, Paolo. *Op. cit.,* 1993.

science occidentale, mais dont les enjeux et présupposés sont politiques et religieux, avant d'être scientifiques[987]. Enfin, cette quête de la pluralité apparaît de la façon la plus claire dans le dernier chapitre de notre thèse, où nous avons montré que, dans les { API } et plus largement dans les écrits d'écran, il n'y a pas que des humains, comme dans tout texte par ailleurs. Il fallait donc proposer des pistes épistémologiques pour développer une sémiotique qui soit à même de reconnaître cette part non-humaine dans les textes numériques.

API,
architexte,
calcul,
code,
discrétisation,
énonciation
computationnelle
formalisme,
HTML

Voir glossaire tome II, p. 3-25

Apports dans le champ scientifique

Qu'apporte cette thèse aux Sciences de l'Information et de la Communication et plus spécifiquement aux champs scientifiques dans lesquels nous nous étions positionnés ? Les quelques paragraphes ci-dessus ont déjà donné quelques réponses à cette question, mais nous souhaitons ici faire un examen plus systématique des apports notre travail. Ils sont au nombre de quatre.

Tout d'abord, nous avons analysé le { code informatique } depuis la perspective de la sémiotique des écrits d'écran. Si la notion d'{ architexte } semble tout à fait adaptée pour qualifier du { code informatique }, cela n'avait pas été fait à notre connaissance. Bien que cette sémiotique soit historiquement issue de travaux sur le { code informatique[988] }, son développement scientifique et institutionnel n'a pas débouché sur des travaux prenant à bras le corps ce type d'écriture. Notre thèse est donc un apport important à ce champ des SIC, à deux titres. Premièrement parce que nous avons pris comme objet du { code } { HTML }. Deuxièmement parce que nous avons proposé une méthodologie pour aborder ce { code } en déjouant certaines idéologies de la machine qui lui sont associées et en faisant un retour critique sur la notion d'{ architexte }, prolongeant certaines remarques adressées par ailleurs sur « l'anthropocentrisme » de ce concept[989].

987 En cela nous rejoignons et précisons les analyses de Milad Doueihi : le numérique a repris à la religion ses prétentions universalistes. Doueihi, Milad. *Op. cit.*, 2008, p. 23 ; Doueihi, Milad. *Op. cit.*, 2011, p. 41.

988 Souchier, Emmanuël et Pomian, Joanna. Informatique et pratiques écrivantes. *Op. cit.*, p. 121-130.

989 Collomb, Cléo. *Op. cit.*, 2016, p. 285-286. Notre argument est un peu différent : nous remarquons simplement que la formulation du concept d'architexte et sa racine grecque d'*archè*, parce qu'elle mêle indissociablement pouvoir et origine, reconduit un ensourcellement du code informatique. Il nous paraît préférable, pour éviter l'écueil possible d'une remontée à l'infini des couches architextuelles, de passer par la théorie de l'énonciation éditoriale pour comprendre le pouvoir précis d'un code informatique.

Le deuxième apport de notre travail concerne l'humanisme numérique. Il peut sembler paradoxal au premier abord de continuer à se réclamer d'un quelconque humanisme alors que nous avons consacré tout le dernier chapitre de notre thèse à élaborer une sémiotique non-anthropocentrée des médias informatisés. Ce serait confondre trop rapidement humanisme et anthropocentrisme. L'humanisme numérique n'est pas une manière de réaffirmer le primat de l'humain sur la technique. C'est une tentative de prendre en compte le fait qu'une technique – le numérique – transforme profondément les pratiques, objets et valeurs sur lesquels s'est bâtie la culture occidentale. De ce point de vue, l'humanisme numérique n'est pas incompatible avec une réévaluation des méthodes (la sémiotique) qui ont servies jusque à maintenant à analyser des textes, bien au contraire. De plus, Milad Doueihi propose l'idée que le numérique est un « sacre de l'hybride[990] », ce dont notre thèse est une preuve par l'exemple. Les { API } sont un lieu d'hybridation entre culture classique et culture informatique du texte ; elles sont également des outils qui produisent des textes nous mettant, nous humains, en présence d'entités étrangères. L'hybridation est donc double.

Notre thèse constitue enfin un apport à la théorie du support telle que développée par Bruno Bachimont[991]. Notre travail est en effet un cas d'analyse des effets de la { discrétisation } du { calcul } – propre à la raison computationnelle[992] – sur les formes des textes de réseau. Mais il ne s'agit pas que de cela, car la nouveauté de notre approche tient en deux points. Premièrement, notre focale sur la combinatoire a permis de préciser certains attendus de cette « raison computationnelle ». Car si le { calcul } { formel } produit en effet un nouveau type de rationnalité, encore faut-il préciser quelles formes prend cette rationnalité, ce qu'elle produit précisément. Selon nous, ces effets sont en partie dus à la combinatoire, qui produit du côté des imaginaires une mystique de l'épuisement ; du côté des formes sémiotiques une { énonciation computationnelle } entendue comme l'actualisation d'un des possibles combinatoires parcourus par le { calcul }. La seconde nouveauté apportée par cette thèse à la théorie du support, c'est que nous avons cherché en permanence à l'articuler avec la théorie des écrits d'écran. Entre technique et sémiotique, entre formes et outils d'écriture… et donc entre raison computationnelle et raison graphique, nous n'avons eu de cesse de passer d'un

990 Doueihi, Milad. *Op. cit.*, p. 11-16.

991 Bachimont, Bruno. *Op. cit.*, 2010..

992 Bachimont, Bruno. Intelligence artificielle et écriture dynamique: de la raison graphique à la raison computationnelle. Dans : Fabbri, Paolo et Petitot, Jean (dir.), op. cit.. Paris : Grasset, 2000 ; Bachimont, Bruno. *Op. cit.*, 2004.

niveau à un autre. Ce faisant, nous nous sommes situé aux seuils respectifs de l'une (la sémiotique des écrits d'écran) et de l'autre (la théorie du support). Là où la première s'arrête bien souvent – le { code informatique } – nous avons analysé dans le détail son rôle éditorial ; Quand la seconde passe en général rapidement sur la vie sociale des signes et leur « orthotéticité[993] », nous avons montré la façon dont la { discrétisation } et le { formalisme } induits par le numérique participent aujourd'hui d'une { industrialisation } de l'écriture et des pratiques lettrés.

architecture de Von Neumann,
***bug*,**
calcul,
code,
discrétisation,
énonciation computationnelle
formalisme,
industrialisation,
logiciel,
matériel,
programmation,
programme,
technologie de l'intellect / intellectuelle
thélémacentrisme

Voir glossaire tome II, p. 3-25

Chemins qui ne mènent nulle part

Il nous reste désormais à indiquer quels sont, parmi les chemins arpentés dans cette thèse, ceux qu'il nous plairait d'explorer. Ces chemins sont, comme le dit la belle expression allemande, des « chemins de bois » (*holzwege*[994]) : des chemins tracés par les bûcherons qui, à partir d'un point donné de la fôret, tracent un chemin, non pas pour aller quelque part, mais simplement pour continuer leur travail, avant de revenir à leur point de départ. Voici donc quels seraient nos « chemins de bois », cette liste étant là encore non-exhaustive : on pourrait toujours tracer un nouveau chemin…

Premier chemin, le plus évident car celui sur lequel nous avons passé le plus du temps : construire une sémiotique du { *bug* }, dans le prolongement de notre dernier chapitre. Plus largement, il serait opportun, à partir des bases proposées dans cette thèse, de s'interroger sur les modalités de textualisation de l'activité computationnelle. On pourrait ainsi affiner l'hypothèse de l'invisibilisation systématique de l'{ énonciation computationnelle } et mieux comprendre comment les textes computationnels donnent à lire les machines, leur activité et la part qu'elles jouent dans l'énonciation.

Deuxième chemin, en complément du premier : réinterroger le { thélémacentrisme } en y réintroduisant la matière et son imaginaire. Nous avons montré dans cette thèse comment l'aspect de commande du { code informatique } nourrit le fantasme d'une écriture qui soit consignation de la volonté : la machine nous obéirait intégralement, elle ne serait que servante docile. Nous aurions certainement tout intérêt à relier cet acquis à la question de la matérialité, car du point de vue imaginaire, la matière

993 Bachimont, Bruno. Intelligence artificielle et écriture dynamique: de la raison graphique à la raison computationnelle. Dans : Fabbri, Paolo et Petitot, Jean (dir.), *Op. cit*.

994 Heidegger, Martin. *Chemins qui ne mènent nulle part*. Paris Gallimard, 2006 [1962].

est liée à la volonté : « [...] les rêveries matérielles changent la dimension de nos puissances ; elles nous donnent des impressions démiurgiques ; elles nous donnent des illusions de la toute-puissance[995] ». Mais comme le souligne Bachelard, il s'agit bien là d'une illusion : la confrontation avec la matière, son travail par l'outil émousse cette illusion et affine un autre type de volonté, cette fois débarassée du fantasme de toute-puissance. Le { thélémacentrisme } pourrait être le résultat d'une telle « rêverie matérielle » poussée à son paroxysme : en négligeant le { matériel } – le *hardware*, ce qui est *dur* –, on entretient l'imaginaire d'une volonté toute puissante, aux prises avec une machine plastique – *soft* – qui s'adapte à tous les contextes. Il faudrait donc mener une enquête croisant les imaginaires de la matière informatique (souplesse du { logiciel }, dureté du { calcul }), les modalités d'écriture que cela permet (adaptabilité, plasticité…) et la façon dont cela construit notre rapport entre un ordinateur et ce que nous voulons qu'il fasse pour nous.

Troisième chemin, cette fois historique : nous aurions intérêt à approfondir, d'un point de vue de l'histoire de l'informatique et de l'écriture, le passage à l'{ architecture de Von Neumann }. Il nous semble en effet que la scripturisation de la { programmation } est un moment important dans l'histoire de l'informatique, car elle implique un tout autre rapport à la machine et radicalise le mouvement d'abstraction que nous avons soulevé dans cette thèse… C'est en somme interroger le moment où le { programme } devient un texte, car c'est un moment charnière où non seulement l'équilibre interne de l'informatique bascule vers une plus grande place de la logique { formelle } ; mais également un moment où la { programmation } devient une pratique d'écriture, profitant alors de tous les avantages de cette { technologie intellectuelle }.

995 Bachelard, Gaston. La terre et les rêveries de la volonté. Paris : Librairie José Corti, 1948, p. 24.

BIBLIOGRAPHIE

Cette bibliographie suit la norme ISO 690 – AFNOR Z 44-0051.

Nous avons distingué sept catégories : les ouvrages, articles de revues et carnets de recherche ; les encyclopédies ou articles d'encyclopédie ; les rapports techniques ou ministériels ; les articles de presse ; les diverses références web (sites marchands, blogs personnels, etc.) ; les articles de loi, diaporamas ou correspondances ; les captations vidéos. Lorsque nous avons consulté des rééditions, la date de la première édition (en français, dans le cas d'un livre traduit) est indiquée entre crochets, immédiatement après la date de réédition.

Pour les ouvrages ou les articles de revues publiés directement en version web – sans version papier disponible – nous avons placé la mention [En ligne] immédiatement après le titre. Dans le cas d'une numérisation d'un imprimé, [En ligne] est placé après la référence complète. Dans le cas d'une republication d'un imprimé en version électronique, les deux ISBN ou ISSN sont indiqués. La mention [manuscrit en ligne] signifie que la version consultée est en prépublication ou n'est pas la version publiée référencée dans la bibliographie.

Toutes les références en ligne sont en libre accès. Elles ont toutes, sauf une, été consultées au 1er septembre 2017.

1 Voir la synthèse de cette norme faite par Véronique Pierre. La norme ISO 690 (Z 44-005). 2007. [En ligne] [mis en ligne le 23 mai 2007] [dernière modification le 14 avril 2009] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://revues.refer.org/telechargement/fiche-bibliographie.pdf.

Ouvrages, articles de revues, carnets de recherche

A

Abramson, Bruce. Promoting Innovation in the Software Industry: A First Principles Approach to Intellectual Property Reform. *Boston University Journal of Science and Technology Law*. 2002, vol. 81, p. 75-156. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.bu.edu/law/journals-archive/scitech/volume81/abramson.pdf.

Akrich, Madeleine. Comment décrire les objets techniques ? *Techniques & Culture*. 1987, nº 9, p. 49-64. Republication en ligne : *Techniques & Culture*. 2010, nº 54-55. [Mis en ligne le 30 janvier 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://tc.revues.org/4999. DOI : 10.4000/tc.4999. ISSN : 1952-420X.

Akrich, Madeleine, Callon, Michel et Latour, Bruno. *Sociologie de la traduction : textes fondateurs*. Paris : Presses de l'École des Mines, 2006. ISBN : 978-2-91176-275-8.

Allard, Laurence. L'impossible politique des communautés à l'âge de l'expressivisme digital. *Cahiers Sens public*. 2010, nº 7-8, p. 105-126. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.cairn.info/revue-cahiers-sens-public-2008-3-page-105.htm. ISSN : 1775-4356.

Angé, Caroline. *La question du sens : écrire et lire le fragment : du texte à l'hypertexte*. Thèse de doctorat. Paris : Université Paris 13, 2005.

Angé, Caroline. Le fragment comme forme texte : à propos de *Fragments d'un discours amoureux*. *Communication & langages*. 2007, nº 152, p. 23-34. [En ligne] [mis en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/colan_0336-1500_2007_num_152_1_4653. DOI : 10.3406/colan.2007.4653. ISSN : 1778-7459.

Angé, Caroline (dir). Empreintes de l'hypertexte. Rétrospective et évolution. Cachan : Lavoisier, 2011. *Les Cahiers du numérique*, vol. 7, nº 3-4,. ISBN : 978-2-74623-977-7. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-les-cahiers-du-numerique-2011-3.htm. ISSN : 1622-1494.

Aspray, William. *John Von Neumann and the Origins of Modern Computing*. Cambridge : MIT Press, 1990. ISBN : 978-0-26201-121-1.

Auroux, Sylvain. *La révolution technologique de la grammatisation. Introduction à l'histoire des sciences du langage*. Liège : Mardaga, 1994. ISBN : 978-2-87009-565-2.

B

Bachelard, Gaston. *La terre et les rêveries de la volonté.* Paris : Librairie José Corti, 1948.

Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Technologies, Idéologies, Pratiques*. 1999, nº 4, p. 195-225. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.utc.fr/~bachimon/Publications_attachments/Hypotexte.pdf.

Bachimont, Bruno. Intelligence artificielle et écriture dynamique : de la raison graphique à la raison computationnelle. Dans : Fabbri, Paolo et Petitot, Jean (dir.), *Au nom du sens. Autour de l'œuvre d'Umberto Eco*. Paris : Grasset, 2000, [s-p]. ISBN : 978-2-24653-761-8. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.utc.fr/~bachimon/Publications_attachments/BachimontCerisy1996.pdf.

Bachimont, Bruno. *Arts et sciences du numérique. Ingénierie des connaissances et critique de la raison computationnelle*. Habilitation à diriger des recherches. Compiègne : Université de Technologie de Compiègne, 2004. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.utc.fr/~bachimon/Livresettheses_attachments/HabilitationBB.pdf.

Bachimont, Bruno. Nouvelles tendances applicatives : de l'indexation à l'éditorialisation. Dans : Gros, Patrick (dir.), *L'indexation multimédia. Description et recherche automatiques.* Cachan : Hermès Sciences – Lavoisier, 2007, p. 313-326. ISBN : 978-2-74621-492-7. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://cours.ebsi.umontreal.ca/sci6116/Ressources_files/BachimontFormatHerme%C-C%80s.pdf.

Bachimont, Bruno. *Le sens de la technique : le numérique et le calcul*. Paris : Les Belles Lettres, 2010. ISBN : 978-2-35088-035-8.

BACHIMONT, Bruno. Pour une critique phénoménologique de la raison computationnelle. *E-dossiers de l'audiovisuel* [En ligne]. 2012. [Mis en ligne en janvier 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.ina-expert.com/e-dossier-de-l-audiovisuel-l-education-aux-cultures-de-l-information/pour-une-critique-phenomenologique-de-la-raison-computationnelle.html.

BARDINI, Thierry, BROYE, Lionel, CITTON, Yves, GALLIGO, Igor, GUEZ, Emmanuel, GUESS, Jeff, JULIEN, Quentin, KRZYWKOWSKI, Isabelle, LECHNER, Marie, MASURE, Anthony, PANDELAKIS, Pia, THIBAULT, Ghislain et VARGOZ, Frédérique. *Manifeste médiarchéologiste* [En ligne]. [Mis en ligne le 5 octobre 2016] [dernière modification le 19 avril 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://pamal.org/wiki/Manifeste_M%C3%A9diarch%C3%A9ologiste.

BARTHES, Roland. Éléments de sémiologie. *Communications*. 1964, nº 4, p. 91-135. [En ligne] [mis en ligne le 21 mars 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/comm_0588-8018_1964_num_4_1_1029. DOI : 10.3406/comm.1964.1029. ISSN : 2102-5924.

BÉGUIN-VERBRUGGE, Annette. *Images en texte, images du texte : dispositifs graphiques et communication écrite*. Villeneuve-d'Ascq : Presses Universitaires du Septentrion, 2006. ISBN : 978-2-85939-938-2.

BEINEKER, Lowell et WILSON, Robin. Modern graph theory. Dans : WILSON, Robin et WATKINS, John J. (dir.), *Combinatorics: Ancient and Modern*. Oxford : Oxford University Press, 2013, p. 331-352. ISBN : 978-0-19965-659-2.

BELL, Daniel. *The Coming of Post-Industrial Society: a Venture in Social Forecasting*. New York : Basic Books, 1973. ISBN : 978-0-46501-281-7.

BENSLIMANE, Djamal, DUSTDAR, Schahram et SHETH, Amit P. Services Mashups: The New Generation of Web Applications. *IEEE Internet Computing*. 2008, vol. 12, nº 5, p. 13-15. [En ligne] [mis en ligne en octobre 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://works.bepress.com/amit_sheth/292/. DOI : 10.1109/MIC.2008.110.

BENVENISTE, Émile. *Le vocabulaire des institutions indo-européennes. 1 : économie, parenté, société*. Paris : Les Éditions de Minuit, 1969. ISBN : 978-2-70730-050-0.

BERRY, David M. *The philosophy of software : code and mediation in the digital age*. Basingstoke : Palgrave Macmillan, 2011. ISBN : 978-0-23024-418-4.

Billeter, Jean-François. *Leçons sur Tchouang-Tseu*. Paris : Allia, 2015 [2002]. ISBN : 978-2-84485-793-4.

Blanchard, Gérard. Textes, images et lieux de mémoire. *Communication & langages*. 1975, nº 28, p. 45-69. [En ligne] [mis en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/colan_0336-1500_1975_num_28_1_4249. DOI : 10.3406/colan.1975.4249. ISSN : 1778-7459.

Blanchard, Gérard. Informations : Rencontres de Lure / en télématique. *Communication & langages*. 1981, nº 49, p. 114. [En ligne] [mis en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : www.persee.fr/doc/colan_0336-1500_1981_num_49_1_1446. ISSN : 1778-7459.

Bodle, Robert. Regimes of Sharing. *Information, Communication & Society*. 2011, vol. 14, nº 3, p. 320-337.

Bolter, David J. *Writing space: the computer, hypertext and the history of writing*. Hillsdale : L. Erlbaum Associates, 2001 [1991]. ISBN : 978-0-80582-918-1.

Borel, Henri. *Wu Wei : Le Non-Agir d'après Lao Tse et le Taoïsme*. Traduit du néérlandais par Pierre Bernard. Nice : Nataraj, 2016 [1913]. ISBN : 978-2-91146-623-6.

Bottéro, Jean. *Mésopotamie : l'écriture, la raison et les dieux*. Paris : Gallimard, 1987 [1984]. ISBN : 978-2-07070-879-6.

Bouchardon, Serge. *La valeur heuristique de la littérature numérique*. Paris : Hermann, 2014. ISBN : 978-2-70568-802-8.

Boullier, Dominique, Ghitalla, Franck, Le Douarin, Laurence, Neau, Aurélie et Gkouskou-Giannakou, Pergia. *L'outre-lecture : manipuler, (s')approprier, interpréter le Web*. Paris : Bibliothèque Publique d'Information, 2003. ISBN : 978-2-84246-081-5. [Version électronique en ligne] [mis en ligne le 6 juin 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://books.openedition.org/bibpompidou/463. ISBN : 978-2-84246-160-7 ; DOI : 10.4000/books.bibpompidou.463.

Bourdieu, Pierre et Passeron, Jean-Claude. *La reproduction : éléments pour une théorie du système d'enseignement*. Paris : Les Éditions de Minuit, 1970. ISBN : 978-2-70730-226-7.

Bucher, Taina. Objects of Intense Feeling: The Case of the Twitter API. *Computational Culture* [En ligne]. 2013, nº 3. [Mis en ligne le 16 novembre 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api. ISSN : 2047-2390.

Bullynck, Maarten, Daylight, Edgar G. et De Mol, Liesbeth. Why did computer science make a hero out of Turing? *Communications of the ACM*. 2015, vol. 58, nº 3, p. 37-39.

C

Campbell-Kelly, Martin. Software as an economic activity. Dans : Hashagen, Ulf, Keil-Slawik, Reinhard et Norberg, Arthur L (dir.), *History of computing: software issues. International Conference on the History of Computing, ICHC 2000, April 5-7, 2000, Heinz Nixdorf MuseumsForum, Paderborn, Germany*. Berlin : Springer-Verlag Berlin Heidelberg, 2002, p. 185-202. ISBN : 978-3-54042-664-6.

Campbell-Kelly, Martin. From Theory to Practice: The Invention of Programming, 1947-51. Dans : Jones, Cliff B. et Lloyd, John L. (dir.), *Dependable and Historic Computing: Essays Dedicated to Brian Randell on the Occasion of His 75th Birthday*. Berlin : Springer Berlin Heidelberg, 2011, p. 23-37. ISBN : 978-3-64224-540-4.

Campbell-Kelly, Martin et Aspray, William. *Computer: a history of the information machine*. New York : Basic Books, 1996. ISBN : 978-0-46502-989-1.

Candel, Étienne. L'imaginaire du « portail » : le cas de Rezo.net. *Communication & langages*. 2005, nº 146, p. 19-34. [En ligne] [mis en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/colan_0336-1500_2005_num_146_1_3374. DOI : 10.3406/colan.2005.3374. ISSN : 1778-7459.

Candel, Étienne. *Autoriser une pratique, légitimer une écriture, composer une culture : les conditions de possibilité d'une critique littéraire participative sur Internet : étude éditoriale de six sites amateurs*. Thèse de doctorat. Paris : Université Paris-Sorbonne, 2007.

Candel, Étienne. *Textualiser les interfaces. Opérativité et épistémologie d'une requalification*. Habilitation à diriger des recherches. Paris : Paris-Sorbonne, 2015.

Candel, Étienne et Gomez-Mejia, Gustavo. Littératures de salon : Des « régimes sociaux » du littéraire dans les « réseaux en ligne ». Dans : Saleh, Imad, Leleu-Merviel, Sylvie, Jeanneret, Yves, Massou, Luc et Bouhai, Nasreddine (dir.), *H2PTM'09. Rétrospective et Perspective 1989 – 2009*. Cachan : Hermès Science – Lavoisier, 2009, p. 205-218. ISBN : 978-2-74622-491-9.

Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *L'économie des écritures sur le Web. Volume 1 : traces d'usage dans un corpus de sites de tourisme*. Cachan : Hermès Science – Lavoisier, 2012, p. 165-201. ISBN : 978-2-74623-284-6.

Cardon, Dominique. Dans l'esprit du PageRank. Une enquête sur l'algorithme de Google. *Réseaux*. 2013, nº 177, p. 63-95. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-reseaux-2013-1-page-63.htm. DOI : 10.3917/res.177.0063. ISSN : 1777-5809.

Cardon, Dominique (dir.). Politique des algorithmes : les métriques du web. *Réseaux* nº 177. Paris : La Découverte, 2013. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-reseaux-2013-1.htm. ISBN : 978-2-70717-548-9.

Cardon, Dominique et Casilli, Antonio A. *Qu'est-ce que le digital labor ?* Bry-sur-Marne : Institut National de l'Audiovisuel, 2015. ISBN : 978-2-86938-229-9.

Casemajor, Nathalie. Matérialisme numérique et trajectoires d'objets : les artefacts numériques en circulation. *French Journal for Media Research* [En ligne]. 2014, nº 1. [Mis en ligne le 8 janvier 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://frenchjournalformediaresearch.com/index.php?id=263. ISSN : 2264-4733.

Casilli, Antonio A. Qui a fait élire Trump ? Pas les algorithmes, mais des millions de « tâcherons du clic » sous-payés. Dans : *Antonio A. Casilli* [En ligne]. Billet du 17 novembre 2016. [Mis en ligne le 17 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.casilli.fr/2016/11/17/qui-a-fait-elire-trump-pas-les-algorithmes-mais-des-millions-de-tacherons-du-clic-sous-payes/.

CERUZZI, Paul E. *A History of Modern Computing*. Cambridge : MIT Press, 2003 [1998]. ISBN : 978-0-26253-203-4.

CHARTIER, Roger. *L'ordre des livres : lecteurs, auteurs, bibliothèques en Europe entre XIV^e et XVIII^e siècle*. Aix-en-Provence : Alinea, 1992. ISBN : 978-2-74010-024-0.

CHENG, Anne. Le corpus canonique confucéen. Dans : JACOB, Christian et GIARD, Luce (dir.), *Des Alexandries. I, Du livre au texte*. Paris : Bibliothèque Nationale de France, 2001, p. 163-177. ISBN : 978-2-71772-177-5.

CHRISTIN, Anne-Marie. *L'image écrite ou la déraison graphique*. Paris : Flammarion, 1995. ISBN : 978-2-08012-635-1.

CHUN, Wendy Hui Kyong. On "Sourcery," or Code as Fetish. *Configurations*. 2008, vol. 16, n° 3, p. 299-324.

CLÉMENT, Jean. L'hypertexte, une technologie intellectuelle à l'ère de la complexité. Dans : BROSSAUD, Claire et REBER, Bernard (dir.), *Nouvelles technologies cognitives et épistémologie*. Cachan : Hermès Science – Lavoisier, 2007, [s. p.]. ISBN : 978-2-74621-661-7.

COHEN, I. Bernard et WELCH, Gregory W (dir.). *Makin' numbers: Howard Aiken and the computer*. Cambridge : MIT Press, 1999. ISBN : 978-0-26203-263-6.

COLEMAN, Gabriella. Code Is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers. *Cultural Anthropology*. 2012, vol. 24, n° 3, p. 420-454.

COLLOMB, Cléo. *Google, une entreprise de mise en foule de la multitude*. [Manuscrit en ligne]. 2015. [Mis en ligne 11 janvier 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-01253751/document. *Working paper or preprint*. Identifiant : <hal-01253751>.

COLLOMB, Cléo. *Un concept technologique de trace numérique*. Thèse de doctorat. Compiègne : Université de Technologie de Compiègne et Bruxelles : Université Libre de Bruxelles, 2016.

Collomb, Cléo et Goyet, Samuel. Meeting the machine halfway : Vers une sémiopolitique de l'agir computationnel. Communication au colloque international *Reconfiguring Humans and Non-Humans: images, texts and beyond*. Université de Jyväskyllä, Finlande, 29-30 octobre 2015. [Manuscrit en ligne] [mis en ligne le 11 janvier 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-01253444/document. *Working paper or preprint*. Identifiant : < hal-01253444>.

Cormerais, Franck et Gilbert, Jacques Athanase. Entretien avec Emmanuël Souchier. *Études digitales : le texte à venir*. 2016, nº 1, p. 189-214.

Cotte, Dominique. Écrits de réseaux, écrits en strates. Sens, technique, logique. *Hermès*. 2004, nº 39, p. 109-115. [En ligne] [mis en ligne le 10 octobre 2007] [consulté le 1er septembre 2017]. Disponible à l'adresse : documents.irevues.inist.fr/handle/2042/9471. DOI : 10.4267/2042/9471. ISSN : 1963-1006.

Crozat, Stéphane, Bachimont, Bruno, Cailleau, Isabelle, Bouchardon, Serge et Gaillard, Ludovic. Éléments pour une théorie opérationnelle de l'écriture numérique. *Document numérique*. 2012, vol. 14, nº 3, p. 9-33. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-document-numerique-2011-3-page-9.htm. ISSN : 1963-1014.

D

Dahl, Ole-Johan, Dijkstra, Edsger W. et Hoare, Charles Anthony R. (dir.). *Structured programming*. London : Academic Press, 1972. ISBN : 978-0-12200-550-3.

Davallon, Jean. Objet concret, objet scientifique, objet de recherche. *Hermès*. 2004, nº 38, p. 30-37. [En ligne] [mis en ligne le 8 octobre 2007] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://documents.irevues.inist.fr/handle/2042/9421. DOI : 10.4267/2042/9421. ISSN : 1963-1006.

Davallon, Jean (dir.). *L'économie des écritures sur le Web, volume 1 : Traces d'usages dans un corpus de sites de tourisme*. Cachan : Hermès Science – Lavoisier, 2012. ISBN : 978-2-74623-284-6.

DAVALLON, Jean, DESPRÉS-LONNET, Marie, JEANNERET, Yves, LE MAREC, Joëlle et SOUCHIER, Emmanuël (dir.). *Lire, écrire, récrire : Objets, signes et pratiques des médias informatisés*. Paris : Éditions de la Bibliothèque Publique d'Information, 2003. ISBN : 978-2-84246-071-6. [Version électronique en ligne] [mis en ligne le 15 mai 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://books.openedition.org/bibpompidou/394. ISBN : 978-2-84246-162-1. DOI : 10.4000/books.bibpompidou.394.

DE CERTEAU, Michel. *L'invention du quotidien. 1, arts de faire*. Paris : Gallimard, 2010 [1990]. ISBN : 978-2-07032-576-4.

DÉLÉAGE, Pierre. *Inventer l'écriture : rituels prophétiques et chamaniques des Indiens d'Amerique du Nord, XVIIe-XIXe siècles*. Paris : Les Belles Lettres, 2013. ISBN : 978-2-25115-001-7.

DE MOL, Liesbeth. Generating, solving and the mathematics of Homo Sapiens. Emil Post's views on computation. Dans : ZENIL, Hector (dir.), *A Computable Universe. Understanding and Exploring Nature as Computation*. River Edge : World Scientific, 2012, p. 45-62. ISBN : 978-9-81437-429-3. [Manuscrit en ligne] [mis en ligne le 14 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://hal.univ-lille3.fr/hal-01396500/document. Identifiant : < hal-01396500>.

DE MOL, Liesbeth, CARLÉ, Martin et BULLYNCK, Maarten. Haskell before Haskell: an alternative lesson in practical logics of the ENIAC. *Journal of Logic and Computation*. 2015, vol. 25, nº 4, p. 1011-1046.

DELEUZE, Gilles. L'épuisé. Dans : BECKETT, Samuel, *Quad et autres pièces pour la télévision*. Paris : Les Éditions de Minuit, 1992, p. 57-105. ISBN : 978-2-70731-389-0.

DERRIDA, Jacques. *De la Grammatologie*. Paris : Les Éditions de Minuit, 1967. ISBN : 978-2-70730-012-6.

DESEILLIGNY, Oriane. Du journal intime au blog : quelles métamorphoses du texte ? *Communication & langages*. 2008, nº 155, p. 45-62. [En ligne] [mis en ligne le 4 mars 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : www.persee.fr/doc/colan_0336-1500_2008_num_155_1_5374. ISSN : 1778-7459.

DESPRET, Vinciane. *Au bonheur des morts : Récits de ceux qui restent*. Paris : Les Empêcheurs de penser en rond – La Découverte, 2015. ISBN : 978-2-35925-125-8.

Desrosières, Alain. *Gouverner par les nombres*. Paris : Presses de l'École des Mines, 2008. ISBN : 978-2-35671-005-5.

Detienne, Marcel. *Les maîtres de vérité dans la Grèce archaïque*. Paris : Pocket, 1995 [Maspero, 1967]. ISBN : 978-2-26606-072-4.

Dijkstra, Edsger W. Letters to the Editor: Go to Statement Considered Harmful. *Communications of the ACM*. 1968, vol. 11, nº 3, p. 147-148. [Tapuscrit en ligne]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cs.utexas.edu/users/EWD/ewd02xx/EWD215.PDF.

Dolphijn, Rick et Tuin, Iris van der (dir.). *New Materialism: Interviews & Cartographies*. Ann Arbor : Open Humanites Press, 2012. ISBN : 978-1-60785-204-9. [Version électronique en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://quod.lib.umich.edu/o/ohp/11515701.0001.001. DOI : http://dx.doi.org/10.3998/ohp.11515701.0001.001.

Doueihi, Milad. *La grande conversion numérique*. Paris : Éditions du Seuil, 2008. ISBN : 978-2-02096-490-6.

Doueihi, Milad. *Pour un humanisme numérique*. Paris : Éditions du Seuil, 2011. ISBN : 978-2-02100-089-4.

Doueihi, Milad. *Qu'est-ce que le numérique ?* Paris : Presses Universitaires de France, 2013. ISBN : 978-2-13062-718-0.

Ducrot, Oswald. Esquisse d'une théorie polyphonique de l'énonciation. Dans : Ducrot, Oswald, *Le dire et le dit*. Paris : Les Éditions de Minuit, 1984, p. 171-233. ISBN : 978-2-70731-003-3.

Dukan, Michèle. *La réglure des manuscrits hébreux au Moyen-Âge*. Paris : Editions du Centre National de la Recherche Scientifique, 1988. ISBN : 978-2-22204-100-9.

E

Eco, Umberto. *La quête d'une langue parfaite dans l'histoire de la culture européenne : leçon inaugurale faite le vendredi 2 octobre 1992 au Collège de France, chaire européenne*. Paris : Collège de France, 1992. ISBN : 978-2-72260-006-5.

Eco, Umberto. *De l'arbre au labyrinthe. Études historiques sur le signe et l'interprétation*. Traduit de l'italien par Hélène Sauvage. Paris : Grasset, 2011 [2010]. ISBN : 978-2-25315-631-4.

Ensmenger, Nathan. *The computer boys take over: computers, programmers, and the politics of technical expertise*. Cambridge : MIT Press, 2010. ISBN : 978-0-26205-093-7.

Ensmenger, Nathan. The Multiple Meanings of a Flowchart. *Information & Culture*. 2016, vol. 51, nº 3, p. 321-351.

Ertzscheid, Olivier. Un algorithme est un éditorialiste comme les autres. Dans : *affordance.info* [En ligne]. Billet du 15 novembre 2016. [Mis en ligne le 15 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.affordance.info/mon_weblog/2016/11/un-algorithme-est-un-editorialiste-comme-les-autres.html. ISSN : 2260-1856.

F

Fabbri, Paolo. Le tournant sémiotique. Cachan : Hermès Science – Lavoisier, 2008. ISBN : 978-2-74622-137-6.

Fauré, Christian. La pharmacologie des API. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *Pour en finir avec la mécroissance : quelques réflexions d'Ars Industrialis*. Paris : Flammarion, 2009, p. 244-245. ISBN : 978-2-08123-563-2.

Flanagin, Andrew J., Farinola, Wendy Jo Maynard et Metzger, Miriam J. The technical code of the internet/world wide web. *Critical Studies in Media Communication*. 2000, vol. 17, nº 4, p. 409-428.

Foucault, Michel. *L'archéologie du savoir*. Paris : Gallimard, 1969. ISBN : 978-2-07026-999-0.

Fournout, Olivier. *Théorie de la communication et éthique relationnelle : du texte au dialogue*. Cachan : Hermès Science – Lavoisier, 2012. ISBN : 978- 2-74623-296-9.

Fuller, Matthew (dir.). *Software Studies: a lexicon*. Cambridge : MIT Press, 2008. ISBN : 978-0-26206-274-9.

G

Galloway, Alexander R. *Protocol: How Control Exists after Decentralization*. Cambridge : MIT Press, 2004. ISBN : 978-0-26207-247-5.

Galloway, Alexander R. Language Wants To Be Overlooked: On Software and Ideology. *Journal of Visual Culture*. 2006, vol. 5, nº 3, p. 315-331. [En ligne] [mis en ligne le 1er décembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://journals.sagepub.com/doi/abs/10.1177/1470412906070519. DOI : 10.1177/1470412906070519. ISSN : 1741-2994.

Gardey, Delphine. *Écrire, calculer, classer : comment une révolution de papier a transformé les sociétés contemporaines (1800-1940)*. Paris : La Découverte, 2008. ISBN : 978-2-70715-367-8.

Gauthier, Richard L. et Ponto, Stephen D. *Designing Systems Programs*. Englewood Cliffs : Prentice-Hall, 1970. ISBN : 978-0-13201-962-0.

Genette, Gérard. *Seuils*. Paris : Éditions du Seuil, 1987. ISBN : 978-2-02009-525-9.

Georges, Fanny, Seilles, Antoine, Artignan, Guillaume, Arnaud, Bérenger, Rodriguez, Nancy, Sallantin, Jean et Hascoët, Mountaz. *Sémiotique et visualisation de l'identité numérique : une étude comparée de Facebook et Myspace* [manuscrit en ligne]. 2009. [Mis en ligne le 25 août 2009] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-00410952. Identifiant : < hal-00410952>. *Working paper or preprint*

Giffard, Alain. Des lectures industrielles. Dans : Stiegler, Bernard, Giffard, Alain et Fauré, Christian, *Pour en finir avec la mécroissance : quelques réflexions d'Ars Industrialis*. Paris : Flammarion, 2009, p. 117-216. ISBN : 978-2-08123-563-2.

Gillespie, Tarleton. The Relevance of Algorithms. Dans : Boczkowski, Pablo J. et Foot, Kirsten A. (dir.), *Media Technologies*. Cambridge : The MIT Press, 2014, p. 167-194. ISBN : 978-0-26252-537-4. [Manuscrit en ligne]. [Mis en ligne le 26 novembre 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.tarletongillespie.org/essays/Gillespie%20-%20The%20Relevance%20of%20Algorithms.pdf.

Glassner, Jean-Jacques. *Ecrire à Sumer : l'invention du cunéiforme*. Paris : Éditions du Seuil, 2000. ISBN : 978-2-02038-506-0.

Goëta, Samuel. *Instaurer des données, instaurer des publics : une enquête sociologique dans les coulisses de l'open data*. Thèse de doctorat. Paris : École Nationale Supérieure des Télécoms, 2016. [En ligne] [mis en ligne le lundi 6 février 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://pastel.archives-ouvertes.fr/tel-01458098. Identifiant : <tel-01458098>.

Gomez-Mejia, Gustavo. *De l'industrie culturelle aux fabriques de soi ? Enjeux identitaires des productions culturelles sur le Web contemporain*. Thèse de doctorat. Paris : Paris-Sorbonne, 2011.

Gomez-Mejia, Gustavo et Candel, Étienne. Signes passeurs et signes du web : le bouton *like*, ou les ressorts d'un clic. Dans : Barats, Christine, *Manuel d'analyse du web en sciences humaines et sociales*. Paris : Armand Colin, 2013, p. 141-146. ISBN : 978-2-20028-627-9.

Goody, Jack. *La raison graphique. La domestication de la pensée sauvage*. Traduit de l'anglais par Jean Bazin et Alban Bensa. Paris : Les Éditions de Minuit, 2007 [1979]. ISBN : 978-2-70730-240-3.

Goody, Jack. *La logique de l'écriture : aux origines des sociétés humaines*. Paris : Armand Colin, 1986. ISBN : 978-2-20036-114-3.

Goody, Jack. *Pouvoirs et savoirs de l'écrit*. Traduit de l'anglais par Claire Maniez. Paris : La Dispute, 2007. ISBN : 978-2-84303-143-4.

Goyet, Samuel. *Facebook à l'épreuve de la mort. L'écriture du deuil à travers la fonction « groupes » : le cas « hommage à Bixente Lopez »*. Mémoire de Master. Paris : Paris-Sorbonne, 2011.

Goyet, Samuel. *Google Glass au regard de son API : visions & régulations d'une innovation technique* [En ligne]. Communication au colloque international *Digital Intelligence*, Nantes, 17-19 septembre 2014. [Mis en ligne le 8 décembre 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.univ-nantes.fr/medias/fichier/paper_36___s_goyet___google_glass_1415432706656.pdf.

Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Communication +1* [En ligne]. 2016, vol. 5, nº 2. [Mis en ligne le 13 septembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://scholarworks.umass.edu/cpo/vol5/iss1/2. DOI : 10.7275/R5H70CR1.

Gras, Stéphan-Eloïse. *L'écoute en ligne : Figures du sujet écoutant et mutations des espaces musicaux sur Internet*. Thèse de doctorat. Paris : Paris-Sorbonne, 2014.

Grésillon, Almuth. *Eléments de critique génétique : lire les manuscrits modernes*. Paris : Presses Universitaires de France, 1994. ISBN : 978-2-13046-380-1.

Grignon, Thomas. L'expertise communicationnelle au prisme de ses instruments. L'exemple de Google Analytics. *Les Cahiers du RESIPROC*. 2015, nº 3, p. 23-47.

Guénon, René. *La Grande Triade*. Paris : Gallimard, 2016 [1957, 1946 pour l'édition originale]. ISBN : 978-2-07268-960-4.

Halpin, Harry. La souveraineté numérique. *Multitudes*. Traduit de l'anglais par Christophe Degoutin. 2009, nº 35, p. 201-213. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.multitudes.net/La-souverainete-numerique/. ISSN : 1777-5841.

H

Hardt, Michael et Negri, Antonio. *Empire*. Traduit de l'américain par Denis-Armand Canal. Paris : 10/18 – Exils, 2000. ISBN : 978-2-26403-877-7.

Hayles, N. Katherine. *Writing machines*. Cambridge : MIT Press, 2002. ISBN : 978-0-26258-215-5.

Heidegger, Martin. *Chemins qui ne mènent nulle part*. Traduit de l'allemand par Wolfgang Brokmeier. Paris : Gallimard, 2006 [1962]. ISBN : 978-2-07070-562-7.

HERRENSCHMIDT, Clarisse. *Les trois écritures : langue, nombre, code*. Paris : Gallimard, 2007. ISBN : 978-2-07076-025-1.

HEUGUET, Guillaume. *De la musique sur les médias informatisés : axiologies culturelles et systèmes d'usages*. Thèse de doctorat. Paris : Paris-Sorbonne, en cours.

HICKS, Marie. *Programmed inequality: how Britain discarded women technologists and lost its edge in computing*. Cambridge : MIT Press, 2017. ISBN : 978-0-26203-554-5.

HOPPER, Grace. A-2 Compiler and Associated Routines for Use with UNIVAC. Dans : ADAMS, Charles W., GILL, Stanley et COMBELIC, Donn (dir.), *Digital Computers. Advanced Coding Techniques*. Cambridge : Massachussets Institute of Technology, 1954, p. 30-40. [Tapuscrit en ligne] [mis en ligne le 1er octobre 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://archive.org/details/bitsavers_summersmputersAdvancedCodingTechniquesSummer195_5635499. Identifiant : ark:/13960/t2p56w929.

IDEL, Moshé. *Le Golem*. Traduit de l'anglais par Cyrille ASLANOFF. Paris : Les Éditions du Cerf, 1992. ISBN : 978-2-20404-583-4.

IFRAH, Georges. *Histoire universelle des chiffres : l'intelligence des hommes racontée par les nombres et le calcul*. Paris : Robert Laffont, 1994 [1981], 2 tomes. ISBN : 978-2-72428-461-5 (tome I) ; 978-2-72428-900-5 (tome II).

ILLICH, Ivan. *In the Vineyard of the Text: a Commentary to Hugh's Didascalicon*. Chicago : University of Chicago Press, 1996 [1993]. ISBN : 978-0-22637-263-5.

ILLICH, Ivan et SANDERS, Barry. *ABC : l'alphabétisation de l'esprit populaire*. Traduit de l'anglais par Maud SISSUNG. Paris – Montréal : L'Harmattan – Boréal, 1990 [1988]. ISBN : 978-2-89052-322-7.

INGOLD, Tim. *Lines: a brief history*. London : Routledge, 2007. ISBN : 978-0-20396-115-5.

J

Jacob, Christian. Fonder. Dans : Jacob, Christian et Giard, Luce (dir.), *Des Alexandries. I, Du livre au texte*. Paris : Bibliothèque Nationale de France, 2001, p. 103-114. ISBN : 978-2-71772-177-5.

Jacob, Christian. La carte des mondes lettrés. Dans : Jacob, Christian et Giard, Luce (dir.), *Des Alexandries. I, Du livre au texte*. Paris : Bibliothèque Nationale de France, 2001, p. 11-40. ISBN : 978-2-71772-177-5.

Jacob, Christian (dir.). *Lieux de savoir. 1 : Espaces et communautés*. Paris : Albin Michel, 2007. ISBN : 978-2-22617-904-3.

Jacob, Christian. *Lieux de savoir. 2 : Les mains de l'intellect*. Paris : Albin Michel, 2010. ISBN : 978-2-22618-729-1.

Jacob, Christian et Giard, Luce (dir.). *Des Alexandries. I, Du livre au texte*. Paris : Bibliothèque Nationale de France, 2001. ISBN : 978-2-71772-177-5.

Jahjah, Marc. *Les* marginalia *de lecture dans les « réseaux sociaux » du livre (2008-2014) : mutations, formes, imaginaires*. Thèse de doctorat. Paris : École des Hautes Études en Sciences Sociales, 2014.

Jahjah, Marc. De la bibliographie matérielle aux « Digital Studies » ? *Revue française des sciences de l'information et de la communication* [En ligne]. 2016, nº 8. [Mis en ligne le 30 mars 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://rfsic.revues.org/1968. DOI : 10.4000/rfsic.1968. ISSN : 2263-0856.

Jahjah, Marc. Des énoncés sans énonciateurs ? Du surlignement à la citation dans le dispositif *Kindle* d'Amazon. *Semen*. 2016, nº 41, p. 107-129.

Jeanne-Perrier, Valérie. Des outils d'écriture aux pouvoirs exorbitants ? *Réseaux*. 2006, vol. 137, nº 3, p. 97-131. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.cairn.info/revue-reseaux1-2006-3-page-97.htm DOI : 10.3917/res.137.0097. ISSN : 1777-5809.

Jeanneret, Yves. *Y a-t-il (vraiment) des technologies de l'information ?* Villeneuve d'Ascq : Presses Universitaires du Septentrion, 2007 [2000]. ISBN : 978-2-75740-019-7.

Jeanneret, Yves. *Penser la trivialité. Volume 1, la vie triviale des êtres culturels*. Cachan : Hermès Science – Lavoisier, 2008. ISBN : 978-2-74621-878-9.

Jeanneret, Yves. *Critique de la trivialité : les médiations de la communication, enjeu de pouvoir*. Paris : Éditions Non Standard, 2014. ISBN : 978-2-95428-525-2.

JEANNERET, Yves et OLLIVIER, Bruno. L'invention problématique d'un champ. *Hermès*. 2004, nº 38, p. 27-29. [En ligne] [mis en ligne le 8 octobre 2007] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://hdl.handle.net/2042/9420. DOI : 10.4267/2042/9420. ISSN : 1963-1006.

