Analyse de documents et du comportement des utilisateurs pour améliorer l'accès à l'information

Axel Jean-Caurant

UNIVERSITÉ DE LA ROCHELLE

ÉCOLE DOCTORALE EUCLIDE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de La Rochelle
Mention : Informatique et Applications

Présentée et soutenue par

Axel Jean-Caurant

Analyse de documents et du comportement des utilisateurs pour améliorer l'accès à l'information

Thèse dirigée par Vincent Courboulay
et Jean-Christophe Burie

préparée au laboratoire L3i

soutenue le 8 octobre 2018

Jury :

Rapporteurs :	Véronique Eglin	-	LIRIS (Lyon)
	Aris Xanthos	-	UNIL (Lausanne)
Examinateurs :	Fatiha Idmhand	-	CRLA (Poitiers)
	Antoine Doucet	-	L3i (La Rochelle)
	Charles Illouz	-	CRHIA (La Rochelle)

Remerciements

Je tiens avant tout à remercier mes deux directeurs, Vincent Courboulay et Jean-Christophe Burie, de m'avoir donné l'opportunité de réaliser cette thèse. Merci également pour les différentes relectures de ce manuscrit qui m'ont permis de le finaliser. Je remercie également Véronique Eglin et Aris Xanthos qui ont accepté le rôle de rapporteur pour cette thèse.

Merci ensuite à Cyrille, avec qui j'ai beaucoup échangé ces quatre dernières années. D'un point de vue professionnel, mes recherches n'aurait sans doute pas pris le même chemin sans les nombreuses discussions que nous avons eues. D'un point de vue personnel, les nombreuses pauses café passées à râler ensemble sur des sujets aussi divers que variés m'amusent toujours autant.

Merci à Nouredine avec qui j'ai rédigé un papier de recherche et qui a été d'une aide très précieuse tout au long de ma thèse. Aussi bien pour la rédaction que pour les discussions scientifiques de haut-vol.

Je tiens aussi à remercier les personnes que j'ai croisé durant ces dernières années au laboratoire, aussi bien les différents chercheurs et doctorants que le personnel administratif.

Je souhaite aussi remercier chaleureusement mes parents, Christophe, Isabelle, Théo, Sandra, Eliott et tous ceux que je pourrais oublier.

Finalement, je remercie Chloé qui m'a supporté et encouragé dans ce projet, comme elle le fait dans la vie.

Résumé

L'augmentation constante du nombre de documents disponibles et des moyens d'accès transforme les pratiques de recherche d'information. Depuis quelques années, de plus en plus de plateformes de recherche d'information à destination des chercheurs ou du grand public font leur apparition sur la toile. Ce flot d'information est bien évidemment une opportunité pour les utilisateurs mais ils sont maintenant confrontés à de nouveaux problèmes. Auparavant, la principale problématique des chercheurs était de savoir si une information existait. Aujourd'hui, il est plutôt question de savoir comment accéder à une information pertinente. Pour résoudre ce problème, deux leviers d'action seront étudiés dans cette thèse.

Nous pensons qu'il est avant tout important d'identifier l'usage qui est fait des principaux moyens d'accès à l'information. Être capable d'interpréter le comportement des utilisateurs est une étape nécessaire pour d'abord identifier ce que ces derniers comprennent des systèmes de recherche, et ensuite ce qui doit être approfondi. En effet, la plupart de ces systèmes agissent comme des boîtes noires qui masquent les différents processus sous-jacents. Si ces mécanismes n'ont pas besoin d'être entièrement maitrisés par les utilisateurs, ils ont cependant un impact majeur qui doit être pris en compte dans l'exploitation des résultats. Pourquoi le moteur de recherche me renvoie-t-il ces résultats ? Pourquoi ce document est-il plus pertinent qu'un autre ? Ces questions apparemment banales sont pourtant essentielles à une recherche d'information critique. Nous pensons que les utilisateurs ont le droit et le devoir de s'interroger sur la pertinence des outils informatiques mis à leur disposition. Pour les aider dans cette tâche, nous avons développé une plateforme de recherche d'information en ligne à double usage. Elle peut tout d'abord être utilisée pour l'observation et la compréhension du comportement des utilisateurs. De plus, elle peut aussi être utilisée comme support pédagogique, pour mettre en évidence les différents biais de recherche auxquels les utilisateurs sont confrontés.

Dans le même temps, ces outils doivent être améliorés. Nous prenons dans cette thèse l'exemple de la qualité des documents qui a un impact certain sur leur accessibilité. La quantité de documents disponibles ne cessant d'augmenter, les opérateurs humains sont de moins en moins capables de les corriger manuellement et de s'assurer de leur qualité. Il est donc nécessaire de mettre en place de nouvelles stratégies pour améliorer le fonctionnement des systèmes de recherche. Nous proposons dans cette thèse une méthode pour automatiquement identifier et corriger certaines erreurs générées par les processus automatiques d'extraction d'information (en particulier l'OCR).

Mots clés : Bibliothèques numériques, Observation des utilisateurs, Accessibilité, Correction Post-OCR

Abstract

The constant increase of available documents and tools to access them has led to a change of research practices. For a few years now, more and more information retrieval platforms are made available online to the scientific community or the public. This data deluge is a great opportunity for users seeking information. However, it comes with new problems and new challenges to overcome. Formerly, the main issue for researchers was to identify if a particular resource existed. Today, the challenge is more about finding how to access pertinent information. We have identified two distinct levers to limit the impact of this new search paradigm.

First, we believe that it is necessary to analyze how the different search platforms are used. To be able to understand and read into users behavior is a necessary step to comprehend what users understand, and to identify what they need to get an in-depth understanding of the operation of such platforms. Indeed, most systems act as black boxes which conceal the underlying transformations applied on data. Users do not need to understand in details how those algorithms work. However, because those algorithms have a major impact on the accessibility of information, and need to be taken into account during the exploitation of search results. Why is the search engine returning those particular results ? Why is this document more pertinent than another ? Such seemingly naive questions are nonetheless essential to undertake an analytical approach of the information search and retrieval task. We think that users have a right and a duty to question themselves about the relevance of such and such tool at their disposal. To help them cope with these issues, we developped a dual-use information search platform. On the one hand, it can be used to observe and understad user behavior. On the other hand, it can be used as a pedagogical medium to highlight research biases users can be exposed to.

Ath the same time, we believe that the tools themselves must be improved. In the second part of this thesis, we study the impact that the quality of documents can have on their accessibility. Because of the increase of documents available online, human operators are less and less able to insure their quality. Thus, there is a need to set up new strategies to improve the way search platform operate and process documents. We propose a new method to automatically identify and correct errors generated by information extraction process such as OCR.