JEANNERET, Yves et SOUCHIER, Emmanuël. Pour une poétique de l'écrit d'écran. *Xoana*. 1999, nº 6, p. 97-107.

JOUËT, Josiane. Retour critique sur la sociologie des usages. *Réseaux*. 2000, vol. 18, nº 100, p. 487-521. [En ligne] [mis en ligne le 7 juin 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/reso_0751-7971_2000_num_18_100_2235. DOI : 10.3406/reso.2000.2235. ISSN : 1777-5809.

K

KIRCHER, Athanase. *Ars Magna Sciendi Sive Combinatoria*. Amsterdam : Jansonius & Waesberge, 1669. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://diglib.hab.de/drucke/6-3-quod-2f/start.htm.

KIRSCHENBAUM, Matthew G. *Track changes: a literary history of word processing*. Cambridge : Belknap Press, 2016. ISBN : 978-0-67441-707-6.

KITTLER, Friedrich. There is No Software. *CTheory: Theory, Technology, Culture*. 1995, nº 32. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://journals.uvic.ca/index.php/ctheory/article/view/14655/5522. ISSN : 1190-9153.

KITTLER, Friedrich. *Gramophone, film, typewriter*. Traduit de l'allemand par Geoffrey WINTHROP-YOUNG et Michael WUTZ. Stanford : Stanford University Press, 1999 [1986 pour la version originale]. ISBN : 978-0-80473-233-8.

KITTLER, Friedrich. *Mode protégé*. Traduit de l'allemand par Frédérique VARGOZ. Dijon : Les Presses du Réel, 2015. ISBN : 978-2-84066-792-6.

KLOCK-FONTANILLE, Isabelle. Penser l'écriture : corps, supports et pratiques. *Communication & langages*. 2014, nº 182, p. 29-43.

KNUTH, Donald E. Literate Programming. *The Computer Journal*. 1984, vol. 27, nº 2, p. 97-111. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.literateprogramming.com/knuthweb.pdf.

Knuth, Donald E. *Éléments pour une histoire de l'informatique*. Traduit de l'anglais par Patrick Cégielski. Stanford : CSLI Publication – Paris : Société Mathématique de France, 2011. ISBN : 978-1-57586-622-2.

Knuth, Donald E. Two thousand years of combinatorics. Dans : Wilson, Robin et Watkins, John J. (dir.), *Combinatorics: Ancient and Modern*. Oxford : Oxford University Press, 2013, p. 3-5. ISBN : 978-0-19965-659-2.

Kraemer, Felicitas, Overveld, Kees et Peterson, Martin. Is there an ethics of algorithms? *Ethics and Information Technology*. 2011, vol. 13, n° 3, p. 251-260. [En ligne] [mis en ligne le 3 juillet 2010] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://link.springer.com/article/10.1007/s10676-010-9233-7. DOI : 10.1007/s10676-010-9233-7. 0000. ISSN : 1572-8439.

Krakowiak, Sacha. Le modèle d'architecture de von Neumann. *Interstices* [En ligne]. 2011. [Mis en ligne le 18 novembre 2011] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://interstices.info/modele-architecture-ordinateurs. ISSN : 2270-6224.

Kryza, Joasia et Grzesiek, Sedek. Source Code. Dans Fuller, Matthew (dir.). *Software Studies: a lexicon*. Cambridge : MIT Press, 2008, p. 236-243. ISBN : 978-0-26206-274-9.

L

LANDOW, George P. *Hypertext: the convergence of contemporary critical theory and technology*. Baltimore : Johns Hopkins University Press, 1992. ISBN : 978-0-80184-281-6.

LASSÈGUE, Jean. *Pour une anthropologie sémiotique ; recherches sur le concept de Forme symbolique*. Habilitation à diriger des recherches. Paris : Paris-Sorbonne, 2010. [Manuscrit en ligne] [mis en ligne le 15 juin 2010] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.formes-symboliques.org/IMG/pdf/doc-217.pdf.

LASSÈGUE, Jean et LONGO, Giuseppe. What is Turing's Comparison between Mechanism and Writing Worth? Dans : COOPER, S. Barry, DAWAR, Anuj et LÖWE, Benedikt (dir.), *How the World Computes*: *Turing centenary conference and 8th conference on computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012: proceedings*. Berlin : Springer Berlin Heidelberg, 2012, p. 450-461. ISBN : 978-3-64230-869-7. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.di.ens.fr/users/longo/files/PhilosophyAndCognition/lass-longo.pdf.

LATOUR, Bruno et LOWE, Adam. The Migration of the Aura – or How to Explore the Original Through Its Facsimiles. Dans : BARTSCHERER, Thomas et COOVER, Roderick (dir.), *Switching codes: thinking through digital technology in the humanities and the arts*. Chicago : University of Chicago Press, 2011, p. 275-297. ISBN : 978-0-22603-8-315. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.bruno-latour.fr/sites/default/files/108-ADAM-FACSIMILES-GB.pdf.

LAUFER, Roger (dir.). *Le Texte et son inscription*. Paris : Editions du Conseil National de la Recherche Scientifique, 1989. ISBN : 978-2-22204-218-1.

LE DEUFF, Olivier. Littératies informationnelles, médiatiques et numériques : de la concurrence à la convergence ? *Études de communication. Langages, information, médiations*. 2012, nº 38, p. 131-147.

LE MAREC, Joëlle. *Ce que le « terrain » fait aux concepts : vers une théorie des composites*. Habilitation à diriger des recherches. Paris : Université Paris Diderot – Paris 7, 2002. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://science.societe.free.fr/documents/pdf/HDR_Le_Marec.pdf.

Le Marec, Joëlle. Le public, le tact et les savoirs de contact. *Communication & langages*. 2013, nº 175, p. 3-25. [En ligne] [mis en ligne le 10 mai 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.necplus.eu/abstract_S0336150013011010. DOI : 10.4074/S0336150013011010. ISSN : 1778-7459.

Lee, Benjamin et LiPuma, Edward. Cultures of Circulation: The Imaginations of Modernity. *Public Culture*. 2002, vol. 14, nº 1, p. 191-213.

Leibniz, Gottfried Wilhelm. *De l'horizon de la doctrine humaine (1693) ; La restitution universelle (Apokatastasis panton, 1715)*. Textes inédits traduits du latin et présentés par Michel Fichant. Paris : Vrin, 1991. ISBN : 978-2-71161-081-5.

Leroi-Gourhan, André. *Le geste et la parole. 1 : Technique et langage*. Paris : Albin Michel, 2008 [1964]. ISBN : 978-2-22601-728-4.

Leroi-Gourhan, André. *Le geste et la parole. 2 : La mémoire et les rythmes*. Paris : Albin Michel, 1991 [1964]. ISBN : 978-2-22602-324-7.

Lessig, Lawrence. *Code and other laws of cyberspace*. New York : Basic Books, 1999. ISBN : 978-0-46503-913-5.

Loeve, Sacha et Normand, Mickaël. How to Trust a Molecule? The Case of Cyclodextrins Entering the Nanorealm. Dans : Zülsdorf, Torben B., Wienroth, Matthias, Coenen, Christopher, Fiedeler, Ulrich, Ferrari, Arianna et Milburn, Colin (dir.), *Quantum Engagements. Social Reflections of Nanoscience and Emerging Technologies*. Berlin : AKA Verlag, 2011, p. 195-216. ISBN : 978-3-89838-659-3. [En ligne] [mis en ligne le 11 janvier 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/hal-00658878. Identifiant : <hal-00658878>.

Lordon, Frédéric. *Capitalisme, désir et servitude : Marx et Spinoza*. Paris : La Fabrique éditions, 2013 [2010]. ISBN : 978-2-35872-013-7.

Lovelace, Ada. Sketch of the analytical engine invented by *Charles Babbage*, Esq. By *L-F. Menabrea*, of Turin, Officer of the Military Engineers. Dans Taylor, Richard (dir.), *Scientific memoirs, selected from the transactions of foreign academies of science and learned societies and from foreign journals*. Volume III. London : Taylor and Francis, 1843, p. 666-731. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://repository.ou.edu/uuid/6235e086-c11a-56f6-b50d-1b1f5aaa3f5e.

Lulle, Raymond. *L'art bref*. Milan : Archè, 1987 [1307]. ISBN 978-1-87252-015-0.

M

Mabi, Clément. *Le débat CNDP et ses publics à l'épreuve du numérique : entre espoirs d'inclusion et contournement de la critique sociale*. Thèse de doctorat. Compiègne : Université de Technologie de Compiègne, 2014. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://bibliotheque.utc.fr/EXPLOITATION/doc/IFD/2014COMP2148.

Mackenzie, Adrian. The Performativity of Code : Software and Cultures of Circulation. *Theory, Culture & Society*. 2005, vol. 22, nº 1, p. 71-92. [Manuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.lancaster.ac.uk/staff/mackenza/papers/code_performativity.pdf.

McKenzie, Donald F. *La bibliographie et la sociologie des textes*. Traduit de l'anglais par Marc Amfreville. Paris : Cercle de la librairie, 1991. ISBN : 978-2-76540-475-0.

Manovich, Lev. *The language of New Media*. Cambridge : MIT Press, 2000. ISBN : 978-0-26213-374-6.

Manovich, Lev. *Software takes command: extending the language of new media*. London : Bloomsbury Publishing, 2013. ISBN : 978-1-62356-672-2.

Masure, Anthony. *Le design des programmes : des façons de faire du numérique*. Thèse de doctorat. Paris : Paris 1, 2014.

Mœglin, Pierre. *Outils et médias éducatifs : une approche communicationnelle*. Grenoble : Presses Universitaires de Grenoble, 2005. ISBN : 978-2-70610-986-7.

Mœglin, Pierre. A la recherche de l'industrialisation du tutorat à distance. *Distances et savoirs*. 2005, vol. 3, nº 2, p. 251-265. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-distances-et-savoirs-2005-2-page-251.htm. ISSN : 1965-0167.

Montfort, Nick, Baudoin, Patsy, Bell, John, Bogost, Ian, Douglass, Jeremy, Marino, Mark C., Mateas, Michael, Reas, Casey, Sample, Mark et Vawter, Noah. *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*. Cambridge : MIT Press, 2012. ISBN : 978-0-26201-846-3.

Moulier-Boutang, Yann. La pollinisation humaine à l'ère numérique. *Labyrinthe*. 2014, nº 40, p. 125-128. [En ligne] [mis en ligne le 1er mars 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://labyrinthe.revues.org/4329. DOI : 10.4000/labyrinthe.4329. ISSN : 1950-6031.

N

Névot, Aurélie. *Comme le sel, je suis le cours de l'eau : le chamanisme à écriture des Yi du Yunnan, Chine*. Nanterre : Société d'ethnologie, 2008. ISBN : 978-2-90116-183-7.

P

Parnas, David L. On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM*. 1972, vol. 15, nº 12, p. 1053-1058. [En ligne] [mis en ligne le 1er mai 2012] [consulté le 1er septembre 2017]. DOI : 10.1145/361598.361623. Disponible à l'adresse : http://repository.cmu.edu/compsci/1980/.

Parikka, Jussi. *Digital contagions: a media archaeology of computer viruses*. New York : Peter Lang, 2007. ISBN : 978-0-82048-837-0.

Paveau, Marie-Anne. Bourdieu me manque. À propos de la technologie intellectuelle. Dans : *La pensée du discours* [En ligne]. Billet du 8 octobre 2010. [Mis en ligne le 8 octobre 2010] [Dernière mise à jour le 20 février 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://penseedudiscours.hypotheses.org/1218. ISSN : 2258-3068.

Pédauque, Roger T. *Le document à la lumière du numérique*. Caen : C & F éditions, 2006. ISBN : 978-2-91582-504-1.

Perec, Georges. Approches de quoi ? Dans : *L'infra-ordinaire*. Paris : Éditions du Seuil, 1989, p. 9-33.

Perret, Jean-Baptiste. Y a-t-il des objets plus communicationnels que d'autres ? *Hermès*. 2004, nº 38, p. 121-128. [En ligne] [mis en ligne le 8 octobre 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://documents.irevues.inist.fr/handle/2042/9435. DOI : 10.4267/2042/9435. ISSN : 1963-1006.

Perkins, Franklin. *Leibniz and China. A Commerce of Light*. Cambridge : Cambridge University Press, 2004. ISBN : 978-0-52183-024-9.

Plantin, Jean-Christophe. Les cartes numériques comme support de remédiation d'une catastrophe nucléaire. Le processus de cartographie de radiation suite à l'accident nucléaire de Fukushima Daiichi. *Réseaux*. 2014, nº 187, p. 163-193. [En ligne] [consulté le 9 septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-reseaux-2014-5-page-163.htm. DOI : 10.3917/res.187.0163. ISSN : 1777-5809.

PLATON. *Phèdre*. Traduit du grec ancien par Luc BRISSON. Paris : Flammarion, 2008. ISBN : 978-2-08071-268-4.

PORTER, Theodore M. *Trust in numbers: the pursuit of objectivity in science and public life*. Princeton : Princeton University Press, 1995. ISBN : 978-0-69103-766-0.

POST, Emil L. Finite combinatory processes—formulation. *The Journal of Symbolic Logic*. 1936, vol. 1, nº 3, p. 103-105.

Q

QUENEAU, Raymond. *Cent mille millards de poèmes*. Paris : Gallimard, 1961. ISBN : 978-2-07010-467-3.

R

RAMUNNI, Girolamo. *La physique du calcul : histoire de l'ordinateur*. Paris : Hachette, 1989. ISBN : 978-2-01012-417-4.

RANDELL, Brian. *On Alan Turing and the origins of digital computers.* Newcastle : University of Newcastle, 1972. ISBN : 978-0-90238-326-5.

RIEDER, Bernhard. Entre marché et communauté : une discussion de la culture participative à l'exemple de Google Maps. Communication donnée au colloque *Ludovia 2008 : Do It Yourself 2.0*, Ax-les-Thermes, août 2008. [Manuscrit en ligne] [mis en ligne le 13 octobre 2008] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://archivesic.ccsd.cnrs.fr/sic_00329899. Identifiant : < sic_00329899>.

ROBERT, Pascal. Qu'est-ce qu'une technologie intellectuelle ? *Communication & langages*. 2000, nº 123, p. 97-114. [En ligne] [mise en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/colan_0336-1500_2000_num_123_1_2992. DOI : 10.3406/colan.2000.2992. ISSN : 1778-7459.

ROBERT, Pascal et SOUCHIER, Emmanuël. La carte, un média entre sémiotique et politique. La carte au rivage des SIC. *Communication & langages*. 2008, nº 158, p. 25-29. [En ligne] [mis en ligne le 12 mars 2009] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.necplus.eu/abstract_S0336150008004031. DOI : 10.4074/S0336150008004031. ISSN : 1778-7459.

ROJAS, Estrella. New figures of web textualities: from semio-technical

forms toward a social approach of digital practices. *Archival Science*. 2009, vol. 8, nº 3, p. 227-246.

Rossi, Paolo. *Clavis universalis : arts de la mémoire, logique combinatoire et langue universelle de Lulle à Leibniz*. Traduit de l'italien par Patrick Vighetti. Grenoble : Millon, 1993. ISBN : 978-2-90561-489-6.

Rouvroy, Antoinette et Berns, Thomas. Détecter et prévenir : de la digitalisation des corps et de la docilité des normes. Dans : Lebeer, Guy et Moriau, Jacques (dir.), *(Se) gouverner. Entre souci de soi et action publique*. Francfort : Peter Lang, 2010, p. 157-184. ISBN : 978-9-05201-601-6. [Manuscrit en ligne] [mis en ligne le 19 juillet 2009] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://works.bepress.com/antoinette_rouvroy/30/.

Rouvroy, Antoinette et Berns, Thomas. Le nouveau pouvoir statistique. *Multitudes*. 2010, nº 40, p. 88-103. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-multitudes-2010-1-page-88.htm. DOI : 10.3917/mult.040.0088. ISSN : 0292-0107.

Ruppli, Mireille et Thorel-Cailleteau, Sylvie. *Mallarmé : la grammaire & le grimoire*. Genève : Droz, 2005. ISBN : 978-2-60001-004-7.

S

Samuelson, Pamela. The Uneasy Case for Software Copyrights Revisited. *George Washington Law Review*. 2011, vol. 79, nº 6, p. 1746-1782. [En ligne] [mis en ligne le 20 septembre 2012] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://scholarship.law.berkeley.edu/facpubs/1545/.

Samuelson, Pamela. Oracle v. Google: Are APIs Copyrightable? *Communications of the ACM*. 2012, vol. 55, nº 11, p. 25-27. [En ligne] [mis en ligne le 31 juillet 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://scholarship.law.berkeley.edu/facpubs/2354/. DOI : 10.1145/2366316.2366325. ISSN : 1557-7317.

Samuelson, Pamela. The Past, Present and Future of Software Copyright Interoperability Rules in the European Union and United States. *European Intellectual Property Review*. 2012, vol. 34, nº 4, p. 229-236. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://works.bepress.com/pamela_samuelson/219/.

SCHOLEM, Gershom. *Les grands courants de la mystique juive : la Merkaba, La Gnose, la kabbale, le zohar, le sabbatianisme et le hassidisme*. Traduit de l'anglais par Marie-Madeleine DAVY. Paris : Payot, 2014 [1950]. ISBN : 978-2-22891-026-2.

SHOOK, Edwin W. Tikal Stela 29. *Expedition*. 1960, vol. 2, n° 2, p. 29-37. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.penn.museum/sites/expedition/tikal-stela-29/. ISSN : 0014-4738.

SIEG, Wilfried, SZABÓ, Máté et MCLAUGHLIN, Dawn. Why Post Did [Not] Have Turing's Thesis. Dans : OMODEO, Eugenio G. et POLICRITI, Alberto (dir.), *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*. Berlin : Springer International Publishing, 2016, p. 175-208. ISBN : 978-3-31941-841-4.

SIRE, Guillaume. *Concevoir le Web et ses conflits* [Article en prépublication]. À paraître.

SOUCHIER, Emmanuël. L'écrit d'écran, pratiques d'écriture & informatique. *Communication & langages*. 1996, n° 107, p. 105-119. [En ligne] [mis en ligne le 15 octobre 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/colan_0336-1500_1996_num_107_1_2662. DOI : 10.3406/colan.1996.2662. ISSN : 1778-7459.

SOUCHIER, Emmanuël. L'image du texte : pour une théorie de l'énonciation éditoriale. *Les Cahiers de médiologie*. 1998, vol. 2, n° 6, p. 137-145. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.cairn.info/revue-les-cahiers-de-mediologie-1998-2-page-137.html. DOI : 10.3917/cdm.006.0137. ISSN : 1777-5604.

SOUCHIER, Emmanuël. Histoires de pages et pages d'histoire. Dans : ZALI, Anne (dir.), *L'Aventure des écritures : la page*. Paris : Bibliothèque Nationale de France, 1999, p. 19-55. ISBN : 978-2-71772-090-7.

SOUCHIER, Emmanuël. L'espace de la presse quotidienne comme « théâtre de mémoire », forme instituante de la doxa contemporaine. Dans : CHEVALIER, Yves et JUANALS, Brigitte (dir.), *Espaces physiques, espaces mentaux : identités et échanges*. Lille : Université Charles-de-Gaulle, 2007, p. 99-119. ISBN : 978-2-84467-095-4.

SOUCHIER, Emmanuël. Le livre au risque de l'écrit d'écran et des écrits de réseaux. Dans : ZALI, Anne (dir.), *La grande aventure du livre : de la tablette d'argile à la tablette numérique*. Paris : Bibliothèque Nationale de France – Hatier, 2013, p. 176-183. ISBN : 978-2-21896-764-1.

Souchier, Emmanuël. Le carnaval typographique de Balzac. Premiers éléments pour une théorie de l'irréductibilité sémiotique. *Communication & langages*. 2015, nº 185, p. 3-22.

Souchier, Emmanuël et Jeanneret, Yves. Écriture numérique ou médias informatisés ? *Pour la Science*. 2002, nº 33, p. 100-105.

Souchier, Emmanuël et Pomian, Joanna. Informatique et pratiques écrivantes. *Traverses*. 1988, nº 43, p. 121-130.

Souza, Cleidson R. et Redmiles, David F. On The Roles of APIs in the Coordination of Collaborative Software Development. *Journal of Computer Supported Cooperative Work*. 2009, vol. 18, nº 5-6, p. 445–475. [En ligne] [mis en ligne le 16 septembre 2009] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://link.springer.com/article/10.1007/s10606-009-9101-3. DOI : 10.1007/s10606-009-9101-3. ISSN : 1573-7551.

Steiner, Pierre. Philosophie, technologie et cognition. État des lieux et perspectives. *Intellectica*. 2010, vol. 1-2, nº 53-54, p. 7-40.

Steinmueller, Edward W. The International Software Industry : An Analysis and Interpretive History. Dans : Mowery, David C (dir.), *The international computer software industry: a comparative study of industry evolution and structure*. Oxford : Oxford University Press, 1996, p. 15-52. ISBN : 978-0-19509-410-7.

Stiegler, Bernard. *La technique et le temps. 1, La faute d'Epiméthée*. Paris : Galilée, 1994. ISBN : 978-2-71860-440-4.

Stiegler, Bernard. *La technique et le temps. 2, La désorientation*. Paris : Galilée, 1996. ISBN : 978-2-71860-468-9.

Stiegler, Bernard. Leroi-Gourhan : l'inorganique organisé. *Les Cahiers de médiologie*. 1998, vol. 2, nº 6, p. 187-194. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cairn.info/revue-les-cahiers-de-mediologie-1998-2-page-187.htm. DOI : 10.3917/cdm.006.0187. ISSN : 1777-5604.

T

TARDY, Cécile et JEANNERET, Yves (dir.). *L'écriture des médias informatisés : espaces de pratiques*. Cachan : Hermès Science – Lavoisier, 2007. ISBN : 978-2-74621-653-2.

TEDRE, Matti. *The science of computing: shaping a discipline*. Boca Raton : CRC Press, 2015. ISBN : 978-1-48221-769-8.

TODOROV, Tzvetan. *Mikhaïl Bakhtine : le principe dialogique*. Paris : Éditions du Seuil, 1981. ISBN : 978-2-02005-830-8.

TURING, Alan M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*. 1937, vol. s2-42, nº 1, p. 230-265. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf.

TURING, Alan M. Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE) [1946]. Dans : CARPENTER, B. E et DORAN, R. W, *A.M. Turing's ACE report of 1946 and other papers*. Cambridge : MIT Press, 1986, p. 20-105. ISBN : 978-0-26203-114-1.

TURING, Alan M. Computing Machinery and Intelligence. *Mind*. 1950, vol. 59, nº 236, p. 433-460. [En ligne] [mis en ligne le 12 janvier 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://academic.oup.com/mind/article/LIX/236/433/986238/I-COMPUTING-MACHINERY-AND-INTELLIGENCE. DOI : 10.1093/mind/LIX.236.433. ISSN : 1460-2113.

U

URICCHIO, William. The algorithmic turn: Photosynth, augmented reality and the changing implications of the image. *Visual Studies*. 2011, vol. 26, nº 1, p. 25-35. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://web.mit.edu/uricchio/Public/pdfs/pdfs/Algorithmic_Turn.pdf.

V

Valéry, Paul. *Cahiers. Tome 23, 1940*. Paris : Centre National de la Recherche Scientifique, 1957-1960.

Valluy, Jérôme. Editorialisation. Dans : *Terra HN* [En ligne]. Publication du 18 mai 2015. [Mis en ligne le 18 mai 2015] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.reseau-terra.eu/article1333.html. ISSN : 8089-9790.

Vandermeersch, Léon. Origine et évolution de l'achilléomancie chinoise. *Comptes rendus des séances de l'Académie des Inscriptions et Belles-Lettres*. 1990, vol. 134, nº 4, p. 949-963. [En ligne] [mis en ligne le 5 juin 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.persee.fr/doc/crai_0065-0536_1990_num_134_4_14929. DOI : 10.3406/crai.1990.14929. ISSN : 1969-6663.

Vegetti, Mario. Dans l'ombre de Thoth. Dynamiques de l'écriture chez Platon. Dans : Detienne, Marcel (dir.), *Les Savoirs de l'écriture en Grèce ancienne*. Lille : Presses Universitaires de Lille, 1988, p. 387-419. ISBN : 978-2-85939-322-9.

Vitali Rosati, Marcello. Qu'est-ce que l'éditorialisation ? *Sens Public* [En ligne]. 2016. [Mis en ligne le 18 mars 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.sens-public.org/article1184.html. ISSN : 2104-3272.

W

Wheeler, David J. The use of sub-routines in programmes. Dans : *Proceedings of the 1952 ACM national meeting (Pittsburgh)*. New York : ACM Press, 1952, p. 235-236.

Wilkes, Maurice V, Gill, Stanley et Wheeler, David J. *The Preparation of Programs for an Electronic Digital Computer*. Cambridge : Addison-Wesley Press, 1951.

Williams, Michael R. *A History of Computing Technology*. Englewood Cliffs : Prentice-Hall, 1985. ISBN : 978-0-13389-917-7.

Wilson, Robin et Watkins, John J. (dir.). *Combinatorics: Ancient and Modern*. Oxford : Oxford University Press, 2013. ISBN : 978-0-19965-659-2.

Y

YATES, Frances A. *The Art of Memory*. Londres : Random House, 2014 [Chicago : University of Chicago Press, 1966]. ISBN : 978-1-84792-292-2.

YATES, Frances A. *Raymond Lulle et Giordano Bruno*. Traduit de l'anglais par Muriel ZAGHA. Paris : Presses Universitaires de France, 1999. ISBN : 978-2-13047-575-0.

YOO, Christopher S. et BLANCHETTE, Jean-François (dir.). *Regulating the cloud: policy for computing infrastructure*. Cambridge : The MIT Press, 2015. ISBN : 978-0-26202-940-7.

Encyclopédies et articles d'encyclopédies

Algorithme de tracé de segment de Bresenham, 2017. *Wikipedia*. [En ligne] [mis en ligne le 26 mai 2006] [dernière modification le 7 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Algorithme_de_trac%C3%A9_de_segment_de_Bresenham.

Article « Application program interface (API) ». Dans : Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New York : Facts On File, 2009 [2003], p. 20-21. ISBN : 978-0-81606-382-6.

Article « Application software ». Dans : Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 22. ISBN : 978-0-81606-382-6.

Article « Library, program ». Dans : Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 276. ISBN : 978-0-81606-382-6.

Article « Object-Oriented Programming ». Dans : Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 339-341.

Bidgoli, Hossein (dir.), *The Handbook of Computer Networks. Volume 3 : Distributed networks, Network planning, Control, Management, and New Trends and Applications*. Hoboken : John Wiley & Sons, 2008. ISBN : 978-0-47178-460-9.

Calvet, Jean-Louis. Article « Double articulation ». Dans : *Encyclopædia Universalis* [En ligne]. [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.universalis.fr/encyclopedie/double-articulation/.

Carson, Georges S. Computer Graphics : standards. Dans : Reilly, Edwin D., Ralston, Anthony et Hemmendinger, David (dir.), *Encyclopedia of Computer Science*. Londres : Nature Publishing Group, 2000 [1976], p. 378-340. ISBN : 978-0-33377-879-1.

Commission Nationale Informatique et Libertés. Article « Algorithme ». [En ligne] [consulté le 22 août 2017]. Disponible à l'adresse : https://www.cnil.fr/fr/definition/algorithme.

Computer Associates International, Inc. v. Altai, Inc., 2017. *Wikipédia*. [En ligne] [mis en ligne le 28 septembre 2009] [dernière modification le 11 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Computer_Associates_International,_Inc._v._Altai,_Inc.

De Biasi, Pierre-Marc. Article « Théorie de l'intertextualité ». Dans : *Encyclopædia Universalis* [En ligne]. [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.universalis.fr/encyclopedie/theorie-de-l-intertextualite/.

Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 276. ISBN : 978-0-81606-382-6.

Hexagramme, 2016. *Wikipedia*. [En ligne] [mis en ligne le 14 mars 2012] [dernière modification le 8 novembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Hexagramme.

Illingworth, Valerie. (dir.). *Dictionnaire d'informatique*. Traduit de l'anglais par Saint-Dizier, Édith. Paris : Hermann – Technique et Documentation Lavoisier, 1991. ISBN : 978-2-85206-758-5.

Oracle America, Inc. v. Google, Inc. 2017. *Wikipédia*. [En ligne] [mis en ligne le 10 mai 2014] [dernière modification le 24 août 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Oracle_America,_Inc._v._Google,_Inc.

Quantified Self, 2017. *Wikipedia*. [En ligne] [mis en ligne le 26 mai 2006] [dernière modification 6 mars 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Quantified_self.

Reilly, Edwin D., Ralston, Anthony et Hemmendinger, David (dir.), *Encyclopedia of Computer Science*. Londres : Nature Publishing Group, 2000 [1976]. ISBN : 978-0-33377-879-1.

Thimpleby, Harold W. Java. Dans : Ralston, Anthony, REILLY, Edwin D et Hemmendinger, David (dir.), *Encyclopedia of computer science*. Londres : Nature Publishing Group, 2000 [1976], p. 937-941. ISBN : 978-0-33377-879-1.

Tunstall, Michael, Markantonakis, Konstantinos, Sauveron, Damien et Mayes, Keith. Smart Cards: Communication Protocols and Applications. Dans : Bidgoli, Hossein (dir.), *The Handbook of Computer Networks. Volume 3 : Distributed networks, Network planning, Control, Management, and New Trends and Applications*. Hoboken : John Wiley & Sons, 2008, p. 251-268. ISBN : 978-0-47178-460-9.

Volonino, Linda et Dalal, Pragati. Network Middleware. Dans : Bidgoli, Hossein (dir.), *The Handbook of Computer Networks. Volume 3 : Distributed networks, Network planning, Control, Management, and New Trends and Applications*. Hoboken : John Wiley & Sons, 2008, p. 33-44. ISBN : 978-0-47178-460-9.

Rapports techniques et ministériels

Curry, Haskell B. et Wyatt, Willa A. *A Study of inverse interpolation of the ENIAC*. Rapport n°AD0640621 du 19 août 1946. Aberdeen : Army Ballistic Research Lab. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.dtic.mil/get-tr-doc/pdf?AD=AD0640621.

Eckert, John A., Mauchly, John W., Goldstine, Herman H. et Brainerd, John G. *Description of the ENIAC and Comments on Electronic Digital Computing Machines*. Rapport n° AD498867 du 30 novembre 1945. Philadelphie : Moore School of Electrical Engineering. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.dtic.mil/get-tr-doc/pdf?AD=AD0498867.

Goldstine, Herman H. et Von Neumann, John. *Planning and Coding of Problems for an Electronic Computing Instrument: report on the mathematical and logical aspects of an electronic computing instrument. Volume 1-3, part II*. Princeton : Institute for Advanced Study, 1947. [Tapuscrit en ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://library.ias.edu/files/pdfs/ecp/planningcodingof0103inst.pdf.

Krim, Tariq. *Les développeurs, un atout pour la France*. Rapport adressé au Ministère de l'Économie, du Redressement productif et du Numérique, 6 mars 2014. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.ladocumentationfrancaise.fr/var/storage/rapports-publics/144000190.pdf.

OCDE, « La Littératie à l'ère de l'information ». Rapport, 2000. [En ligne] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.oecd.org/fr/edu/innovation-education/39438013.pdf.

Von Neumann, John. *First draft of a report on the EDVAC.* Rapport du 30 juin 1945. Philadelphie : Moore School of Electrical Engineering, University of Pennsylvania. [Tapuscrit en ligne] [mis en ligne le 27 janvier 2011] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://library.si.edu/digital-library/book/firstdraftofrepo00vonn. DOI : 10.5479/sil.538961.39088011475779. Version électronique en ligne, éditée par Godfrey, Michael D. [Mis en ligne le 15 janvier 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://archive.org/details/vnedvac.

Articles de presse

Guillaud, Hubert. Limiter le pouvoir des algorithmes. *InternetActu.net* [En ligne]. Article du 10 octobre 2013. [Mis en ligne le 10 octobre 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.internetactu.net/2013/10/10/limiter-le-pouvoir-des-algorithmes/.

Haupt, Michael. "Data is the New Oil" — A Ludicrous Proposition. Dans : *Twenty One Hundred* [En ligne]. Article du 2 mai 2016. [Mis en ligne le 2 mai 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://medium.com/twenty-one-hundred/data-is-the-new-oil-a-ludicrous-proposition-1d91bba4f294#.vjyvcwnp0.

Steiner, Georges. The end of bookishness ? *Times Literary Suplement*. 8-16 juillet 1988, p. 754.

The world's most valuable resource is no longer oil, but data. *The Economist* [En ligne]. Article du 6 mai 2017. [Mis en ligne le 6 mai 2017] [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.economist.com/news/leaders/21721656-data-economy-demands-new-approach-antitrust-rules-worlds-most-valuable-resource.

Autres références web

Apidays. Site officiel [En ligne]. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.apidays.io/.

Chevalier, Paul-Antoine. Retour sur le 1er data camp de l'Assemblée Nationale. Dans : *Le blog d'Etalab* [En ligne]. Billet du 5 décembre 2016. [Mis en ligne le 5 décembre 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.etalab.gouv.fr/retour-sur-le-1er-data-camp-de-lassemblee-nationale.

Facebook. Centre de ressources marketing [En ligne]. 2016. [consulté le 1er septembre 2017]. Disponible à l'adresse : https://fr.facebookbrand.com/.

Google. Introducing our Geo Developers Blog. Dans : *Google Geo Developers* [En ligne]. Article du 28 mai 2008. [Mis en ligne le 28 mai 2008] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://maps-apis.googleblog.com/2008/05/.

Gouvernement français. La French Tech : une ambition collective pour les start-up françaises. [En ligne] 2017. [Dernière modification le 15 mai 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.gouvernement.fr/action/la-french-tech-une-ambition-collective-pour-les-start-up-francaises.

Guess, Jeff et Wagon, Gwenola. *Media Mediums* [En ligne]. 2013 – présent. [consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.mediamediums.net/fr/projects

Lane, Kin. *The Api Evangelist blog* [En ligne]. Blog personnel. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://apievangelist.com/blog/.

Open Classrooms. Introduction aux algorigrammes [En ligne]. 2013. [Mis à jour le 10 janvier 2013] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://openclassrooms.com/courses/introduction-aux-algorigrammes.

Open Graph Protocol. 2017 [En ligne]. [Dernière modification le 27 juillet 2017] [consulté le 1er septembre 2017]. Disponible à l'adresse : ogp.me.

Programmable Web. *ProgrammableWeb – APIs, Mashups, and the Web as Platform* [En ligne]. Site officiel. [s. d.]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.programmableweb.com/.

Valla, Clément. *Postcards From Google Earth* [En ligne]. 2010-Présent. [Consulté le 2 septembre 2017]. Disponible à l'adresse : http://www.postcards-from-google-earth.com/.

Yoast. Yoast SEO for WordPress plugin. Dans : *Yoast. SEO for* everyone [En ligne]. 2003-Présent. [consulté le 1er septembre 2017]. Disponible à l'adresse : https://yoast.com/wordpress/plugins/seo/#title-description.

Lois, diaporamas, correspondances

De Mol, Liesbeth. *Code source sans code : le cas de l'ENIAC*. Communication donnée dans le cadre du séminaire « Codes Sources », LIP6, UPMC-Paris 6, 22 juin 2016. Support de présentation disponible en ligne. [Mis en ligne le 14 juillet 2016] [consulté le 1er septembre 2017]. Disponible à l'adresse : https://hal.archives-ouvertes.fr/cel-01345599/document. Identifiant : <cel-01345599>.

Descartes, René. *À Beeckman*. Lettre du 29 avril 1619. [En ligne] [mis en ligne en avril 2013] [consulté le 1er septembre 2017]. Traduit du latin vers l'anglais par Jonathan Bennett. Disponible à l'adresse : http://www.earlymoderntexts.com/assets/pdfs/descartes1619_1.pdf, p. 3.

United States Copyright Office. Copyright Law of the United States, title 17. [En ligne]. Décembre 2016. [consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.copyright.gov/title17/.

Captations vidéos

Knuth, Donald E. *2014 Kailath Lecture: Let's Not Dumb Down the History of Computing Science* [captation vidéo en ligne]. Conférence donnée à l'université de Stanford le 7 mai 2014. [Mis en ligne le 30 mai 2014] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://kailathlecture.stanford.edu/2014KailathLecture.html. Durée : 52'50".

Facebook. *f8 Conference Keynote Speech – Recorded 21st April 2010* [captation vidéo en ligne]. Conférence donnée à Palo Alto le 21 avril 2010. [Mis en ligne le 21 avril 2010] [consulté le 1er septembre 2017]. Disponible à l'adresse : http://original.livestream.com/f8conference/video/pla_e7a096b4-3ef9-466d-9a37-d920c31040aa. Durée : 43'45".

INDEX THÉMATIQUE

A

B

C

✦ D ✦

E

✦ F ✦

G

H

I

L

M

N

O

P

R

S

T

U

W

INDEX DES NOMS

J

K

L

S

T

V

W

Y

De briques et de blocs. La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Résumé

Boutons « J'aime », tweets ancrés… Toutes ces formes sont générées par des interfaces de programmation ou API, outils d'écriture informatique qui ont integré la chaîne de production des textes de réseau contemporains. Cette thèse interroge la fonction éditoriale des API, soit leur rôle dans la production, la standardisation et la circulation des « petites formes » des textes de réseau. Avec comme corpus les API de Facebook et de Twitter ainsi que les petites formes qu'elles permettent de produire, notre analyse techno-sémiotique s'articule en cinq chapitres. Le premier est une généalogie des API du point de vue de l'écriture combinatoire. Nous montrons que cette conception de l'écriture est un trait saillant de la programmation et de l'informatique. Le second chapitre interroge les imaginaires de l'écriture informatique, entre chiffre, combinatoire et méthode scientifique universelle. Le troisième chapitre est une analyse des conséquences sémiotiques de cet universalisme combinatoire, où nous montrons que les API proposent une conception du texte comme ensemble abstrait de blocs combinables. Abstraction du texte qui sert une « économie des passages », objet de notre quatrième chapitre, dans laquelle les API sont des lieux d'industrialisation d'une « pratique lettrée » : elles établissent des critères de lisibilité et de reproductibilité du texte. Parmi ces critères, nous notons une invisibilisation du rôle pourtant fondamental du calcul informatique. Nous proposons donc, dans un cinquième chapitre, des pistes pour développer une sémiotique qui prenne en compte le calcul comme mode d'expression propre aux médias numériques.

Mots-clés : Calcul ; numérique ; informatique ; combinatoire ; éditorialisation ; interfaces de programmation ; API ; écrits d'écran ; sémiotique ; écriture ; industrialisation de l'écriture ; techno-sémiotique ; code informatique.

Of Bits and Blocks. The publishing function of web Application Programming Interfaces (APIs): from a combinatorial science to an industry of text-processing.

Summary

« Like » buttons, embedded tweets… All of these visual forms are produced by Application Programming Interfaces (APIs). APIs are digital writing tools which have become part of the publishing process of contemporary web pages. This thesis aims at understanding the « publishing function » of APIs : their role in the production, standardization and circulation of the « little forms » of online texts. Focused on Facebook's and Twitter's APIs, our work is divided into five chapters. The first one is a genealogy of the APIs, starting from their combinatorial aspect, a conception of writing which trace back to early programming and the invention of computer science. The second chapter is an inquiry about the imaginaries of calculus as a kind of writing, torn between the imaginary of numbers, of combinatorics and the search for a universal scientific method. The third chapter is a study of the semiotic consequences of this combinatorial universalism. We show how APIs are based on an idea of text as an abstract, modular object. This abstraction of the text is beneficial to an « economy of passages ». In this economy where circulation produce value, APIs are a place of « literate practices » (chapter four). They establish visual standards for the readability, production and circulation of online texts. Among these standards, there's a systematic invizibilisation of the action of machines, although calculus is a necessary part of the production of digital texts. Therefore, in the fifth chapter, we give some epistemological elements towards non-anthropocentric semiotics, meaning : semiotics which would take into account computational machines as a part of the utterance of digital texts.

Keywords : Calculus ; digital studies ; computing ; combinatorics ; editorialization ; publishing process ; literary practices ; coding ; written screens ; API ; programming ; semiotics ; writing ; industrialization of writing.

Thèse en **Sciences de l'Information et de la Communication,** préparée au sein du **GRIPIC**
(Groupe de Recherche Interdisciplinaire sur les Processus d'Information et de Communication).

CELSA, UNIVERSITÉ PARIS-SORBONNE — 77 rue de Villiers – 92200 Neuilly-sur-Seine.
École doctorale nº 5 « Concepts et langages » — Maison de la Recherche – 28 rue Serpente, 75006 Paris.

CELSA
UNIVERSITÉ PARIS-SORBONNE

ÉCOLE DOCTORALE nº 5 « Concepts et Langages »

Laboratoire de recherche GRIPIC (EA 1498)

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-SORBONNE

Discipline : Sciences de l'Information et de la Communication

Présentée et soutenue par :

Samuel GOYET

le 22 novembre 2017

De briques et de blocs.

La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Tome II : glossaire et annexes

Sous la direction de :

M. Emmanuël SOUCHIER – Professeur des universités, CELSA Université Paris-Sorbonne.

Membres du jury :

M. Bruno BACHIMONT – Professeur des universités, Université de Technologie de Compiègne.

M^me^ Marie-Paule CANI – Professeure des université, École Polytechnique.

M. Jean DAVALLON – Professeur émérite, Université d'Avignon et des Pays de Vaucluse.

M. Milad DOUEIHI – Professeur des universités, Université Paris-Sorbonne.

M^me^ Joëlle LE MAREC – Professeure des universités , CELSA Université Paris-Sorbonne.

M^me^ Marie-Anne PAVEAU – Professeure des universités, Université Paris 13.

M. Emmanuël SOUCHIER – Professeur des universités, CELSA Université Paris-Sorbonne.

GLOSSAIRE

Ce glossaire a pour vocation d'expliquer et de définir certains termes techniques utilisés dans la thèse. Tous les mots mis entre accolades dans la thèse sont définis dans ce glossaire, à l'entrée correspondante.

Les définitions proposées sont des synthèses de ressources en libre accès en ligne : encyclopédies, articles de revues, sites d'apprentissage de la programmation, documentation technique…. Pour ne pas alourdir le propos, nous avons choisi de ne pas mentionner toutes les sources utilisées pour créer chaque définition. En revanche, quand nous citons explicitement un auteur (entre guillemets) ou que nous avons utilisé une source imprimée, nous mentionnons le nom de l'auteur ainsi que le titre de l'ouvrage. Les références complètes peuvent être retrouvées dans la bibliographie de la thèse.

Les mots « formel » et « formalisation » (ainsi que leurs formes accordées) dans le texte de la thèse renvoient à l'entrée de glossaire « Formalisme ». Lorsque le mot « langage » est employé dans la thèse, seule l'acception « langage de programmation » est intégrée au glossaire. Il en va de même pour le mot « objet ».

A

ABRÉVIATEUR D'URL : Service en ligne permettant de réduire la longueur d'une adresse web. Par exemple, l'adr esse « https://www.timeshighereducation.com/news/academics-meet--declare-support-universities-hit-conflict » deviendra, dans sa version abrégée « bit.ly/1KckTkj ». Les abréviateurs d'URL sont particulièrement utilisés dans des sites comme Twitter (où ils sont même intégrés dans la plateforme, qui raccourcit automatiquement les URL), car ils permettent d'économiser le nombre de caractères utilisés, Twitter limitant la taille des messages à cent quarante caractères. Il est à noter que l'adresse URL résultant de cette abréviation porte la marque de l'abréviateur utilisé. Dans notre exemple (bit.ly), le service utilisé est Bitly.

AGIR COMPUTATIONNEL : Mode d'action spécifique aux ordinateurs et plus largement à toute machine computationnelle. Il se caractérise par l'exploitation systématique, par le calcul, de toutes les possibilités d'un ensemble combinatoire dont les limites sont définies en amont par un être humain. Cette activité ne suppose pas une volonté autonome, mais un automatisme – une faculté de mouvement sans intervention humaine. Voir *Machine computationnelle*.

ALGORITHME : Un algorithme est « […] la description d'une suite d'étapes permettant d'obtenir un résultat à partir d'éléments fournis en entrée ». L'algorithme n'est pas propre à l'informatique, mais est un concept central de cette discipline en vertu du fait que les ordinateurs sont, par leur fonctionnement calculatoire et combinatoire, des machines particulièrement adaptées pour exécuter un algorithme. Pour qu'un algorithme puisse être mis en œuvre par un ordinateur, il faut qu'il soit exprimé dans un langage informatique. L'algorithme est un élément central de la culture numérique, car il est aujourd'hui impliqué dans énormément d'activités menées par le biais de médias informatisés : tri et retouches d'images, imagerie médicale et diagnostic, compression du son et de l'image, recherche dans des corpus massifs de textes…

Source : Commission Nationale Informatique et Libertés. Article « Algorithme » [en ligne]. [Consulté le 1er septembre 2017]. Disponible à l'adresse : https://www.cnil.fr/fr/definition/algorithme.

API *(APPLICATION PROGRAMMING INTERFACE)* : Interface de programmation. « Ensemble de routines standard facilitant le développement d'applications ou la manipulation d'opérations sur une plateforme ». Dans le cas des API web de notre corpus, elles permettent un accès structuré, sous certaines conditions, à deux types d'objets : des blocs de code préécrits permettant la génération de *widgets* ; certaines des données détenues par Facebook ou Twitter et pouvant alors être utilisée dans des applications. Une API est constituée de l'infrastructure technique permettant l'accès – l'interface à proprement parler, les serveurs et protocoles d'authentification pour accéder aux données – et de la documentation, soit l'ensemble des textes qui expliquent comment sont structurées les données et comment y accéder. Voir *Widget*.

Source : Doueihi, Milad. *Pour un humanisme numérique*. Paris : Éditions du Seuil, 2011, p. 169.

APPEL (API) : Voir *Requête (API)*.

APPLICATION (INFORMATIQUE) : Logiciel informatique permettant à l'utilisateur d'effectuer une tâche spécialisée. Par exemple, Microsoft Word est une application de traitement de texte : elle sert à composer un texte. Un navigateur est un autre exemple d'application. On distinguera les applications des systèmes d'exploitation, qui sont des logiciels mais qui ne permettent pas une utilisation spécialisée. Ce sont au contraire des logiciels généralistes qui organisent les échanges entre les composants matériels et les applications, permettant à ces dernières de fonctionner. Voir *Logiciel ; Système d'exploitation*.

Source : Article « Application software ». Dans Henderson, Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 22.

ARCHITECTURE DE VON NEUMANN : Mode d'organisation des ordinateurs développé par John Von Neumann et son équipe au début des années 1940. Exposé pour la première fois dans le *First Draft of a Report on the EDVAC*, il est ensuite largement adopté dès 1946. L'architecture de Von Neumann consiste à organiser un ordinateur en cinq unités qui ont autant de fonctions : la mémoire (qui stocke les données) ; l'unité de commande (qui vérifie l'ordre des instructions) ; l'unité logique et arithmétique (qui effectue les calculs) ; l'unité d'entrée de données (qui permet de les mettre en mémoire) ; l'unité de sortie de données (qui restitue les résultats). La grande innovation de cette architecture est qu'elle stocke sous la même forme les instructions et les données du calcul. C'est ce qu'on appelle un « ordinateur à programme en mémoire interne », au contraire d'un ordinateur comme l'ENIAC où le programme était sous forme externe (un arrangement de câbles reliant les différentes parties de la machine). Parce qu'elle permet d'internaliser le programme, l'architecture de Von Neumann permet donc de faire du programme un texte. C'est ce que nous avons appelé la « scripturisation de la programmation ». Pour plus d'informations, voir *Partie I. 2. A., p. 114*.

ARCHITECTURE MATÉRIELLE : Agencement des composants d'un ordinateur selon leurs fonctions et interactions. Il existe plusieurs architectures matérielles, selon les constructeurs et les usages prévus des machines, chaque architecture matérielle faisant varier les performances de ces dernières.

ARCHITEXTE : Concept forgé par Emmanuël Souchier et Yves Jeanneret pour désigner les « [...] outils qui permettent l'existence de l'écrit à l'écran et qui, non contents de représenter la structure du texte, en commandent l'exécution et la réalisation ». Un traitement de texte, un champ de recherche Google, un logiciel de montage vidéo, une API sont des architextes : ce sont des outils qui permettent d'écrire grâce à un ordinateur. Ces outils ont deux particularités : ce sont des « outils d'écriture écrits », ils permettent donc l'écriture en la formatant dans le même temps ; ils sont propre à l'informatique en ceci que cette technique d'écriture est fondée sur une rupture sémiotique et sémantique. C'est cette rupture qui nécessite des outils de conversion – les architextes – entre le niveau de l'inscription (la matérialité du disque dur ou de tout support de mémoire) et celui de l'affichage (l'écran, l'interface).

Sources : Jeanneret Yves et Souchier Emmanuël. Pour une poétique de l'écrit d'écran. *Xoana*. 1999, n°6, p. 103 ; Souchier Emmanuël. Le livre au risque de l'écrit d'écran et des écrits de réseaux. Dans : Zali, Anne (Dir.), *La grande aventure du livre : de la tablette d'argile à la tablette numérique*. Paris : Bibliothèque Nationale de France – Hatier, 2013, p. 176-183.

AUTOTHÉTICITÉ : L'autothéticité est l'une des caractéristiques du signe informatique qui, en vertu de ses propriétés formelles, « [...] ne renvoie qu'à sa propre effectivité ». Le nom d'une variable dans un programme, par exemple, n'a pas besoin de renvoyer à un élément du monde réél ou même d'être un mot existant de la langue pour être efficace techniquement. Il a simplement besoin d'être non-ambigu et de rester le même dans tout le programme. En revanche, il est plus simple pour des développeurs d'utiliser des termes intelligibles, en empruntant à la langue de rédaction du programme. De ce point de vue – celui que Bachimont nomme « orthothétique » – le signe informatique renvoie à quelque chose qui lui est extérieur : à la culture des développeurs et à leurs critères d'intelligibilité d'un texte. Voir *Variable ; Programme*.

Source : Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Technologies, Idéologies, Pratiques*. 1999, n° 4, p. 201.

✦ B ✦

BALISE (INFORMATIQUE) : en informatique, plus spécifiquement dans le contexte des langages de balisage, une balise représente une unité syntaxique utilisée pour indiquer à l'ordinateur les différentes séquences d'un flux de caractères. La machine peut ainsi lire les balises et isoler des séquences de caractères pour leur appliquer des actions spécifiques : placement dans la page, appliquer des styles de caractère, ou au contraire ne pas afficher la séquence concernée. Une balise, à l'instar d'une parenthèse, doit donc être ouverte puis fermée, ce sans quoi la machine ne discriminerait pas la balise du reste du texte. En HTML par exemple (qui est un langage à balise), les balises `<li></li>` indiquent les éléments d'une liste. Tout ce qui se trouve entre ces balises sera donc automatiquement inclus dans une liste. Voir *HTML ; Langage de programmation*.

BIBLIOTHÈQUE LOGICIELLE : En programmation, les bibliothèques logicielles (*library* ou encore *program library*) sont des opérations standards précodées auquel un développeur peut faire appel, lors de l'écriture de son programme, pour gagner du temps ou pour pouvoir

intégrer de nouvelles fonctionnalités. Par exemple, dans le logiciel de création graphique *Processing*, il est nécessaire d'importer une bibliothèque logicielle pour pouvoir manipuler des polices de caractères, ce que ne permet pas le logiciel sans cette bibliothèque. L'appel à ces bibliothèques se fait soit en amont de l'écriture, par importation dans le logiciel, soit au cours de l'écriture, en formulant un appel à la librairie directement dans le code source. La mise au point de ces bibliothèques est aussi ancienne que la programmation informatique, les premières bibliothèques logicielles étant théorisées dès les années 1940. Voir *Code source ; Programme*.

Source : Article « Library, program ». Dans Henderson Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 276.

BINAIRE (LANGAGE) : Le langage binaire désigne la représentation conventionnelle de la façon dont un courant électrique circule dans les composants d'un ordinateur. Bien souvent, il consiste en une suite de 0 et de 1, aussi appelé langage machine. Une machine contemporaine composée de portes logiques est entièrement binaire, car ses composants organisent le courant électrique selon deux modalités : tension supérieure ou inférieure à cinq volts. Le binaire n'est pas inhérent à l'informatique (les premières machines comme le Harvard Mark I étaient électromécaniques et certains des composants étaient en base dix) mais il a été adopté dans les années 1940 pour optimiser la vitesse de calcul et la fabrication des machines. Voir *Porte logique*.

BIOS : *Basic Input / Output System*. Le BIOS est un programme élémentaire, stocké dans la mémoire morte d'un ordinateur ou à même un composant (on parle alors de *firmware*, ou micrologiciel), qui initie les premières opérations de la machine : reconnaissance des périphériques, lancement du système d'exploitation… Il est la première couche logicielle, avant même le sytème d'exploitation, même s'il est encore très lié à la matérialité de la machine sur laquelle il est exécuté. Voir *Logiciel ; Matériel ; Programme*.

BOUCLE (PROGRAMMATION) : Une boucle est, dans un programme informatique, une suite d'instructions qui se répète sous conditions jusqu'à obtention d'un résultat déterminé en amont. On peut par exemple programmer une boucle qui opère un décompte de 10 à 0, en prenant une situation de départ (x = 10), en appliquant une opération (si x > 0, alors retrancher 1 à x) et en déterminant une situation d'arrivée (x = 0). De fait, la boucle est une structure algorithmique. Elle est très utilisée en programmation, car elle permet une économie de l'écriture : une boucle permet d'écrire une suite d'opérations en une seule instruction, plutôt que de détailler chaque opération. Voir *Algorithme*.

BUG : Mot anglais signifiant « insecte ». Initialement employé en ingénierie pour désigner un défaut dans un mécanisme, un *bug* est, en informatique, une erreur qui empêche le programme ou la machine de fonctionner. Le *bug* peut être d'origine logicielle (une erreur logique dans l'écriture du programme, une bibliothèque à laquelle le programme ne peut accéder, etc.), matérielle (un composant défectueux) ou à l'intersection des deux (un pilote non mis à jour qui empêche un logiciel d'avoir accès à un périphérique comme une imprimante par exemple). Voir *Logiciel ; Matériel*.

C

CALCUL (INFORMATIQUE) : Suite finie d'opérations discrètes effectuées automatiquement par le biais d'un nombre fini de symboles univoques. Ces opérations consistent en une manipulation de ces symboles suivant les règles de la logique formelle. Le calcul ainsi entendu est fondé sur la logique hilbertienne et sa mécanisation – ouvrant la voie à l'informatique contemporaine – est notamment l'apanage d'Alan Turing.

CARTE GRAPHIQUE : Composant matériel chargé de convertir les données calculées par le processeur en images affichées sur un écran.

CARTE PERFORÉE : Carte découpée en cases pouvant servir à la programmation informatique. Chaque instruction donnée à la machine est encodée sous une forme binaire (1/0), forme que traduit matériellement la perforation de la carte (un trou pour 0, un plein pour 1). Un programme peut ainsi être écrit comme une suite de cartes perforées. Les cartes perforées apparaissent avec les premiers automates et trouvent une application industrielle notable au début du XIX^e^ siècle avec les métiers à tisser Jacquard : les motifs à tisser sont encodés sous la forme de cartes perforées que la machine peut lire. Le concept sera ensuite repris par Charles Babbage pour sa machine analytique, puis par IBM au milieu des années 1930, utilisant le format Hollerith standard de quatre-vingts colonnes. Les premiers ordinateurs modernes tels le Harvard Mark I étaient réalisés en partenariat avec IBM. Le format Hollerith a donc été utilisé pour sa programmation. C'est ainsi que les cartes perforées sont devenues des outils de programmation informatique. Voir *Programme*.

Source : Williams, Michael R. *A History of Computing Technology*. Englewood Cliffs : Prentice-Hall, 1985, p. 253-258.

CMS (*CONTENT MANAGEMENT SYSTEM*) : Systèmes de gestion de contenus. On désigne par cet acronyme des outils d'écriture de sites web qui permettent la conception et la mise en ligne d'un site par le biais d'un éditeur de texte. L'édition du code HTML et la mise en ligne des fichiers par le protocole FTP sont prises en charge par le CMS. Un CMS est donc une couche logicielle qui facilite l'écriture d'un site web, car il n'est plus nécessaire de coder directement la page en langage HTML. Voir *Code (informatique) ; HTML*.

CODE (INFORMATIQUE) : Texte « [...] détaillant formellement le déroulement d'un programme informatique ». Le code informatique comporte une part destinée à la machine (le code comme suite d'instructions à exécuter) et une part destinée aux humains (le code comme texte où l'on explique les choix faits dans le programme, sa structure, etc.). Le code est toujours rédigé en suivant la syntaxe propre à un langage de programmation. Voir *Langage de programmation*.

Source : Doueihi, Milad. *Pour un humanisme numérique*. Paris : Éditions du Seuil, 2011, p. 170.

CODAGE / ENCODAGE (INFORMATIQUE) : Action de spécifier la structure d'un format de données ou les instructions qu'une machine peut exécuter. L'encodage consiste donc à construire la correspondance (le code) entre une séquence de caractère binaire (le langage machine) et ce que cette séquence

déclenche comme action, où à quel type de donnée elle renvoie (dans le cas d'un format de données). Voir *Binaire (langage) ; Format*.

CODE SOURCE : Ensemble des documents détaillant le déroulement d'un programme informatique. Le code source est ce qui permet, une fois compilé (c'est-à-dire converti en langage machine) puis exécuté, d'ouvrir un programme informatique. Voir *Code (informatique) ; Langage machine ; Programme*.

COMMENTAIRES (CODE) : Éléments du code informatique qui sont ignorés par la machine, mais qui apparaissent dans le code et peuvent donc être lus par des humains. Les commentaires servent en général à expliciter les choix des développeurs, apporter des compléments d'information sur la structure du programme, mais ils peuvent avoir de multiples autres usages. Voir *Code (informatique)*.

CONVERTIBILITÉ : Faculté permise par le calcul de poser un plan d'équivalence entre deux objets ou deux valeurs. On peut alors en faire le dénombrement. Par exemple, lorsque l'on calcule les jours en faisant des entailles dans une écorce de bois, chaque entaille représente un jour. On pose ainsi une équivalence entre un objet matériel (l'entaille) et une durée de temps (une journée).

CÔTÉ CLIENT : Lorsqu'un logiciel échange des données avec un serveur, par exemple dans le cas d'un navigateur Internet qui cherche à accéder à une page web, toutes les opérations qui sont effectuées par l'ordinateur de l'utilisateur (l'ordinateur faisant la requête au serveur) sont dites « côté client ». Un service de stockage de documents en ligne peut mettre en place un encryptage côté client, ce qui signifie que l'encryptage des données est effectué par l'ordinateur expédiant les données et non par le service en ligne. De la même façon, dans le protocole HTTP utilisé pour le Web, le navigateur (côté client) envoie une requête à un serveur, qui lui renvoie un document en HTML (le code source de la page demandée). Le code source est ensuite interprété côté client, par le navigateur. Voir *côté serveur ; HTTP*.

CÔTÉ SERVEUR : Lorsqu'un logiciel échange des données avec un serveur, toutes les opérations qui sont effectuées par le serveur sont appelées « côté serveur » et le logiciel ne peut y accéder. Par exemple, lorsque l'API de Facebook génère un jeton d'identification pour autoriser une application à accéder aux données d'un utilisateur de la plateforme, l'encryptage ainsi que l'identification du jeton a lieu côté serveur. Notre navigateur (côté client) ne peut y avoir accès, ce sans quoi il y aurait de sérieuses failles de sécurité : nous pourrions accéder à de nombreuses données personnelles en récoltant les différents jetons. Voir *Côté client ; Jeton*.

CSS (*CASCADING STYLE SHEET*) : Feuilles de style en cascade. Document servant uniquement à spécifier la mise en forme d'un texte numérique. Le concept de CSS repose sur la stricte séparation, dans un texte, entre son contenu linguistique et sa mise en forme. Le contenu linguistique du texte est écrit dans un document (par exemple un fichier XML ou HTML) et sa mise en forme dans un autre (le CSS). Le CSS est aujourd'hui un format standard

du Web. L'aspect graphique d'une page web est donc essentiellement décidé par le CSS et non par le code HTML. Voir *HTML*.

D

DÉVELOPPEUR : Personne dont le métier est d'écrire des programmes informatiques à l'aide de langages de programmation. Il existe certaines spécialités, selon le langage de programmation de prédilection et/ou le secteur industriel concerné : développeur web, développeur multimédia… Voir *Langage de programmation*.

DISCRET (MATHÉMATIQUE/INFORMATIQUE) : On appelle ensemble discret un ensemble dont les parties sont des éléments séparés les uns des autres. Le discret s'oppose au continu. Par exemple, l'alphabet est un ensemble discret au sens où on passe de A à B, puis de B à C sans étape intermédiaire. L'ensemble des entiers naturels (1, 2, 3, etc.) est un ensemble discret. En revanche, l'ensemble des nombres réels (nombre auxquels on peut rajouter des décimales) est un ensemble continu. Entre 1 et 2, il existe une infinité de valeurs : 1,1 ; 1,11 ; 1,111, etc.

DISCRÉTISATION (INFORMATIQUE) : La discrétisation est l'action de rendre discret, c'est-à-dire « […] *l'opération selon laquelle un contenu est inscrit en un langage constitué d'unités discrètes indépendantes les unes des autres* […] ». La dicrétisation est une opération fondamentale en informatique en ceci qu'elle est une condition de possibilité du calcul. Ce qu'on nomme « numérisation » est donc avant tout une discrétisation. Par exemple, lorsqu'une peinture (qui est de l'ordre du continu : chaque coup de pinceau se fond avec l'autre pour composer l'image) est numérisée, on transforme l'ensemble continu (les marques de peinture) en un ensemble discret (un agencement de pixels). C'est cette discrétisation qui permet la manipulation calculatoire par un ordinateur. Voir *Calcul (informatique) ; Pixel*.

Source : Bachimont, Bruno. De l'hypertexte à l'hypotexte : les parcours de la mémoire documentaire. *Téchnologies, Idéologies, Pratiques*. 1999, n° 4, p. 200.

DOCUMENTATION (API) : Documents expliquant la structure des données, ainsi que les méthodes d'accès à celles-ci, disponibles à travers une API. Dans le cas des API web, la documentation est disponible en ligne, sur un site dédié mis en place par la plateforme ayant ouvert une API.

DOM (*DOCUMENT OBJECT MODEL*) : Le DOM est une interface intégrée aux navigateurs web qui représente le contenu d'une page sous la forme d'une arborescence d'objets. Il permet ainsi à des scripts, par exemple, de modifier le contenu de la page web, en passant par le DOM. Voir *Objet (informatique)*.

E

ÉCRITURE-CALCUL AUTOMATIQUE : Écriture propre aux machines computationnelles telles qu'elles ont été théorisées depuis le modèle de Turing en 1937. L'écriture-calcul possède cinq propriétés. C'est une écriture formelle, monosémique, discrète, combinatoire et unidimensionnelle. C'est sous ces conditions que cette écriture peut devenir un calcul automatisable, c'est-à-

dire réalisable par une machine. Tous les ordinateurs contemporains sont des machines d'écriture, au sens où ils écrivent selon le modèle de l'écriture-calcul. La représentation typique de cette écriture-calcul est le langage binaire : 01001000010001010001111 0…. L'écriture-calcul produit ce que Bruno Bachimont nomme la « raison computationnelle ». Voir *Calcul ; Langage binaire.*

Source : Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Communication +1* [en ligne]. 2016, vol. 5, nº 2.

ÉCRITURE-TEXTE : Type d'écriture propre aux êtres humains et à leur appareil perceptif. L'écriture-texte a quatre caractéristiques. C'est une écriture combinatoire, bi ou tridimensionnelle, dont l'interprétation est contextuelle et dont les signes sont fondamentalement polysémiques. Cette écriture produit des textes, c'est-à-dire des objets sur lesquels sont inscrits un agencement de signes visibles et lisibles. Le texte est ouvert à de multiples interprétations, en vertu des deux dernières caractéristiques de l'écriture-texte. Cette dernière produit ce que Jack Goody nomme la « raison graphique ». L'écriture-texte désigne la plupart, si ce n'est toutes, les pratiques d'écriture humaines. Ce terme a comme principale utilité de faire saisir, par contraste, l'altérité constitutive de l'écriture-calcul. Voir *Écriture-calcul.*

Source : Goyet, Samuel et Collomb, Cléo. Do Computers Write on Electric Screens? *Communication +1* [en ligne]. 2016, vol. 5, nº 2.

ÉNONCIATION COMPUTATIONNELLE : Dans un écrit d'écran, part de l'énonciation qui revient aux entités computationnelles dans le texte lisible. L'énonciation computationnelle désigne le fait que tout écrit d'écran est le résultat, au moins en partie, de l'exploration de tous les possibles d'un ensemble combinatoire par le biais du calcul formel. Ainsi, ce qu'on lit à l'écran est l'un des possibles retenus par les entités computationnelles à l'œuvre dans la production de ce texte. Par exemple, la page de résultat Google est générée par un algorithme (le PageRank) qui scanne un ensemble de documents pour les discriminer et les classer par ordre de pertinence – telle que définie par le PageRank. Il choisira de mettre en avant les dix ou quinze documents les plus pertinents. Le même tri pourrait être effectué par un humain, mais il y a une différence radicale avec le tri humain : l'algorithme scanne effectivement tous les documents à sa disposition ; cette exploration se fait par le biais d'un calcul combinatoire, permettant une telle exhaustivité. Cette différence de nature (le calcul) permet une différence de degré (masse des volumes scannés). Voir *Calcul ; Entités computationnelles.*

ÉNONCIATION MACHINIQUE : Dans un texte, part de l'énonciation qui revient aux machines ayant été impliquées dans la production de ce texte. Les caractéristiques de cette énonciation dépendent de la machine utilisée. Par exemple, dans un livre réalisé grâce à une presse mécanique, le papier portera la marque de ce pressage et le texte lu sera sémiotiquement différent d'un texte imprimé en *offset,* qui repose sur un procédé technique différent, similaire à la décalcomanie. Dans le cas des écrits d'écran, l'énonciation machinique est caractérisée par l'automatisme – l'ordinateur modifie par lui-même son état physique, même s'il n'en est pas à l'initiative – et par le calcul. Voir *Énonciation computationnelle ; Calcul.*

ENTITÉS COMPUTATIONNELLES : Entités logicielles ou matérielles qui réalisent les opérations de calcul au sein d'une machine computationnelle. Les entités computationnelles constituent donc une première couche d'abstraction par rapport à la matérialité des machines computationnelles. Un logiciel, un algorithme, un abréviateur d'URL sont des exemples d'entités computationnelles. Ce sont ces entités qui sont en nombre indéfini dans une machine. Voir *Machine computationnelle ; Calcul.*

✦ F ✦

FONCTION (PROGRAMMATION) : Dans un programme informatique, bloc de code sous forme algorithmique qui permet d'accomplir une tâche répétitive. Une fonction commence toujours par la déclaration de son type, de ses valeurs initiales. Le corps de la fonction décrit les règles à appliquer à ces valeurs, la fonction retourne ensuite un résultat. Bien souvent, une fonction obéit à une structure conditionnelle (si x, alors y, etc.) et dans tous les cas son fonctionnement est algorithmique : elle applique des règles formelles de calcul à des données de départ pour retourner un résultat. Du point de vue pratique, une fonction permet d'économiser du temps dans l'écriture des programmes et de rendre ces derniers plus lisibles : on peut repérer en un coup d'œil quelles sont les étapes logiques du programme en regardant ses différentes fonctions. Voir *algorithme.*

FORMALISME / FORMEL (CALCUL) : Est formel « [...] ce qui est exprimable par le biais de symboles univoques », dont le sens contextuel est évacué au profit de la seule forme du symbole. C'est pourquoi nous parlons de symboles et non de signes. Le formalisme est en effet fondé sur la dualité du signe linguistique, entre signifiant et le signifié. La notation formelle évacue la question du signifié – ce à quoi le signe renvoie – pour ne se préoccuper que de la forme du signe – son signifiant. Le signe ainsi évidé – devenu symbole – il peut se prêter à des manipulations automatisables comme le calcul, où chaque symbole se voit attribué un seul signifié. Le formalisme est donc à la base de la monosémie de l'écriture-calcul et par conséquent indispensable au calcul informatique. Par exemple, A ≠ non-A est l'expression formelle d'un rapport qui s'abstrait de ce à quoi renvoie A pour pouvoir appliquer des règles logiques et calculatoires. A peut désigner n'importe quelle idée ou chose du monde, la manipulation n'en restera pas moins vraie d'un point de vue formel. On voit d'ailleurs par cet exemple que le formalisme nécessite la mise au point d'une notation graphique spécifique, adaptée à ces manipulations. Voir *Calcul ; Écriture-calcul.*

Source : Lassègue, Jean. *Pour une anthropologie sémiotique ; recherches sur le concept de Forme symbolique*. Habilitation à diriger des recherches. Paris : Paris-Sorbonne, 2010, p. 8.

FORMAT (INFORMATIQUE) : Façon conventionnelle d'encoder des données. Pour une même suite binaire (par exemple 00011100 pour prendre l'exemple d'un octet, c'est-à-dire huit *bits*, huit unités élémentaires du calcul), ce qu'exprimera cet octet sera différent selon le format utilisé. Dans le cadre d'un logiciel de retouche d'image, il pourra exprimer la coloration de quelques pixels ; dans un logiciel de compression audio, il pourra désigner une certaine fréquence sonore. Le for-

mat est donc ce qui convertit contextuellement et culturellement, dans un usage précis, une suite formelle de symboles binaires. Par extension, on désigne par « format de fichiers » les extensions accolées au nom d'un fichier qui précise l'encodage utilisé pour convertir les données. Voir *Calcul (informatique)*.

FORMES-TEXTES : Nous appelons formes-texte les formes récurrentes, les « formes repères » qui composent tout texte et en facilitent la lecture par reconnaissance de ces formes. Elles se stabilisent graphiquement par usage répété, voire par standardisation industrielle dans le cas des « petites formes ». Le paragraphe est une « forme-texte », le titre également, ainsi que les boutons « J'aime » des écrits de réseau ou une carte ancrée dans un site web. En tant que formes, elles sont travaillées pour être visibles et reconnaissables, par exemple par l'usage du blanc ou du cadrage. En tant que parties agençables d'un texte, nous appelons ces formes « formes-textes ». Le même terme est employé par Caroline Angé dans son travail sur le fragment. Si notre point de départ sémiotique est le même, nous avons cherché à parler des formes-textes de façon plus globale que la seule question du fragment, c'est pourquoi notre utilisation du terme est un peu différente.

Sources : Angé, Caroline. Le fragment comme forme texte : à propos de Fragments d'un discours amoureux. *Communication & langages*. 2007, nº 152, p. 23-34 ; Pédauque, Roger T. *Le document à la lumière du numérique*. Caen : C & F éditions, 2006, p. 106.

G

GRAPHE : Représentation cartographique et formelle de systèmes sous la forme de nœuds reliés par des liens. On peut par exemple faire le graphe d'une ville, d'un écosystème, des relations entre profils socionumériques.

GRAPHE SOCIAL : Outil mis au point par Facebook en 2010 qui représente sous la forme d'un graphe les relations qu'entretient un utilisateur Facebook avec l'ensemble des autres profils et pages Facebook. Le Graphe Social, via le protocole *Open Graph*, s'est depuis étendu à tout le Web. En théorie, ce graphe peut donc intégrer n'importe quelle interaction entre un utilisateur Facebook et une page web, à condition que, dans son code source, la page consultée ait été décrite au format *Open Graph*. Les boutons et autres *widgets* sont notamment l'un des vecteurs privilégiés d'extension de ce Graphe Social, puisque cliquer sur un bouton Facebook sur une page web intègre automatiquement cette page au graphe social. Voir *Code Source ; Graphe ; Widget*.

H

HEXADÉCIMAL : Système de numérotation en base seize, qui utilise les neuf premiers entiers naturel en notation arabe (de 0 à 9) complétés par les six premières lettres de l'alphabet de A à F, A ayant donc comme valeur « 11 », B « 12 » et ainsi de suite jusqu'à F=16. L'hexadécimal est particulièrement employé en informatique comme format de données, car il permet une écriture plus ramassée – et donc plus lisible par un humain – du langage bi-

naire, en vertu du fait que l'hexadécimal permet de rassembler quatre chiffres en base deux (1100) en un chiffre en base seize. Dans l'exemple que nous venons de donner, 1100 est égal à douze en base dix, celle de notre notation ordinaire (0x20 + 0x21 + 1x22 + 1x24 = 0 + 0 + 4 + 8 = 12, sachant que l'écriture informatique se lit de droite à gauche). En hexadécimal, «1100» peut donc s'écrire « B ». On voit par là que c'est un format plus ramassé que la base dix, tout en étant un multiple de quatre, ce qui permet de représenter facilement les octets (groupe de huit *bits*, standard employé en informatique). Voir *Binaire (Langage) ;* voir *Chapitre II. 1. B. b* p. 170 pour une explication plus détaillée du concept de base.

HTML (*HYPERTEXT MARKUP LANGAGE*) : Langage à balise, standard de description des pages web. Le code écrit en HTML décrit la structure globale de la page web, son contenu linguistique, textes, images ou vidéos. Souvent, le document HTML est appelé « code source » de la page. Il n'est toutefois pas le seul document impliqué dans la production d'une page web. Il est complété par différents scripts, par le CSS et par le DOM. Voir *Balise ; CSS ; DOM ; Langage de programmation ; Script*.

HTTP (*HYPERTEXT TRANSFER PROTOCOL*) : Protocole de transmission des fichiers HTML. Il est organisé en clients et serveurs, les clients faisant des requêtes aux serveurs, qui retournent les documents demandés s'ils sont disponibles. Lorsqu'on ouvre un navigateur et que nous entrons une adresse dans la barre de navigation, cela revient en termes techniques à ouvrir un client HTTP (le navigateur) qui va demander un document (le code source de la page HTML qui se trouve à l'adresse entrée dans la barre de navigation) à un serveur, lequel va transmettre au client le document. Le client va alors se charger lire le document HTML et de l'afficher, ce qui produit la page web lue par l'internaute. Les API web sont ainsi nommées car l'échange de données entre l'API et le développeur passe par le protocole HTTP et plus spécifiquement par l'architecture REST (*Representational State Transfer*), intégrée au protocole HTTP. Voir *Code source ; Côté client ; Côté serveur ; HTTP ; Web.*

INDUSTRIALISATION : Processus de production composé selon Pierre Mœglin de trois dynamiques : le « [...] recours à des systèmes techniques faisant [...] l'économie de la force et du temps de travail humain » ; « [...] l'adoption de méthodes d'organisation et de gestion [...] » afin d'optimiser l'utilisation des systèmes techniques adoptés ; « l'avènement d'un état d'esprit [...] privilégiant l'utilisation de tous les moyens humains et techniques pour concourir au rendement et à la productivité ». Yves Jeanneret résume ces trois dynamiques par le triptyque instrumentation ; standardisation, idéologisation. Voir *Standardisation*.

Sources : Mœglin, Pierre. À la recherche de l'industrialisation du tutorat à distance. *Distances et savoirs*. 2005, vol. 3, nº 2, p. 251 ; Jeanneret, Yves. *Critique de la trivialité : les médiations de la communication, enjeu de pouvoir*. Paris : Éditions Non Standard, 2014, p. 10-11.

INTERFACE (INFORMATIQUE) : Dispositif de mise en contact entre deux composants informatiques, qu'ils soient des composants logiciels ou matériels. Une interface spécifie, par son organisation,

les méthodes d'accès aux composants et les conditions d'échange de données entre ces deux composants. Une interface peut être crée entre deux composants matériels (par exemple un câble adaptateur entre un écran de télévision et un ordinateur), entre un composant matériel et logiciel (par exemple un logiciel d'accès aux données d'un disque dur) ou entre deux logiciels, par exemple une API. Voir *Logiciel ; Matériel.*

INTERFACE GRAPHIQUE : Historiquement, on nomme interface graphique (de *Graphical User Interface*) une interface entre humains et ordinateurs qui passe par un régime visuel métaphorique, à l'image des systèmes d'exploitation actuels qui font ouvrir des « dossiers » sur un « bureau », etc. Plus largement, nous entendons par interface graphique tout dispositif de mise en contact entre humains et machines informatiques qui convertit, dans les termes de la raison graphique, les opérations de calcul de la machine. De cette façon, une carte perforée, une ligne de commande ou une fenêtre Windows sont toutes des interfaces graphiques. Voir *Calcul ; Interface.*

INTEROPÉRABILITÉ (INFORMATIQUE) : Capacité d'un objet informatique (logiciel, matériel, document, format de données…) à pouvoir fonctionner dans d'autres environnements, avec d'autres systèmes, d'autres logiciels, sur d'autres machines. Par exemple, on peut dire qu'un document Word est peu interopérable, car il est fort probable que l'ouverture de ce fichier dans un autre logiciel que Microsoft Word provoque une perte de données. Les formats de données et les interfaces – dont les interfaces de programmation – sont essentiels pour construire l'interopérabilité. Ils permettent d'accéder à la structure de l'objet afin d'adapter cet objet aux différents environnements techniques dans lequels il serait amené à circuler. Voir *Interface ; Format.*

IRRÉDUCTIBILITÉ COMPUTATIONNELLE : Part de l'énonciation des écrits d'écran qui revient en propre au calcul et qui fait la spécificité de ces textes. Tout texte produit par une machine computationnelle portera la marque de ce calcul, soit le fait que le texte affiché est l'un des possibles actualisés parmi un ensemble de textes potentiels actualisables (car parcourus par le calcul). Très concrètement, cela signifie que tout texte numérique se caractérise par le fait qu'il pourrait être agencé autrement, en vertu des propriétés techniques du calcul. Voir *Calcul ; Machine computationnelle.*

Source : Collomb, Cléo. *Un concept technologique de trace numérique*. Thèse de doctorat. Compiègne : Université de Technologie de Compiègne et Bruxelles : Université Libre de Bruxelles, 2016, p. 297, 330 ; Chapitre V. 5. C de la présente thèse.

J

JAVASCRIPT : Langage de programmation crée en 1995 par Brendan Eich, alors ingénieur informatique chez Netscape Communications. Particulièrement utilisé sur le Web, le JavaScript permet d'intégrer dans des documents HTML des mini-programmes, ou « scripts ». Ces scripts provoquent des évènements conditionés à certaines actions de l'utilisateur – alors que l'HTML ne le permet pas, étant un langage de description. Les scripts sont exécutés côté client, par le navigateur web. Le JavaScript a fait l'objet d'une standardisation internationale en 1998 (ISO/CEI 16262). Voir *Côté client ; HTML ; Scripts.*

JETON (API) : Clé encryptée, générée automatiquement par une API lors de l'authentification d'un développeur externe ou d'un internaute *via* une application qui demande l'accès aux données personnelles de ce dernier. Le fonctionnement des jetons est similaire au *symbolon* grecs. Un jeton est composé de deux parties : un code détenu par l'API ; un code détenu par le développeur. Lorsque le développeur ou l'application formule une requête à l'API, cette dernière lit le jeton, vérifie si ce jeton existe, si le code correspond au code détenu par l'API. Si les codes correspondent, l'API retourne les données. Le jeton permet aussi d'identifier, selon la nature de l'encryptage, le niveau de confidentialité des données auxquelles le développeur a accès. Voir *API ; Requête (API)*.

JSON : *JavaScript Object Notation*. Format de représentation de données sous la forme d'objet, en suivant la syntaxe du JavaScript. Le JSON est le format principal choisi par les API de notre corpus pour la restitution des données. Le JSON a l'avantage d'être un format très manipulable, car directement intégrable dans JavaScript et donc dans une page web. Il y'a par conséquent moins de conversions et de manipulations à faire entre les résultats de la requête API et l'utilisation de ces résultats dans une application web. Voir *Format ; JavaScript ; Objet (informatique)*.

L

LANGAGE ASSEMBLEUR : Langage de programmation de bas niveau, première conversion du langage machine binaire en instructions sous une forme plus lisibles pour l'humain, par exemple en hexadécimal ou par des mots permettant de mémoriser plus facilement à quelle opération machinique correspond la séquence de caractères. Par extension, on nomme « assembleur » le programme qui fait la conversion des instructions depuis langage assembleur vers le langage machine. Apparus dès les années 1950, les langages assembleurs ont la particularité d'être spécifiques à chaque type de processeur, au contraire des langages de haut niveau. Voir *Architecture matérielle ; Binaire (langage) ; Langage de bas niveau ; Langage de haut niveau*.

LANGAGE DE BAS NIVEAU : Langage de programmation proche, dans sa syntaxe, du langage machine. Le langage assembleur est, par exemple, un langage de bas niveau. Les opérations que permettent les langages de bas niveau sont nombreuses et particulièrement rapides car les instructions données en bas niveau ne nécessitent pas autant de conversions qu'avec des langages de haut niveau. Les risques d'erreur sont également moins importants. En revanche, la programmation demande un savoir-faire plus spécialisé, car la syntaxe des langages de bas niveau est très éloignée des langages naturels. Voir *Binaire (Langage) ; Langage assembleur ; Langage de haut niveau*.

LANGAGE DE HAUT NIVEAU : Langage de programmation dont la syntaxe est proche des langages naturels comme l'anglais, ou utilisant des symboles mathématiques courant. La mise au point de ces langages dès le début des années 1950 (Fortran en 1957 ; Lisp, Algol en 1958 ; Cobol en 1960) a été très importante pour le développement de la programmation, parce qu'ils automatisent la conversion des instructions en langage machine. Ainsi, ils ne nécessitent pas un savoir-faire aussi spécialisé que le langage assembleur par exemple. De plus, les programmes écrits avec des langages de haut niveau ont l'avantage de pouvoir être exécutés sur plusieurs machines, selon le mode de compilation utilisé. Voir *Langage de bas niveau ; Langage assembleur.*

LANGAGE MACHINE : Voir *Binaire (langage).*

LANGAGE (DE PROGRAMMATION) : Syntaxe utilisée pour représenter les instructions d'un programme informatique. Il existe de nombreux langages de programmation, chacun ayant ses spécificités et son degré d'abstraction vis-à-vis de la machine. La plupart des langages sont procéduraux – c'est-à-dire qu'ils contiennent des séquences de calcul s'effectuant sous certaines conditions logiques – mais il existe aussi des langages déclaratifs comme l'HTML. Malgré une base logique et formelle commune, chaque langage de programmation possède un style, induit une certaine façon de programmer selon sa construction et sa complexité. Certains langages sont dédiés à des tâches spécialisés (le PHP pour gérer les échanges entre une page web et une base de données par exemple), mais la plupart sont généralistes. Il existe des paradigmes de programmation selon les langages de programmation dominant d'une époque, comme c'est le cas par exemple avec la programmation orientée objet, paradigme auxquel correspondent les langages orientés objet : C++, Java, etc. Voir *Langage de bas niveau ; Langage de haut niveau ; Langages orientés objet ; Programme.*

LANGAGES ORIENTÉS OBJET : Types de langage de programmation qui reposent sur le concept d'objet. Dans ces langages, la programmation se fait principalement en définissant des « objets » au sens informatique du terme : des entités ayant certaines propriétés et des méthodes d'accès à ces propriétés. Programmer, dans le modèle de la programmation orientée objet, c'est écrire les différentes interactions entre des objets. Un langage orienté objet propose une syntaxe qui permet d'écrire ces objets, leurs propriétés et méthodes d'accès. Le premier langage orienté objet, Simula 67, a été écrit à la fin des années 1960, suivi par Smalltalk dans les années 1970. Mais c'est dans les années 1980 et 1990 que la programmation orientée objet est devenue le modèle dominant, notamment autour du succès des langages C++, Java puis Visual Basic. Voir *Objet (informatique) : Langage de programmation.*

Source : Article « Object-Oriented Programming » dans Henderson Harry (dir.). *Encyclopedia of Computer Science and Technology*. New-York : Facts on File, 2009 [2003], p. 339-341.

LITTÉRATIE : « Aptitude à comprendre et à utiliser l'information écrite dans la vie courante, au travail et dans la collectivité en vue d'atteindre des buts personnels et d'étendre ses connaissances et capacités. » La littératie ne se limite pas à l'apprentissage technique de la lecture

et de l'écriture, elle inclut également la capacité à utiliser ces techniques pour produire un raisonnement. Le concept de littératie est particulièrement plastique et s'adapte selon les environnements d'écriture / lecture. On parle ainsi aujourd'hui de « littératie numérique » pour désigner l'ensemble des aptitudes nécessaires à la production de connaissances dans un environnement numérique. Les contours de cette littératie restent toutefois débattus.

Sources : OCDE, « La Littératie à l'ère de l'information » [en ligne]. Rapport, 2000. [Consulté le 1er septembre 2017]. Disponible à l'adresse : http://www.oecd.org/fr/edu/innovation-education/39438013.pdf ; Le Deuff, Olivier. Littératies informationnelles, médiatiques et numériques : de la concurrence à la convergence ? *Études de communication. Langages, information, médiations*. 2012, nº 38, p. 131-147.

LOGICIEL (*SOFTWARE*) : Ensemble des instructions, de leur séquence, ainsi que des données nécessaires à leur réalisation, qui « [...]ordonne à un ordinateur de faire une tâche spécifique ». Le logiciel est cet ensemble tripartite (instructions ; séquencage ; données) qui permet l' usage spécifique d'un ordinateur. De ce point de vue, un système d'exploitation, un navigateur web ou un outil de gestion de base de données sont tous des logiciels. L'écriture des logiciels est aujourd'hui un marché industriel, avec ses méthodes, ses outils, ses compétences spécialisés. Un logiciel peut être dit propriétaire (l'utilisateur n'a aucun droit sur le code source du logiciel), libre (le code peut être consulté par tous et modifié) ou ouvert (le code source est lisible, mais pas forcément modifiable). Le logiciel (*software*) est une couche d'abstraction dans l'usage d'un ordinateur qui s'oppose au matériel (*hardware*). Le logiciel est historiquement lié aux composants matériels de la machine sur lequel il s'exécute, mais l'évolution des rapports entre logiciel et matériel se caractérise par un idéal d'abstraction de la matérialité. Cette abstraction est bien idéale – un logiciel est toujours dépendant d'une machine – mais puissante du point de vue imaginaire et économique. Voir *Application ; Code (informatique) ; Code Source ; Matériel ; Programme*.

Source : Ceruzzi, Paul E. *A History of Modern Computing*. Cambridge : MIT Press, 2003 [1998], p. 80.

LOGIQUE BOOLÉENNE : Logique formelle suivant la syntaxe du logicien anglais Georges Boole (1815-1864). L'algèbre booléen est une formalisation binaire des raisonnements logiques, en utilisant notamment des opérateurs de conjonction (« ET ») ou de disjonction (« OU »). On peut ainsi manipuler des raisonnements logiques par le biais d'un calcul binaire. C'est cette possibilité de l'algèbre de Boole qui sera reprise en informatique et qui est aujourd'hui implémentée directement dans les circuits électroniques. Voir *Architecture matérielle ; Formalisme*.

M

MACHINE COMPUTATIONNELLE : « Par machine computationnelle, nous entendons l'ensemble des objets techniques dont le fonctionnement repose sur du calcul binaire intégré dans une machinerie électronique. » Un ordinateur, une tablette numérique, un téléphone intelligent sont des machines computationnelles. Précisons que dans cette thèse, le terme machine computationnelle désigne la machine physique, matérielle et non l'ensemble des programmes de la machine. Voir *Calcul*.

Source : Collomb, Cléo et Goyet, Samuel. Meeting the machine halfway : Vers une sémiopolitique de l'agir computationnel. Communication au colloque international *Reconfiguring Humans and Non-Humans: images, texts and beyond*. Université de Jyväskyllä, Finlande, 29-30 octobre 2015. [Manuscrit en ligne].

MACHINE AUTOMATIQUE : Objet technique qui, grâce à des moyens mécaniques, hydrauliques ou électroniques, peut suivre une séquence d'actions programmées sans intervention humaine lors du déroulement de cette séquence. Une horloge, un automate, ou un métier à tisser Jacquard sont des exemples de machines automatiques. Si toute machine computationnelle est une machine automatique, l'inverse n'est pas vrai : toute machine automatique n'est pas computationnelle. Dans les exemples que nous venons de donner, aucune de ces machines n'est computationnelle, en ceci que leur fonctionnement ne repose pas sur du calcul formel mais sur des principes mécaniques – même si on peut par ailleurs formaliser leur fonctionnement pour le faire réaliser par une machine computationnelle. Voir *Machine computationnelle.*

MASQUAGE D'INFORMATION (***INFORMATION HIDING***) : Principe de programmation mis au point par David Parnas dans le paradigme de la programmation modulaire. Dans ce paradigme, un programme est découpé en différents modules, selon les tâches à accomplir dans le programme. On attribue à chaque développeur ou équipe de développeurs l'écriture d'un module. Le masquage d'information consiste à construire pour chaque module une interface qui permet de comprendre la fonction globale du module sans avoir besoin de comprendre tous les processus en jeu. Un développeur peut alors se contenter de connaître l'interface et le rôle du module. Il va ainsi pouvoir se concentrer sur la partie du programme dont il a la charge. Voir *Interface (informatique) ; Programme.*

MATÉRIEL (***HARDWARE***) : Ensemble des composants électroniques qui constituent une machine computationnelle. Le *hardware* se compose de l'écran, du clavier, du processeur, de la carte graphique, des disques durs… Ces composants matériels doivent être complétés par des logiciels, pour que l'utilisateur puisse intervenir sur cette matérialité. Voir *Logiciel (software).*

MÉTADONNÉES : Ensemble des données décrivant d'autres données, par exemple la structure d'un fichier, sa date de création, etc.

MIDDLEWARE : Mot valise composée de *software* et de *middle*. Littéralement « logiciel de l'entre-deux » ou « logiciel au milieu ». On appelle *middleware* tous les logiciels servant à connecter deux applications afin d'en assurer le bon fonctionnement mutuel. Un *middleware* n'est pas une interface à proprement parler : c'est un programme entre deux autres programmes et non un moyen d'accès à l'un de ces programmes. Les *middleware* sont devenus particulièrement importants en informatique avec le développement des réseaux comme Internet, qui demandent le fonctionnement en concorde de plusieurs machines et logiciels. Voir *Interface (informatique) ; Logiciel.*

Source : Volonino, Linda et Dalal, Pragati. Network Middleware. Dans : Bidgoli, Hossein (dir.), *The Handbook of Computer Networks. Volume 3 : Distributed networks, Network planning, Control, Management, and New Trends and Applications*. Hoboken : John Wiley & Sons, 2008, p. 33-44.

O

OBJET (INFORMATIQUE) : Représentation d'un objet physique sous forme informatique, c'est-à-dire formelle et selon la syntaxe d'un langage de programmation. Un objet est une entité qui a des attributs (ses caractéristiques) et des méthodes (les façons d'accéder à ces attributs). L'objet est un concept central dans le paradigme de la programmation orientée objet, où coder consiste à créer ces objets et ensuite en écrire les différentes manipulations. Par exemple, on peut récupérer grâce à l'API de Twitter des tweets en JSON (*JavaScript Object Notation*). Le document transmis par l'API est une représentation du tweet sous forme d'objet, avec ses différents attributs : date, auteur, contenu linguistique, etc. Cette représentation respecte la syntaxe du JavaScript.Voir *JavaScript ; JSON ; Langage de programmation ; Langage orienté objet*.

ORDINOGRAMME : Représentation du déroulé d'un programme sous forme de diagramme. Chaque étape logique du programme est représentée par un système de boîtes reliées par des flèches, ce système étant standardisé. L'ordinogramme est une technologie intellectuelle qui permet de représenter synthétiquement la complexité d'un programme informatique. Il est toutefois compliqué à utiliser lorsque les programmes excèdent une certaine taille et donc une certaine complexité. Voir *Programme*.

P

PETITES FORMES : « [...] formes récurrentes, de dimensions restreintes, qui lissent la composition des écrans ». Le concept de petites formes part du constat empirique de la dissémination de certaines formes sur les pages web : nuages de tags, index, cartes, formulaire de recherche... Toutes ces formes sont des « [...] éléments de la grammaire éditoriale [...] » des sites web contemporain et participent à l'industrialisation de l'écriture du Web. Ce sont en effets des formes à la fois standards – pour pouvoir se reproduire et se stabiliser sémiotiquement – et personnalisables, qui puisent dans différentes cultures du texte. Les API sont l'une des procédures techniques par lesquelles ces petites formes sont industrialisées. Voir *Industrialisation ; Standardisation*.

Source : Candel, Étienne, Jeanne-Perrier, Valérie et Souchier, Emmanuël. Petites formes, grands desseins. D'une grammaire des énoncés éditoriaux à la standardisation des écritures. Dans : Davallon, Jean (dir.), *L'économie des écritures sur le Web. Volume 1 : traces d'usage dans un corpus de sites de tourisme*. Cachan : Hermès Science – Lavoisier, 2012, p. 166-167.

PIXEL : Contraction de l'anglais *picture element* (élément d'une image). Le *pixel* est l'unité de mesure élémentaire d'une image matricielle, c'est-à-dire définie selon des repères orthonormés. La résolution d'un écran d'ordinateur est mesurée en *pixels*, signifiant que l'image est découpée selon des coordonnées de largeur et de hauteur, puis à chaque coordonnée est attribuée une couleur. C'est la carte graphique qui est chargée de calculer et d'attribuer la couleur à chaque *pixel*. Le *pixel* est un exemple de la discrétisation nécessaire au calcul informatique : une image ne peut s'afficher sur un écran d'ordinateur que parce qu'elle est trans-

formée en une suite discrète d'unités élémentaires. C'est la condition pour que la carte graphique puisse calculer cette image et donc l'afficher. Voir *Carte graphique ; Discrétisation*.

PLUGIN : Logiciel complémentaire qui se greffe sur un autre pour y ajouter une fonctionnalité. Le *plugin* a comme particularité de ne pas être indépendant de son logiciel hôte. Par exemple, le *plugin* « Adblock Plus », qui permet de ne pas afficher les publicités sur le Web, ne peut pas être installé seul. Il doit nécessairement être installé en complément d'un navigateur.