Keywords : Digital libraries, User observation, Accessibility, Post-OCR correction

Table des figures

1.1 Pyramide conceptuelle du savoir 2
1.2 Interactions entre utilisateur, système et documents 4

2.1 Aperçu d'un tiroir de classement du Mondaneum 11
2.2 Principe de fonctionnement du PageRank 13
2.3 Transformation d'un texte avant son indexation 15
2.4 Aperçu des métadonnées associées à la page Wikipedia de la ville de Paris 19
2.5 Agent conversationnel utilisant une base de connaissance 19
2.6 Exemple de métadonnées sous la forme d'un graphe RDF 23
2.7 Interface d'interrogation de la base de données du TPIY 27
2.8 Expositions disponibles sur Europeana 29
2.9 Page d'une ressource de la bibliothèque numérique Isidore 30
2.10 Interface de recherche de Gallica 31

3.1 Architecture logicielle utilisée pour le développement de notre plate-forme d'étude 38
3.2 Organisation des données dans le modèle conceptuel PCDM 40
3.3 Organisation des données dans le modèle conceptuel Hydra Works . 41
3.4 Résultats affichés par le système après une requête 42
3.5 Page de visualisation d'une ressource 43
3.6 Schéma d'une boîte noire 54
3.7 Interface de modifications des paramètres du moteur de recherche . . 56
3.8 Courbe de pertinence dans la page des résultats 57
3.9 Interface de visualisation des indicateurs personnels 58
3.10 Interface de visualisation des indicateurs collectifs 58

4.1 Exemple d'un arbre syntaxique 69
4.2 Exemple d'entrée du thésaurus WordNet 70

4.3 Aperçu des catégories Wikipedia 71
4.4 Aperçu d'une ontologie sur les automobiles 71
4.5 Importance des personnages dans la pièce *Hamlet* de Shakespeare. . 74
4.6 Outil de visualisation des n-grammes dans une collection de documents numérisés 78
4.7 Construction des exemples d'apprentissage 80
4.8 Réseau de neurones utilisé pour l'architecture CBOW avec un seul mot de contexte. 81
4.9 Aspect du réseau de neurones lorsque plusieurs mots de contexte sont pris en compte. 83
4.10 Aspect du réseau de neurones lorsque l'architecture Skip-Gram est utilisée. 84
4.11 Aspect du réseau de neurones lorsque l'architecture PV-DM est utilisée. 86
4.12 Aspect du réseau de neurones lorsque l'architecture PV-DBOW est utilisée. 86

5.1 Exemple de création d'un document bruité artificiellement 93
5.2 Alignement du texte d'un document original et de sa version bruitée 94
5.3 Versions bruitées de quelques entités nommées identifiées dans le corpus 94
5.4 Chaîne de traitement pour corriger les entités nommées dans un corpus bruité 97
5.5 Graphe des entités nommées similaires. 102
5.6 Exemple d'une courbe de Lorenz 112
5.7 Comparaison de la distribution de l'accessibilité des documents après un processus OCR 113
5.8 Comparaison de la distribution de l'accessibilité des documents ayant subi un processus OCR après application de notre méthode 114

A.1 Schéma d'un neurone biologique 122
A.2 Schéma d'un neurone formel. 123
A.3 Exemple d'un réseau de neurones avec trois couches. 124
A.4 Graphiques et équations de trois fonctions d'activation souvent utilisées dans divers réseaux de neurones. 125

Liste des tableaux

2.1 Schéma fictif de métadonnées utilisé dans le graphe 2.6 23
2.2 Fonctionnalités de différentes bibliothèques numériques 32

3.1 Valeurs des indicateurs pour les quatre tâches de recherche d'information 50
3.2 Différences statistiques entre deux types de tâches de recherche d'information 51

5.1 Impact de l'OCR sur la détection des entités nommées. 95
5.2 Èvaluation par différentes mesures du clustering des entités nommées et de la qualité de la correction du corpus bruité 107

A.1 Récapitulatif des similarités entre neurone biologique et neurone formel. 123

B.1 Cas possibles pour l'opérateur lex_{min}. 133
B.3 Valeurs de B selon les cas 1 et 1-1. 134
B.5 Valeurs de B selon les cas 2 et 2-1. 135
B.7 Valeurs de B selon les cas 2 et 2-2. 135
B.9 Comparaison entre A et B. 136
B.11 Comparaison de A et B dans le cas (1). 137
B.13 Comparaison de A et B pour le cas (2). 137
B.15 Cas possibles pour l'opérateur lex_{max}. 137
B.17 Valeurs de B selon les cas 1 et 1-1. 138
B.19 Valeurs de B selon les cas 2 et 2-1. 139
B.21 Valeurs de B selon les cas 2 et 2-2. 139
B.23 Comparaison de A et B. 140
B.25 Comparaison de A et B dans le cas (1). 141
B.27 Comparaison de A et B dans le cas (2). 141

Table des matières

Remerciements i

Résumé iii

Abstract v

Table des figures vii

Liste des tableaux ix

Table des matières xi

1 Introduction générale 1

I Bibliothèques Numériques et Utilisateurs 7

2 De nombreux moyens d'accès à l'information 9

2.1 Introduction . . . 9

2.2 Historique des pratiques de recherche . . . 10

2.3 Fonctionnement d'un moteur de recherche . . . 12

2.3.1 Prétraitement des données . . . 13

2.3.2 Indexation et évaluation des requêtes . . . 14

2.3.3 Fonctions additionnelles . . . 17

2.4 Importance des métadonnées . . . 18

2.4.1 Différents types de métadonnées . . . 20

2.4.1.1 Métadonnées descriptives . . . 20

2.4.1.2 Métadonnées structurelles . . . 20

2.4.1.3 Métadonnées administratives . . . 21

2.4.2 Représentation des métadonnées . . . 21

2.5 Différentes plateformes de recherche d'information . . . 22

2.5.1 Moteur de recherche Web . . . 25

2.5.2 Bases de données en ligne . . . 26

2.5.3 Bibliothèques numériques . . . 26

2.5.3.1 Europeana . . . 28
2.5.3.2 Isidore . . . 29
2.5.3.3 Gallica . . . 31
2.6 Conclusion . . . 32

3 Réduire le fossé entre les utilisateurs et les données 35
3.1 Introduction . . . 35
3.2 Plateforme d'étude . . . 37
3.2.1 Technologies utilisées . . . 37
3.2.2 Modélisation des données . . . 39
3.2.3 Présentation du fonctionnement . . . 42
3.3 Ce que le système comprend des utilisateurs . . . 44
3.3.1 Modèles de comportement pour la recherche d'information . . . 44
3.3.2 Observation des utilisateurs . . . 45
3.3.2.1 Actions à observer . . . 46
3.3.2.2 Implémentation . . . 47
3.3.2.3 Indicateurs comportementaux . . . 48
3.3.3 Expérimentations . . . 48
3.3.3.1 Contexte et contraintes de l'étude . . . 49
3.3.3.2 Résultats . . . 49
3.4 Ce que les utilisateurs comprennent du système . . . 53
3.4.1 Biais méthodologiques . . . 53
3.4.2 Fonctionnement et implémentation . . . 55
3.5 Conclusion . . . 59