POLYCHRÉSIE : Mot composé des termes grecs *poly* (plusieurs) et *kreisten* (user de). Yves Jeanneret désigne par ce terme la capacité qu'ont les objets communicationnels à « [...] soutenir différentes logiques sociales » et à « [...] correspondre à plusieurs usages différents à la fois » C'est la polychrésie de ces objets qui expliquent leurs multiples réappropriations et transformations, en somme leur vie sociale. Par exemple, une API est polychrésique au sens où elle peut à la fois – et en même temps – servir à défendre une vision de l'économie collaborative, à écrire des pages web, à structurer des relations de dépendance économiques avec des développeurs.

Sources : Jeanneret, Yves. *Penser la trivialité. Volume 1, la vie triviale des êtres culturels*. Cachan : Hermès Science – Lavoisier, 2008, p. 84 ; Jeanneret, Yves. *Critique de la trivialité : les médiations de la communication, enjeu de pouvoir*. Paris : Éditions Non Standard, 2014, p. 12.

PORTE LOGIQUE : Composant informatique qui laisse passer ou retient le courant électrique selon les règles de la logique booléenne. Par exemple, une porte « ET » recevra deux influx électriques en entrée et produira en sortie un influx électrique à condition que les deux influx en entrée aient la même valeur (0 ou 1). On peut ainsi construire une machine comme un ensemble de propositions logiques intégrées dans des composants électroniques. Voir *Logique booléenne*.

PROGRAMMATION : Action d'écrire des programmes informatiques, le plus souvent à l'aide de divers langages de programmation. Voir *Programme ; Langages de programmation*.

PROGRAMME : Ensemble d'actions automatisées écrit à l'aide d'une syntaxe permettant l'exécution de ces actions par une machine. Un programme est ici quasi-synonyme de logiciel, mais nous insistons sur le fait que, si le logiciel est propre à l'informatique car reposant sur du calcul formel, le programme ne l'est pas. Ainsi, la partition d'un orgue de Barbarie ou le paquet de cartes perforées d'un métier Jacquard sont des programmes. Ce ne sont pas pour autant des logiciels. Voir *Logiciel*.

PROGRAMMATION MODULAIRE : Paradigme de programmation qui consiste à diviser un programme en « modules », chaque module étant une étape du programme. Ces modules sont ensuite dotés d'une interface afin que les développeurs écrivant le programme aient accès aux infos essentielles concernant chaque module. C'est le principe de « masquage d'information ». La programmation modulaire permet ainsi de mieux gérer la production de logiciels particulièrement lourds et nécessitant le travail de plusieurs équipes de développeurs. En attribuant à chaque équipe un module, on peut diviser et optimiser le travail des développeurs. Voir *Masquage d'information*.

PROGRAMMATION ORIENTÉE OBJET : Paradigme de programmation fondé sur le concept d'objet. Voir *Langage orienté objet ; Objet (informatique) ; Programmation.*

R

REQUÊTE (API) : Dans le cas des API web, la requête API désigne le fragment de code par lequel un développeur demande l'autorisation à l'API de récupérer des données. Pour être effectif, l'appel doit respecter une certaine syntaxe – décrite dans la documentation de l'API – et doit, dans la plupart des cas, inclure le jeton d'identification du développeur. L'API répond ensuite à cette requête et retourne soit les données demandées, soit un code d'erreur si la requête n'a pas aboutie. Les API sont dites web car le protocole de transmission de données utilisé pendant l'échange est le protocole HTTP, protocole standard du Web. Voir *Jeton ; Documentation.*

S

SCRIPT : Programme informatique, en général assez court, qui a comme particularité de pouvoir être interprété directement par un logiciel et donc de ne pas devoir être compilé (c'est-à-dire converti en langage machine afin de pouvoir être exécuté par la machine). Les scripts sont un moyen facile et rapide de formuler des instructions à un ordinateur, sans devoir passer par toute la phase de compilation. JavaScript est un exemple de langage de scripts spécialisé pour le Web. Voir *Binaire (langage).*

SIGNE PASSEUR : Yves Jeanneret et Emmanuël Souchier proposent d'appeler « signes passeurs » tous les signes des écrits d'écran qui permettent d'agir sur le texte à l'écran : icône cliquable, boutons, champ de recherche, flèches de navigation, « liens hypertextes »… Un signe passeur a trois caractéristiques : sa signification dépend de son emplacement dans la page ; il permet d'actualiser un texte possible ; il se désigne comme signe passeur et anticipe, dans sa forme, le texte auquel il donne accès. La notion de « signe passeur » fut élaborée pour prendre en compte l'épaisseur sémiotique des écrits d'écran tout en considérant leurs spécificités techniques, au contraire d'un terme comme « lien hypertexte » qui tend à naturaliser certaines conceptions du texte et de l'écriture.

Source : Jeanneret Yves et Souchier Emmanuël. Pour une poétique de l'écrit d'écran. *Xoana*, n°6, 1999, p. 100.

STANDARDISATION : L'une des dynamiques de l'industrialisation telle que définie par Pierre Mœglin. La standardisation désigne la mise au point de techniques et de procédures pour optimiser la reproduction des formes du texte que permet un outil technique. Par exemple, l'uniformisation des différents formats de papier (A4, A3, etc.) avec différents grammages eux aussi uniformisés sont des processus de standardisation des supports d'écriture qui facilitent la reproduction et la diffusion des textes en intégrant directement les outils de la chaîne éditoriale : traitements de texte, imprimantes… Voir *Industrialisation.*

Sources : Mœglin, Pierre. À la recherche de l'industrialisation du tutorat à distance. *Distances et savoirs*. 2005, vol. 3, n° 2, p. 251.

SUBROUTINE : Séquence d'un programme informatique qui est isolée puis écrite de telle façon à pouvoir être réutilisée dans d'autres programmes. La *subroutine* naît du constat que, dans un programme (ou « routine »), il arrive bien souvent de répéter des séquences de calcul utilisées dans d'autres programmes. En repérant ces séquences routinières, on peut alors constituer une bibliothèque de micro-séquences de calcul remobilisables sans qu'on ait à les réécrire constamment. La *subroutine* est un concept de base de programmation dès la fin des années 1940. Voir *Programme*.

SYSTÈME D'EXPLOITATION : Logiciel chargé d'organiser les relations entres applications et couches matérielles. Le système d'exploitation est en relation avec les composants matériels (carte graphique, mémoire, disque dur…) et les composants logiciels (applications, pilotes, protocoles réseau…), puis sert d'intermédiaire en gérant les besoins en ressources des logiciels. Le système d'exploitation est le premier logiciel qui s'exécute sur une machine et le plus important au sens où il conditionne le bon fonctionnement des autres logiciels. Windows, MacOS, GNU/Linux, Android sont des exemples de systèmes d'exploitation. Voir *Application ; Logiciel*.

T

TCP/IP : *Transfer Control Protocol / Internet Protocol*. Protocoles utilisés pour transmettre des données sur le réseau Internet. Le TCP/IP est composé de quatre couches, chacune assurant une fonction différente. La couche physique décrit le matériel utilisé pour la transmission des données (type de connexion, intensité des signaux, etc.). La couche de liaison de données définit la manière dont les données sont transportées. La couche réseau gère l'acheminement des données d'un point A à un point B du réseau. La couche application décode les informations reçues *via* la couche réseau selon le port utilisé. C'est au niveau de la couche application que se situe le protocole HTTP. Voir *HTTP*.

TECHNOLOGIE DE L'INTELLECT / TECHNOLOGIE INTELLECTUELLE : « […] outil régulé de gestion du nombre (de la complexité) opérant une traduction de l'évènement en document grâce à l'opération de conversion des dimensions ». Terme élaboré par Daniel Bell et Jack Goody dans les années 1970. Le concept de technologie intellectuelle, particulièrement au sens de Jack Goody, rend compte du fait que certaines techniques – par leurs caractéristiques propres ou les opérations qu'elles permettent – transforment les modes de pensée. Goody prend l'exemple de l'écriture et plus particulièrement de la forme du tableau. Selon lui l'écriture, parce qu'elle permet le classement d'entités sur un espace bidimensionnel, peut, dans certains contextes, produire des modes de pensée classificatoire comme la logique artistotélicienne. Une carte est une technologie intellectuelle, une base de données est une technologie intellectuelle. Dans cette thèse, nous défendons que la combinatoire est une technologie intellectuelle : elle est un outil de gestion de la complexité qui produit des formes spécifiques de pensée.

Source : Robert, Pascal. Qu'est-ce qu'une technologie intellectuelle *Communication & langages*. 2000, nº 123, p. 103.

TEXTE COMPUTATIONNEL : Agencement de signes produit par le biais du calcul formel. En vertu de l'hypothèse d'une « irréductibilité sémiotique », on peut penser que tout texte computationnel porte la marque du calcul. Les API sont une des entités computationnelles impliquées dans la production d'un texte computationnel. Comme tout texte, cet agencement est aussi une mise en visibilité ou au contraire une invisibilisation de certains acteurs du texte. Le terme de texte computationnel est quasi-synonyme d'écrit d'écran, mais déplace la focale vers le calcul (qui est une forme d'écriture) et les différents acteurs techniques du texte, précisément en raison de l'hypothèse d'une invisibilisation de ces acteurs. Voir *Calcul ; Machines computationnelles ; Entités computationnelles.*

Source : Souchier, Emmanuël. Le carnaval typographique de Balzac. Premiers éléments pour une théorie de l'irréductibilité sémiotique. *Communication & langages*. 2015, nº 185, p. 3-22.

THÉLÉMACENTRISME : De *théléma* (volonté). Idéologie de l'écriture propre au code informatique, qui repose sur une double croyance : que le code est le décalque d'une volonté humaine ; que la machine exécute l'intégralité du code écrit. Ainsi, le thélémacentrisme conduit à aborder les écrits d'écran comme le seul produit d'une volonté humaine, niant le rôle constitutif de la médiation technique. Le thélémacentrisme est une reformulation du logocentrisme, en l'adaptant aux spécificités techniques du code informatique. Le code est en effet une écriture de commande : on écrit des instructions à une machine. D'un point de vue imaginaire, cette spécificité technique peut entraîner l'idée que la machine ne fait que suivre la volonté de l'humain écrivant du code. Voir *Code informatique.*

TIMELINE : Littéralement : frise chronologique. Forme utilisée par Twitter pour organiser les tweets selon un ordre antéchronologique : les tweets les plus récents sont en haut de l'écran et viennent s'empiler sur les plus anciens au fur et à mesure de l'actualisation de la page. Voir *Tweet*.

TWEET : Littéralement : gazouillis. Message publié grâce à la plateforme Twitter. Du point de vue sémiotique, un tweet est la forme propre à Twitter, composée d'une photo de profil, d'un nom d'utilisateur, d'un court texte (limité à cent quarante caractères) ou d'une image, ainsi que de boutons de manipulation (répondre, aimer, republier). Le tweet est un exemple d'écriture architextuelle sous contrainte : on ne peut pas écrire sur ce site en dehors de la contrainte des cent quarante caractères, mais c'est précisément cette contrainte qui rend cette forme particulièrement circulante. Voir *Architexte*.

V

VALEUR BOOLÉENNE : En programmation, on nomme valeur booléenne une valeur qui ne peut avoir que deux états : vrai ou faux. Une variable booléenne aura donc comme valeur possible : *true* (vrai) ou *false* (faux), au contraire d'une variable *string* (chaîne) qui acceptera comme valeur toute chaîne de caractères. Voir *Variable (programmation).*

VARIABLE (PROGRAMMATION) : En informatique, on appelle variable un endroit dans un programme où stocker des informations pour la suite du programme. Pour cela, on associe une valeur (l'information que l'on veut stocker) à un

nom, que l'on écrira dans le programme chaque fois qu'on aura besoin de l'information stockée. « Déclarer » une variable, c'est indiquer à l'ordinateur quel sera le nom de la variable et sa valeur associée. Une fois la variable déclarée, l'ordinateur alloue un espace mémoire à cette variable. On peut ainsi réutiliser cette variable dans le programme. Selon les langages de programmation, la valeur des variables peut être de différents types. Par exemple, on distingue en JavaScript le type booléen (variable dont la valeur peut être soit juste, soit faux), le type *integer* (dont la valeur est un chiffre) ; le type *string* (dont la valeur est une chaîne de caractères) ; le type *array* (dont la valeur est une liste d'autres variables) ou encore le type *object*, dont la valeur est un objet au sens informatique du terme. Même si le contenu de deux variables est indentique, le type de ces variables est important car il augure de leurs possibles manipulations. Prenons deux variables : `name = true; name = "true"`. La première est de type booléen et veut dire : « il existe une variable `"name"` dont la valeur est vraie ». La seconde est de type *string* et veut dire : « il existe une variable `"name"` dont la valeur est la chaîne de caractères `"true"` ». Alors qu'on pourra compter la longueur de cette dernière ou en convertir le contenu en capitales, on ne pourra pas faire ces opérations avec la variable booléenne, dont le contenu veut simplement dire « vrai ». Voir *Programme ; Valeur booléenne.*

W

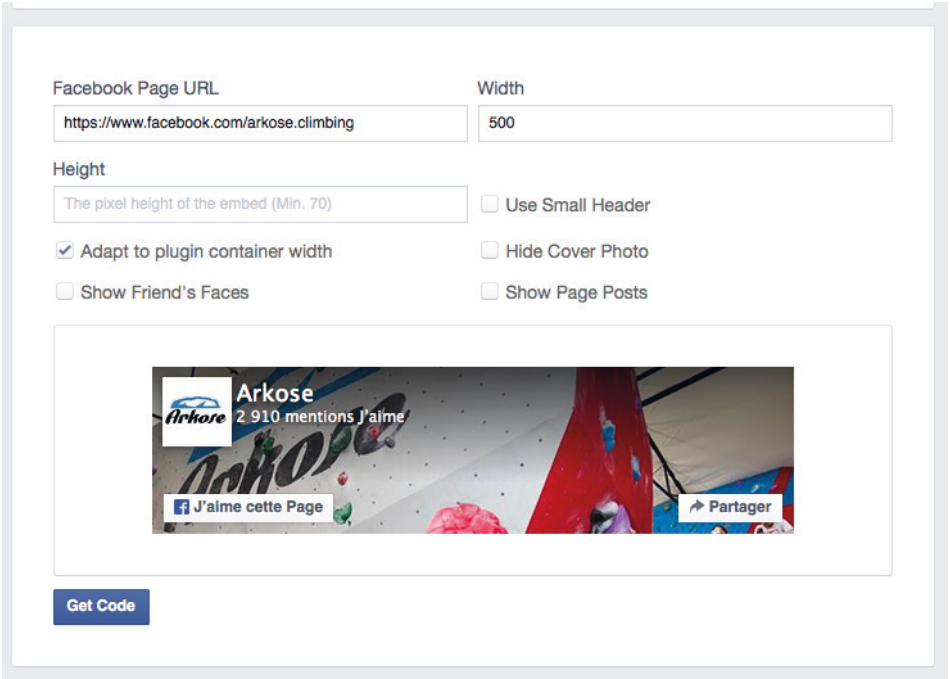
WEB : Diminutif de *World Wide Web*. Réseau public de documents hypertextes. Le Web est une partie du réseau Internet. Il est exclusivement utilisé pour relier différents documents hypertextes écrits au format HTML et les consulter grâce à un logiciel dédié : le navigateur. Le Web est crée en 1991 par Tim Berners-Lee et Robert Cailliau, travaillant alors au CERN à Genève. Voir *HTML.*

WIDGET : Mot-valise composé de *gadget* et de *window*. Le terme a deux sens, tous les deux tirés du design d'interface. Un *widget* est un élément visuel de base d'une interface graphique avec lequel un utilisateur peut interagir. Par exemple, une fenêtre est un *widget*, au même titre qu'une barre de défilement ou qu'un bouton « Valider ». En combinant différents *widgets*, on arrive ainsi à composer une interface complète. Par extension, le terme *widget* a pris un second sens : celui d'un petit module de fonctionnalités, par exemple une fonction « Horloge », « Météo » ou « Post-it™ » qu'on peut manipuler sous la forme d'une petite fenêtre indépendante. La composition de certains sites web comme Netvibes repose entièrement sur l'agencement de *widgets*, mais on trouve aussi des *widgets* par défaut dans nombre de systèmes d'exploitation. Voir *Interface graphique.*

TABLE DES ANNEXES

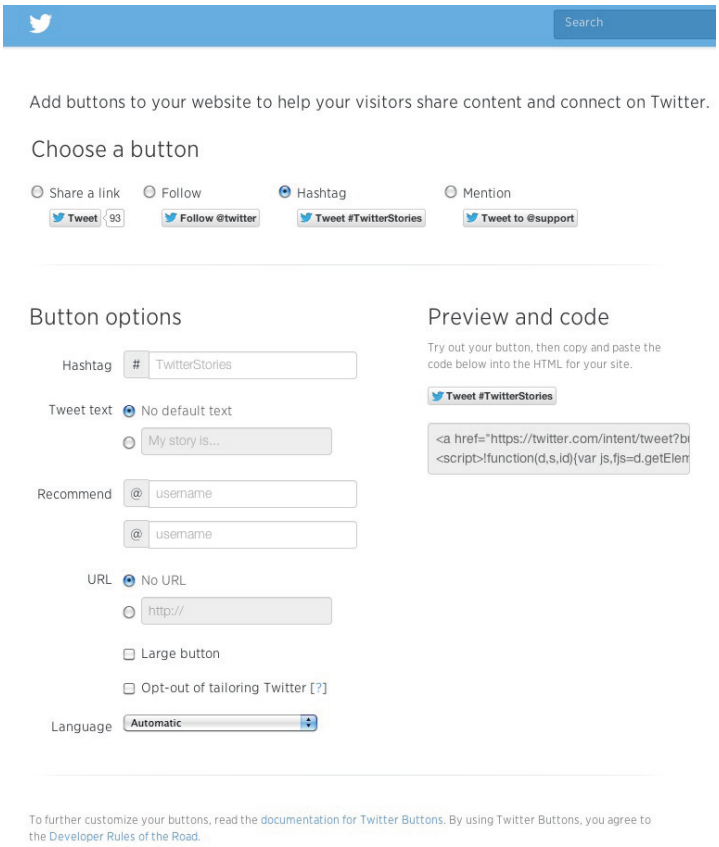
ANNEXES

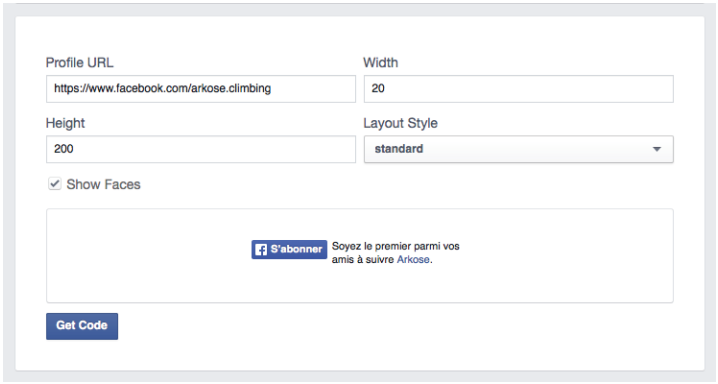
Annexe n° 1 Exemples de personnalisation de *widgets*.

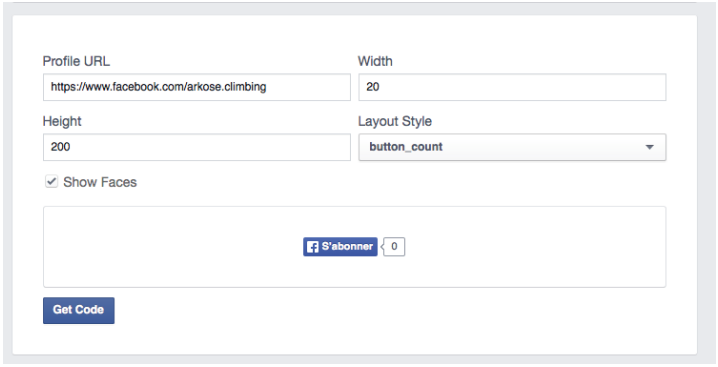


Capture de la page developers.facebook.com/docs/plugins/page-plugin (détail), réalisée le 29 juillet 2015 ; capture de la page twitter.com/widgets/user-timeline (détail), réalisée le 15 juillet 2015.

Annexe n° 2 Exemples de personnalisation de boutons.







Capture de la page about.twitter.com/resources/buttons#hashtag (détail), réalisée le 14 juillet 2015. Captures de la page developers.facebook.com/docs/plugins/follow-button (détail), réalisée le 29 juillet 2015.

Annexe n° 3 Histoire de l'informatique, quelques dates clés (1936-1955)

La présente chronologie concerne les débuts de l'informatique, des premières conceptualisations théoriques à la fin des années 1930 à la sortie de l'après-guerre. La séquence s'ouvre en 1936, avec l'article d'Emil Post sur les processus combinatoires et celui de Turing sur les nombres calculables. Elle se termine avec la mise hors de fonction de l'ENIAC, le 2 octobre 1955. Nous couvrons ainsi l'essentiel de la séquence retenue le chapitre I. 2. A. de notre thèse (1946-1951) ainsi que les années 1936-1937 que nous abordons dans le chapitre I. 3.

Cette chronologie n'est pas exhaustive. Elle a vocation à accompagner la lecture du premier chapitre de notre thèse, afin de faire comprendre rapidement le contexte scientifique et industriel des textes que nous analysons. Elle est le résultat de choix (focale sur l'ENIAC, sur les textes de Von Neumann et de Turing, et plus globalement sur l'informatique telle qu'elle se développe dans les universités nord-américaines au début des années 1950) explicités dans le corps de la thèse.

1936
– *Finite Combinatory Processes – Formulation 1,* Emil Post
– premier brevet du Z1, par Konrad Zuse.

1937
– *On Computable Numbers, with an application to the Entscheidung problem,* d'Alan M. Turing

– premier schéma d'un ordinateur électronique entièrement binaire par John Vincent Atanasoff et Clifford Berry

1939
premier prototype de l'ABC *(Atanasoff Berry Calculator)*

1941
Premier modèle opérationnel du Z3, calculateur électronique de Konrad Zuse.

1942
Première version opérationnelle de l'ABC.

1943
Première version fonctionnelle du Harvard Mark I à Endicott (Etats-Unis)

1943 :
Projet ENIAC approuvé par l'armée états-unienne.

Décembre 1943 : Colossus opérationnel à Bletchley Park (Angleterre).

1943
début du projet Whirlwind (ordinateur réalisée pour la Marine états-unienne par le MIT)

1944
Janvier : rapport de Mauchly et Eckert sur une mémoire magnétique ou électronique pour l'ENIAC

1944
Juillet :
– premiers éléments fonctionnels de l'ENIAC.

– première visite de Von Neuman à la *Moore School of Electronic* (Unniversité de Pennsylvanie).

1945
ENIAC pleinement opérationnel.

30 juin : *First Draft of a Report on the EDVAC* (John Von Neumann)

30 novembre : *Description of the ENIAC and Comments on Electronic Digital Computing Machine,* John A. Eckert, John W. Mauchly, Herman H. Goldstine et John G Brainerd. Première apparition du terme *subroutine*.

1946
19 mars : *Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE),* Alan Turing.

Départ de Von Neumann pour l'I*nstitute of Advanced Studies* pour mettre au point un calculateur électronique, machine qui deviendra l'EDVAC

8 juillet — 31 août : cours *« Theory and Techniques for the design of Electronic Digital Computers »* à la Moore School

19 août : *A Study of inverse extrapolation of the ENIAC.* Haskell B. Curry & Willa A. Wyatt.

Septembre : Fondation du *Royal Society Computer Laboratory* à l'université de Manchester

1947
Harvard Mark II opérationnel.

printemps : première démonstration de l'EDVAC.

– *Planning and Coding of Problems for an Electronic Computing Instrument: report on the mathematical and logical aspects of an electronic computing instrument,* John Von Neumann et Herman H. Goldstine (1947-148)

1948
Janvier : présentation publique du S*elective Sequence Electronic Calculator* (SSEC, IBM)

– IBM 604.

– invention du transistor.

1949
Harvard Mark III opérationnel.

mai : première démonstration de l'EDSAC (Cambridge, Royaume-Uni)

National Physic Laboratory Pilot Ace opérationnel (Londres, Royaume-Uni)

1950
projet Whirlwind fonctionnel

1951
– *The Preparation of Programs for an Electronic Digital Computer,* Maurice V. Wilkes, David J. Wheeler & Stanley Gill

1952
Décembre : livraison des premiers IBM 701, premier calculateur entièrement électronique d'IBM. Démantèlement du SSEC.

1952
Mark IV opérationnel.

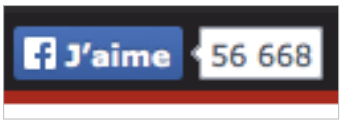
10 juin : EDVAC opérationnel

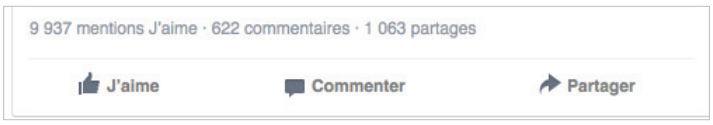
The use of sub-routines in programmes. David J. Wheeler.

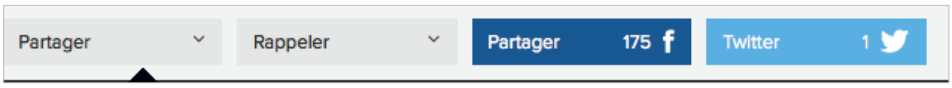
1954
Été : séminaire *Digital Computers. Advanced Coding Techniques* au MIT. Sous la direction de Charles W. Adams, Stanley Gill et Donn Combelic.

1955 :
2 octobre Mise hors de fonction de l'ENIAC

Annexe n° 4 Omniprésence du chiffre et de la statistique dans notre corpus

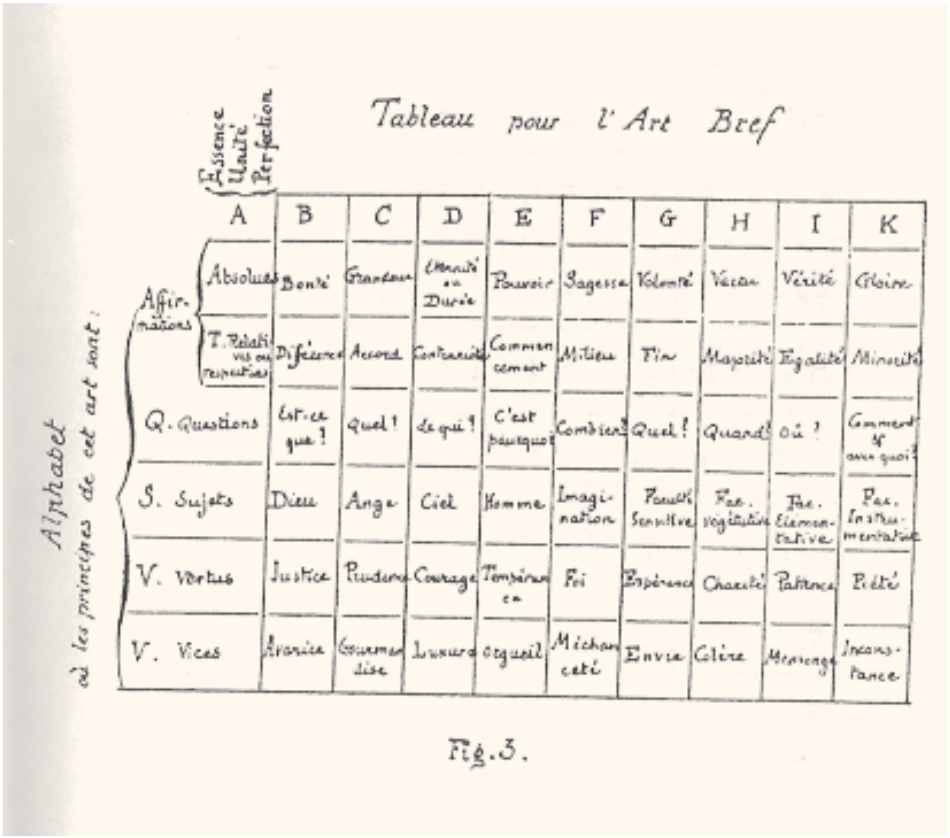






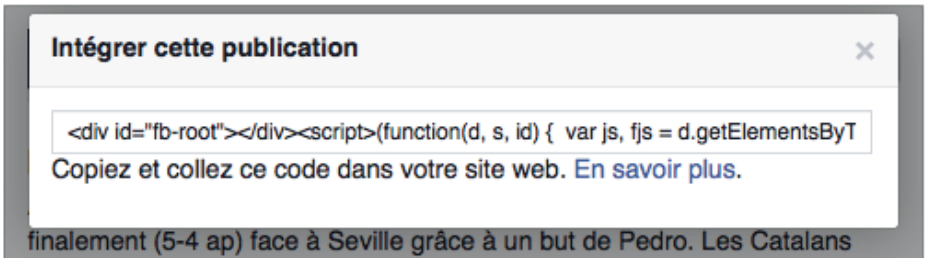
Extraits des sites : cielmondoctorat.tumblr.com/page/3 (capture réalisée le 16 juillet 2015 à 17h47) ; cinema.jeuxactu.com (réalisée le 11 août 2015) ; letalkfoot.fr/video/ligue-des-champions/video/le-resume-de-barcelone-fc-seville-5-4 (capture réalisée le 13 août 2015) ; mmorpg/com/newsroom (capture réalisée le 16 juillet 2015) ; south-park-streaming.com/saison-5/episode-4/scott-tenorman-doit-mourir (capture réalisée le 7 août 2015) ; twitter.com (capture réalisée le 18 juillet 2015 à 17h55, profil personnel) ; www.arte.tv/guide/fr/050347-000/john-von-neumann?autoplay=1 (capture réalisée le 5 août 2015) ; www.betaseries.com (réalisée le 16 juillet 2015) ; www.slate.fr (capture réalisée le 11 août 2015) ?

Annexe n° 5 Table d'encodage de l'*Ars Magna* lullien.



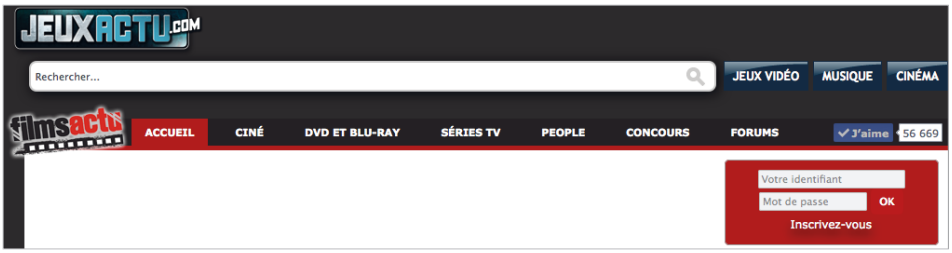
Source : LULLE, Raymond. *L'art bref.* Milan : Archè, 1987, figure 3, p. 9.

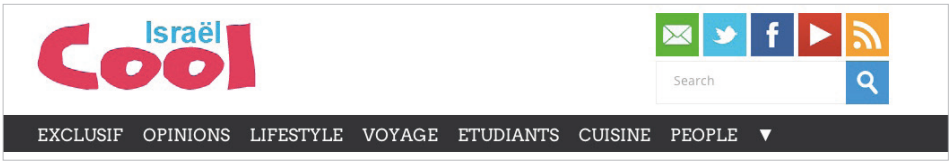
Annexe n° 6 Exemple de procédure d'ancrage d'une publication Facebook dans une page web (le site letalkfoot.fr).

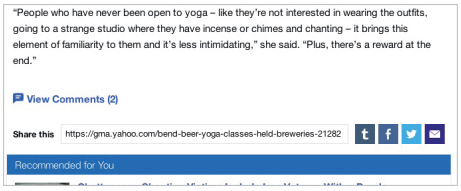


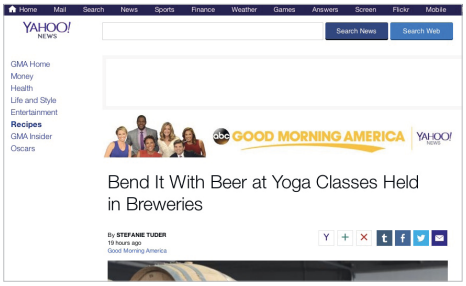
Extraits de la page letalkfoot.fr/video/ligue-des-champions/video/le-resume-de-barcelone-fc-seville-5-4 et de www.facebook.com (profil personnel). Captures réalisées le 13 août 2015.

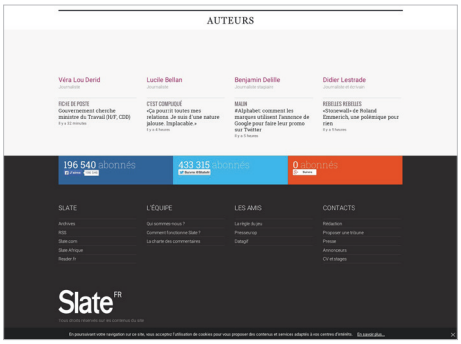
Annexe n° 7 Exemples de boutons placés en tête et en pied de page





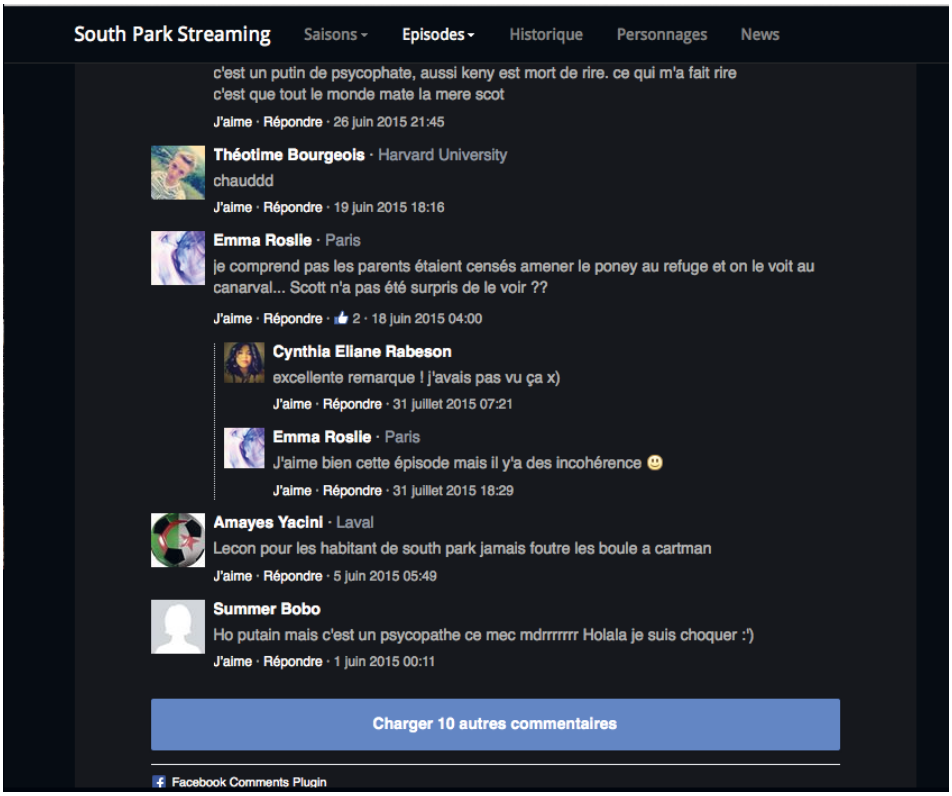






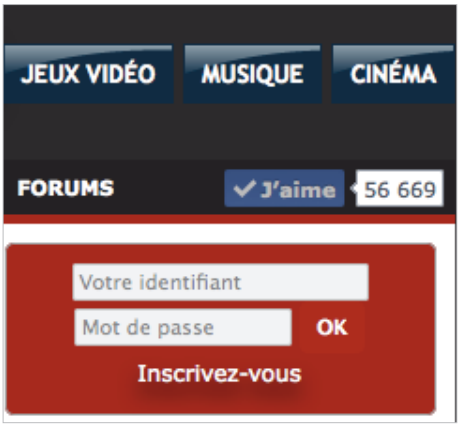
Extraits de www.slate.fr (pied de page, capture réalise le 11 août 2015) ; cinema.jeuxactu.com (en-tête, capture réalisée le 11 août 2015) ; coolisrael.fr/25041/coince-a-paris-tinquiete-la-plage-de-tel-aviv-debarque-chez-toi (en-tête et pied de page, captures réalisées le 16 août 2015) ; http://gma.yahoo.com/bend-beer-yoga-classes-held-breweries-212825058--abc-news-Recipes.html (en-tête et pied de page, captures réalisées le 17 juillet 2015).

Annexe n° 8 Exemple de module commentaires placé à la suite du contenu central de la page.



Extrait du site www.south-park-streaming.com/saison-5/episode-4/scott-tenorman-doit-mourir. Capture réalisée le 7 août 2015.

Annexe n° 9 Exemples de boutons répartis dans des espaces interstices.

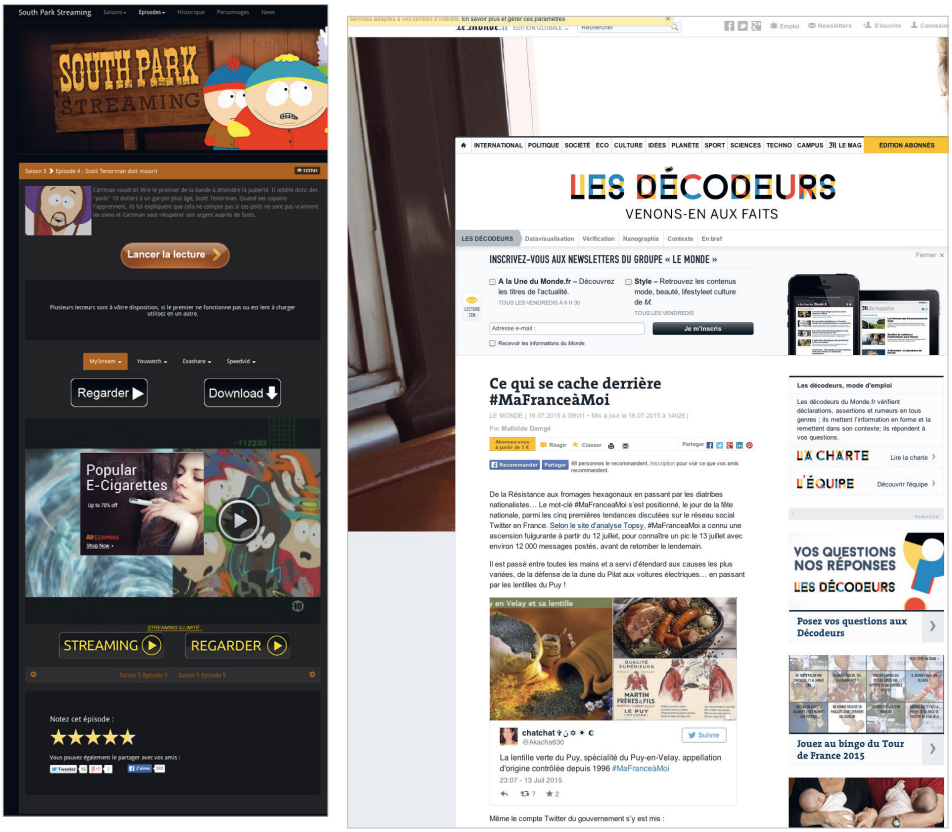


Extraits de coolisrael.fr/25041/coince-a-paris-tinquiete-la-plage-de-tel-aviv-debarque-chez-toi (capture réalisée le 16 août 2015) ; cinema.jeuxactu.com (capture réalisée le 11 août 2015) ; http/www.noisey.vice.com/123pires (capture réalisée le 16 juillet 2015) ; mmorpg.com/newsroom (capture réalisée le 16 juillet 2015) ; www.betaseries.com (capture réalisée le 16 juillet 2015)

Annexe n° 10 Exemple de page construite comme un lieu de mémoire.

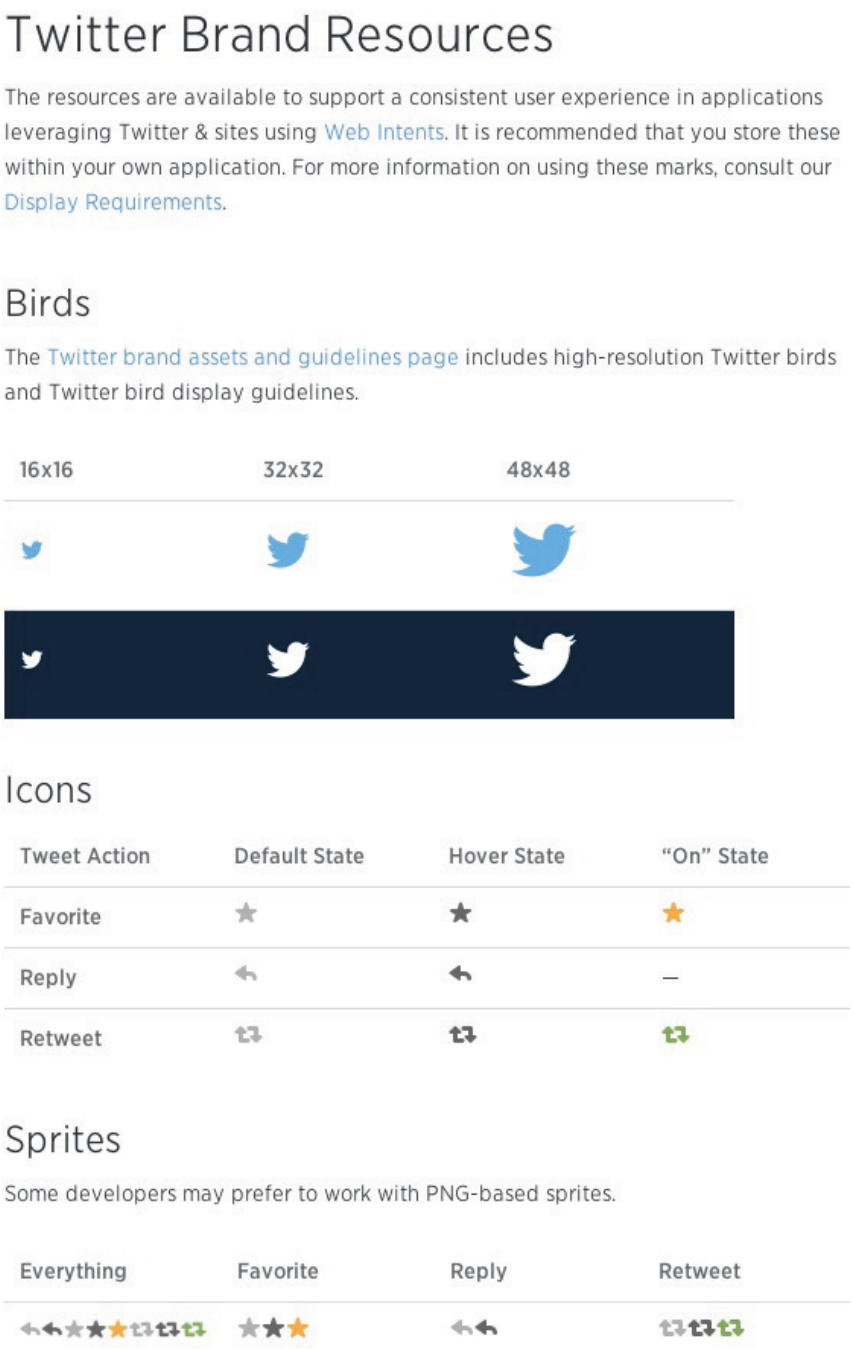
Les boutons en haut à droite s'appliquent au site *Vice France*, qui chapeaute l'énonciation en cours. Le menu du haut permet de naviguer dans le contenu éditorial de *Noisey*, les boutons sur la droite permettent de « suivre » les actualités de *Noisey*, tandis que les boutons sous le titre « les 123 pires artistes de tout les temps » concernent seulement l'article affiché. Capture de la page http/www.noisey.vice.com/123pires (détail), réalisée le 16 juillet 2015.

Annexe n° 11 Exemples d'énonciations en inclusion.



Dans le premier exemple, un tweet est ancré « dans » un article du site Les Décodeurs, lui–même dépendant du site du journal *Le Monde*. Dans le second exemple, une vidéo est ancrée « dans » une publication du site south-park-streaming.com. Captures des pages www.lemonde.fr/les-decodeurs/article-2015-07-16-ma-france-a-moi-le-mot-cle-qui-a-enflamme-twitter-le-14-juillet_4685117_4355770.html (détail, capture réalisée le 17 juillet 2015) ; www.south-park-streaming.com/saison-5/episode-4/scott-tenorman-doit-mourir (capture réalisée le 7 août 2015)

Annexe n° 12 Icônes recommandées pour les *intents* Twitter, micro-formes de la culture anthologique.



Twitter Brand Resources

The resources are available to support a consistent user experience in applications leveraging Twitter & sites using Web Intents. It is recommended that you store these within your own application. For more information on using these marks, consult our Display Requirements.

Birds

The Twitter brand assets and guidelines page includes high-resolution Twitter birds and Twitter bird display guidelines.

16x16	32x32	48x48

Icons

Tweet Action	Default State	Hover State	"On" State
Favorite			
Reply			—
Retweet			

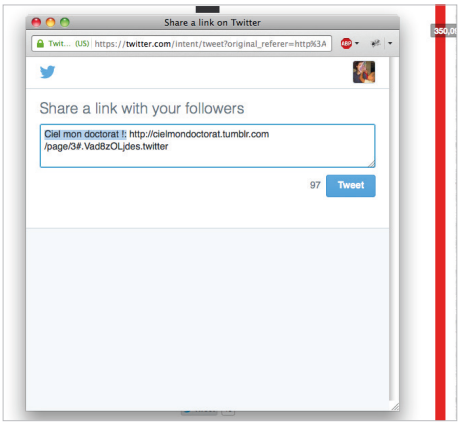
Sprites

Some developers may prefer to work with PNG-based sprites.

Everything	Favorite	Reply	Retweet

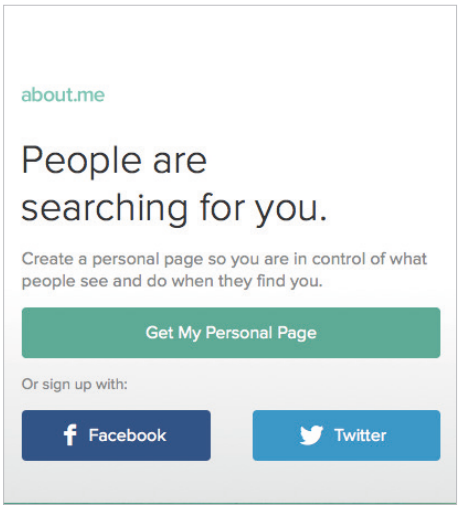
Capture de la page dev.twitter.com/overview/general/image-resources (détail). Capture réalisée le 14 juillet 2015.

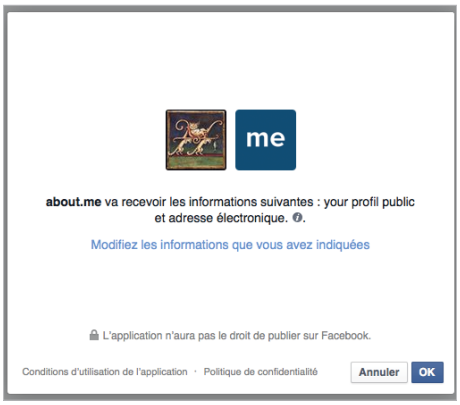
Annexe n° 13 La forte présence d'outils d'éditorialisation dans notre corpus.

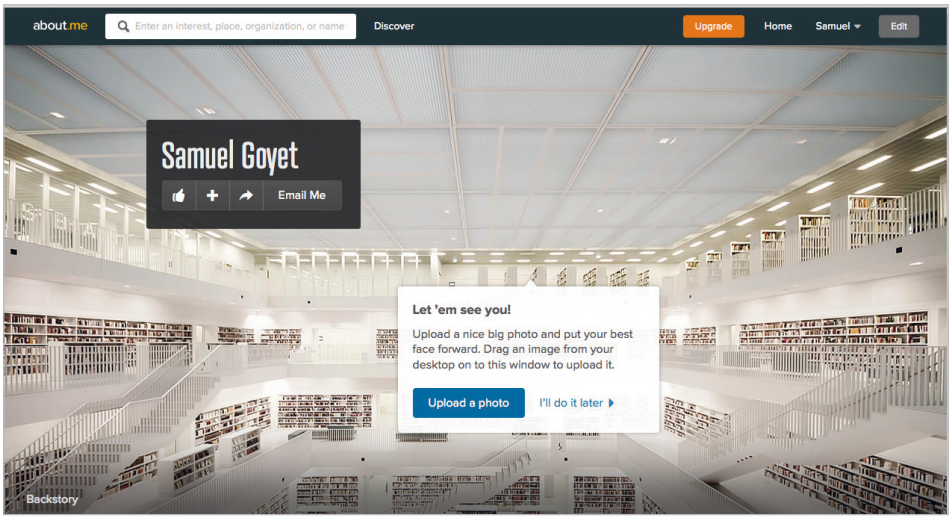


Extraits des pages cinema.jeuxactu.com (capture réalisée le 11 août 2015) ; cielmondoctorat.tumblr.com/page/3, (capture réalisée le 16 juillet 2015 à 11h45) ; rappler.com/nation/89508/mar-roxas-je-jomar-binay-2016 (capture réalisée le 12 août 2015) ; www.senscritique.com (capture réalisée le 7 août 2015).

Annexe n° 14 Création automatique d'un profil *via* la fonction « Facebook *Connect* » sur le site about.me.

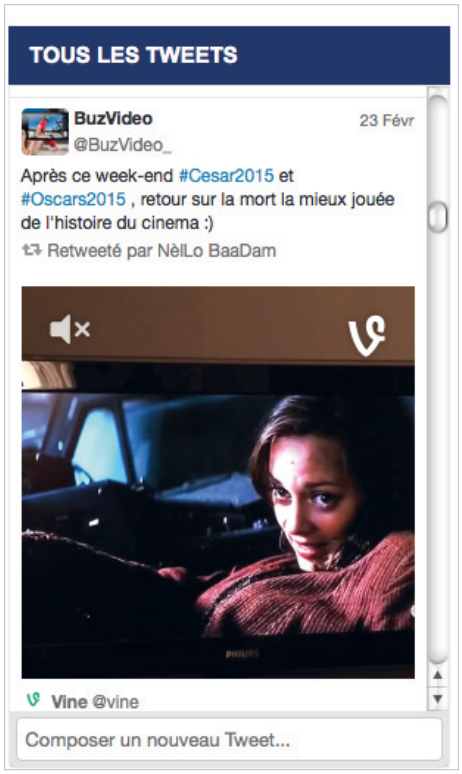






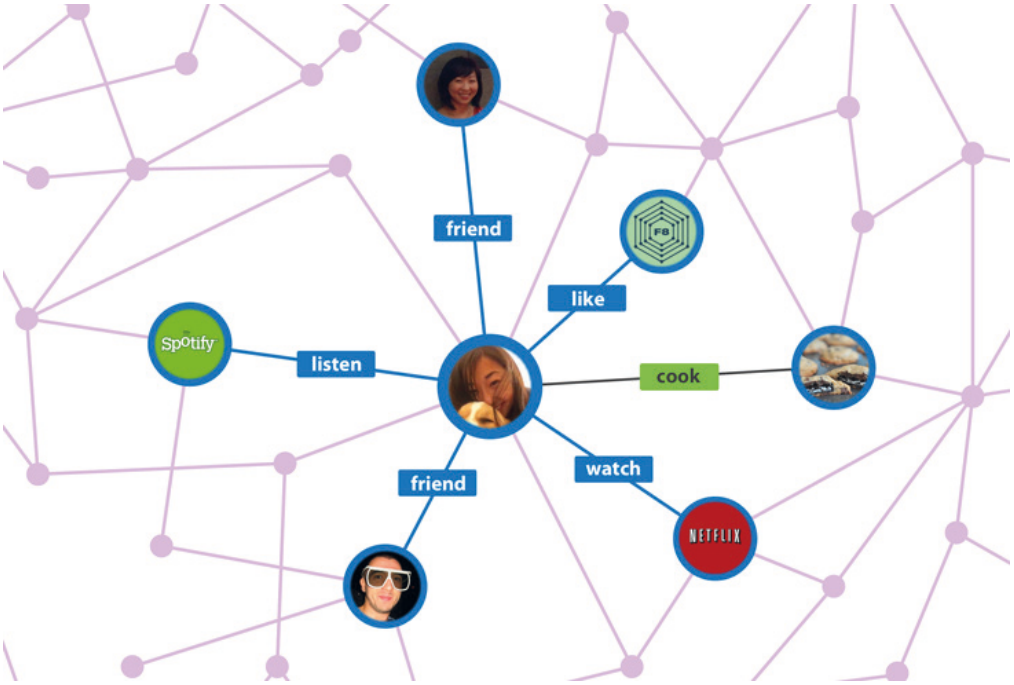
Une fois le site autorisé à avoir accès au graphe social de l'internaute, un profil se crée automatiquement en recomposant sémiotiquement des éléments discrets auquels l'API donne accès. Extraits des sites about.me et facebook.com (profil personnel). Captures réalisées le 12 août 2015.

Annexe n° 15 Exemples de création automatique de formes sémiotiques.



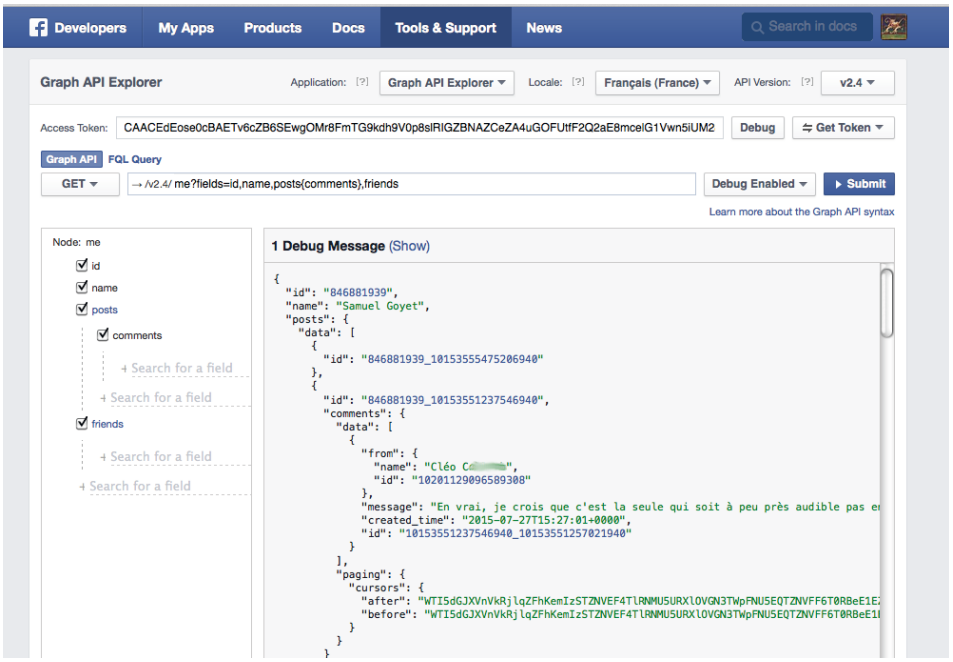
Extraits des sites facebook.com (profil personnel, détail, captures réalisées le 5 et 12 août 2015) ; www.canalplus.fr/c-cinema/c-ceremonie-des-cesar-sur-canal-pid5499-livetweet-ceremonie-cesar-2015.html (détail, capture réalisée le 17 juillet 2015).

Annexe n° 16 Illustration du « graphe social » selon Facebook.



Consultable sur http://img.macg.co/2011/9/1316731172_mgpic_final.jpg. [Consulté le 3 août 2017].

Annexe n° 17 Exemple d'exploration du graphe social (profil personnel) grâce à la *Graph API* de Facebook.



Après nous être identifié auprès de Facebook, nous avons demandé à accéder à nos dernières publications (champ « GET » en haut de la page Graph Explorer) ainsi qu'aux commentaires de nos amis sur le réseau. Le résultat est une représentation JSON de la publication que l'on peut voir par ailleurs sur Facebook. Capture de facebook.com, réalisée le 29 juillet 2015 (détail, profil personnel) ; Capture de developers.facebook.com/graph/api-explorer (détail, profil personnel), réalisée le 29 juillet 2015.

Annexe n° 18 Extraits du code source du site senscritique.com (capture réalisée le 7 août 2015).

```
955                 <div id="fb-root"></div>
956 <div data-rel="fb_container" data-sc-id="157003860988959"></div>
957 <script>
958     // window.fbAsyncInit = function() {
959     //  FB.init({
960     //      appId:'157003860988959',
961     //      cookie:true,
962     //      status:true,
963     //      xfbml:true,
964     //      // channel: 'http://".$domain."/fb_channel.php'
965     //  });
966
967     //  FB.Event.subscribe('auth.statusChange', function(response) {
968     //      if (response.status === 'connected') {
969     //          var uid = response.authResponse.userID;
970     //          var accessToken = response.authResponse.accessToken;
971     //          var expiresIn = response.authResponse.expiresIn;
972     //          // Do login facebook, only if we have the try
973     //          $('[data-sc-facebook-autologin="true"]').each(function() {
974     //              document.location = '/oauthapi/facebook/';
975     //          });
976     //          $('[data-sc-refresh-api-facebook="true"]').each(function() {
977     //              $.ajax({
978     //                  url: '/oauthapi/facebook/renewToken.json',
979     //                  type: 'post',
980     //                  data:{
981     //                      access_token:accessToken,
982     //                      expires_in:expiresIn,
983     //                      uid:uid
984     //                  },
985     //                  dataType: 'json',
986     //                  success: function (data, textStatus, jqXhr) {
987
988     //                  }
989     //              });
990     //          });
991     //      } else {
992     //          var status = (response.status === 'not_authorized') ? 'connected' : 'unknow';
993     //          // Do nothing
994     //          $('[data-sc-facebook-autologin="true"]').each(function() {
995     //              $.ajax({
996     //                  url: '/oauthapi/facebook/status/' + status + '.json',
997     //                  type: 'get',
998     //                  dataType: 'json',
999     //                  success: function (data, textStatus, jqXhr) {
1000
1001     //                  }
1002     //              });
```

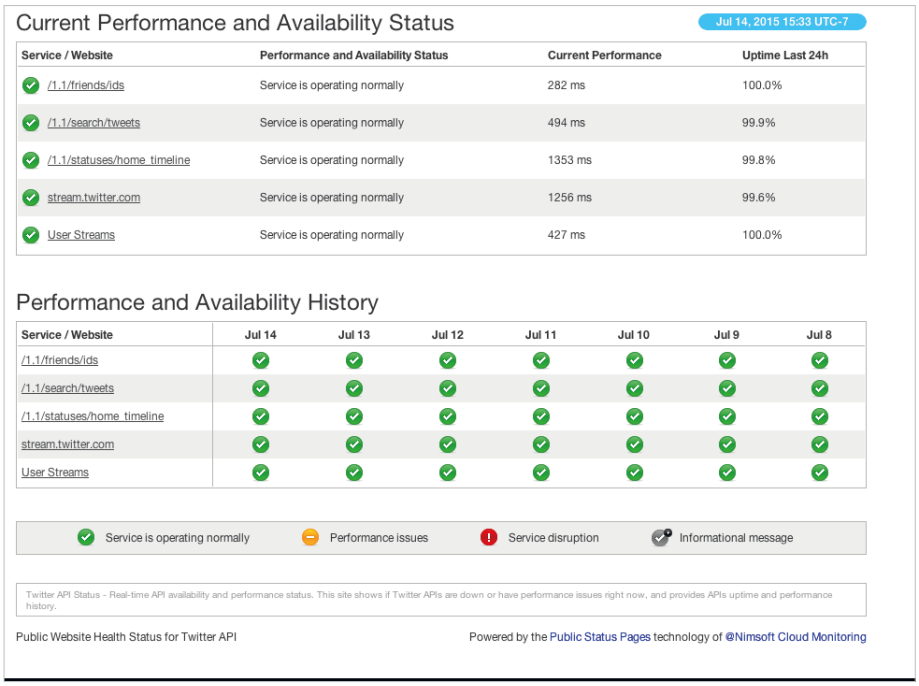
```
1001     //              }
1002     //          });
1003     //      });
1004     //  }
1005     // });
1006     // };
1007
1008     // // Load the SDK Asynchronously
1009     // (function(d){
1010     //  var js, id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
1011     //  js = d.createElement('script'); js.id = id; js.async = true;
1012     //  js.src = "http://connect.facebook.net/fr_FR/all.js";
1013     //  d.getElementsByTagName('head')[0].appendChild(js);
1014     // }(document));
1015 </script>
1016     </body>
1017 </html>
1018
```

Un script commence ligne 957 (balise `<script>`). Il commence par authentifier le site Sens Critique auprès de Facebook (lignes 958 à 965), pour ensuite récupérer les données Facebook de l'internaute en accédant, via l'API, à son graphe social (lignes 967 à 1006) et en lui générant un jeton personnel d'identification (lignes 967 à 984). Les dernières lignes 1007 à 1014 permettent de régler le moment où le navigateur, quand il exécute la page d'accueil de Sens Critique, exécute le script spécifique qui permet de se connecter à l'API. La syntaxe du script respecte rigoureusement la syntaxe de Facebook Connect définie dans la documentation de l'API de Facebook. Voir https://developers.facebook.com/docs/javascript/quickstart [consulté le 3 août 2017].

Annexe n° 19 La représentation informatique d'un tweet, selon l'API de Twitter.

Capture de la page dev.twitter.com/overview/api/tweets, réalisée le 14 juillet 2015.

Annexe n° 20 Représentation de la « santé » de l'API.



Current Performance and Availability Status

Jul 14, 2015 15:33 UTC-7

Service / Website	Performance and Availability Status	Current Performance	Uptime Last 24h
/1.1/friends/ids	Service is operating normally	282 ms	100.0%
/1.1/search/tweets	Service is operating normally	494 ms	99.9%
/1.1/statuses/home_timeline	Service is operating normally	1353 ms	99.8%
stream.twitter.com	Service is operating normally	1256 ms	99.6%
User Streams	Service is operating normally	427 ms	100.0%

Performance and Availability History

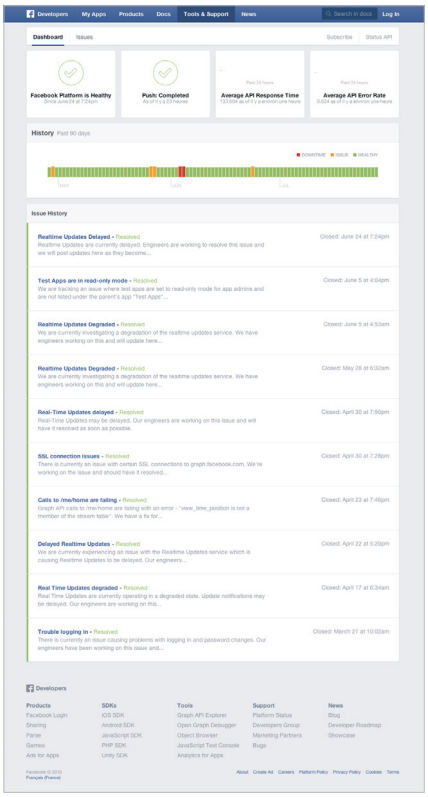
Service / Website	Jul 14	Jul 13	Jul 12	Jul 11	Jul 10	Jul 9	Jul 8
/1.1/friends/ids	✓	✓	✓	✓	✓	✓	✓
/1.1/search/tweets	✓	✓	✓	✓	✓	✓	✓
/1.1/statuses/home_timeline	✓	✓	✓	✓	✓	✓	✓
stream.twitter.com	✓	✓	✓	✓	✓	✓	✓
User Streams	✓	✓	✓	✓	✓	✓	✓

Service is operating normally · Performance issues · Service disruption · Informational message

Twitter API Status - Real-time API availability and performance status. This site shows if Twitter APIs are down or have performance issues right now, and provides APIs uptime and performance history.

Public Website Health Status for Twitter API

Powered by the Public Status Pages technology of @Nimsoft Cloud Monitoring



Le bon fonctionnement technique est attesté par un système de marqueurs au code sémiotique élémentaire : vert quand le temps de réponse des serveurs est optimal ; orange quand il est un peu long et rouge quand quand le serveur ne répond plus. Ces formes combinent la vision synoptique du service (à travers l'image du *dashboard*) et la métaphore de la « bonne santé » (*Health status*) d'un système technique. Capture de la page dev.twitter.com/overview/status (détail), réalisée le 15 juillet 2015 ; capture de la page developers.facebook.com/status/dashboard, réalisée le 29 juillet 2015.

Annexe n° 21 Réponse à une requête API faite pour vérifier le bon fonctionnement de l'API.


```
{
   "current": {
      "health": 1,
      "subject": "Facebook Platform is Healthy"
   },
   "push": {
      "status": "Complete",
      "updated": "2015-07-27T16:00:06-07:00",
      "id": 60060758
   }
}
```

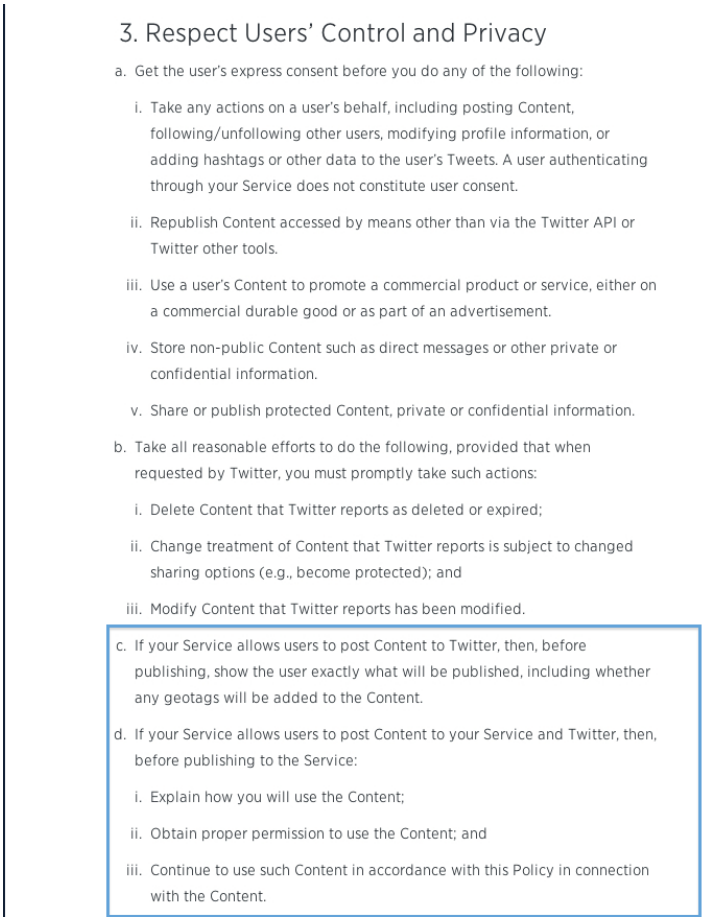

La réponse est un objet JSON où peut à nouveau se lire la métaphore médicale (« *Facebook Platform is healthy* »). Capture réalisée le 29 juillet 2015.

Annexe n° 22 Exemple de la forme contractuelle d'une API.

Par cette fenêtre, une application (ici, about.me) demande l'autorisation à l'utilisateur de Facebook d'accéder à ses données personnelles, et détaille à quels types de données l'application aura accès. Si l'internaute accepte, un jeton d'identification est généré et l'application peut, grâce à l'API de Facebook, récupérer les données de l'utilisateur.

Capture du site facebook.com, profil personnel (détail). Capture réalisée le 12 août 2015.

Annexe n° 23 Exemple de recommandations d'ordre moral dans la documentation de l'API de Twitter.



3. Respect Users' Control and Privacy

a. Get the user's express consent before you do any of the following:

   i. Take any actions on a user's behalf, including posting Content, following/unfollowing other users, modifying profile information, or adding hashtags or other data to the user's Tweets. A user authenticating through your Service does not constitute user consent.

   ii. Republish Content accessed by means other than via the Twitter API or Twitter other tools.

   iii. Use a user's Content to promote a commercial product or service, either on a commercial durable good or as part of an advertisement.

   iv. Store non-public Content such as direct messages or other private or confidential information.

   v. Share or publish protected Content, private or confidential information.

b. Take all reasonable efforts to do the following, provided that when requested by Twitter, you must promptly take such actions:

   i. Delete Content that Twitter reports as deleted or expired;

   ii. Change treatment of Content that Twitter reports is subject to changed sharing options (e.g., become protected); and

   iii. Modify Content that Twitter reports has been modified.

c. If your Service allows users to post Content to Twitter, then, before publishing, show the user exactly what will be published, including whether any geotags will be added to the Content.

d. If your Service allows users to post Content to your Service and Twitter, then, before publishing to the Service:

   i. Explain how you will use the Content;

   ii. Obtain proper permission to use the Content; and

   iii. Continue to use such Content in accordance with this Policy in connection with the Content.

Ici, la section 3 des conditions d'utilisation de l'API, consacré à la vie privée (*privacy*) des utilisateurs et qui repose entièrement sur la notion de consentement (*consent*). Elle mobilise également une rhétorique de la transparence : l'application qui utilise l'API de Twitter pour publier sur la plateforme doit clairement donner à voir ce qui sera publié, et sous quelle forme. (sous-section c et d, encadré bleu).

Capture de la page dev.twitter.com/overview/terms/policy (détail). Capture réalisée le 14 juillet 2015.

Annexe n° 24 La notion de « bon partenaire » (*good partner*) pour Twitter.

6. Be a Good Partner to Twitter

a. Follow the guidelines for using Tweets in broadcast if you display Tweets offline.

b. If you provide Content to third parties, including downloadable datasets of Content or an API that returns Content, you will only distribute or allow download of Tweet IDs and/or User IDs.

   i. You may, however, provide export via non-automated means (e.g., download of spreadsheets or PDF files, or use of a "save as" button) of up to 50,000 public Tweets and/or User Objects per user of your Service, per day.

   ii. Any Content provided to third parties via non-automated file download remains subject to this Policy.

c. Use and display Twitter Marks solely to identify Twitter as the source of Content.

d. Comply with Twitter Brand Assets and Guidelines.

e. Do not do any of the following:

   i. Use a single application API key for multiple use cases or multiple application API keys for the same use case.

   ii. Charge a premium above your Service's standard data and usage rates for access to Content via SMS or USSD.

   iii. Sell or receive monetary or virtual compensation for Tweet actions or the placement of Tweet actions on your Service, such as, but not limited to follow, retweet, favorite, and reply.

   iv. Do not use, access or analyze the Twitter API to monitor or measure the availability, performance, functionality, usage statistics or results of Twitter's products and services or for any other benchmarking or competitive purposes, including without limitation, monitoring or measuring:

      1. the responsiveness of Twitter websites, web pages or other online services; or

      2. aggregate Twitter user metrics such as total number of active users, accounts, user engagements or account engagements.

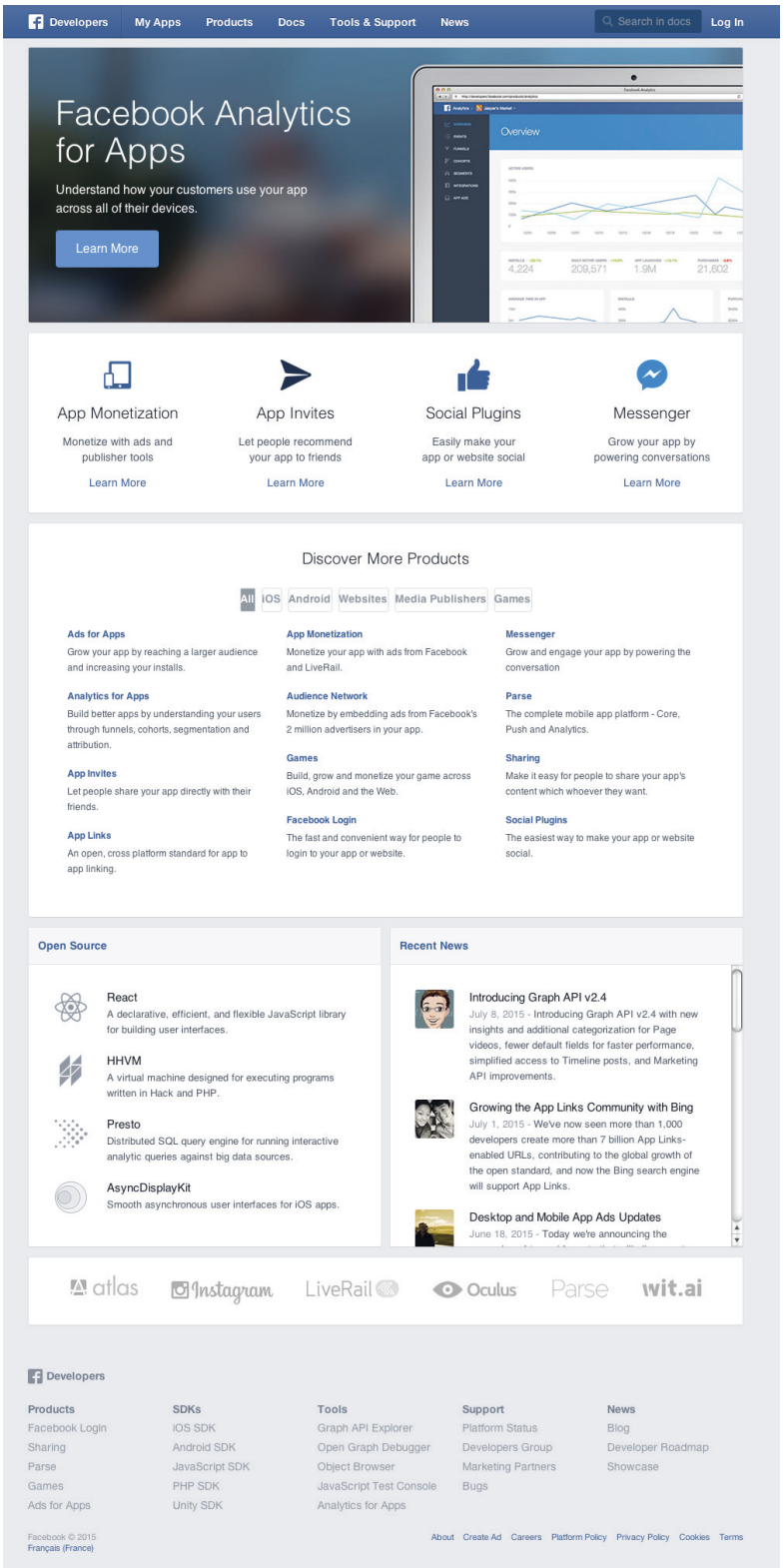
   v. Use Twitter Content, by itself or bundled with third party data, to target users with advertising outside of the Twitter platform, including without limitation on other advertising networks, via data brokers, or through any other advertising or monetization services.

   vi. Use Twitter Marks, or Twitter Certified Products Program badges, or similar marks or names in a manner that creates a false sense of endorsement, sponsorship, or association with Twitter.

   vii. Use the Twitter Verified Account badge, Verified Account status, or any other enhanced user categorization on Twitter Content other than that reported to you by Twitter through the API.

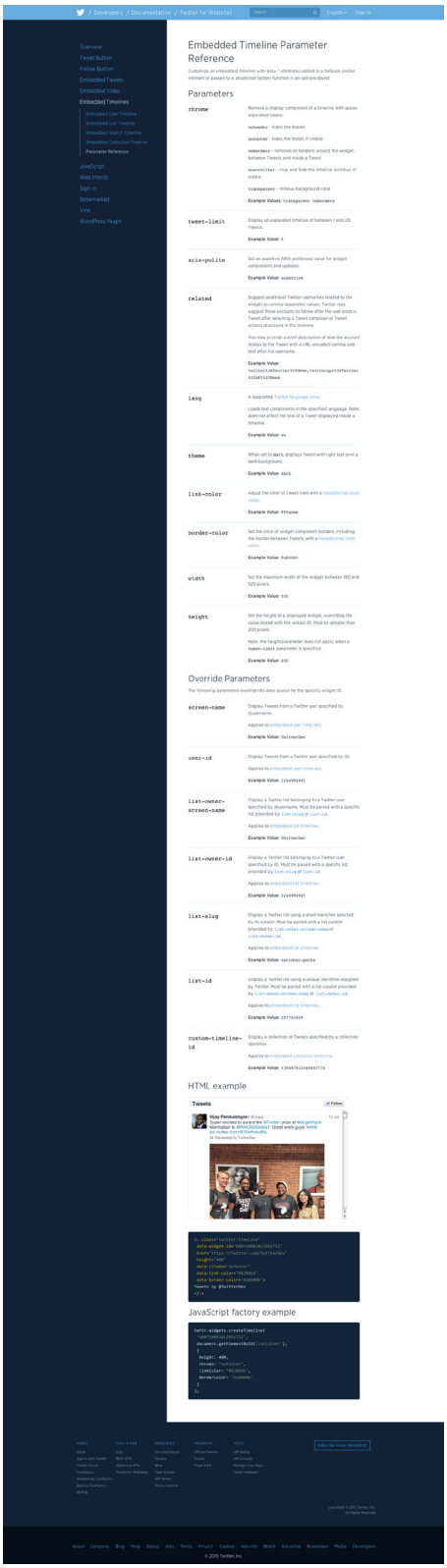
Capture de la page dev.twitter.com/overview/terms/policy (détail). Capture réalisée le 14 juillet 2015.

Annexe n° 25 Exemple de la forme du portail, page d'accueil de l'API de Facebook.



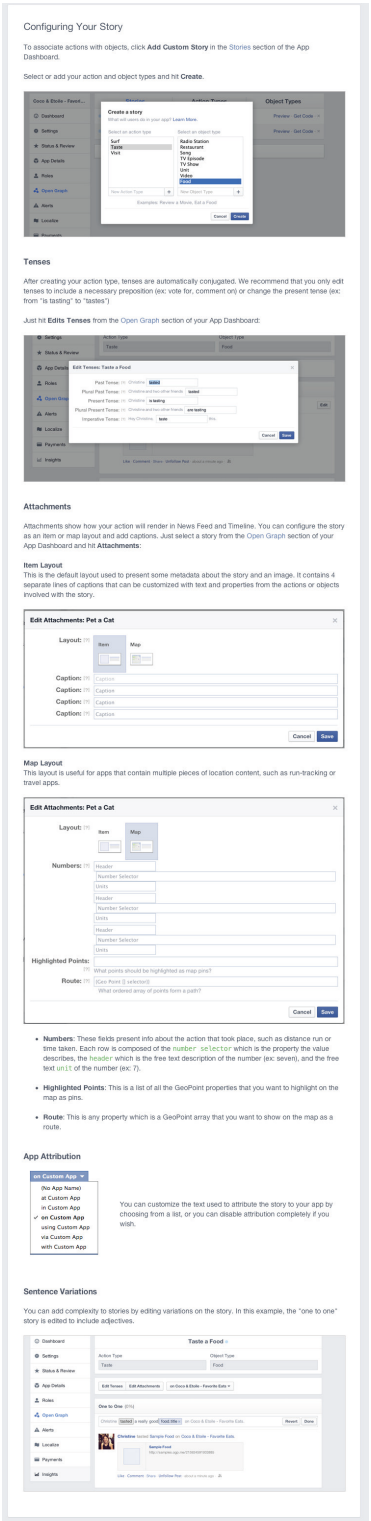
Les quatre rubriques renvoient aux grandes parties de la documentation de l'API. L'ensemble construit la vision d'une API aux multiples réutilisations possibles. Capture de la page dev.facebook.com. Capture réalisée le 28 juillet 2015.

Annexe n° 26 Exemple de la forme de la liste.



Page de la documentation de Twitter qui liste les différents paramètres de l'objet *timeline*. Capture de la page dev.twitter.com/web/embedded-timelines-parameters. Capture réalisée le 14 juillet 2015.

Annexe n° 27 Exemple de la forme du guide.



Page de la documentation de Facebook qui montre comment créer une histoire (*story*) en utilisant l'*Open Graph* API. Chaque étape est expliquée, captures d'écran à l'appui.

Capture de la page developers.facebook.com/docs/sharing/opengraph-custom (détail). Capture réalisée le 28 juillet 2015.

Annexe n° 28 Exemple d'article sur le site du *Times Higher Education*.

La *timeline* Twitter (encadré bleu) est située dans la coin inférieur droit, en marge du texte principal de l'article. Elle occupe une espace assez restreint dans la page.

Capture de la page www.timeshighereducation.co.uk/news/academics-meet--declare-support-universities-hit-conflict. Capture réalisée le 17 juillet 2015.

Annexe n° 29 Composition du corpus (petites formes) : tableau récapitulatif.

Les annexes qui suivent (30 à 95) sont la version imprimée d'une partie de notre corpus : les captures d'écran de petites formes permises par les API de Facebook et de Twitter. Nous n'avons pas inclus la capture d'écran de la page entière des sites hôtes de ces formes, ni le code source correspondant (deuxième partie de notre corpus), ni la documentation des API (troisième partie de notre corpus). Ces captures d'écran, bien trop grandes, auraient été illisibles sur papier. En revanche, nous les avons intégrées dans le CD qui accompagne cette thèse.

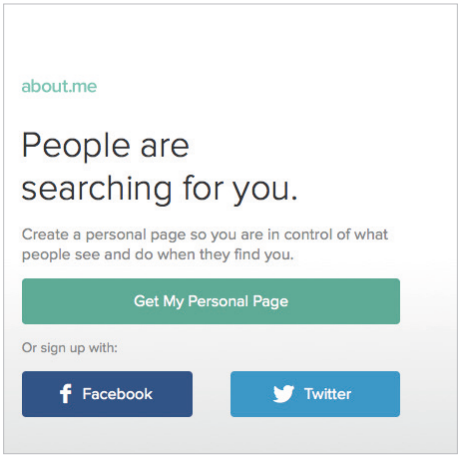
Nous avons donc choisi d'anonymiser nos captures d'écran seulement dans le cas de particuliers (amis, *followers* sur Twitter…) dont le visage et / ou le nom apparaissent du fait de notre identification préalable sur notre profil personnel Twitter ou Facebook. Les noms d'institutions publiques, de marques, de sociétés, d'organes de presse, les visages de célébrités n'ont pas été floutés. Nous avons estimé qu'il s'agissait là d'informations publiques. Les tweets de particuliers intégrés dans une page publique (article de presse par exemple) n'ont pas été anonymisés. Là encore, nous avons jugé qu'il s'agissait d'une rééditorialisation qui rend publics ces tweets, publicité dont la responsabilité incombe à l'auteur de l'article. Bien entendu, nous n'avons pas anonymisé notre propre nom et photo de profil, estimant que notre consentement était implicite.

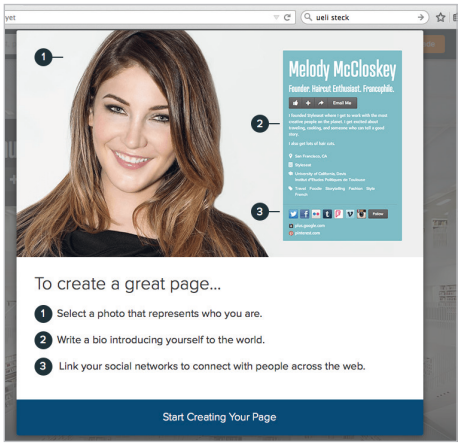
Le tableau suivant résume les soixante-cinq annexes qui suivent. Chaque ligne comprend le numéro d'annexe, l'API qui permet de produire la petite forme, le nom de cette forme, la description de son effet (si c'est un signe passeur) ou de sa forme et enfin le nom abrégé du site sur lequel nous avons trouvé cette petite forme. Nous espérons ainsi donner une vue synthétique de cette partie de notre corpus et permettre une navigation plus facile dans les annexes.

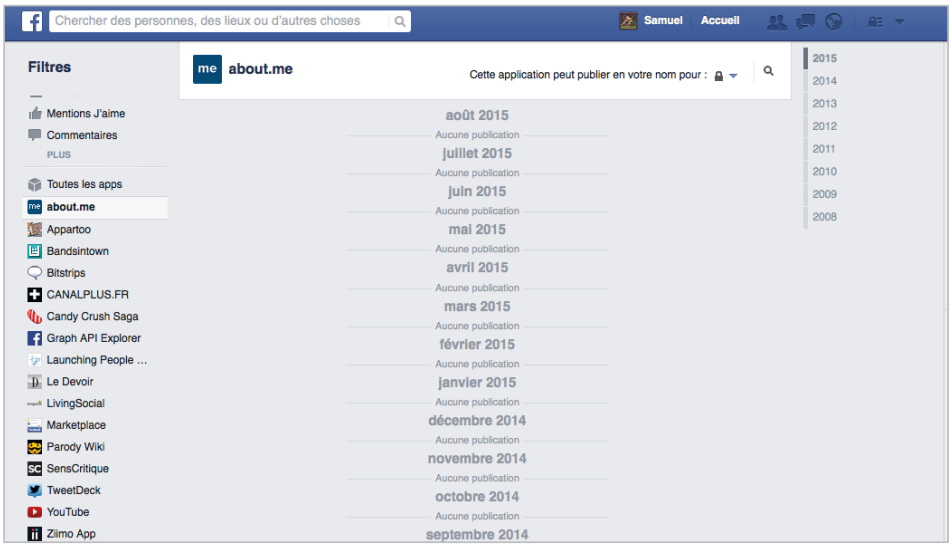
n°	Api Concernée	Nom de la petite forme / de la fonctionnalité	Effet du signe passeur / descriptif de la petite forme	Site source
30	Facebook	Se connecter avec Facebook	Génération automatique d'un profil	About.me
31	Facebook	Se connecter avec Facebook	Génération automatique d'un profil	About.me
32	Facebook	Se connecter avec Facebook	Génération automatique d'un profil	Sens Critique
33	Facebook	Se connecter avec Facebook	Génération automatique d'un profil	Sens Critique
34	Facebook	Bouton « *Like* »	Permet d'« aimer » sur facebook la page hôte	Cinemactu
35	Facebook	Bouton « *Like* »	Permet d'« aimer » sur facebook la page hôte	Slate
36	Facebook	Bouton « *Like* »	Permet d'« aimer » sur facebook la page hôte	South Park streaming
37	Facebook	Bouton « *Send* »	Publie le lien sous forme de message privé sur facebook	Cool Israël
38	Facebook	Bouton « *Share* »	Publie un lien sur facebook	Arte
39	Facebook	Bouton « *Share* »	Publie un lien sur facebook	Sens Critique
40	Facebook	Bouton « *Share* »	Publie un lien sur facebook	Le Tag Parfait
41	Facebook	publication ancrée	Publication facebook republiée sur le site hôte	Le talk foot
42	Facebook	*Feed dialog*	Publie un lien sur facebook	Good Morning America
43	Facebook	*Feed dialog*	Publie un lien sur facebook	Rappler
44	Facebook	*Feed dialog*	Publie un lien sur facebook	Youtube
45	Facebook	*Plugin* commentaires	Commentaires facebook inclus dans le site hôte	Cool Israël
46	Facebook	*Plugin* commentaires	Commentaires facebook inclus dans le site hôte	South park streaming
47	Facebook	*Plugin* page	Lien vers la page facebook du site hôte	Beta series
48	Facebook	*Plugin* page	Lien vers la page facebook du site hôte	Droit finances
49	Facebook	*Plugin* page	Lien vers la page facebook du site hôte	Le talk foot
50	Twitter	Bouton « Follow »	Permet de s'abonner aux publications de la page hote sur twitter	Beta series
51	Twitter	Bouton « *Follow* »	Permet de s'abonner aux publications de la page hote sur twitter	Cinemactu
52	Twitter	Bouton « *Follow* »	Permet de s'abonner aux publications de la page hote sur twitter	Slate
53	Twitter	Bouton « Tweet»	Permet de publier sur twitter	Ciel Mon Doctorat
54	Twitter	Bouton « Tweet»	Permet de publier sur twitter	Mmorpg
55	Twitter	Bouton « Tweet»	Permet de publier sur twitter	Noisey
56	Twitter	Carte application	Lien vers un magasin d'application mobile	Radio Thunder
57	Twitter	Carte application	Lien vers un magasin d'application mobile	Rappler
58	Twitter	Carte application	Lien vers un magasin d'application mobile	Watford
59	Twitter	Carte *gallery*	Organise une gallerie d'images	Asx
60	Twitter	Carte *gallery*	Organise une gallerie d'images	Gulf News
61	Twitter	Carte *gallery*	Organise une gallerie d'images	Nasa
62	Twitter	Carte *player*	Permet de lire une vidéo depuis twitter	Youtube

63	Twitter	Carte *player*	Permet de lire une vidéo depuis twitter	Baby A***
64	Twitter	Carte *player*	Permet de lire une vidéo depuis twitter	Canal Plus
65	Twitter	Carte *player*	Permet de lire une vidéo depuis twitter	Big Browser
66	Twitter	Carte *summary*	Génère un chapeau introductif	*Le Monde*
67	Twitter	Carte *summary*	Génère un chapeau introductif	Frenchweb
68	Twitter	Carte *summary*	Génère un chapeau introductif	Wired
69	Twitter	*List timeline*	*Timeline* organisée selon une liste de contributeurs	Canal Plus
70	Twitter	Embed search	*Timeline* organisée selon un mot-clé	Canal Plus
71	Twitter	*Timeline* ancrée	*Timeline* publiée sur le site hôte	Good Morning America
72	Twitter	*Timeline* ancrée	*Timeline* publiée sur le site hôte	BnF
73	Twitter	*Timeline* ancrée	*Timeline* publiée sur le site hôte	Times Higher Education
74	Twitter	Tweet ancré	Tweet publié sur le site hôte	Big Browser
75	Twitter	Tweet ancré	Tweet publié sur le site hôte	Les Décodeurs
76	Twitter	Tweet ancré	Tweet publié sur le site hôte	Foxoo
77	Twitter	Vidéo ancrée	Vidéo sur twitter publiée sur le site hôte	Big Browser
78	Twitter	Vidéo ancrée	Vidéo sur twitter publiée sur le site hôte	Sports.fr
79	Twitter	*Intent «Favorite»*	Permet de mettre en favoris sur twitter	Big Browser
80	Twitter	*Intent « Favorite »*	Permet de mettre en favoris sur twitter	Canal Plus
81	Twitter	*Intent « Favorite »*	Permet de mettre en favoris sur twitter	Foxoo
82	Twitter	*Intent « Follow »*	Permet de s'abonner au profil twitter de l'auteur de la publication	BnF
83	Twitter	*Intent « Follow »*	Permet de s'abonner au profil twitter de l'auteur de la publication	Times Higher Education
84	Twitter	*Intent « Follow »*	Permet de s'abonner au profil twitter de l'auteur de la publication	Foxoo
85	Twitter	*Intent « reply »*	Permet de répondre sur twitter à l'auteur de la publication	Big Browser
86	Twitter	*Intent « reply »*	Permet de répondre sur twitter à l'auteur de la publication	Canal Plus
87	Twitter	*Intent « reply »*	Permet de répondre sur twitter à l'auteur de la publication	Foxoo
88	Twitter	*Intent* « retweet »	Permet de répondre sur twitter à l'auteur de la publication	Big Browser
89	Twitter	*Intent* « retweet »	Permet de republier sur twitter	Canal plus
90	Twitter	*Intent* « retweet »	Permet de republier sur twitter	Foxoo
91	Twitter	*Intent* « tweet »	Permet de publier sur twitter	Le monde
92	Twitter	*Intent* « tweet »	Permet de publier sur twitter	Nztec
93	Twitter	*Intent* « tweet »	Permet de publier sur twitter	Tedtalks
94	Twitter	*Intent* « user »	Permet de consulter le profil d'un utilisateur twitter	Gouvernement
95	Twitter	*Intent* « user »	Permet de consulter le profil d'un utilisateur twitter	Times Higher Education

Annexe n° 30 Facebook, « Se connecter avec Facebook », site about.me (processus d'identification)

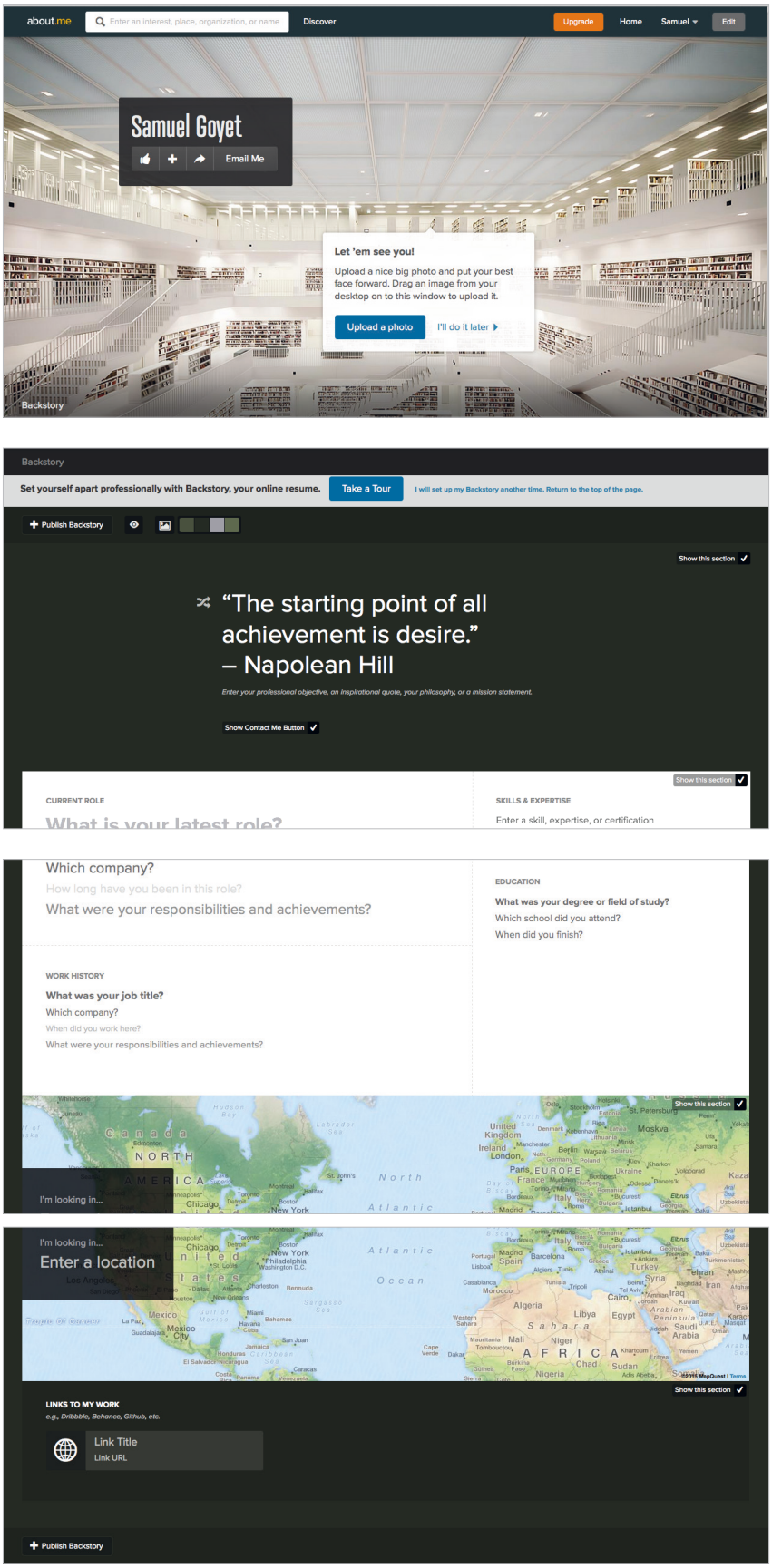






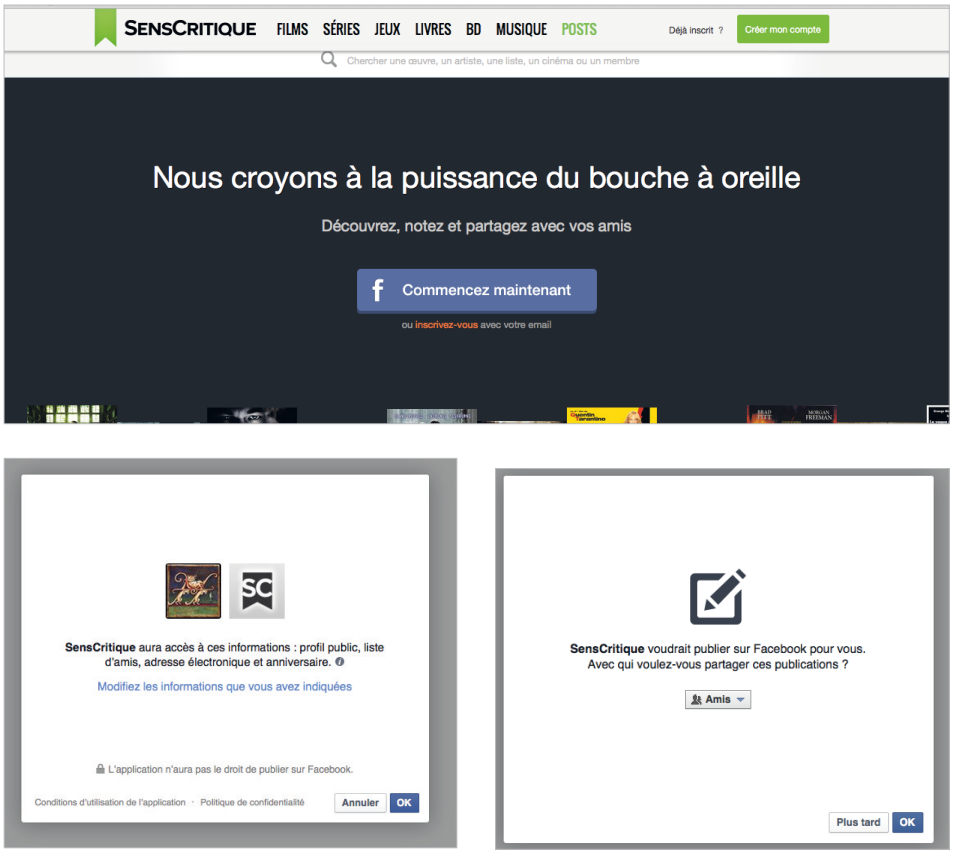
Captures d'écran du site about.me, réalisées 12 août 2015 (détails).