II Données et Algorithmes 61

4 Analyse de documents et extraction de connaissances 63
4.1 Introduction . . . 64
4.2 Différents types d'hétérogénéités . . . 64
4.2.1 Hétérogénéité des médiums . . . 64
4.2.2 Hétérogénéité des types de fichiers . . . 65
4.2.3 Hétérogénéité des contenus . . . 65
4.2.4 Hétérogénéité de la qualité . . . 66
4.3 Analyse et enrichissement . . . 66

4.3.1 Extraction de texte . . . 66
4.3.2 Compréhension de la structure . . . 67
4.3.3 Compréhension sémantique . . . 68
4.3.3.1 Information non structurée et connaissance . . . 69
4.3.3.2 Entités nommées . . . 72
4.3.4 Lecture distante . . . 73
4.3.4.1 Analyse de réseaux . . . 73
4.3.4.2 Modèles thématiques . . . 75
4.4 Méthodes de modélisation et d'analyse du langage . . . 76
4.4.1 Représentation vectorielle des éléments textuels . . . 76
4.4.1.1 Sacs de mots . . . 76
4.4.1.2 N-grammes . . . 77
4.4.2 Word embedding . . . 79
4.4.2.1 Continuous Bag of Words . . . 80
4.4.2.2 Skip Gram . . . 83
4.4.2.3 Optimisations algorithmiques . . . 84
4.4.2.4 Améliorations . . . 85
4.5 Conclusion . . . 87

5 Correction automatique des entités nommées 89
5.1 Introduction . . . 90
5.2 Travaux précédents . . . 91
5.3 Évaluation préalable . . . 92
5.3.1 Création d'une vérité terrain . . . 92
5.3.2 Entités nommées et bruit OCR . . . 94
5.4 Présentation de la méthode . . . 96
5.4.1 Entraînement du modèle contextuel de langage . . . 96
5.4.2 Définition d'une bi-similarité lexicographique . . . 98
5.4.2.1 Similarité contextuelle . . . 98
5.4.2.2 Similarité d'édition . . . 99
5.4.2.3 Combinaison des deux similarités . . . 99
5.4.3 Construction du graphe des entités nommées similaires . . . 101
5.4.4 Clustering . . . 102
5.5 Évaluation . . . 103
5.5.1 Évaluation du clustering . . . 103

5.5.1.1 Mesure MUC . . . 103
5.5.1.2 Mesure B^3 . . . 104
5.5.1.3 Mesure CEAF . . . 105
5.5.1.4 Mesure combinée . . . 106
5.5.2 Évaluation de la correction . . . 108
5.5.2.1 Sélection d'un représentant . . . 108
5.5.2.2 Correction du corpus . . . 108
5.5.3 Évaluation de l'accessibilité des documents . . . 109
5.5.3.1 Mesure d'accessibilité . . . 110
5.5.3.2 Résultats et discussion . . . 112
5.6 Conclusion . . . 115

6 Conclusion générale **117**

A Fonctionnement général d'un réseau de neurones **121**
A.1 Introduction . . . 121
A.1.1 Présentation générale . . . 121
A.1.2 Analogie entre neurone biologique et neurone formel . . . 122
A.2 Architecture en réseau et apprentissage . . . 123
A.2.1 Pré-requis . . . 123
A.2.1.1 Fonctions d'activation . . . 123
A.2.1.2 Entraînement et fonctions de coût . . . 125
A.2.2 Algorithmes . . . 126
A.2.2.1 Prédire un résultat : Forward Propagation . . . 127
A.2.2.2 Mettre à jour les poids : Backpropagation . . . 128
A.3 Discussion . . . 130
A.4 Conclusion . . . 131

B Preuves des propriétés des opérateurs lexicographiques **133**
B.1 Propriétés des fonctions lex_{min} et lex_{max} . . . 133
B.2 Opérateurs de comparaison $\preceq_{lex}$ et $\succeq_{lex}$ . . . 141

Bibliographie **145**

Chapitre 1

Introduction générale

Motivations

Depuis toujours, les civilisations humaines ont produit, stocké et échangé des données, des informations pour transmettre des connaissances. Aujourd'hui, la majorité des données produites par l'humanité sont numériques, stockées dans des ordinateurs, et sont échangées sur des réseaux de télécommunication comme Internet. L'utilisation de l'outil informatique ces dernières années a contribué à augmenter considérablement la quantité de données produites et disponibles. En effet, selon l'étude [Gantz 2011] fournie par l'IDC (International Data Corporation), la taille de "l'Univers digital", c'est-à-dire de l'ensemble des données numériques du monde, augmente de manière exponentielle depuis ces 20 dernières années. En 2010, le seuil symbolique du zetta-octet [1] a été dépassé. Un an plus tard, en 2011, la taille de "l'Univers digital" a approché les 2 zetta-octets. Aujourd'hui, cette progression ne ralentit pas et l'IDC prévoit qu'en 2020, 40 zetta-octets de données seront produites [2].

Ces chiffres sont impressionnants, mais ils représentent uniquement les données brutes. En effet, selon [Brooking 1999] une donnée est un fait, une image, un nombre ou un mot, présenté sans contexte particulier. L'analyse de ces données et leur mise en relation dans un contexte comparable permet de générer de l'information. Cette notion d'information est donc de plus haut niveau que la donnée, qui est à la base de la hiérarchie conceptuelle du savoir (voir figure 1.1) définie dans [Rowley 2007]. C'est sur ces données que s'élaborent les notions d'information et de connaissance. Selon l'interprétation et la représentation mentale que l'on a d'une information, la connaissance que l'on en tire peut être bien différente. Dans [Tsuchiya 1993], l'auteur exprime sa conception de la connaissance ainsi : "l'information ne devient connaissance que lorsqu'elle est comprise par le schéma d'interprétation du receveur qui lui donne un sens". Une information incompatible avec ce schéma n'est généralement pas prise en compte. De plus, selon [Penalva 2002], la connaissance repose sur un engagement, une intention. Autrement dit, elle correspond à ce que l'on veut faire de l'information.