Annexe n° 31 Facebook, « Se connecter avec Facebook », site about.me (après identification)



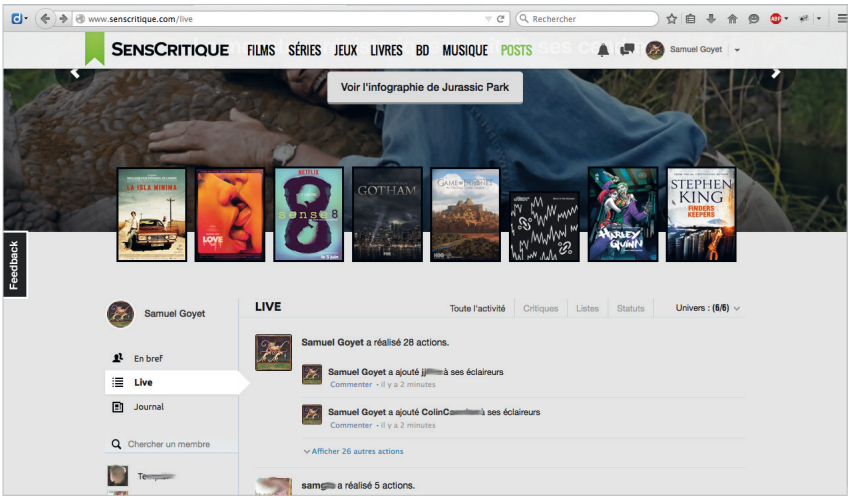
Captures d'écran du site about.me, réalisées 12 août 2015 (détails).

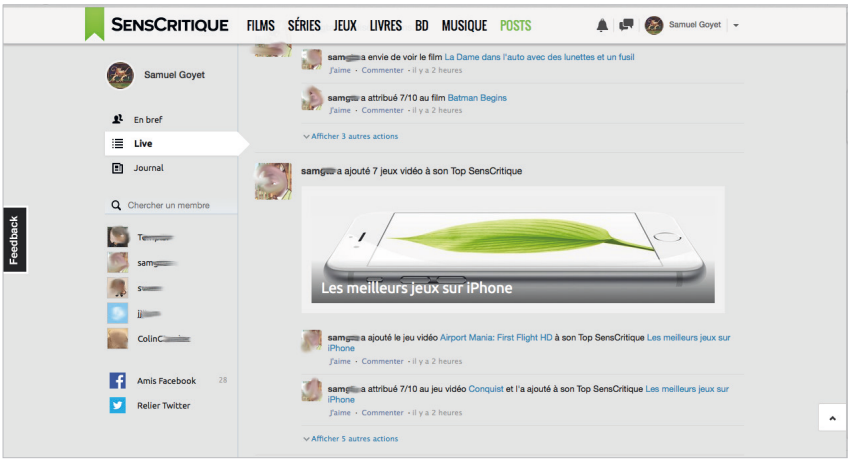
Annexe n° 32 Facebook, « Se connecter avec Facebook », site Sens Critique (processus d'identification)

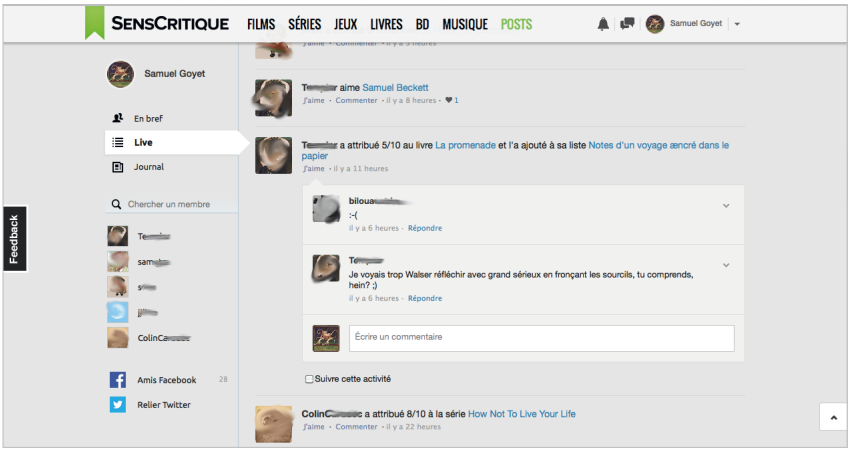


Captures d'écran du site about.me, réalisées 12 août 2015 (détails).

Annexe n° 33 Facebook, « Se connecter avec Facebook », site Sens Critique (après identification)

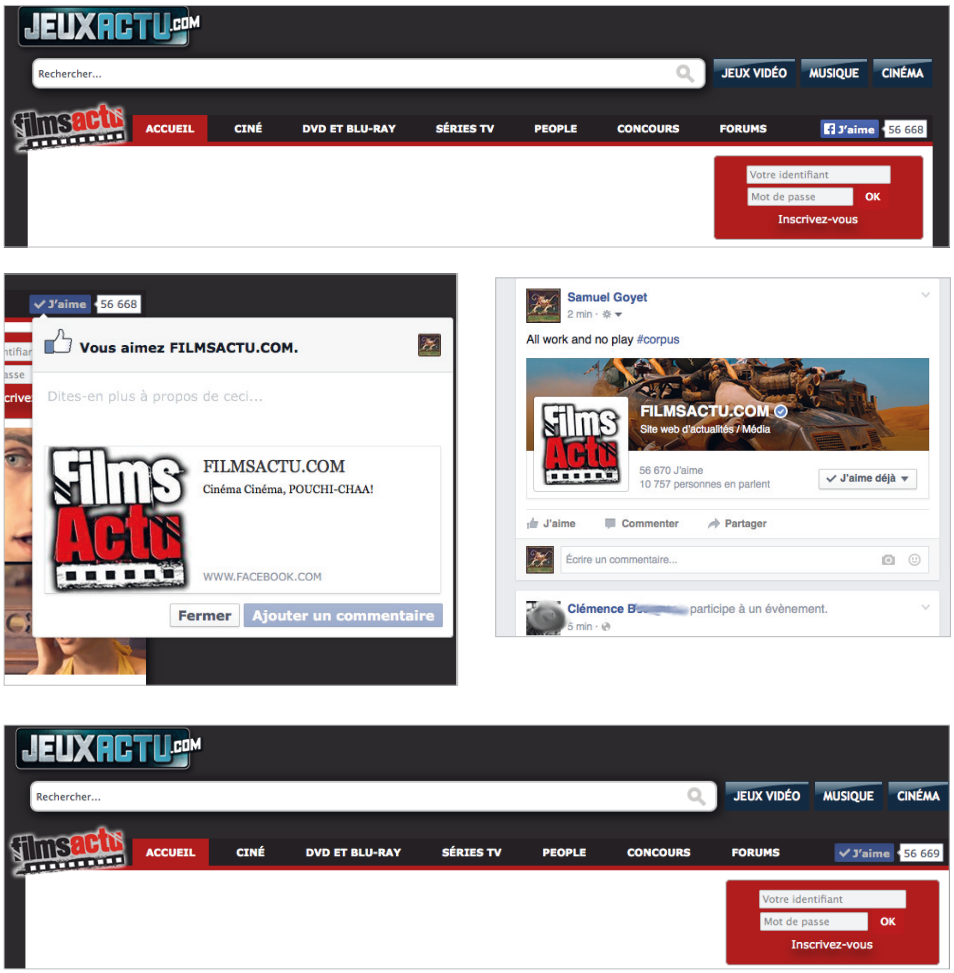






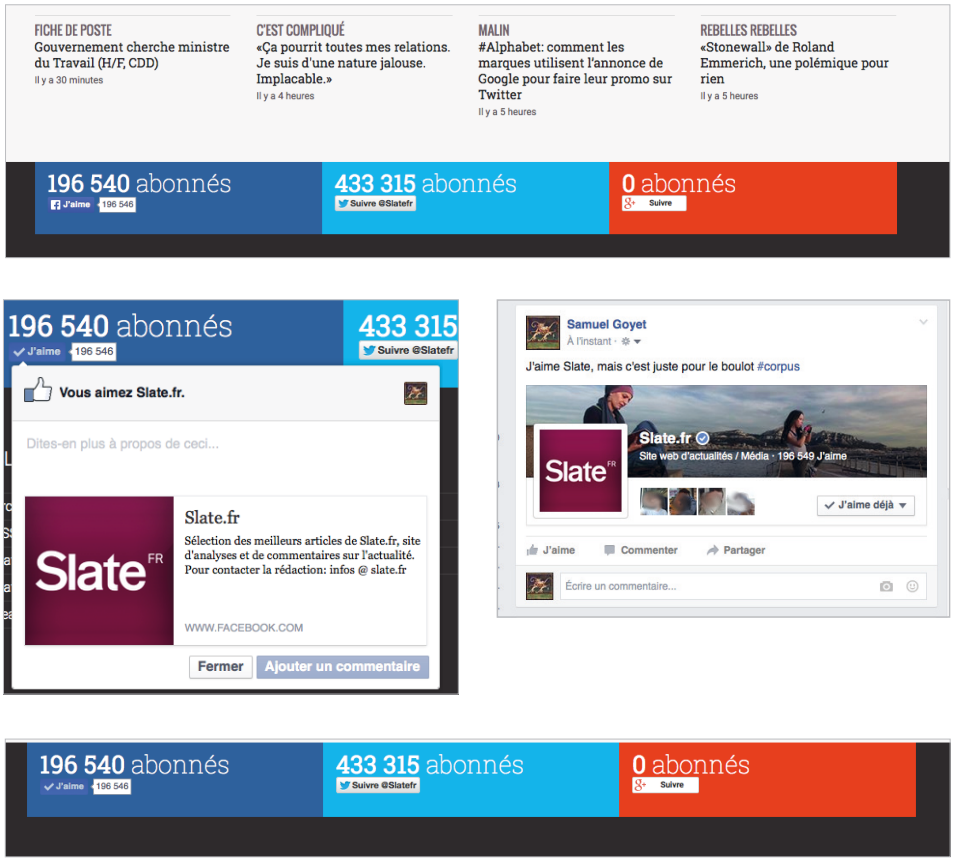
Captures d'écran du site www.senscritique.com, réalisées le 7 août 2015 (détails).

Annexe n° 34 Facebook, bouton « *Like* », site Cinemactu.



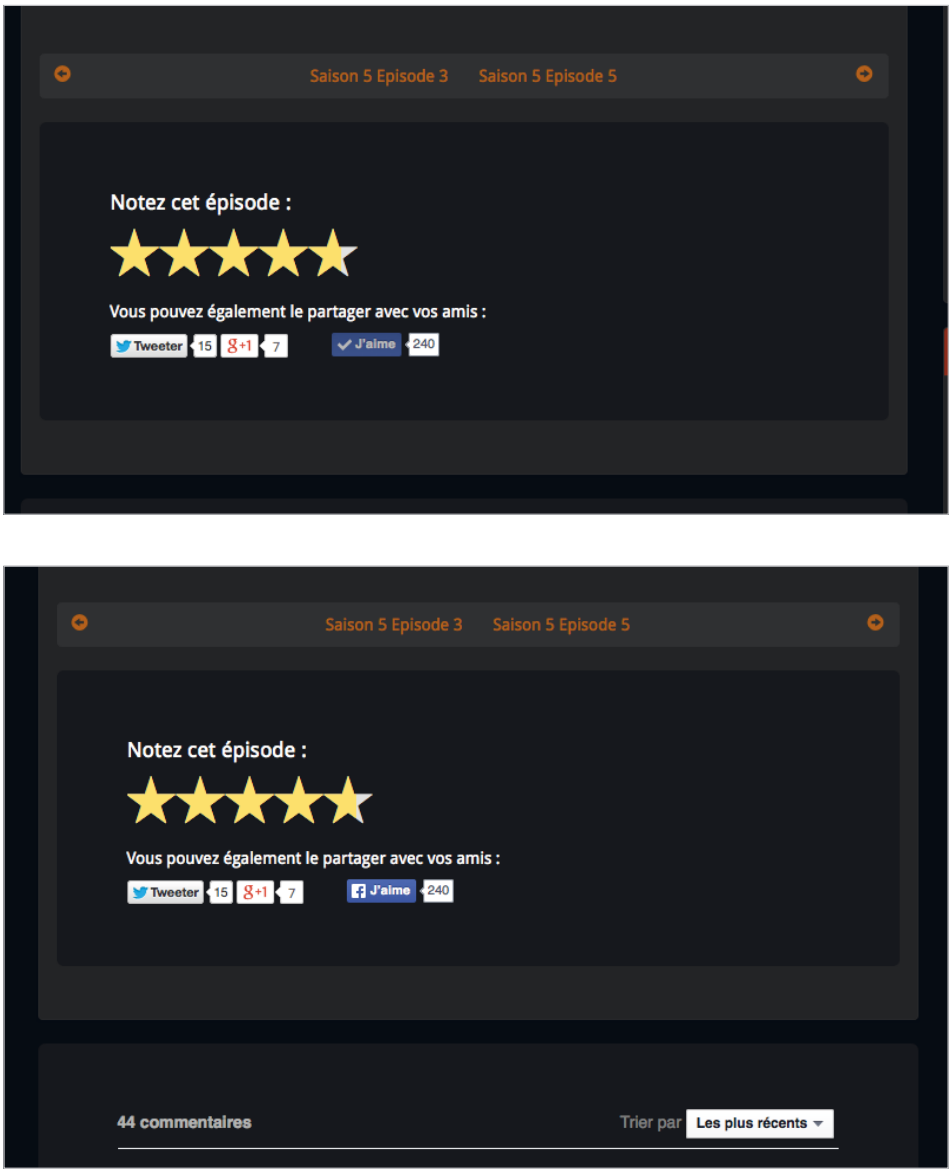
Captures d'écran des pages www.cinema.jeuxactu.com et de facebook.fr (profil personnel), réalisées le 11 août 2015 (détails).

Annexe n° 35 Facebook, bouton « *Like* », site Slate.



Captures d'écran des pages www.slate.fr et de www.facebook.fr (profil personnel), réalisées le 11 août 2015 (détails).

Annexe n° 36 Facebook, bouton « *Like* », site South Park streaming.

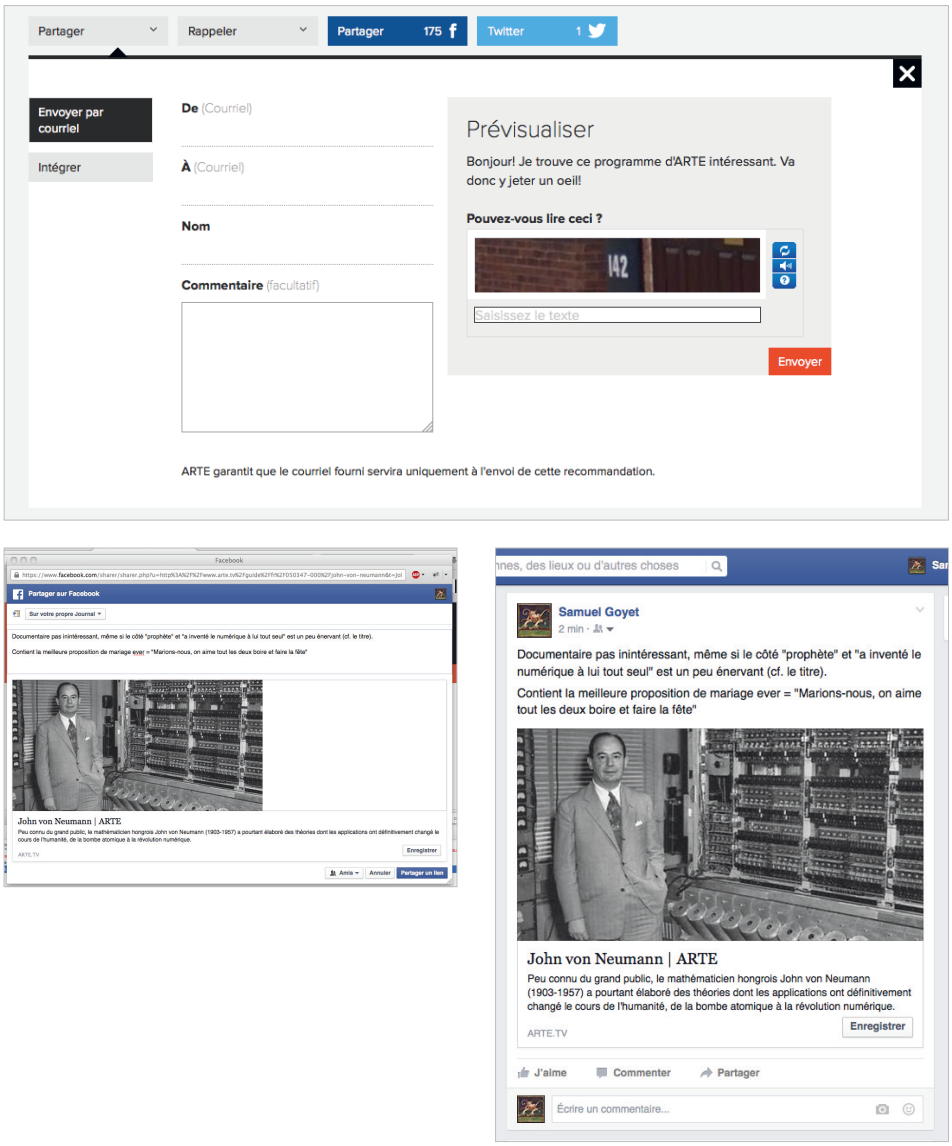


Captures d'écran de la page www.south-park-streaming.com/saison-5/episode-4/scott-tenorman-doit-mourir, réalisées le 7 août 2015 (détails).

Annexe n° 37 Facebook, bouton « *Send* », site Cool Israël.

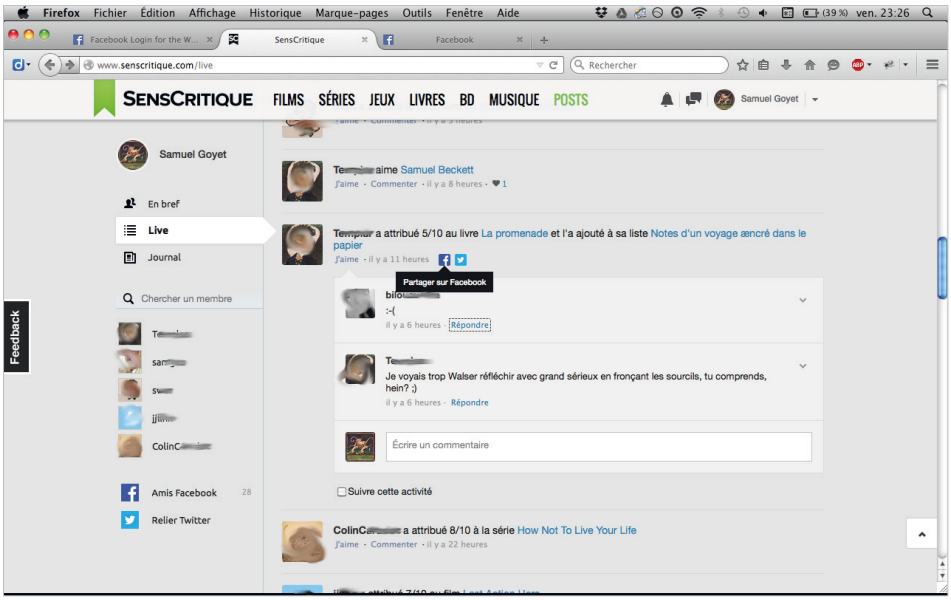
Captures d'écran des pages coolisrael.fr/25041/coince-a-paris-tinquiete-la-plage-de-tel-aviv-de-barque-chez-toi et de ww.facebook.fr (profil personnel), réalisées le 16 août 2015 (détails).

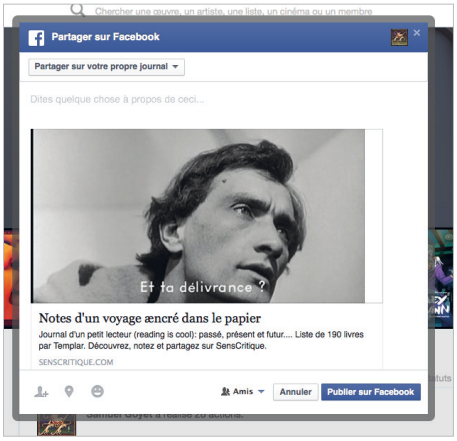
Annexe n° 38 Facebook, bouton « *Share* », site Arte.



Captures d'écran des pages site www.arte.tv/guide/fr/050347-000/john-von-neumann?autoplay=1 et de www.facebook.fr (profil personnel), réalisées le 5 août 2015 (détails).

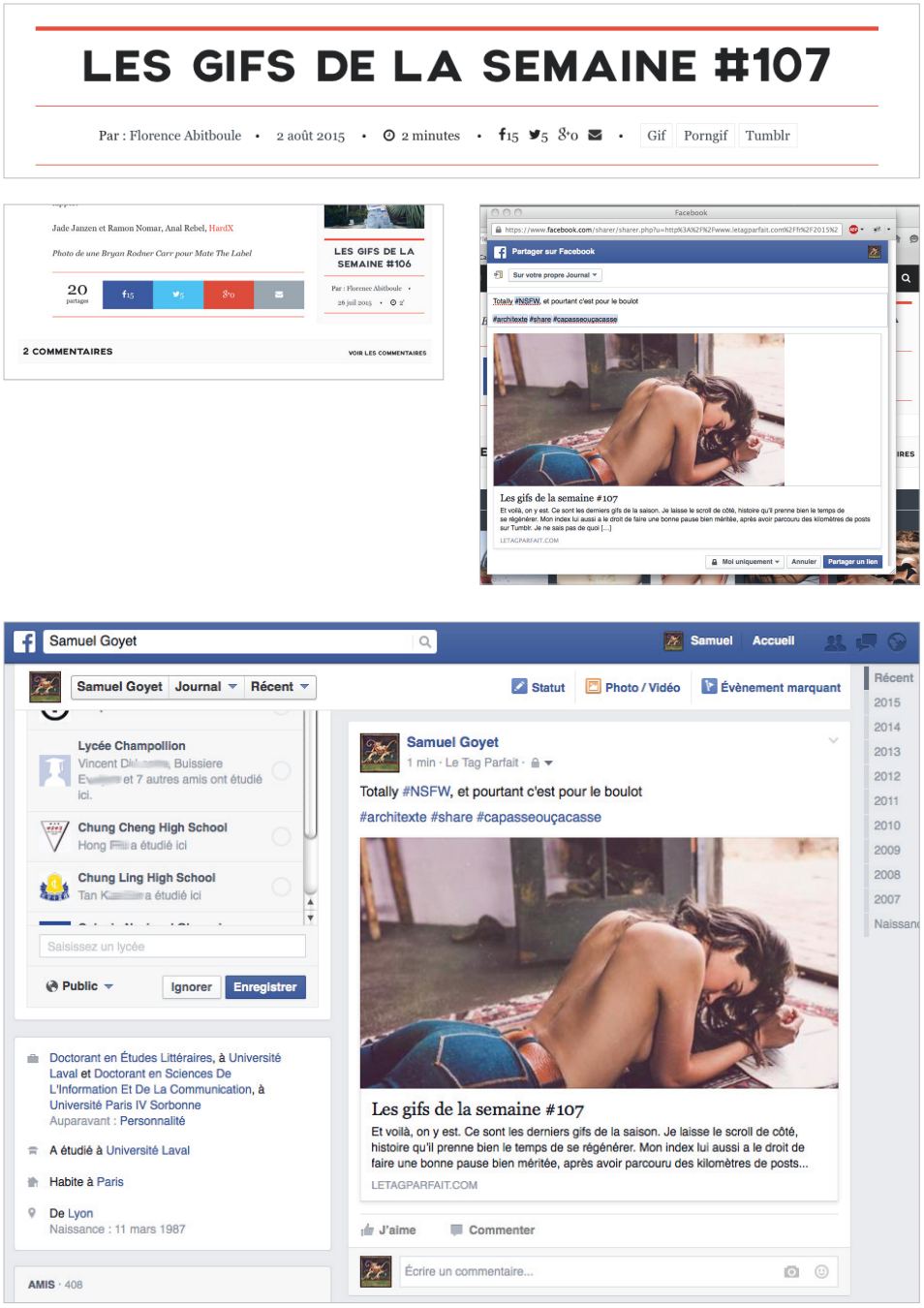
Annexe n° 39 Facebook, bouton « *Share* », site Sens Critique.





Captures d'écran du site www.senscritique.com (après identification) et www.facebook.fr (profil personnel), réalisées le 7 août 2015 (détails).

Annexe n° 40 Facebook, bouton « *Share* », site Le Tag Parfait.

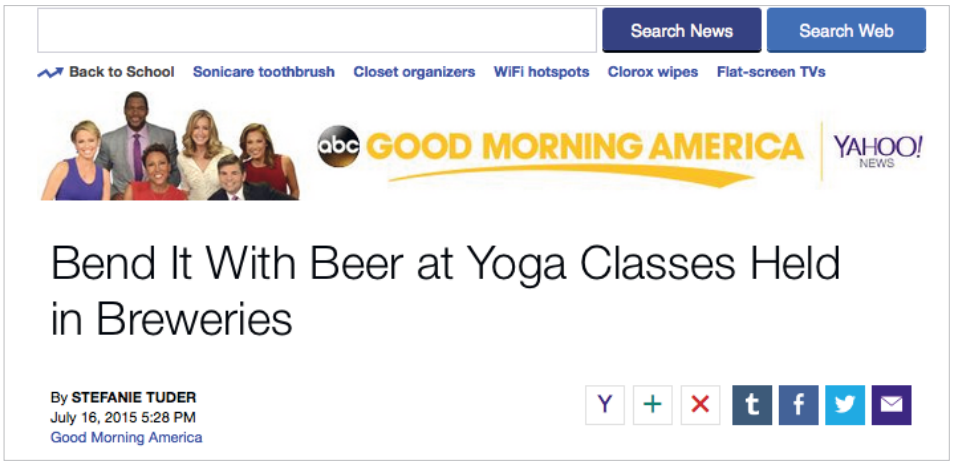


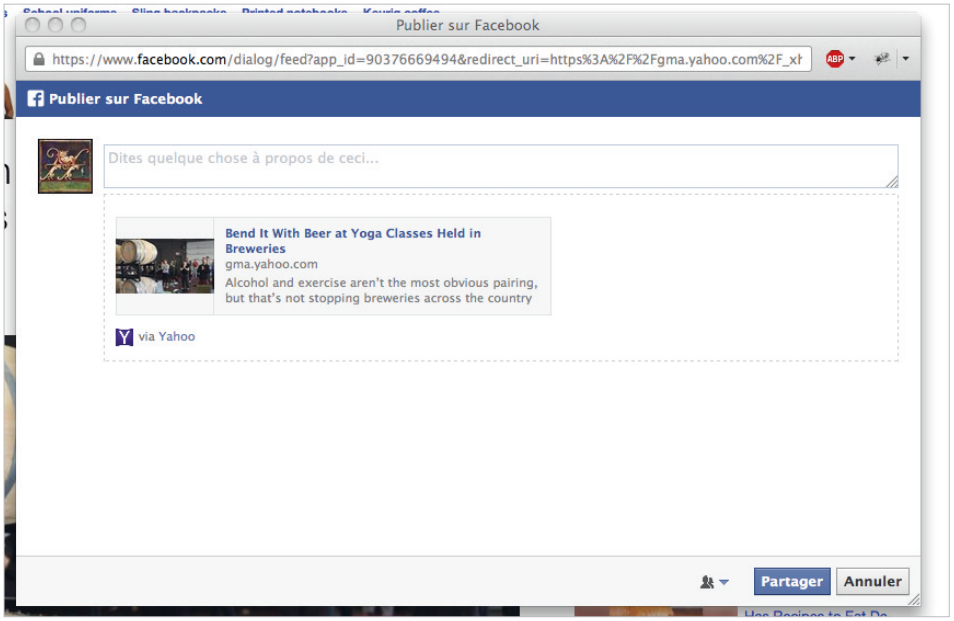
Captures d'écran du site www.letagparfait.com et de www.facebook.fr (profil personnel), réalisées le 11 août 2015 (détails).

Annexe n° 41 Facebook, publication ancrée, site Le Talk Foot.

Captures d'écran des pages letalkfoot.fr/video/ligue-des-champions/video/le-resume-de-barcelone-fc-seville-5-4/ et de www.facebook.fr (profil personnel), réalisées le 13 août 2015 (détails).

Annexe n° 42 Facebook, « *Feed Dialog* », site Good Morning America.



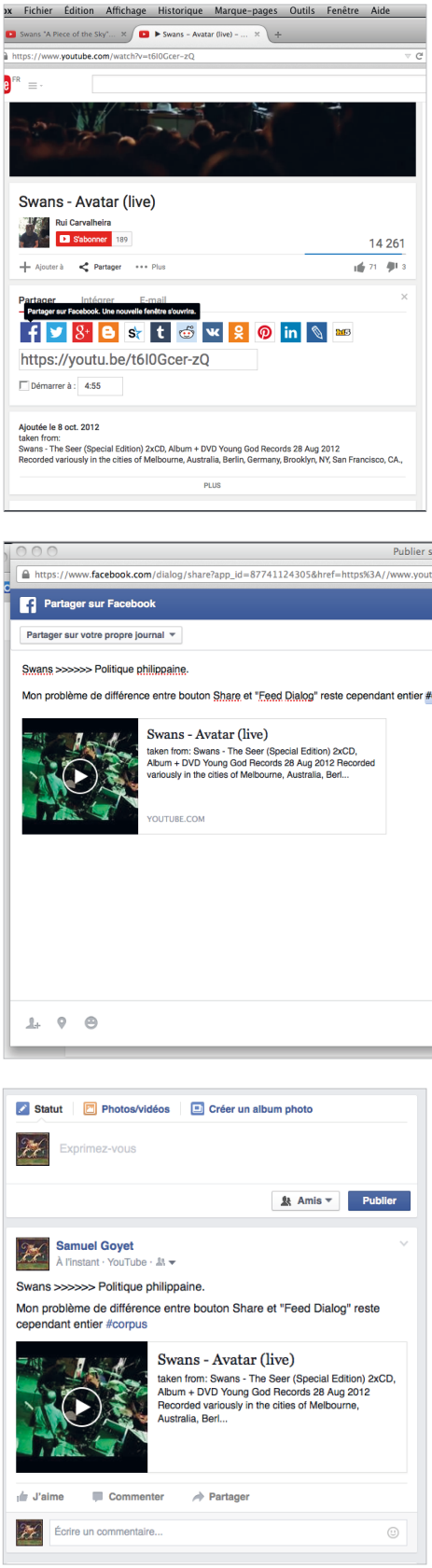


Captures d'écran des pages gma.yahoo.com/bend-beer-yoga-classes-held-breweries/212825058--abc-news-Recipes.html et de www.facebook.fr (profil personnel), réalisées le 13 août 2015 (détails).

Annexe n° 43 Facebook, « *Feed Dialog* », site Rappler.

Captures d'écran des pages rappler.com/nation/89508/mar-roxas-jejomar-binay-2016 et de facebook.fr (profil personnel), réalisées le 12 août 2015 (détails).

Annexe n° 44 Facebook, « *Feed Dialog* », site YouTube.

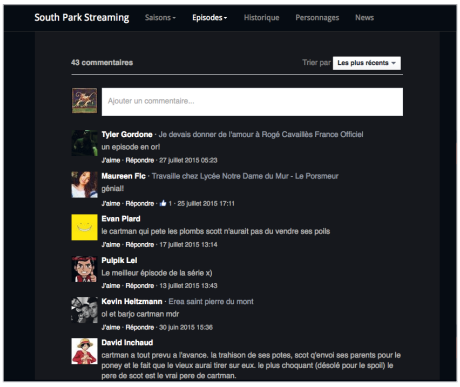


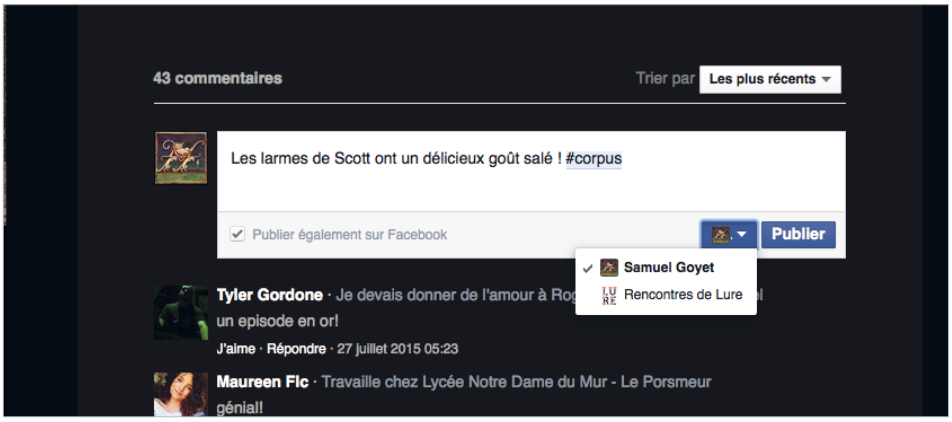
Captures d'écran des pages www.youtube.com/watch?v=t6I0Gcer-zQ et de facebook.fr (profil personnel), réalisées le 12 août 2015 (détails).

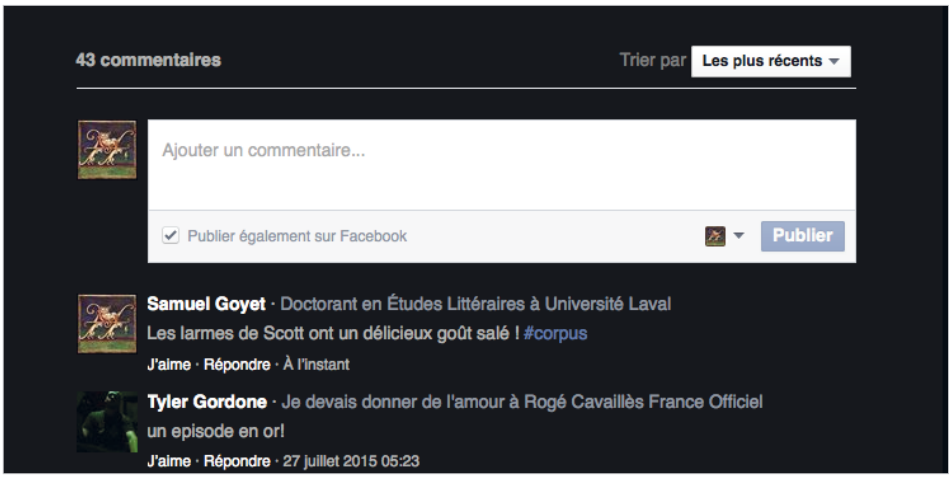
Annexe n° 45 Facebook, *Plugin* commentaires, site Cool Israël.

Captures d'écran de la page coolisrael.fr/25041/coince-a-paris-tinquiete-la-plage-de-tel-aviv-de-barque-chez-toi, réalisées le 16 août 2015 (détails).

Annexe n° 46 Facebook, *Plugin* commentaires, site South Park streaming







Captures d'écran de la page www.south-park-streaming.com/saison-5/episode-4/scott-tenor-man-doit-mourir, réalisées le 7 août 2015 (détails).

Annexe n° 47 Facebook, *Plugin* Page, site Beta Series.

Captures d'écran des pages www.betaseries.com et de facebook.fr (profil personnel), réalisées le 12 août 2015 (détails).

Annexe n° 48 Facebook, *Plugin* Page, site Droit Finances.

Captures d'écran des pages www.droit-finances.commentcamarche.net/faq/6848/preavis-de-depart-d-un-mois-location-modele et de facebook.fr (profil personnel), réalisées le 11 août 2015 (détails).

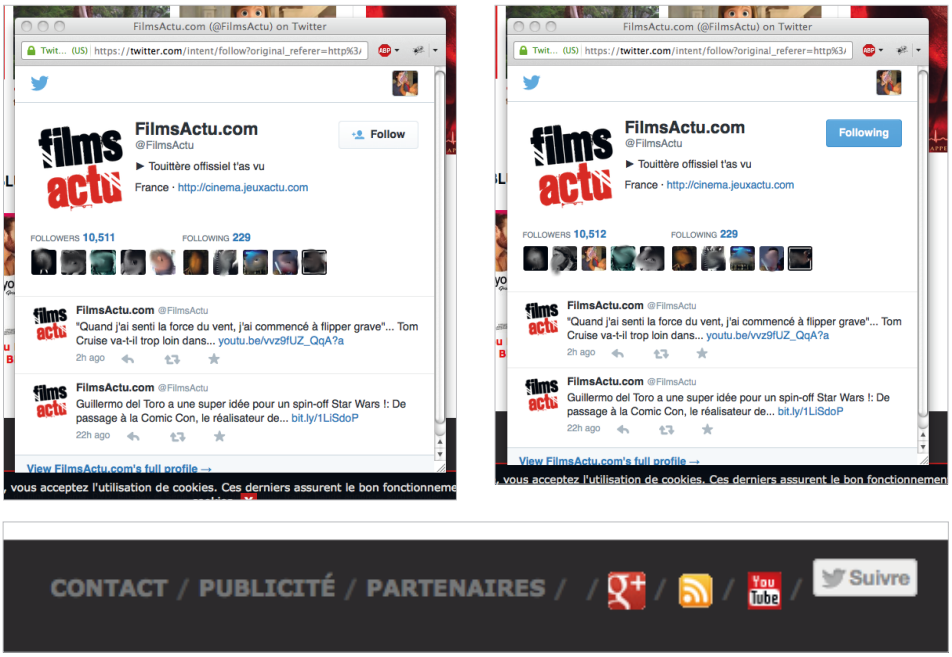
Annexe n° 49 Facebook, *Plugin* Page, site Le Talk Foot.

Captures d'écran des pages www.letalkfoot.fr et de facebook.fr (profil personnel), réalisées le 11 août 2015 (détails).

Annexe n° 50 Twitter, bouton « *Follow* », site Beta Series.

Captures d'écran des pages www.betaseries.com et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 51 Twitter, bouton « *Follow* », site Cinemactu.

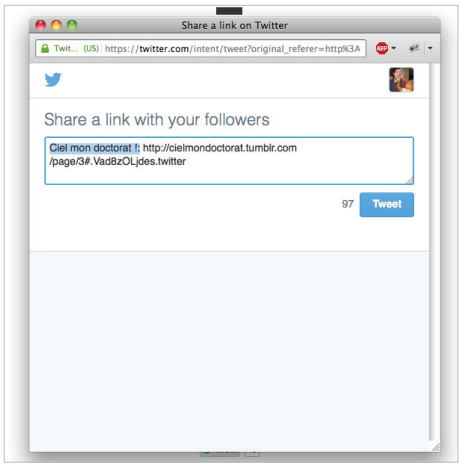


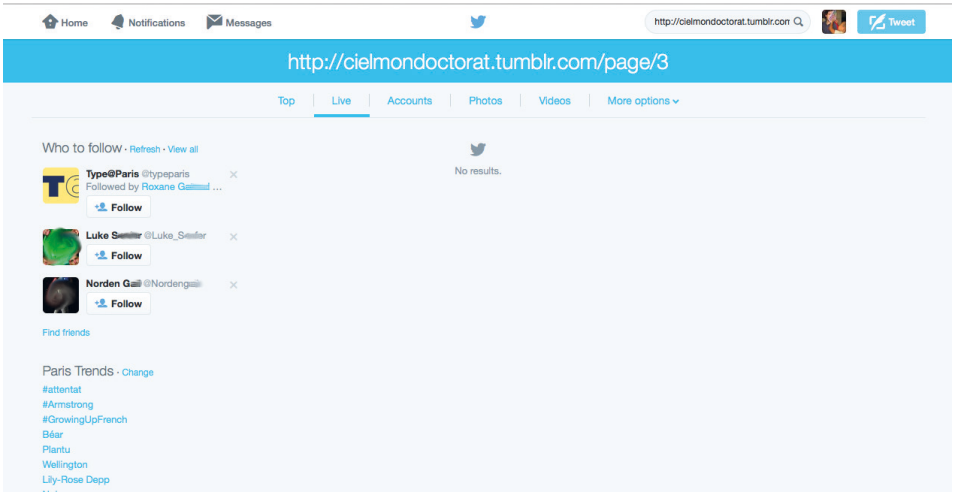
Captures d'écran des pages www.cinema.jeuxactu.com et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 52 Twitter, bouton « *Follow* », site Slate.

Captures d'écran des pages www.slate.fr et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

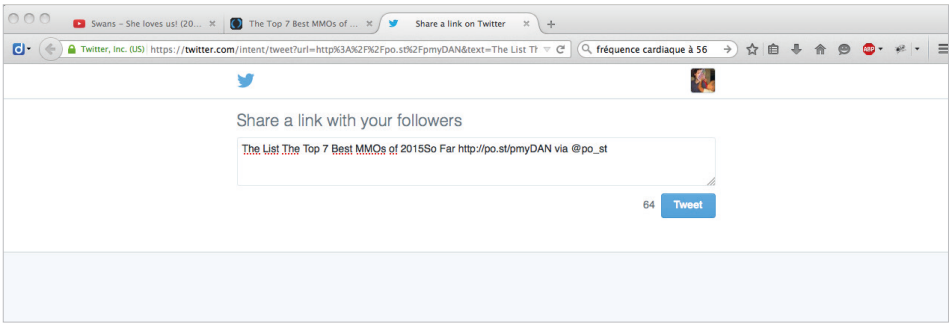
Annexe n° 53 Twitter, bouton « Tweet », site Ciel Mon Doctorat.

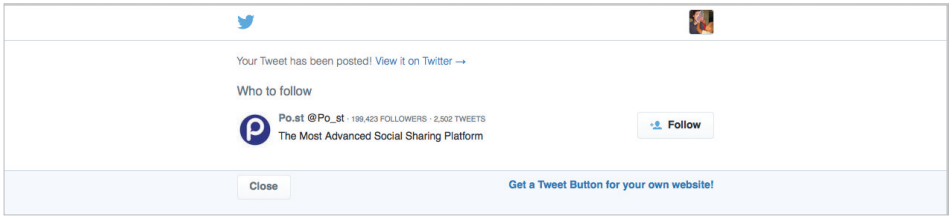


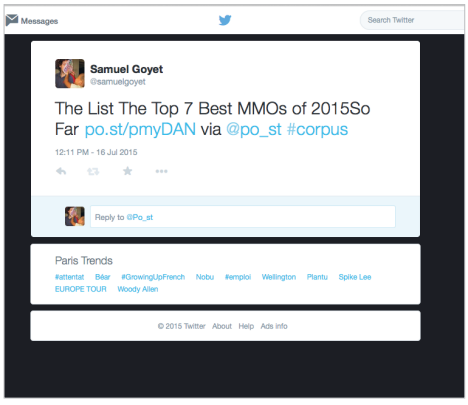


Captures d'écran des pages http://cielmondoctorat.tumblr.com/page/3 et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 54 Twitter, bouton « Tweet », site MMORPG.