Par conséquent, il devient nécessaire d'adapter nos sytèmes d'information pour gérer ce déluge de données. Si le stockage n'est plus vraiment un problème aujourd'hui, le vrai défi est l'exploitation de ces données pour en extraire de l'information

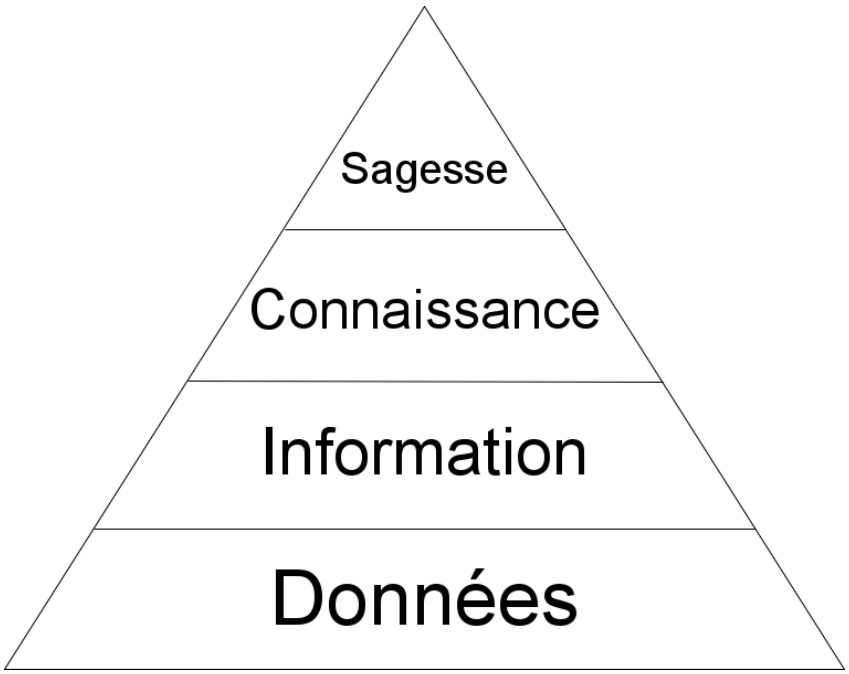
1. Le zetta-octet (Zo) correspond à 1000 milliards de giga-octets (Go). À des fins de comparaison, aujourd'hui, la plupart des disques durs sont capables de stocker aux alentours de 1000 Go.

2. https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

et de la connaissance. En effet, l'existence et la disponibilité des données ne suffit pas à leur donner de la valeur. La principale contrainte est d'être capable d'accéder aux données qui nous intéressent. Le même problème a été décrit pour la mémoire humaine dans [Tulving 1966]. Qui n'a jamais fait l'expérience d'un mot qui reste sur "le bout de la langue" sans pouvoir sortir ? Si le mot est ainsi **disponible** en mémoire, il n'est pas pour autant **accessible**. Cette dichotomie s'applique aussi au domaine des données et des documents. Le problème y est encore plus complexe car non seulement la quantité de données ne cesse de s'accroître, mais les moyens d'accès à ces données sont eux aussi de plus en plus nombreux. Tous ces éléments entrainent une forme de surcharge cognitive qui rend difficile la tâche de recherche d'information.

Si ces problèmes quantitatifs peuvent nuire à l'accessibilité de l'information, ce ne sont pas les seuls responsables. En effet, il faut aussi s'inquiéter de la qualité des données disponibles. Par exemple, lors d'une campagne de numérisation, il n'est pas rare que les documents (manuscrits, enregistrements audio, *etc.*) soient manuellement transcrits par des opérateurs humains. Si la qualité de la transcription est assurée, cela demande toutefois un travail considérable (surtout de nos jours avec le nombre croissant des campagnes de numérisation). Ainsi, il n'est pas rare d'auto-

FIGURE 1.1 – Pyramide conceptuelle du savoir



Dans cette représentation, les données sont des faits, présentés sans contexte particulier. L'analyse de ces données et leur mise en relation dans un contexte commun permet d'en extraire de l'information. On parle de connaissance quand cette information est interprétée et qu'on lui donne du sens. La sagesse est caractérisée par l'utilisation que l'on fait de nos connaissances.

matiser ces traitements par manque de temps ou de ressource. Or, les résultats des algorithmes d'extraction (OCR, ICR, SpeechToText, *etc.*) sont très dépendants de la qualité des sources originales. Ces algorithmes sont susceptibles de générer du bruit informationnel lorsqu'ils font des erreurs de reconnaissance. Le principal problème des différentes plateformes qui exploitent les résultats de ces algorithmes est qu'elles placent au même niveau les données de qualité et les données bruitées. Cela a un impact très fort sur les pratiques des utilisateurs, surtout s'ils ne sont pas conscients des processus de dématérialisation et d'indexation sous-jacents. Par exemple, un utilisateur s'intéressant à un mot clé particulier ne devrait pas conclure, après une recherche infructueuse, qu'aucun document ne contenait ce mot clé. Il est tout à fait possible que ce mot clé particulier ait été mal reconnu par les algorithmes d'extraction d'information. Lorsque les données sont intégrées au système sans contrôle, celui-ci ne peut discriminer les erreurs. Ainsi, même si ce mot est présent dans un document, le système d'information en a une mauvaise représentation. Cet exemple simple est typique du changement de paradigme des processus de recherche de nos jours.

Positionnement et public visé

Tous ces éléments, qui nuisent à l'accessibilité des données et documents, constituent ce que l'on peut appeller du "bruit informationnel". Si l'accessibilité de l'information est un problème très général qui concerne la plupart des utilisateurs, nous allons limiter le cadre de notre étude à l'accessibilité des **documents** au sein d'un **système de recherche d'information**. Le bruit informationnel dans ce domaine est appellé "bruit documentaire". Cette notion, définie dans [Silem 1997], correspond à la qualité de la réponse d'un système de recherche d'information. C'est par exemple le cas lorsque trop de documents sont présentés à l'utilisateur, ou lorsque les documents retournés sont de faible qualité. Ce bruit peut avoir plusieurs causes : une mauvaise qualité des données, un système mal configuré, un utilisateur maladroit dans les requêtes, une trop grande quantité de documents disponibles, *etc.*

La présence de ce bruit informationnel dans un contexte de recherche est particulièrement problématique. C'est notamment le cas dans le domaine des sciences humaines et sociales, où la documentation joue un rôle majeur. De nombreux chercheurs s'intéressent aujourd'hui aux interactions entre informatique et SHS. Cette communauté est rassemblée sous le nom des **humanités numériques**. Des projets très variés sont nés de ces collaborations, mais ils supportent généralement des valeurs communes telles que l'approche critique, l'expérimentation, le travail collaboratif, la diffusion et le partage. Pour les chercheurs en SHS, il est essentiel de considérer le numérique non pas comme un simple outil, mais bien comme une évolution des pratiques [Dacos 2015].

Plusieurs critiques ont cependant récemment émergé, par exemple dans l'article écrit par Lauren F. Klein et Matthew K. Gold [3]. Les auteurs y présentent différents points qui nécessitent des améliorations. Nous en retiendrons particulièrement deux.

3. http://dhdebates.gc.cuny.edu/debates/text/51/comment/29887

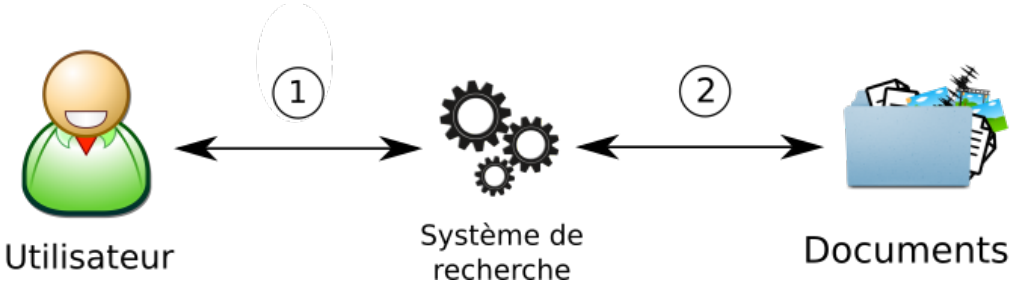
— Beaucoup de chercheurs utilisent des outils numériques sans totalement comprendre comment les données sont transformées. La confiance aveugle dans ces "boîtes noires" limite la possibilité d'une approche critique et réfléchie.

— Il y a un cruel manque d'intérêt concernant les aspects pédagogiques. L'évolution des pratiques vers le "tout numérique" nécessite de former les utilisateurs aux différents outils disponibles et à leur impact sur la recherche.

Les objectifs de cette thèse sont de proposer différentes stratégies pour répondre à ces critiques.

Organisation du manuscrit

FIGURE 1.2 – Interactions entre utilisateur, système et documents



Pour améliorer l'accessibilité des documents dans les plateformes de recherche d'information, nous avons identifié deux leviers d'action principaux. Nous pouvons dans un premier temps étudier la nature des interactions entre les utilisateurs et les systèmes (1). Nous pouvons aussi agir sur l'intégration des documents au sein des plateformes de recherche (2).

La figure 1.2 présente les principales interfaces qui séparent les utilisateurs des documents, et donc de l'information. Cette thèse est divisée en deux parties, chacune se concentrant sur une interface. Nous nous intéressons donc dans la première partie aux interactions entre les utilisateurs et les systèmes de recherche d'information. Nous faisons en effet le postulat qu'il est important de comprendre les pratiques des utilisateurs dans un système de recherche d'information. L'observation de ces pratiques peut nous aider à comprendre comment les utilisateurs conçoivent et utilisent une plateforme d'accès à des documents numériques. Grâce à ces observations, nous pouvons aussi identifier les différents points sur lesquels il est nécessaire de former les utilisateurs. Notre principale contribution est ici **le développement d'une plateforme expérimentale à double usage : l'observation du comportement des utilisateurs et la formation aux outils de recherche populaires**.

La deuxième partie de cette thèse s'intéresse à la manière dont les systèmes de recherche d'information intègrent et mettent à la disposition des utilisateurs des ressources numériques. En effet, si les utilisateurs peuvent améliorer leurs pratiques, ils n'ont aucune influence sur la manière dont ces documents sont intégrés. Nous

présentons donc dans cette partie différents outils d'analyse de documents susceptibles d'être utilisés par les chercheurs. Nous nous intéressons aussi à l'impact que peut avoir la qualité des ressources disponibles sur l'accessibilité des différents documents d'une plateforme de recherche d'information. La principale contribution de cette partie est **le développement d'une méthode pour limiter l'impact de la qualité des documents sur leur accessibilité**.

Première partie

Bibliothèques Numériques et Utilisateurs

Chapitre 2

De nombreux moyens d'accès à l'information

Sommaire

2.1 Introduction . . . 9
2.2 Historique des pratiques de recherche . . . 10
2.3 Fonctionnement d'un moteur de recherche . . . 12
2.3.1 Prétraitement des données . . . 13
2.3.2 Indexation et évaluation des requêtes . . . 14
2.3.3 Fonctions additionnelles . . . 17
2.4 Importance des métadonnées . . . 18
2.4.1 Différents types de métadonnées . . . 20
2.4.1.1 Métadonnées descriptives . . . 20
2.4.1.2 Métadonnées structurelles . . . 20
2.4.1.3 Métadonnées administratives . . . 21
2.4.2 Représentation des métadonnées . . . 21
2.5 Différentes plateformes de recherche d'information . . . 22
2.5.1 Moteur de recherche Web . . . 25
2.5.2 Bases de données en ligne . . . 26
2.5.3 Bibliothèques numériques . . . 26
2.5.3.1 Europeana . . . 28
2.5.3.2 Isidore . . . 29
2.5.3.3 Gallica . . . 31
2.6 Conclusion . . . 32

2.1 Introduction

Selon le chercheur danois Niels Ole Finnemann, une société où les échanges d'information n'ont que peu d'importance ne peut pas exister [Finnemann 2001]. Autrement dit, l'information est au cœur du développement de l'humanité. Comprendre comment sont organisées les données dans un système de recherche ne peut être que bénéfique pour les utilisateurs. Nous pensons en effet qu'il est important de comprendre pourquoi un document est plus pertinent qu'un autre au regard d'une requête. De nombreux paramètres, sur lesquels les utilisateurs n'ont que peu de contrôle, entrent en jeu dans la sélection automatique de documents pertinents.

Parce que ces technologies de recherche sont omniprésentes dans notre société, nous pensons qu'il faut en avoir une vision critique et ne pas leur faire aveuglément confiance. L'objectif de ce chapitre est donc de présenter les différents éléments qui jouent un rôle dans la recherche d'information en ligne de nos jours.

La première section est dédiée à une courte description de l'évolution des technologies de la recherche d'information. L'organisation de l'information est une problématique aussi vieille que l'écriture, et certains paradigmes de classement sont toujours utilisés aujourd'hui. Dans la deuxième section, nous présenteront le fonctionnement général d'un moteur de recherche, les différents traitements opérés sur les documents et les mécanismes d'interrogation. La section 2.4 présente le rôle des métadonnées dans l'organisation d'une base documentaire. Elles ont aussi un rôle prépondérant dans la recherche d'information. Il est donc essentiel d'en comprendre l'utilisation. Finalement, nous présentons dans la section 2.5 les différents types de plateformes ouvertes aux utilisateurs pour la recherche d'information en ligne. Nous nous attardons sur la description des bibliothèques numériques, particulièrement utilisées dans le monde de la recherche.