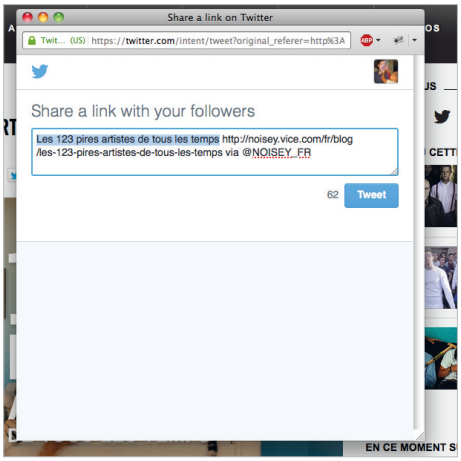


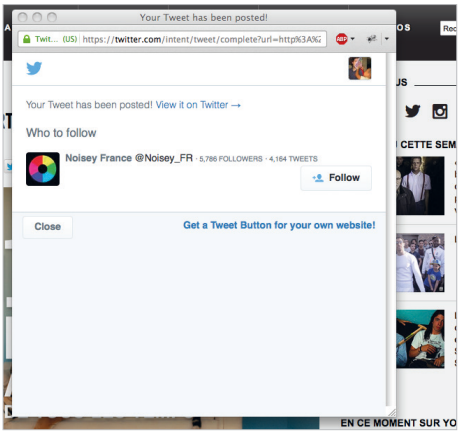




Captures d'écran des pages www.mmorpg/com/newsroom et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 55 Twitter, bouton « Tweet », site Noisey.



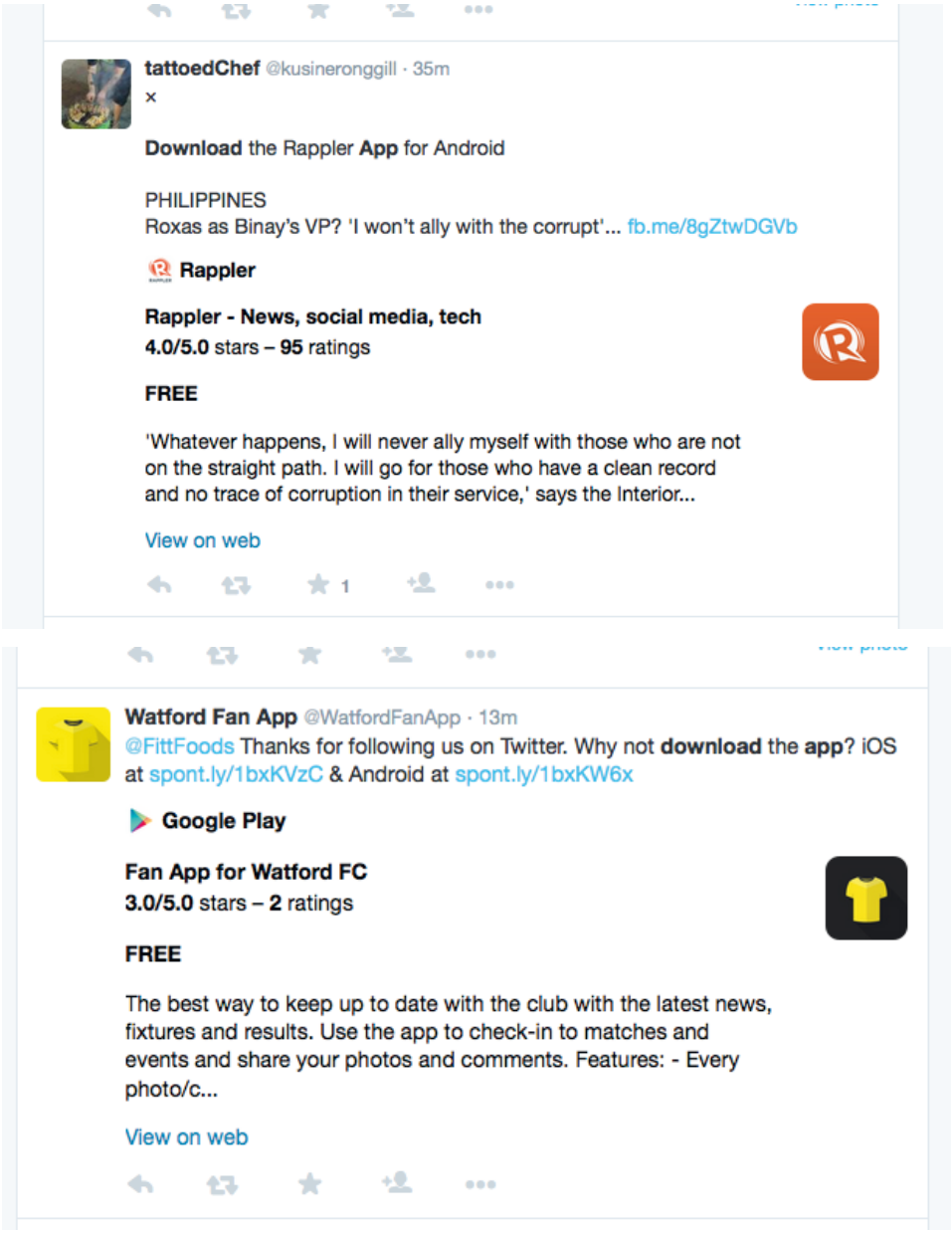


Captures d'écran des pages http/www.noisey.vice.com/123pires et de www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 56 Twitter, Carte application, Radio Thunder.

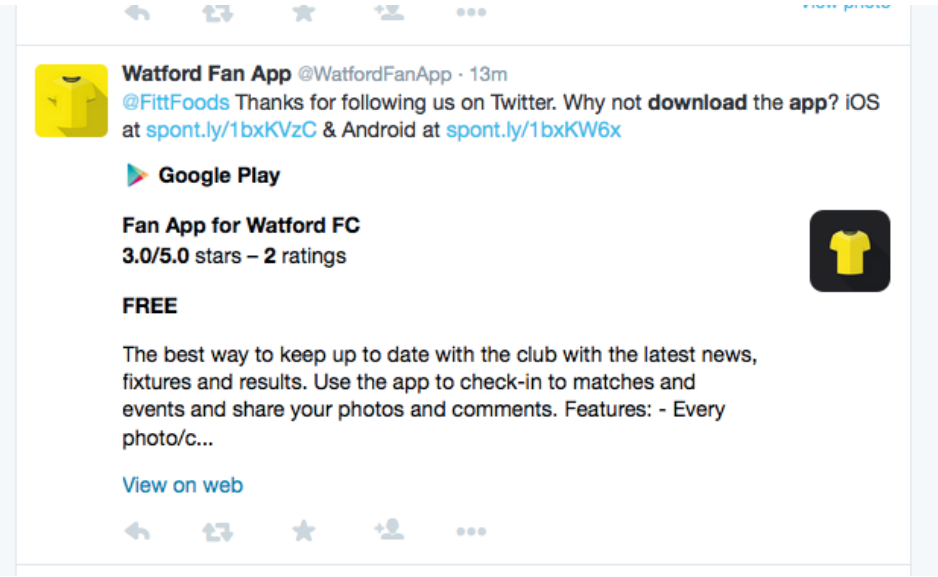
Capture d'écran de la page www.twitter.com (profil personnel), réalisée le 18 juillet 2015 (détails).

Annexe n° 57 Twitter, Carte application, Rappler.



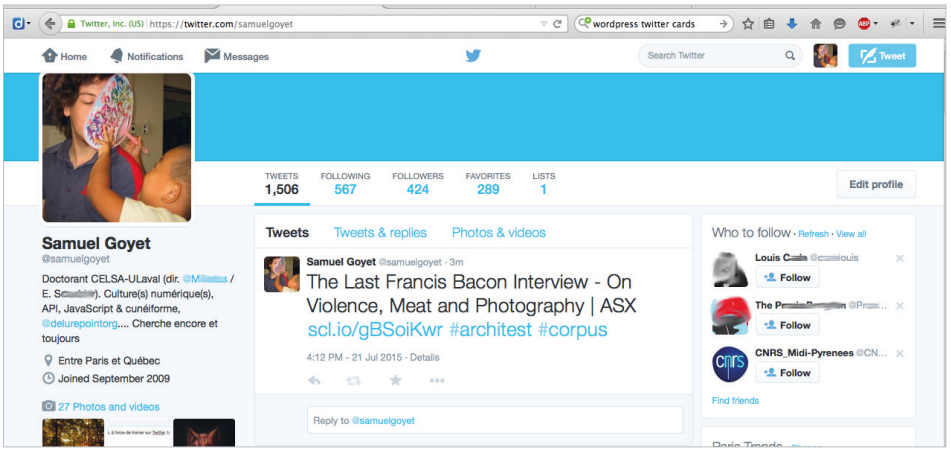
Captures d'écran de la page www.twitter.com (profil personnel), réalisées le 18 juillet 2015 (détails).

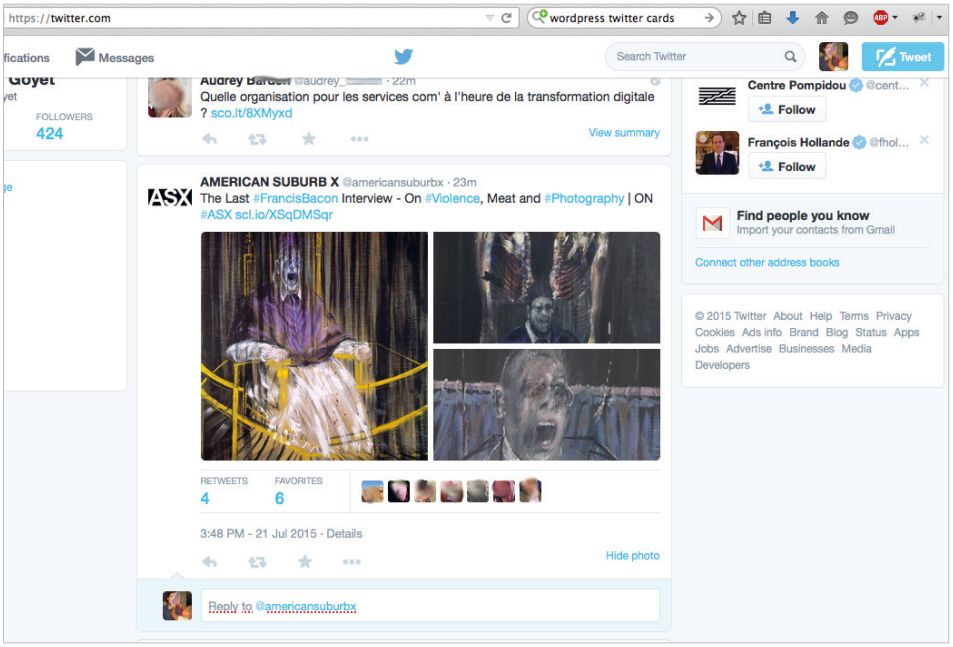
Annexe n° 58 Twitter, Carte application, Watford.



Capture d'écran de la page www.twitter.com (profil personnel), réalisée le 18 juillet 2015 (détails).

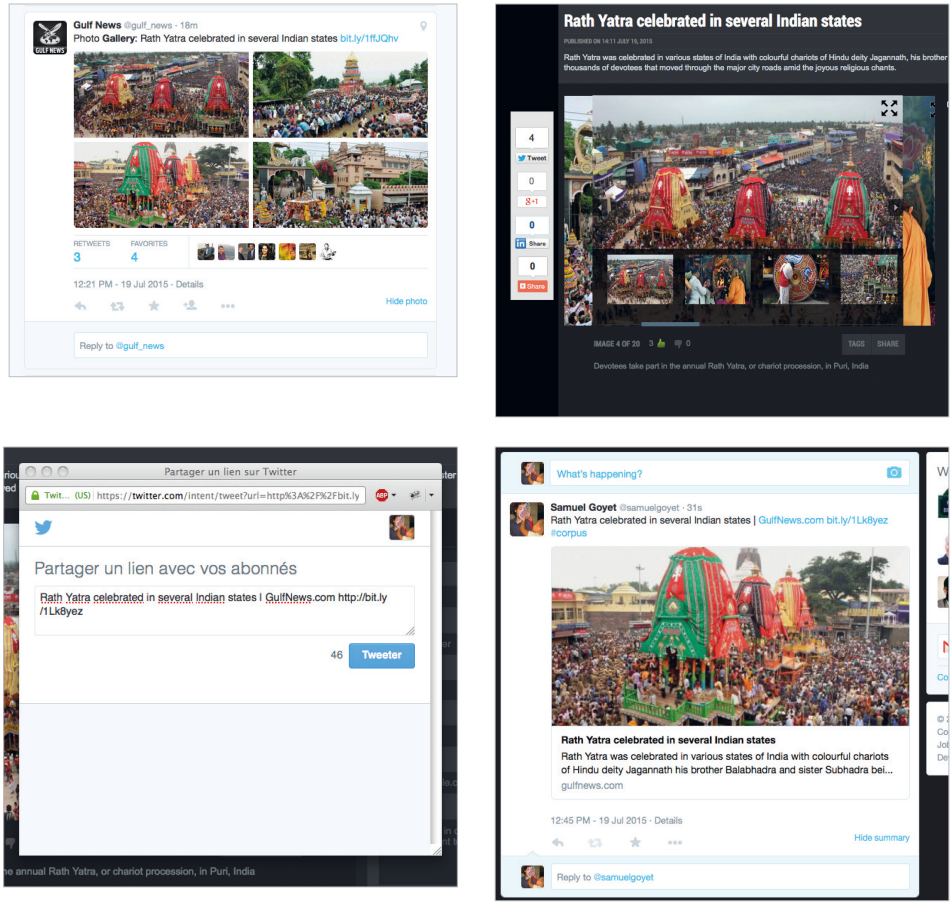
Annexe n° 59 Twitter, Carte *gallery*, site ASX.





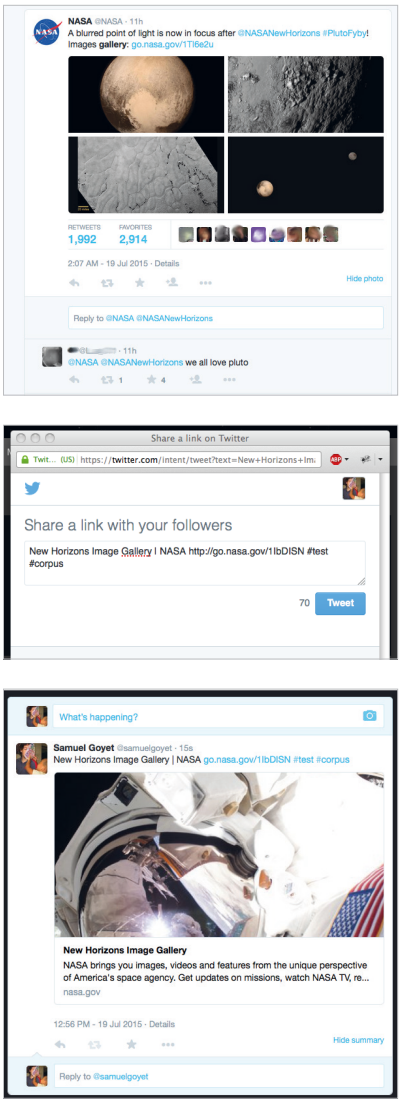
Captures d'écran de la page www.twitter.com (profil personnel), réalisées le 21 juillet 2015 (détails).

Annexe n° 60 Twitter, Carte *gallery*, site Gulf News.



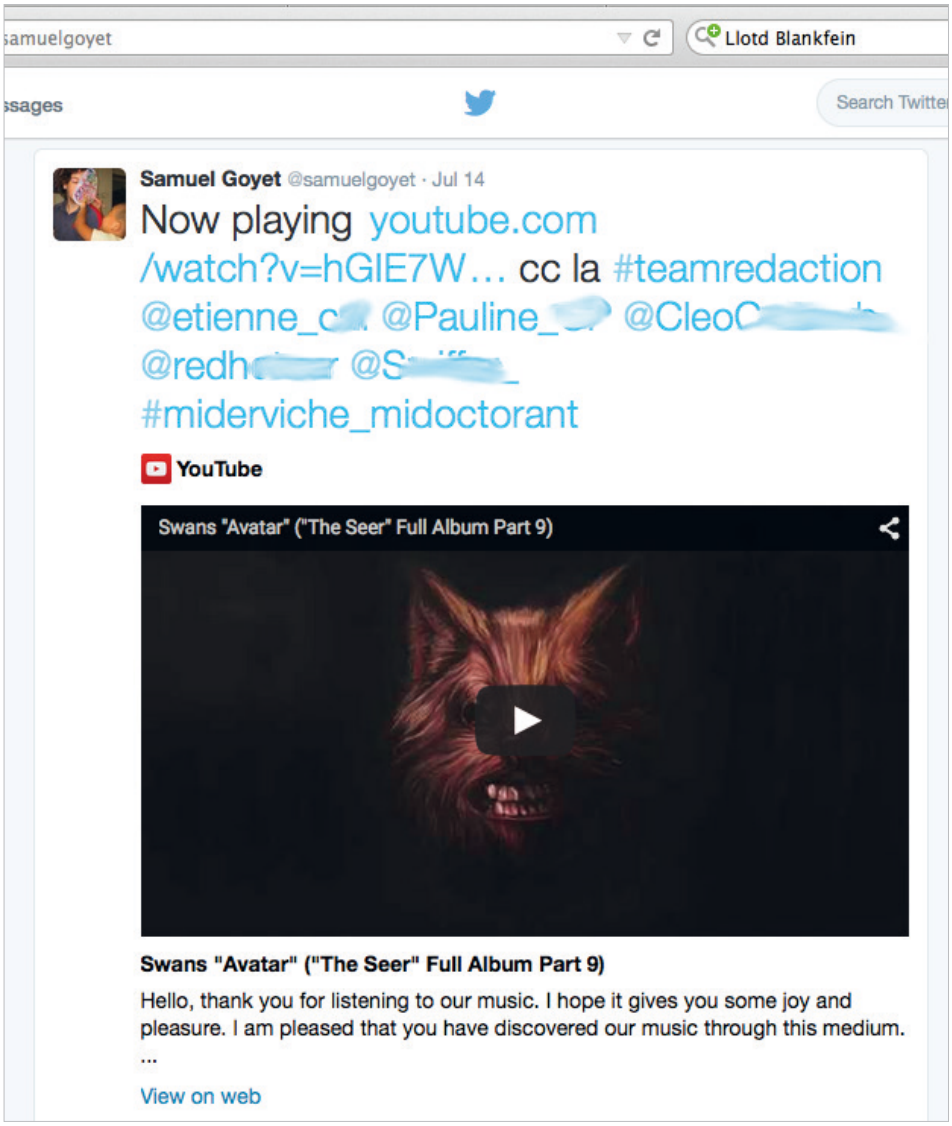
Captures d'écran des pages www.twitter.com (profil personnel) et de www.gulfnews.com/multimedia/framed-culture/rath-yatra-celebrated-in-several-indian-states, réalisées le 19 juillet 2015 (détails).

Annexe n° 61 Twitter, Carte *gallery*, site NASA.



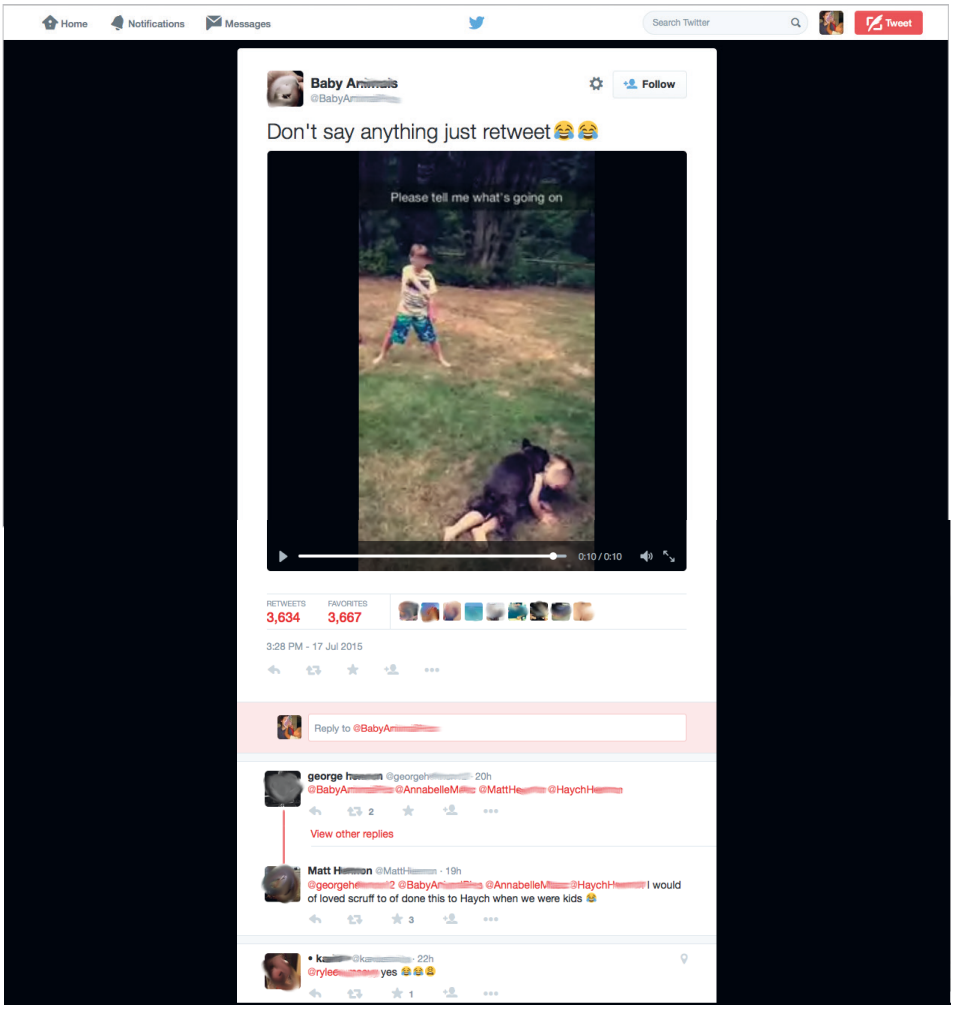
Captures d'écran des pages www.twitter.com (profil personnel) et de www.nasa.gov/mission/pages/newhorizons/images/index.html?linkId=15664795, réalisées le 19 juillet 2015 (détails).

Annexe n° 62 Twitter, Carte *player*, site YouTube.



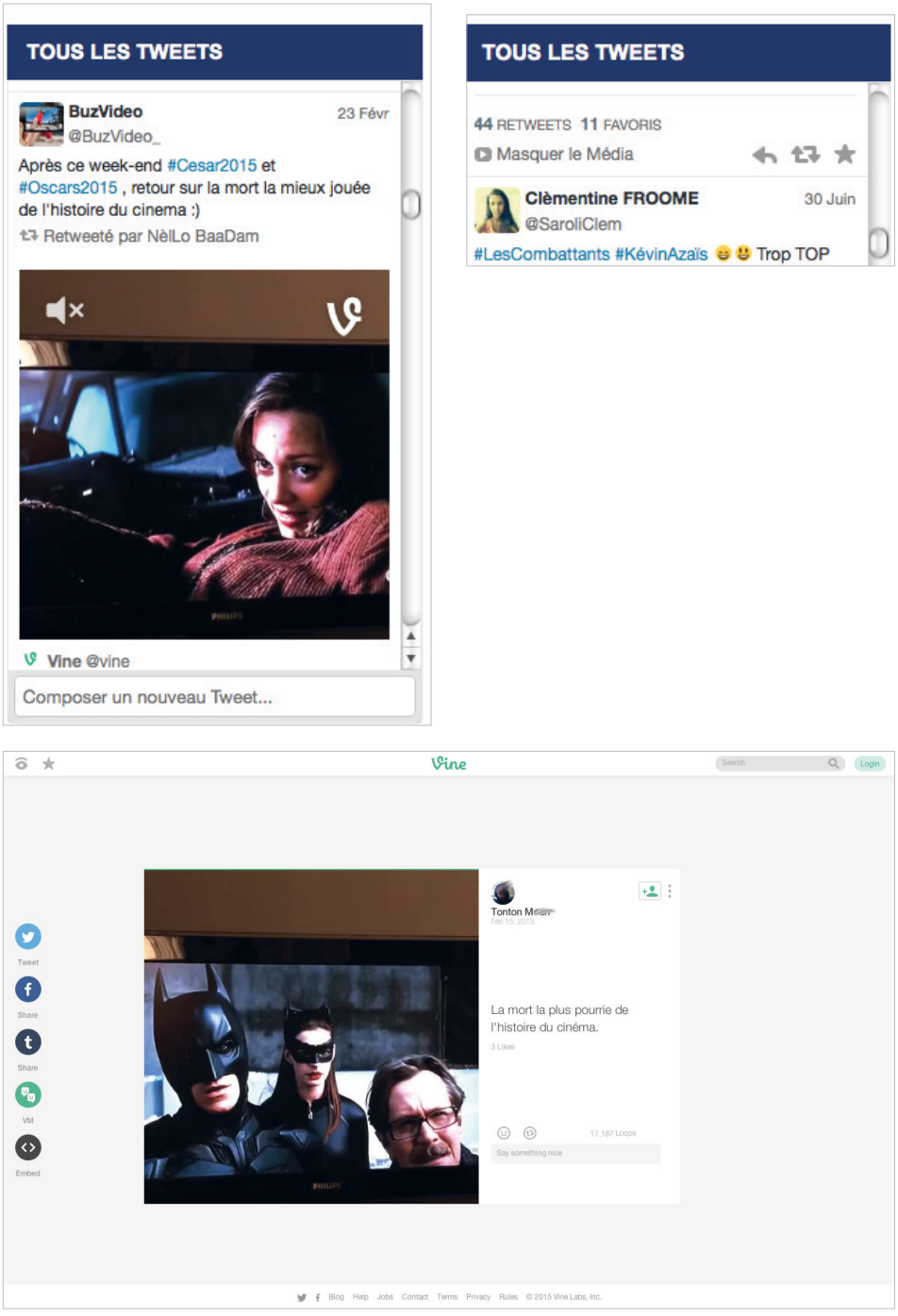
Captures d'écran de la page www.twitter.com (profil personnel) réalisée le 18 juillet 2015 (détails).

Annexe n° 63 Twitter, Carte *player*, site Baby A***.



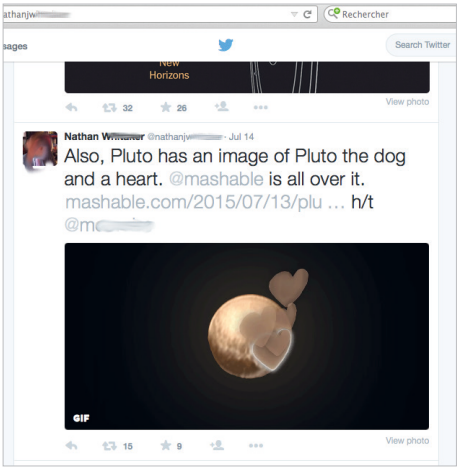
Capture d'écran de la page www.twitter.com (profil personnel) réalisée le 18 juillet 2015 (détails).

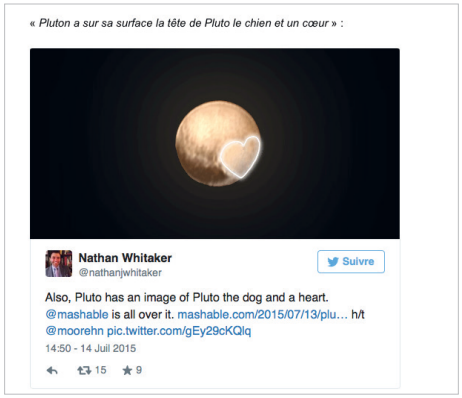
Annexe n° 64 Twitter, Carte *player*, site Canal Plus.

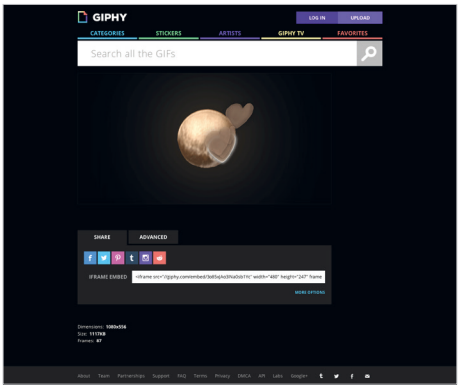


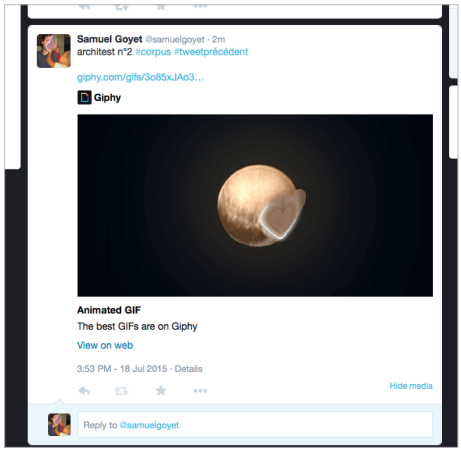
Captures d'écran des pages www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/livetweet-ceremonie-cesar-2015.html et vine.co-v-OP37wZwlHDr, réalisées le 17 juillet 2015 (détails).

Annexe n° 65 Twitter, Carte *Player*, site Big Browser.



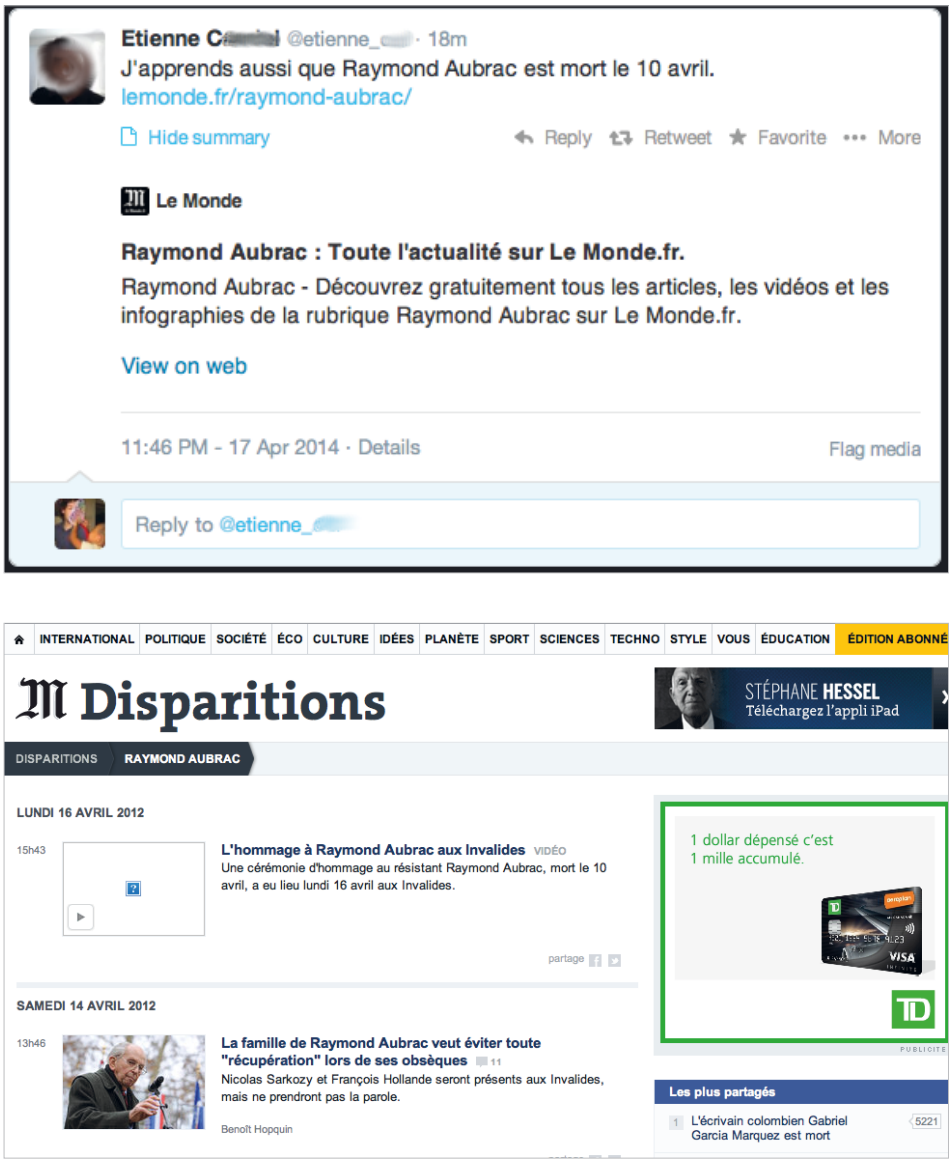






Captures d'écran des pages bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx, giphy.com-gifs-3o85xJAo3INa0sb1Yc(20150718)_site et twitter.com (profil personnel), réalisées le 18 juillet 2015 (détails).

Annexe n° 66 Twitter, Carte *summary*, site Le Monde.

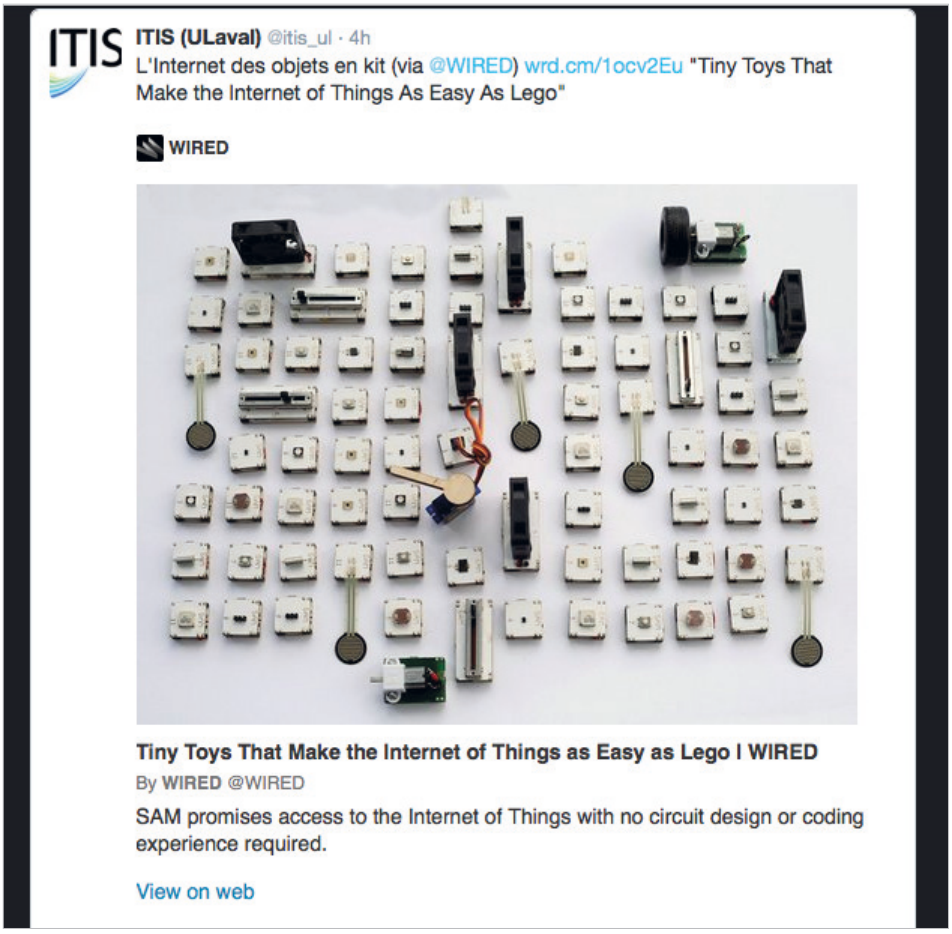


Captures d'écran des pages twitter.com (profil personnel) et lemonde.fr/raymond-aubrac, réalisées le 18 avril 2014 (détails).

Annexe n° 67 Twitter, Carte *summary*, site FrenchWeb.

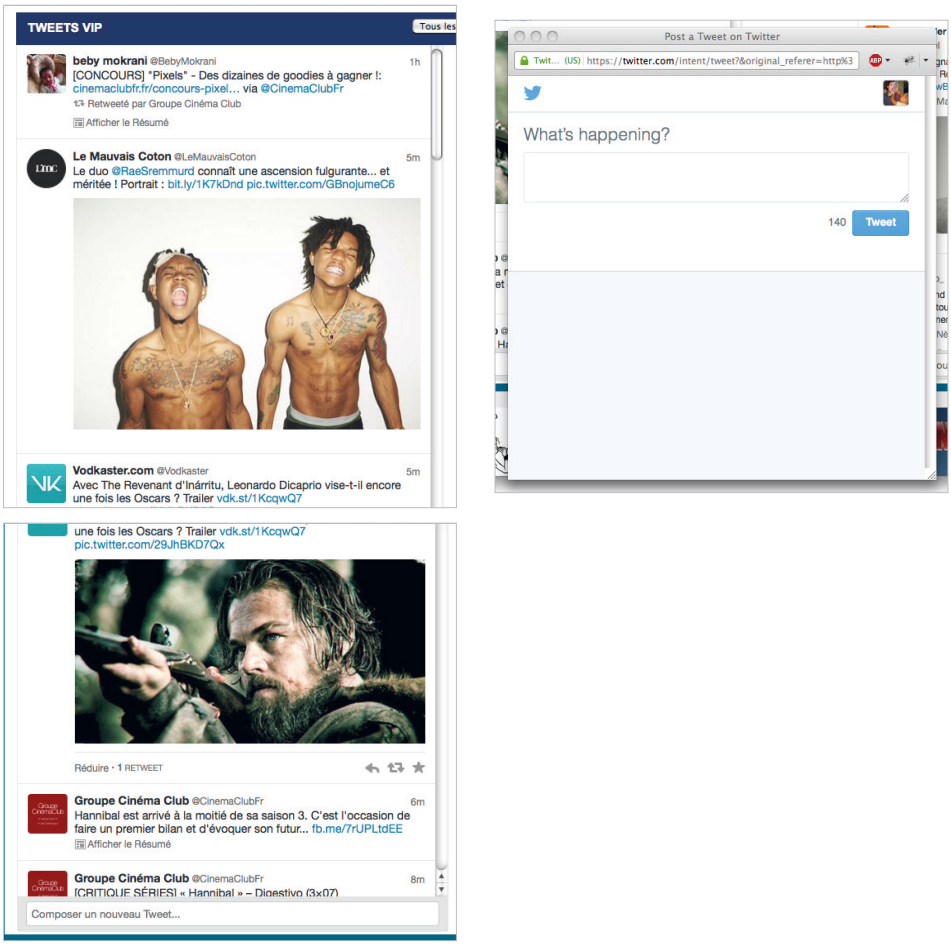
Capture d'écran de la page twitter.com (profil personnel), réalisée le 16 octobre 2014 (détails).

Annexe n° 68 Twitter, Carte *summary*, site Wired.



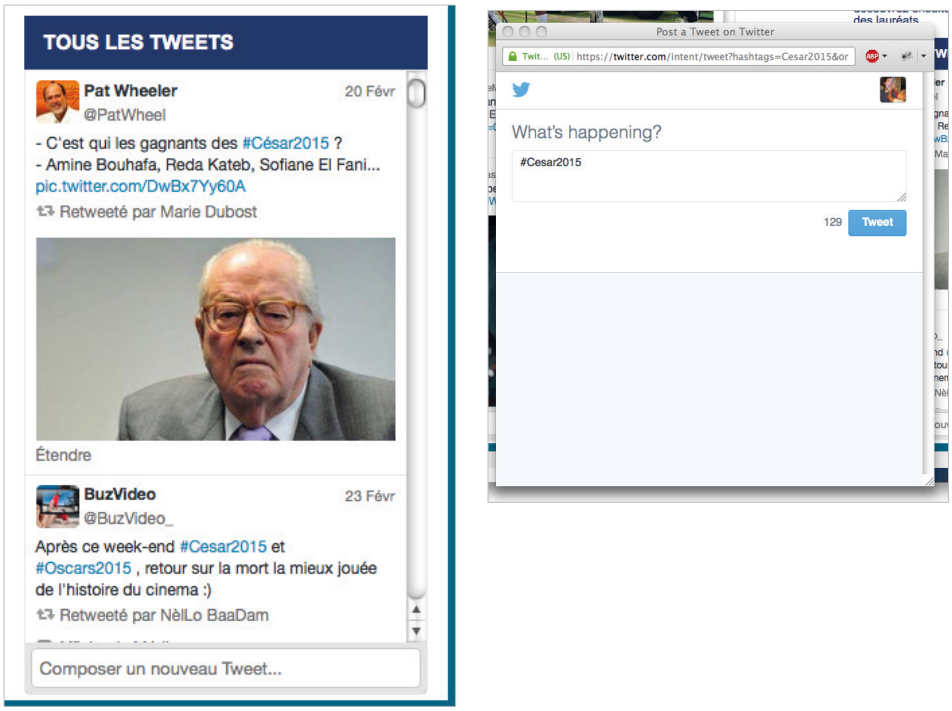
Capture d'écran de la page twitter.com (profil personnel), réalisée le 16 octobre 2014 (détails).

Annexe n° 69 Twitter, *list timeline*, site Canal Plus.



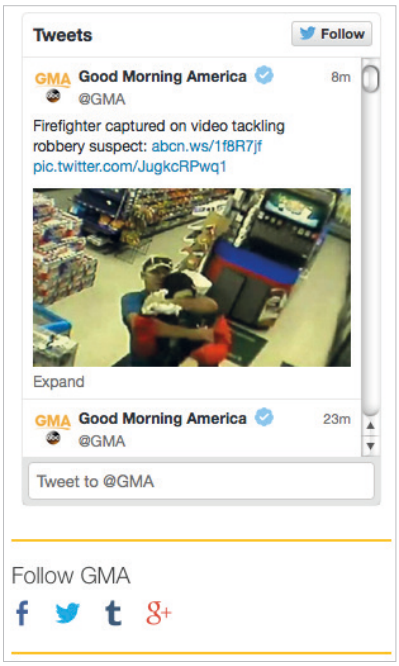
Captures d'écran de la page www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/livetweet-ceremonie-cesar-2015.html, réalisées le 17 juillet 2015 (détails).

Annexe n° 70 Twitter, *embed search*, site Canal Plus.



Captures d'écran de la page www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/livetweet-ceremonie-cesar-2015.html, réalisées le 17 juillet 2015 (détails).

Annexe n° 71 Twitter, *timeline* ancrée, site Good Morning America.

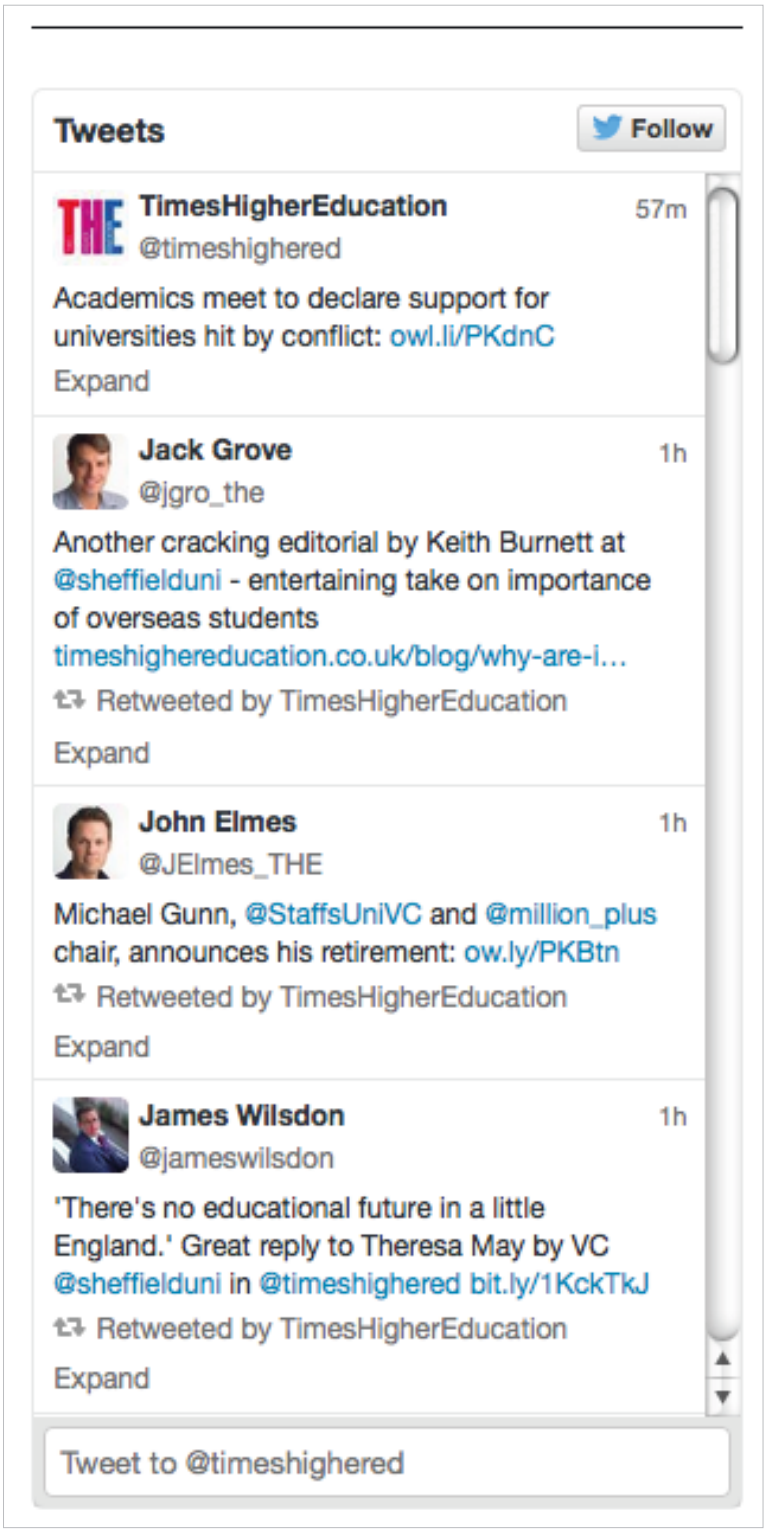


Captures d'écran de la page www.gma.yahoo.com/bend-beer-yoga-classes-held-breweries/212825058--abc-news/Recipes.html, réalisées le 17 juillet 2015 (détails).

Annexe n° 72 Twitter, *timeline* ancrée, site BNF.