2.2 Historique des pratiques de recherche

L'Histoire de la recherche d'information est naturellement liée à l'Histoire de l'information elle-même. L'Humanité a pendant très longtemps échangé des informations de manière orale. Ce n'est qu'au IV$^{\text{ème}}$ millénaire avant J.-C., en Mésopotamie, qu'apparaissent les premiers écrits. Il semble qu'à l'époque, l'écriture soit essentiellement utilisée pour garder une trace des informations administratives et commerciales [Kramer 1988]. Les écrits s'accumulant avec l'âge, il fut vite nécessaire de réfléchir à l'organisation de l'information pour faciliter son accès. Une première façon de faire est le classement par ordre alphabétique qui a probablement été utilisé dans la grande bibliothèque d'Alexandrie [Blum 1991]. Des indexes plus complexes sont ensuite apparus. C'est par exemple le cas de *Naturalis Historia*, une encyclopédie rédigée par Pline l'Ancien et publiée aux alentours de l'an 77. Composée de 37 livres, elle rassemble des informations de culture générale dans différents domaines. Le premier des livres est utilisé comme principal point d'accès. Il contient une table des matières permettant de renvoyer le lecteur aux articles détaillés des autres livres.

À la fin du XIX$^{\text{ème}}$ siècle, deux bibliographes belges développent le système de classification décimale universelle (CDU). Ce système est un exemple de classement par facettes. Le principe est de représenter l'ensemble les connaissances et de les hiérarchiser. Ainsi, 9 classes principales, numérotées de 0 à 9 (la classe 4 est vide), sont au sommet de la hiérarchie : sciences sociales, mathématiques, philosophie, divertissement, *etc.* Ces dernières sont elles-mêmes divisées en 10 parties, qui peuvent encore se séparer. En tout, cette table contient plus de 60 000 subdivisions. Il existe également des symboles spéciaux capables de préciser une requête. Par exemple, la requête "17:7" concerne l'Éthique (catégorie 17) en relation avec l'Art (catégorie 7). Ce système a notamment été utilisé par l'un de ses créateurs, Paul Otlet, pour la réalisation du Mundaneum (aussi appelé Palais Mondial) à Mons en Belgique. Ce bâtiment a pour ambition de rassembler l'ensemble des connaissances humaines sur de

FIGURE 2.1 – Aperçu d'un tiroir de classement du Mondaneum

Au total, plus de 12 millions de cartes et documents sont disponibles à la consultation dans ce bâtiment. Photographie : Marc Wathieu / CC-BY

petites cartes rassemblées dans des tiroirs et classées selon le CDU (voir figure 2.1). Cette entreprise est aujourd'hui considérée comme le premier moteur de recherche de l'Histoire [Jenart 2013].

Vannevar Bush, un ingénieur américain, publie en 1945 un article intitulé "As We May Think"[Bush 1945] qui démontre l'importance d'améliorer les moyens d'accès à l'information scientifique. Il est le premier à proposer l'utilisation des ordinateurs pour cette tâche. Durant les années 1950, de nombreux travaux s'intéressent à cette idée. En 1958, l'utilisation de systèmes de recherche d'information est validée dans la Conférence Internationale sur l'Information Scientifique tenue à Washington. Les premiers systèmes sont basés sur un modèle booléen, c'est-à-dire sur la présence ou l'absence de termes dans une collection de documents. Un opérateur, spécialiste du domaine, est chargé d'identifier les termes importants permettant de discriminer les documents d'un corpus. Cette méthode simple et facilement compréhensible est efficace pour des corpus spécialisés lorsque l'utilisateur a une bonne connaissance du vocabulaire du domaine. Au contraire, si ce dernier manque d'expertise ou si le corpus est trop généraliste, il est difficile d'appliquer une telle méthode. De plus, l'indexation des documents était entièrement manuelle à l'époque. Les termes significatifs de chaque document ne sont donc pas toujours objectifs et ne font que traduire la vision de l'opérateur chargé de l'indexation.

Dans les années 1980, des algorithmes d'indexation automatique font leur apparition (voir section suivante). Très vite, il devient nécessaire de formaliser l'évaluation de ces différents algorithmes pour comparer leurs performances dans une tâche de recherche d'information. C'est pour cette raison que la conférence TREC (*Text REtrieval Conference*) a été créée dans le début des années 1990. Comme les

pratiques des utilisateurs évoluent et que la quantité d'informations disponibles augmente beaucoup avec la popularité du Web, il devient nécessaire d'évaluer la qualité des algorithmes de recherche à grande échelle. De nombreux groupes de recherche se retrouvent donc dans cette conférence pour développer et échanger de nouveaux corpus d'évaluation. Ces derniers, de plus en plus volumineux, permettent d'évaluer les différentes méthodes de manière plus réaliste.

À la fin des années 1990, l'augmentation rapide de la popularité d'Internet et du *World Wide Web*[1], notamment chez les particuliers, a conduit a une multitude de sites disponibles en ligne. Le principe de base du Web est que chaque ressource (document ou site) est identifiée par une adresse unique : une **URL** (*Uniform Resource Locator*). De plus, ces ressources sont liées par ce qu'on appelle des **liens hypertextes**. Chercheurs et ingénieurs se sont vite rendus compte que les différentes pages accessibles sur le Web pouvaient être automatiquement identifiées par un algorithme capable de naviguer grâce aux liens hypertextes [Sanderson 2012]. Cette méthodologie est à l'origine de l'algorithme *PageRank* développé par les créateurs de Google. Dans celui-ci, une note de popularité est associée à chaque site selon le nombre et la qualité des liens qui permettent d'y accéder (voir figure 2.2). Il peut être utilisé pour faciliter la recherche d'information dans tous les corpus qui contiennent des documents reliés entre eux d'une certaine manière. Dans le cas des pages Web, ces liens prennent la forme d'adresse Web. Dans un corpus de littérature scientifique, les citations peuvent avoir le même rôle.

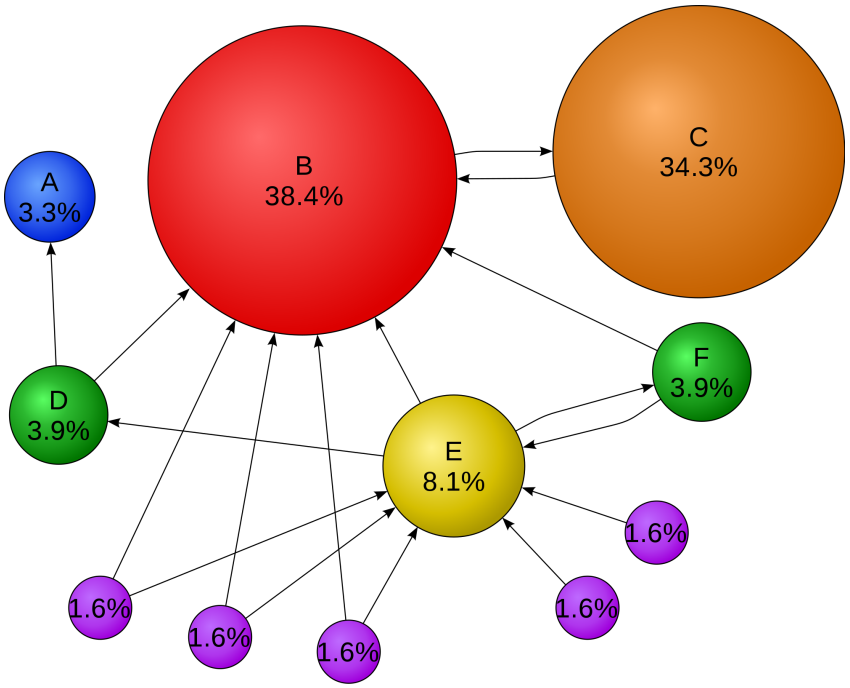
Aujourd'hui, de plus en plus de plateformes de recherche d'information sont disponibles, dans tous les domaines. Pour les utilisateurs, le problème n'est plus de savoir si une information existe, mais bien de trouver l'information pertinente au bon moment. La principale interface entre tous ces documents et les utilisateurs sont les moteurs de recherche. Ces derniers sont capables d'analyser et de répondre aux besoins informationnels d'un utilisateur, exprimés sous la forme d'une requête. Nous présentons leur fonctionnement général dans la section suivante.

2.3 Fonctionnement d'un moteur de recherche

Tous les moteurs de recherche reposent sur des concepts et un objectif communs : répondre au besoin informationnel d'un utilisateur. Ce besoin est exprimé sous la forme d'une requête qui est ensuite traitée par le système, lequel renvoie alors un ensemble de résultats possibles, classés par ordre de **pertinence**. Pour l'utilisateur, ce concept est flou et très peu explicite. À terme, cela peut induire différents biais qui seront explicités dans le chapitre suivant. Avant de pouvoir répondre à une requête, le moteur de recherche doit en amont avoir indexé les données. C'est-à-dire avoir identifié ce qui caractérise chacun des documents. Dans le cas de documents textuels, il est souvent nécessaire de transformer les données avant l'indexation.

1. Ces deux notions ne sont pas équivalentes. Pour résumer rapidement, le Web est une application d'Internet qui permet de publier et consulter différents types de documents (sites, contenus numériques, *etc.*). Internet supporte de nombreux autres services : téléphonie, messagerie, TV, *etc.*

FIGURE 2.2 – Principe de fonctionnement du PageRank



Ce graphe représente un échantillon de sites Web faisant référence les uns aux autres. Plus un site Web est référencé, plus on lui donne d'importance. Ainsi, la plupart des sites ci-dessus font référence au site B. On associe alors à ce site une note de popularité élevée. Le site C est lui aussi important, même si un seul site le référence. C'est une illustration de la "qualité" des liens. Parce que le site B est important, les sites qu'il référence doivent aussi l'être.

2.3.1 Prétraitement des données

Pour être capable de réaliser sa fonction, un moteur de recherche doit être capable de comparer le contenu d'une requête avec celui des documents à sa disposition. Les textes contiennent des informations à plusieurs niveaux de granularité : chapitres, paragraphes, phrases, mots et caractères. Si l'élément le plus basique est le caractère, on ne peut cependant pas lui attribuer de sens. C'est pour cette raison que les moteurs de recherche se concentrent plutôt sur l'analyse des mots. Il est donc nécessaire de transformer les différents textes pour en extraire les groupes de caractères consécutifs formant un "mot". Nous utilisons ici des guillemets car plusieurs types d'information peuvent être extraits. Il est par exemple intéressant de pouvoir reconnaître une adresse mail, une adresse IP, un numéro de téléphone *etc.* Quoi qu'il en soit, la **segmentation** d'un texte n'est pas un problème simple à résoudre. Une approche intuitive consiste à utiliser l'espace comme caractère séparant les mots. Cependant, certains termes ne devraient pas être séparés. C'est notamment le cas de certaines villes (Los Angeles, La Rochelle, *etc.*). En français, l'utilisation de l'apostrophe pose aussi certaines questions. Ainsi, quelle est la segmentation correcte pour "l'avion" ? Est-ce "l ' avion", "l' avion" ou "l'avion" ? La question est d'autant plus

difficile dans certaines langues d'Asie où les différents mots ne sont séparés par aucun caractère. Il convient alors d'étudier des modèles de langages pour espérer obtenir la segmentation correcte. Si la segmentation des mots d'un texte n'est pas un problème évident, la segmentation des phrases ne l'est pas plus. Comme nous l'avons tous appris à l'école élémentaire, le caractère qui permet d'identifier la fin d'une phrase est le point. Cependant, ce caractère peut également apparaître derrière une abréviation par exemple (Dr. pour docteur, Pr. pour professeur, *etc.*).

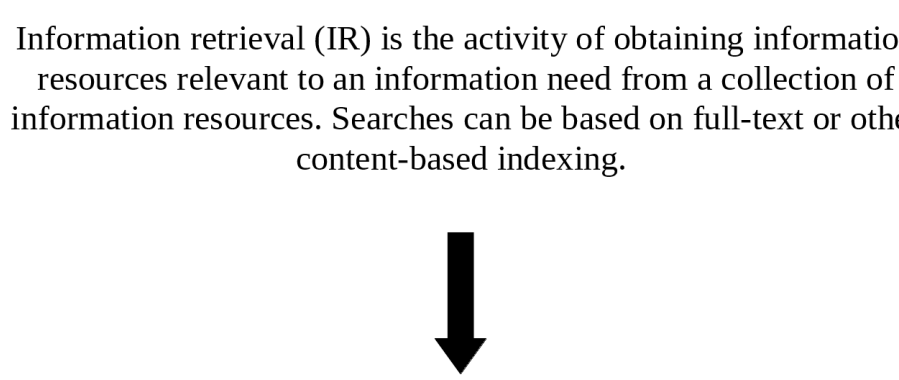
Un autre problème à traiter est la variabilité de certains mots. Selon la grammaire de la langue étudiée, un même mot peut en effet être modifié selon le contexte dans lequel il est utilisé. C'est notamment le cas en français avec la notion de genre ou de nombre. Il semble en effet bénéfique pour la recherche d'information que les termes "petit", "petite", "petits" et "petites" soient indexés de la même manière. La transformation permettant cela est appelée **racinisation** (*stemming* en anglais). Le premier algorithme capable de faire cela est présenté dans [Lovins 1968]. Un autre algorithme très populaire encore aujourd'hui est celui de Porter, présenté dans [Van Rijsbergen 1980]. Développé pour la langue anglaise, ces deux algorithmes font usage d'une liste de règles permettant de tronquer les mots selon leur terminaison. Il est bon de noter que les différentes racines extraites par ce type d'algorithme ne sont pas nécessairement des mots réels. Par exemple, la racine du mot "chercher" est "cherch", qui n'est présent dans aucun dictionnaire.