Capture d'écran de la page www.bnf.fr/acc/x.accueil.html, réalisée le 17 juillet 2015 (détails).

Annexe n° 73 Twitter, *timeline* ancrée, site Times Higher Education.



Capture d'écran de la page www.timeshighereducation.co.uk/news/academics-meet--declare-support-universities-hit-conflict, réalisée le 17 juillet 2015 (détails).

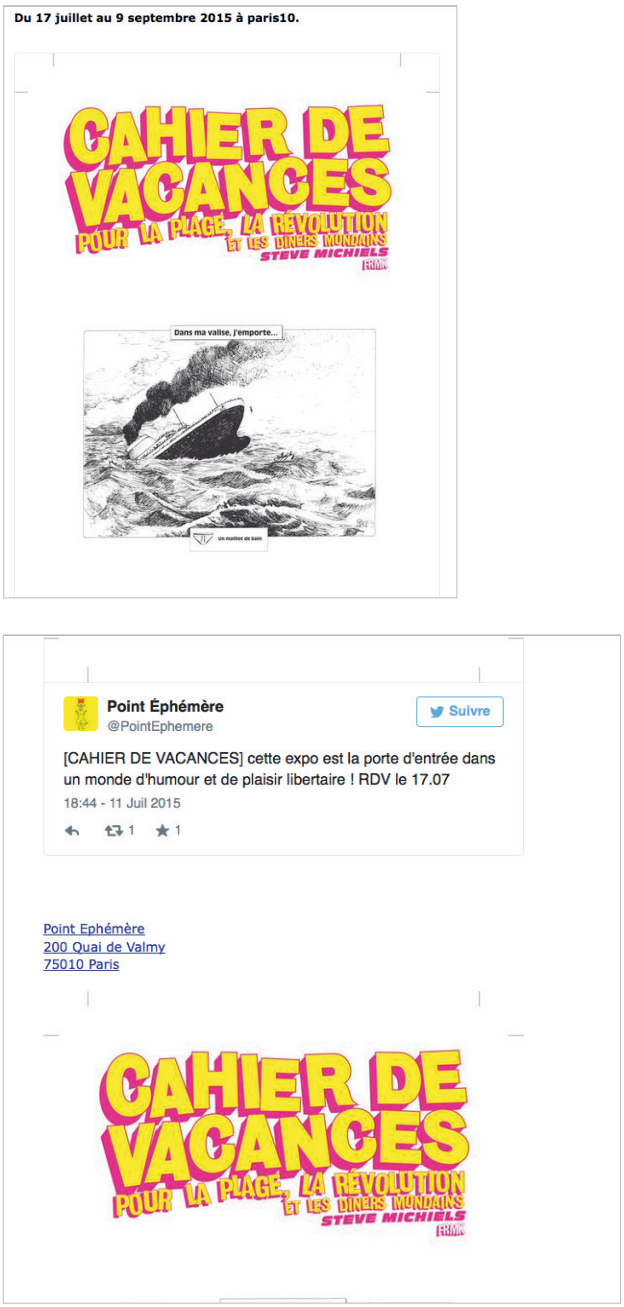
Annexe n° 74 Twitter, tweet ancré, site Big Browser.

Capture d'écran de la page bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx, réalisée le 16 juillet 2015 (détails).

Annexe n° 75 Twitter, tweet ancré, site Les Décodeurs.

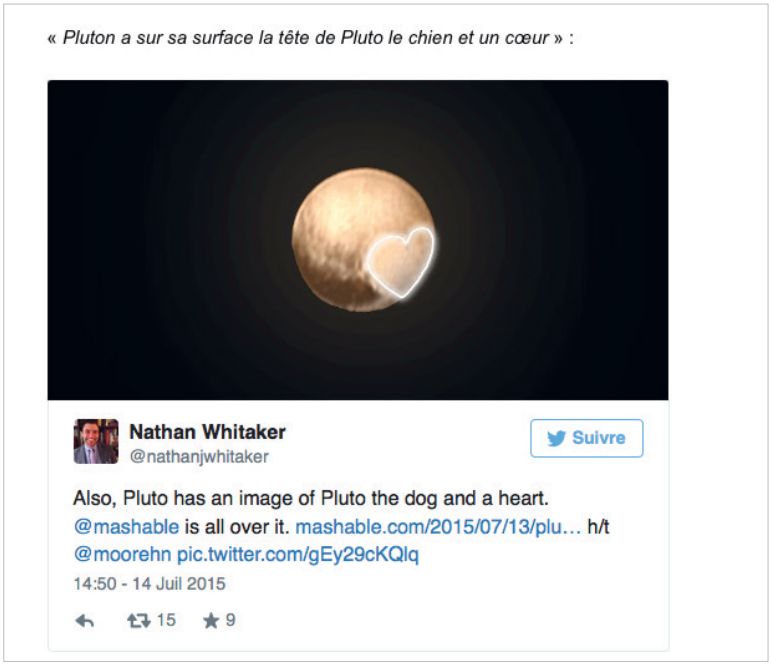
Capture d'écran de la page www.lemonde.fr/les-decodeurs/article/2015/07/16/ma-france-a-moi-le-mot-cle-qui-a-enflamme-twitter-le-14-juillet_4685117_4355770.html, réalisée le 17 juillet 2015 (détails).

Annexe n° 76 Twitter, tweet ancré, site Foxoo.



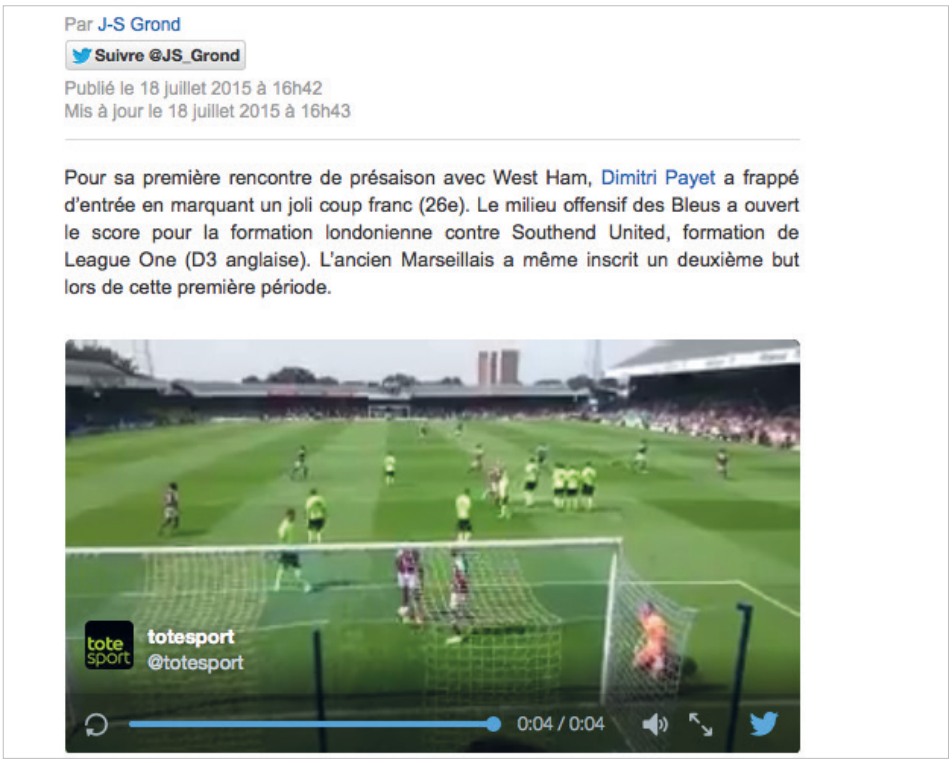
Captures d'écran de la page paris.foxoo.com/exposition/cahier-vacances-pour-plage,revolution-diners-mondains,paris10,nx1507111948306692.html, réalisées le 17 juillet 2015 (détails).

Annexe n° 77 Twitter, vidéo ancrée, site Big Browser.



Captures d'écran de la page bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx, réalisées le 16 juillet 2015 (détails).

Annexe n° 78 Twitter, vidéo ancrée, site Sports.fr.

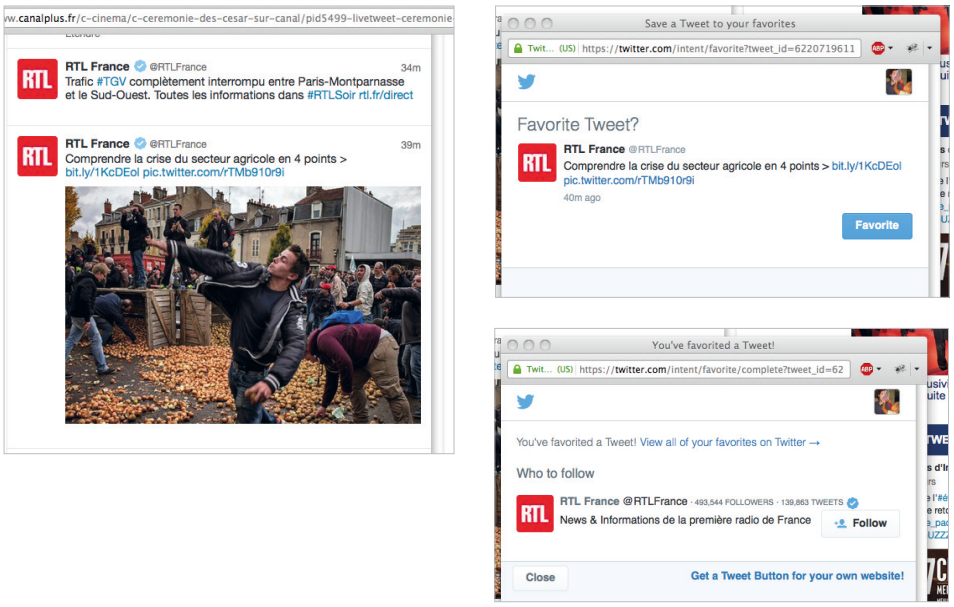


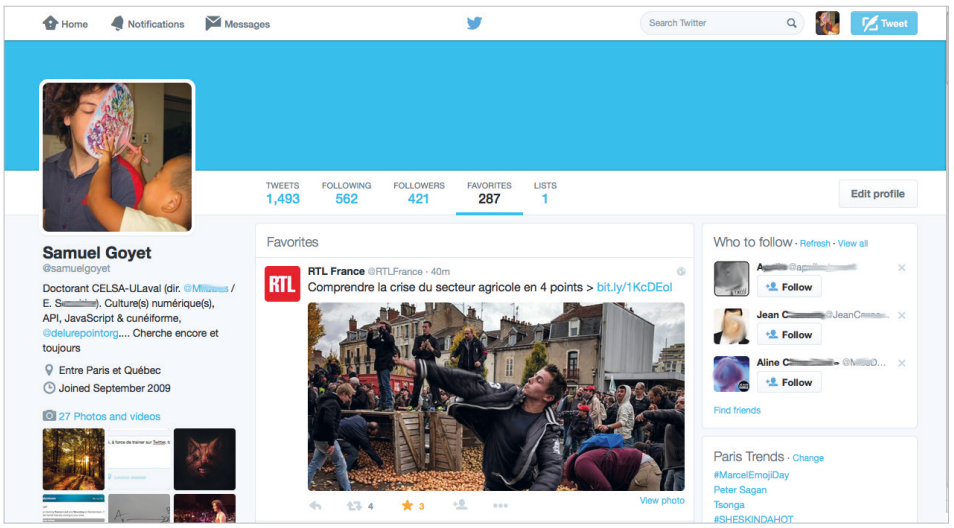
Capture d'écran de la page www.sports.fr/football/angleterre/articles/le-coup-franc-parfait-de-payet-video/1287018/, réalisée le 18 juillet 2015 (détails).

Annexe n° 79 Twitter, *intent* « *Favorite* », site Big Browser.

Captures d'écran des pages bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx et www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

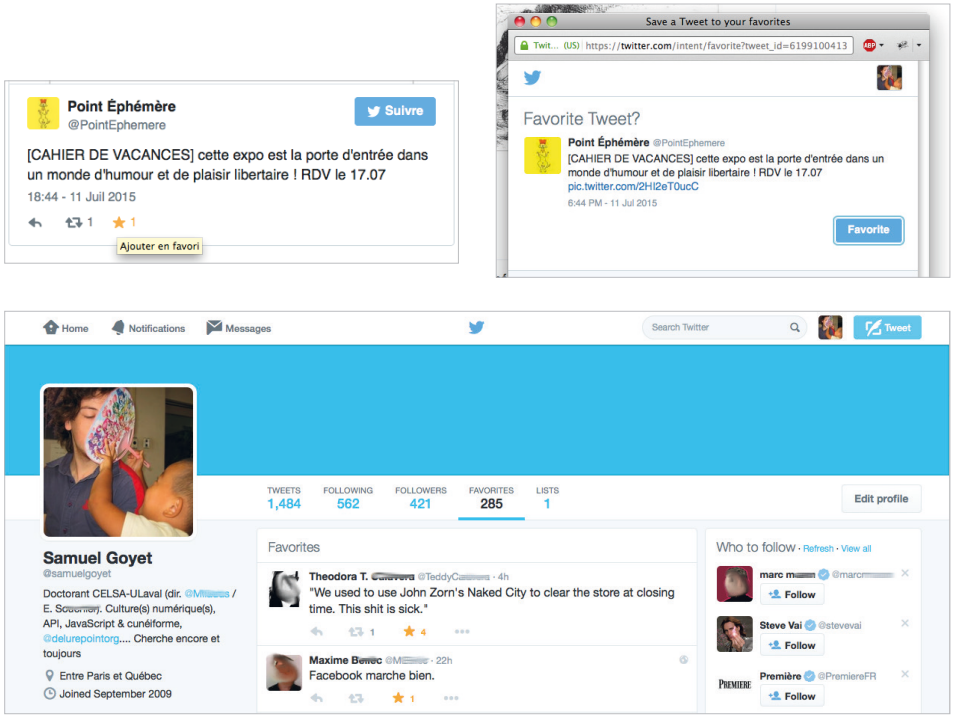
Annexe n° 80 Twitter, *intent* « *Favorite* », site Canal Plus.





Captures d'écran de la page www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/livetweet-ceremonie-cesar-2015.html et de la page www.twitter.com (profil personnel) réalisées le 17 juillet 2015 (détails).

Annexe n° 81 Twitter, *intent « Favorite »*, site Foxoo.

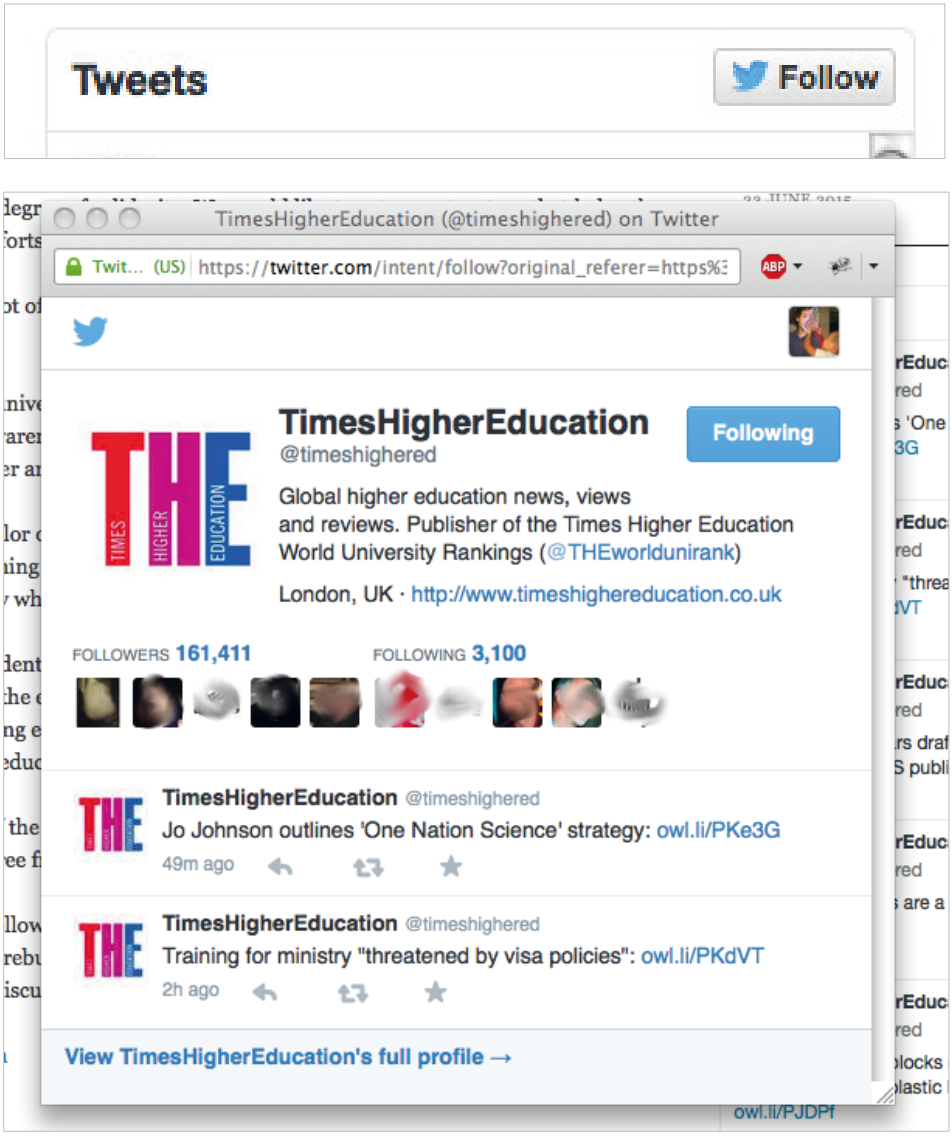


Captures d'écran des pages paris.foxoo.com/exposition/cahier-vacances-pour-plage,revolution-diners-mondains,paris10,nx1507111948306692.html et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 82 Twitter, *intent « Follow »*, site BNF.

Captures d'écran des pages www.bnf.fr/fr/acc-x.accueil.html et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 83 Twitter, *intent* « *Follow* », site Times Higher Education.



Captures d'écran des pages www.timeshighereducation.co.uk/news/academics-meet--declare-support-universities-hit-conflict et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 84 Twitter, *intent* « *Follow* », site Foxoo.

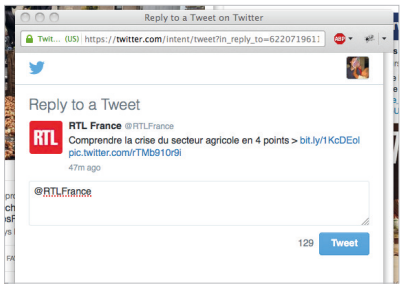
Captures d'écran des pages paris.foxoo.com/exposition/cahier-vacances-pour-plage,revolution-diners-mondains,paris10,nx1507111948306692.html et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

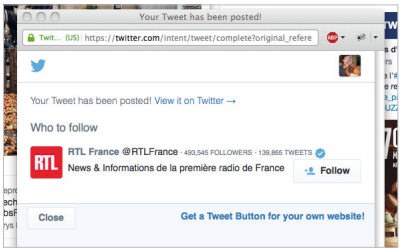
Annexe n° 85 Twitter, *intent* « *Reply* », site Big Browser.

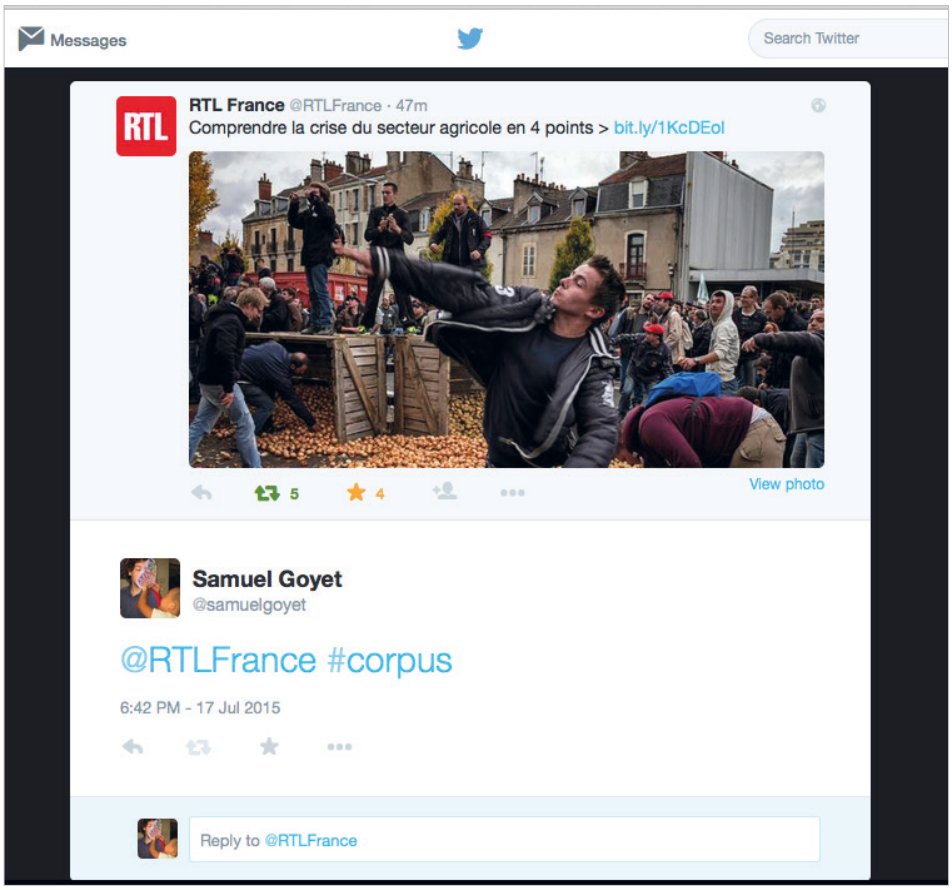
Captures d'écran des pages www.bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx/ et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 86 Twitter, *intent* « *Reply* », site Canal Plus.

Captures d'écran des pages www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/

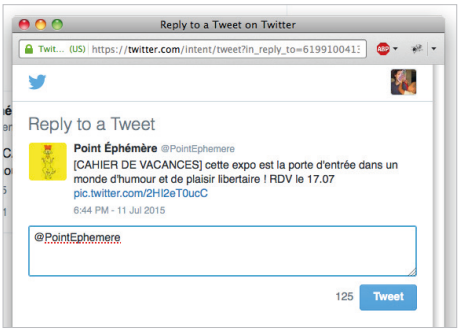






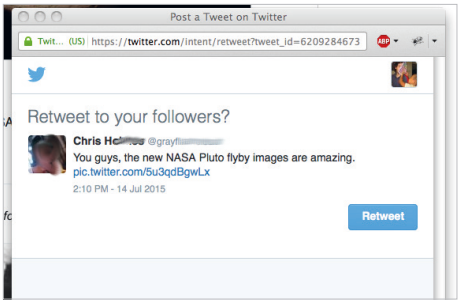
livetweet-ceremonie-cesar-2015.html et www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

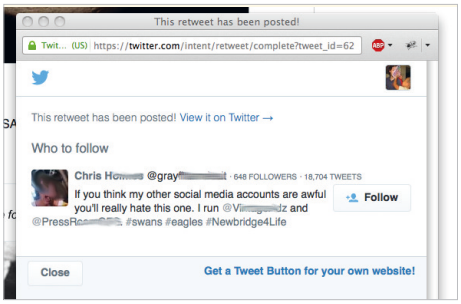
Annexe n° 87 Twitter, *intent* « *Reply* », site Foxoo.

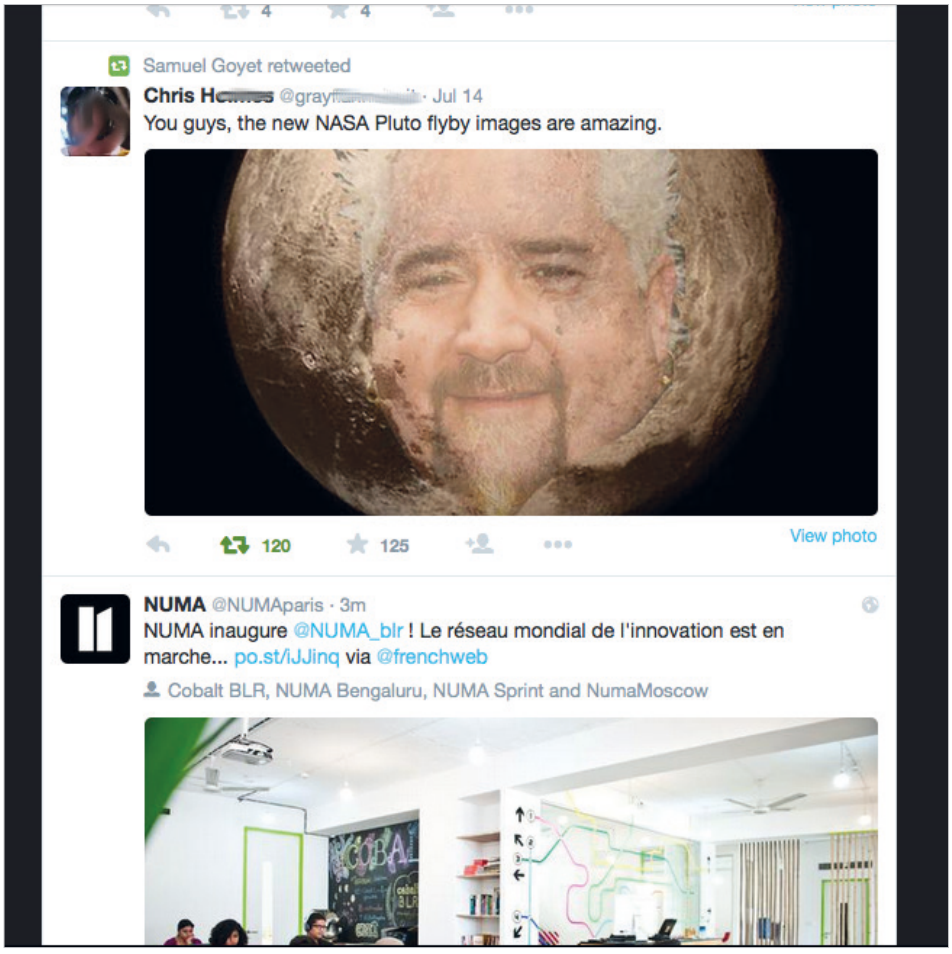


Captures d'écran des pages paris.foxoo.com/exposition/cahier-vacances-pour-plage,revolution-diners-mondains,paris10,nx1507111948306692.html et www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 88 Twitter, *intent* « Retweet », site Big Browser.

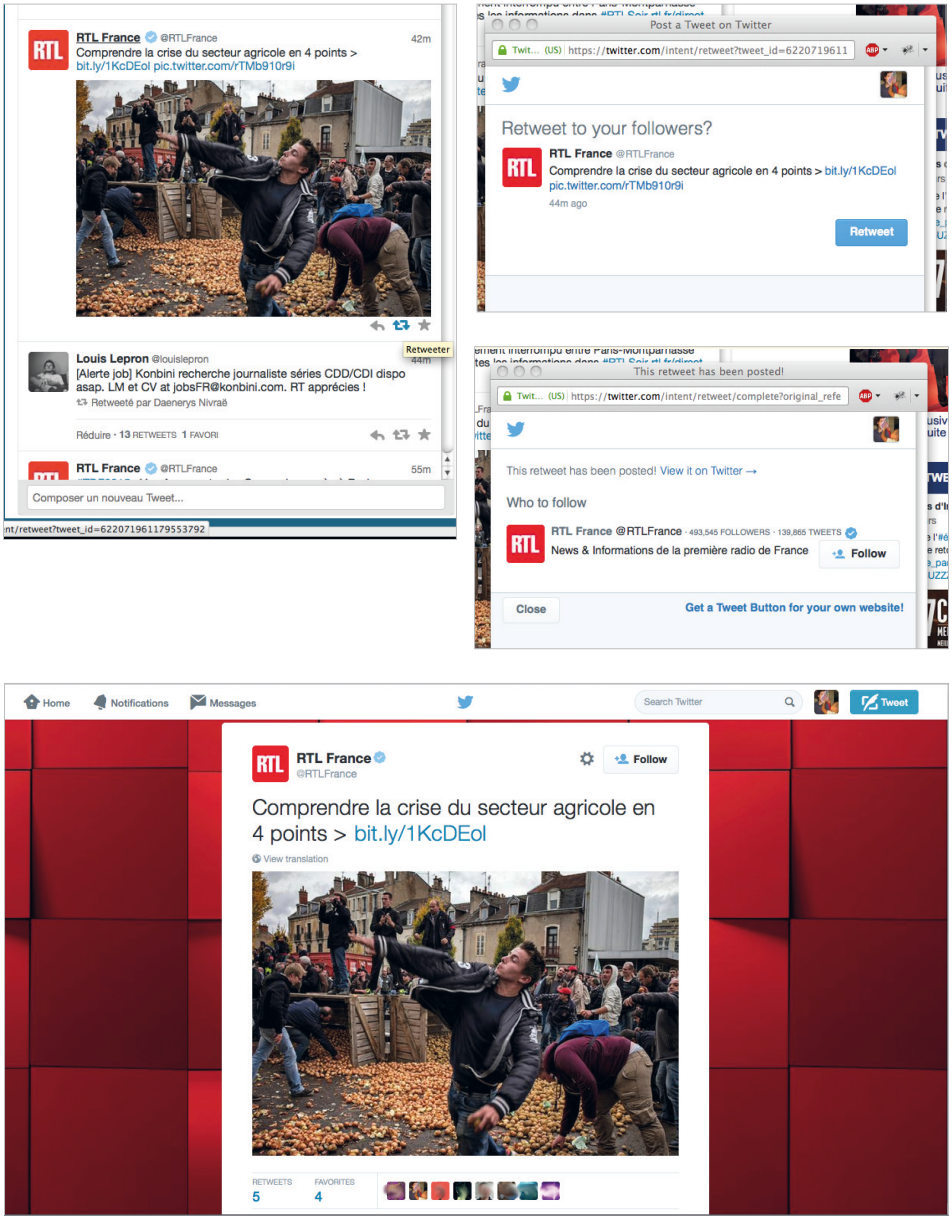






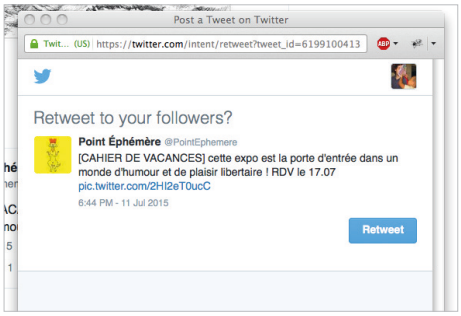
Captures d'écran des pages bigbrowser.blog.lemonde.fr/2015/07/14/pluton-plutot-pluto-ou-plutot-marx et www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 89 Twitter, *intent* « Retweet », site Canal Plus.



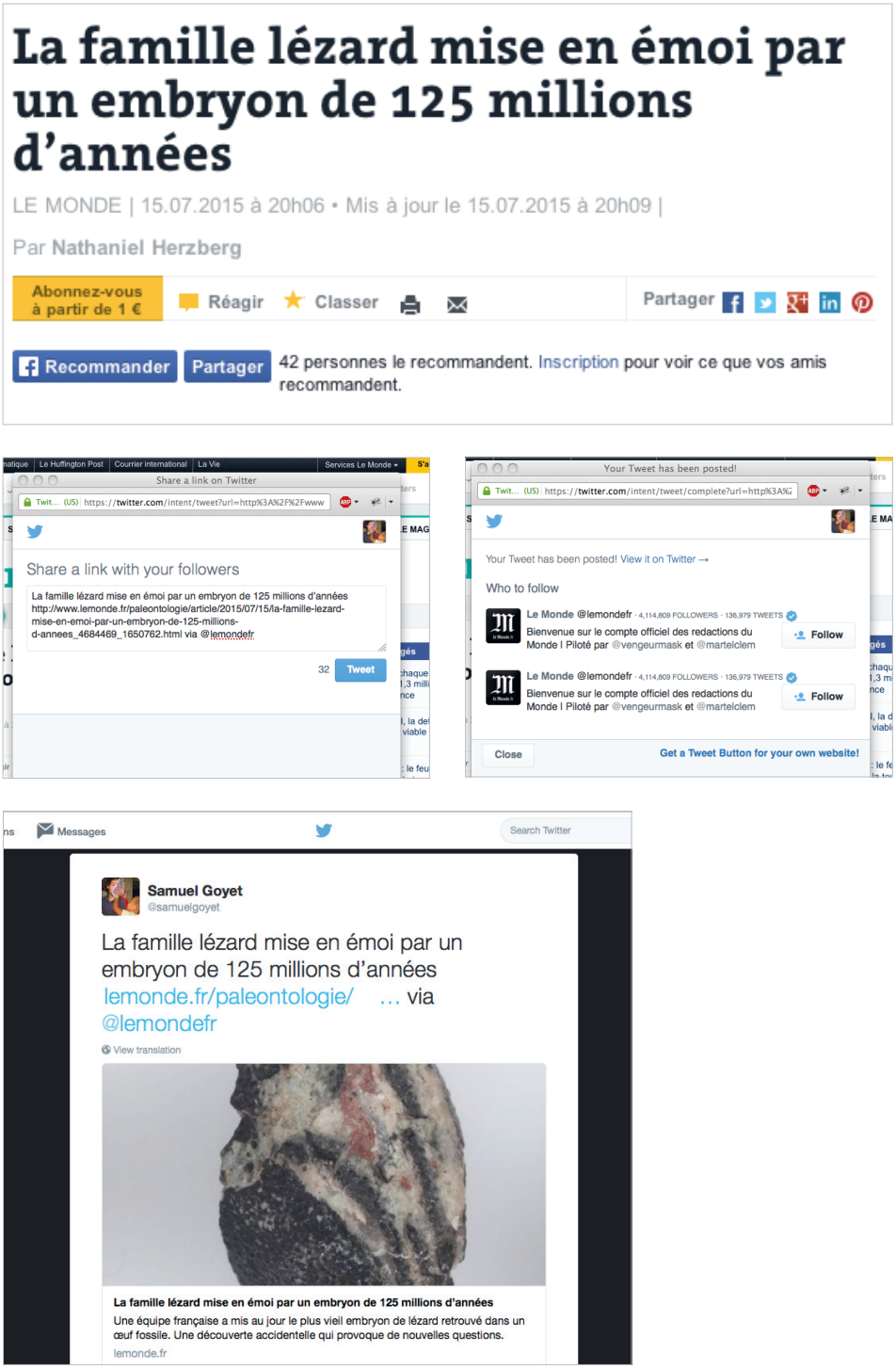
Captures d'écran des pages www.canalplus.fr/c/cinema/c/ceremonie-des-cesar-sur-canal/pid5499/livetweet-ceremonie-cesar-2015.html et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 90 Twitter, *intent* « Retweet », site Foxoo.



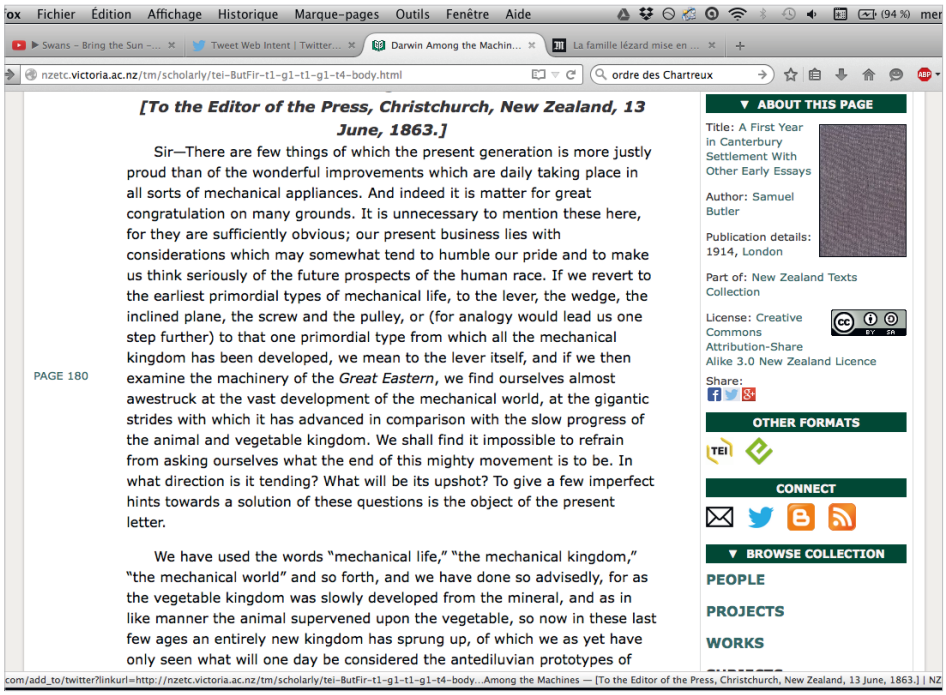
Captures d'écran des pages paris.foxoo.com/exposition/cahier-vacances-pour-plage,revolution-diners-mondains,paris10,nx1507111948306692.html et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 91 Twitter, *intent* « Tweet », site Le Monde.



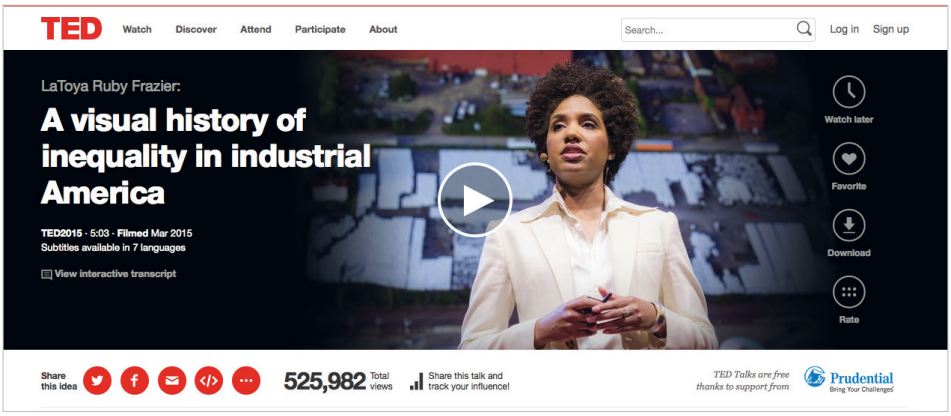
Captures d'écran des pages www.lemonde.fr/paleontologie/article/2015/07/15/la-famille-lezard-mise-en-emoi-par-un-embryon-de-125-millions-d-annees_4684469 et www.twitter.com (profil personnel), réalisées le 15 juillet 2015 (détails).

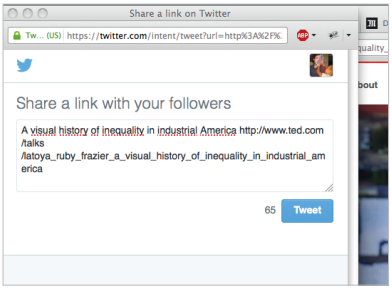
Annexe n° 92 Twitter, *intent* « Tweet », site Nzetc.



Captures d'écran des pages http://nzetc.victoria.ac.nz/tm/scholarly/tei-ButFir-t1-g1-t1-g1-t4-body.html et www.twitter.com (profil personnel), réalisées le 15 juillet 2015 (détails).

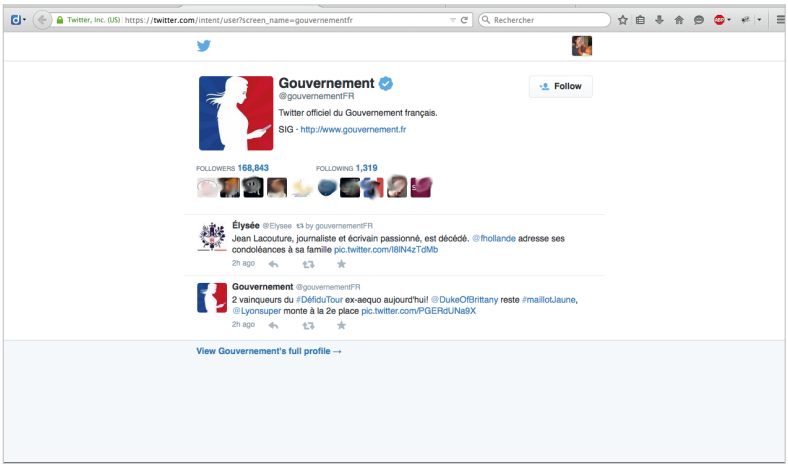
Annexe n° 93 Twitter, *intent* « Tweet », site TedTalks.





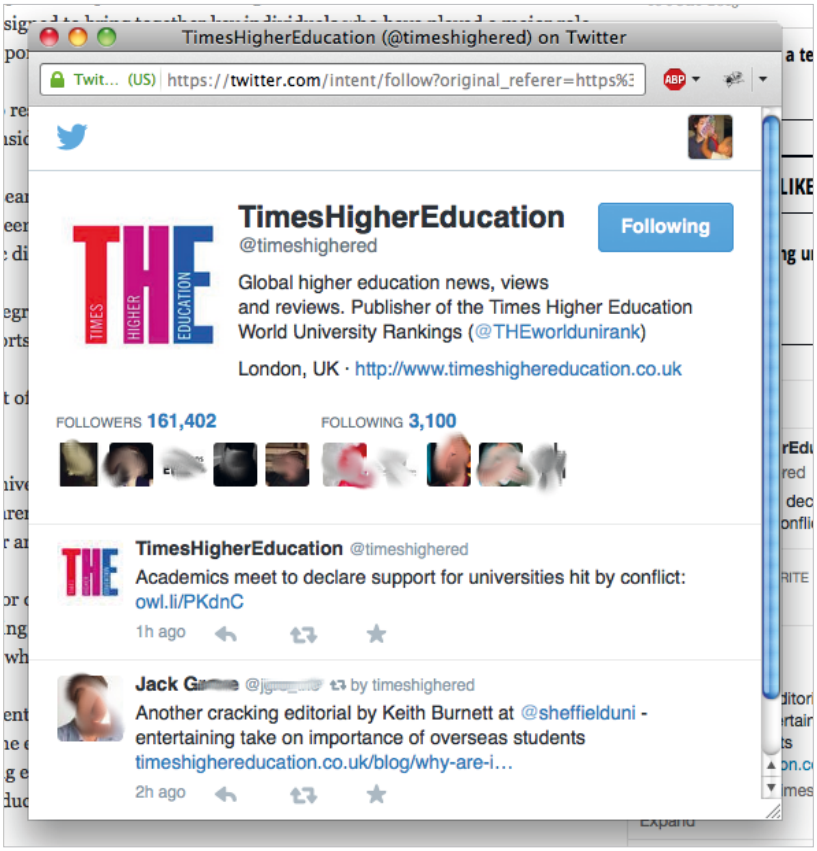
Captures d'écran des pages http://www.ted.com/talks et www.twitter.com (profil personnel), réalisées le 16 juillet 2015 (détails).

Annexe n° 94 Twitter, *intent « User »*, site Gouvernement.



Captures d'écran des pages http://www.gouvernement.fr et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

Annexe n° 95 Twitter, *intent « User »*, site Times Higher Education.



Captures d'écran des pages www.timeshighereducation.co.uk/news/academics-meet--declare-support-universities-hit-conflict et www.twitter.com (profil personnel), réalisées le 17 juillet 2015 (détails).

De briques et de blocs. La fonction éditoriale des interfaces de programmation (API) web : entre science combinatoire et industrie du texte.

Résumé

Boutons « J'aime », tweets ancrés… Toutes ces formes sont générées par des interfaces de programmation ou API, outils d'écriture informatique qui ont integré la chaîne de production des textes de réseau contemporains. Cette thèse interroge la fonction éditoriale des API, soit leur rôle dans la production, la standardisation et la circulation des « petites formes » des textes de réseau. Avec comme corpus les API de Facebook et de Twitter ainsi que les petites formes qu'elles permettent de produire, notre analyse techno-sémiotique s'articule en cinq chapitres. Le premier est une généalogie des API du point de vue de l'écriture combinatoire. Nous montrons que cette conception de l'écriture est un trait saillant de la programmation et de l'informatique. Le second chapitre interroge les imaginaires de l'écriture informatique, entre chiffre, combinatoire et méthode scientifique universelle. Le troisième chapitre est une analyse des conséquences sémiotiques de cet universalisme combinatoire, où nous montrons que les API proposent une conception du texte comme ensemble abstrait de blocs combinables. Abstraction du texte qui sert une « économie des passages », objet de notre quatrième chapitre, dans laquelle les API sont des lieux d'industrialisation d'une « pratique lettrée » : elles établissent des critères de lisibilité et de reproductibilité du texte. Parmi ces critères, nous notons une invisibilisation du rôle pourtant fondamental du calcul informatique. Nous proposons donc, dans un cinquième chapitre, des pistes pour développer une sémiotique qui prenne en compte le calcul comme mode d'expression propre aux médias numériques.

Mots-clés : Calcul ; numérique ; informatique ; combinatoire ; éditorialisation ; interfaces de programmation ; API ; écrits d'écran ; sémiotique ; écriture ; industrialisation de l'écriture ; techno-sémiotique ; code informatique.

Of Bits and Blocks. The publishing function of web Application Programming Interfaces (APIs): from a combinatorial science to an industry of text-processing.

Summary

« Like » buttons, embedded tweets… All of these visual forms are produced by Application Programming Interfaces (APIs). APIs are digital writing tools which have become part of the publishing process of contemporary web pages. This thesis aims at understanding the « publishing function » of APIs : their role in the production, standardization and circulation of the « little forms » of online texts. Focused on Facebook's and Twitter's APIs, our work is divided into five chapters. The first one is a genealogy of the APIs, starting from their combinatorial aspect, a conception of writing which trace back to early programming and the invention of computer science. The second chapter is an inquiry about the imaginaries of calculus as a kind of writing, torn between the imaginary of numbers, of combinatorics and the search for a universal scientific method. The third chapter is a study of the semiotic consequences of this combinatorial universalism. We show how APIs are based on an idea of text as an abstract, modular object. This abstraction of the text is beneficial to an « economy of passages ». In this economy where circulation produce value, APIs are a place of « literate practices » (chapter four). They establish visual standards for the readability, production and circulation of online texts. Among these standards, there's a systematic invizibilisation of the action of machines, although calculus is a necessary part of the production of digital texts. Therefore, in the fifth chapter, we give some epistemological elements towards non-anthropocentric semiotics, meaning : semiotics which would take into account computational machines as a part of the utterance of digital texts.

Keywords : Calculus ; digital studies ; computing ; combinatorics ; editorialization ; publishing process ; literary practices ; coding ; written screens ; API ; programming ; semiotics ; writing ; industrialization of writing.

Thèse en **Sciences de l'Information et de la Communication,** préparée au sein du **GRIPIC**
(Groupe de Recherche Interdisciplinaire sur les Processus d'Information et de Communication).

CELSA, UNIVERSITÉ PARIS-SORBONNE — 77 rue de Villiers – 92200 Neuilly-sur-Seine.
École doctorale nº 5 « Concepts et langages » — Maison de la Recherche – 28 rue Serpente, 75006 Paris.