D'autres traitements sont effectués sur les documents pour améliorer les chances qu'a un utilisateur de trouver une information. Il est par exemple courant de transformer les textes pour qu'ils ne contiennent que des lettres minuscules. De même, certains mots fréquents et peu porteurs d'information sont supprimés. On les appelle "mots vides" (*stop words* en anglais). Ce sont par exemple les déterminants, les pronoms, les articles, *etc.* De plus, certains traitements spécifiques au corpus à traiter peuvent être opérés (suppression de balises HTML par exemple). Finalement, un exemple de texte transformé est présenté dans la figure 2.3. C'est ce texte transformé qui sera indexé par le moteur de recherche.

2.3.2 Indexation et évaluation des requêtes

L'objectif de tout moteur de recherche est de répondre aux besoins informationnels qu'un utilisateur peut exprimer à travers une requête. Pour effectuer cette tâche dans un temps raisonnable pour l'utilisateur, il convient cependant d'organiser l'information des différents documents mis à disposition. En effet, si le moteur de recherche devait à chaque requête parcourir le contenu de l'ensemble des documents, cela prendrait un temps considérable. Pour éviter cela, une solution typique est d'utiliser un index. L'idée est très similaire à l'index d'un livre qui recense les pages dans lesquels apparaissent les mots importants. Durant l'indexation, tous les documents préalablement normalisés grâce aux traitements de la section précédente sont observés. Les différents termes qui les composent sont alors enregistrés dans une liste, aux côtés des identifiants des documents qui les contiennent. De la même manière que l'index d'un livre fait apparaître uniquement les mots sémantiquement importants, un traitement automatique doit pouvoir évaluer l'importance des mots du corpus à indexer.

FIGURE 2.3 – Transformation d'un texte avant son indexation



Plusieurs traitements sont opérés sur le texte original (extrait de la page anglaise de Wikipedia sur la recherche d'information) : suppression des mots vides, passage en minuscule, racinisation, etc. Cette étape permet de faciliter la recherche d'information en se concentrant sur l'essence même des mots. Cela permettra à l'utilisateur de rechercher de l'information sans se soucier des différentes variations qu'un mot peut subir.

Une stratégie généralement adoptée est d'associer à chaque terme du vocabulaire un poids numérique représentant son importance. Ici, la notion d'importance est en rapport avec le caractère discriminatoire d'un mot. Ainsi, un mot important est un mot qui est typique d'une petite proportion des documents d'un corpus. Ce principe est exploité par le calcul du TF*IDF [Salton 1973]. Le principe de base est d'estimer l'importance d'un mot dans un document selon deux critères :

— *Term Frequency* (TF) : la fréquence du mot dans un document donné. Intuitivement, plus un terme est présent dans un document, plus il a d'importance.

— *Inverse Document Frequency* (IDF) : correspond à l'inverse du nombre de document qui contiennent au moins une fois ce mot. Plus le terme est commun dans le corpus, moins l'IDF sera élevé.

Un autre élément à prendre en compte est la taille des documents. En effet, un terme qui apparaît deux fois dans un livre n'a pas la même importance qu'un terme qui apparaît deux fois dans un *tweet*. Pour cette raison, il peut être intéressant de normaliser la fréquence d'un terme selon le nombre total de mots dans le document.

Parce qu'il a été créé pour des humains, l'objectif du TF*IDF est d'imiter la perception qu'un utilisateur peut se faire de la pertinence d'un document donné, par rapport à une requête. Or, il se trouve que la mesure du TF*IDF telle que nous l'avons décrite ne répond pas tout à fait à cet objectif. En effet, ce n'est pas parce qu'un terme apparaît deux fois plus dans un document que dans un autre que ce dernier est deux fois plus important. Pour limiter ce biais, différentes variations

des calculs des composants du TF*IDF sont utilisées. Il est par exemple possible d'effectuer une normalisation logarithmique du TF, qui va permettre de lisser les différences entre documents. Avec ce calcul, un document qui contient deux fois plus un terme donné sera plus important, mais pas deux fois plus important qu'un autre document. Les mêmes considérations sont valables pour l'IDF, et différents calculs peuvent être utilisés pour lisser les résultats et se rapprocher d'une notion de pertinence comparable à celle d'un humain.

Une autre évolution populaire du TF*IDF est l'algorithme BM25, basé sur l'idée d'un modèle probabiliste de pertinence développée dans [Robertson 1976], et dont la première utilisation est référencée dans [Mitev 1985]. La principale évolution de cet algorithme est de limiter encore plus l'impact de la fréquence des termes sur la pertinence des documents. À la différence du TF*IDF, où le score TF augmente presque linéairement avec l'augmentation de la fréquence d'un terme, le score TF calculé par BM25 est limité par une valeur asymptotique qu'il ne pourra jamais dépasser. De plus, comme dans le TF*IDF classique, la longueur du document est aussi prise en compte. Cependant, elle est comparée avec la longueur moyenne des documents du corpus. Ainsi, le score TF d'un terme dans un document particulièrement long augmentera moins vite avec sa fréquence que dans un autre document. Au contraire, un terme très présent dans un document plus court va rapidement faire saturer le score TF à sa valeur asymptotique.

Ces différents algorithmes de pondération sont utilisés pour évaluer les différentes requêtes émises par les utilisateurs. La plupart des moteurs de recherche combinent différents modèles pour classer les documents selon leur pertinence par rapport à une requête donnée. Le modèle booléen peut ainsi être utilisé pour rapidement discriminer les documents susceptibles d'être pertinents. Par exemple, la requête **(chat OR tigre) AND NOT chien** s'intéresse uniquement aux documents qui contiennent le mot "chat" ou le mot "tigre" et qui ne contiennent pas le mot "chien". Il convient finalement de classer les documents restants selon leur pertinence par rapport à cette requête. Pour cela, il est d'abord nécessaire d'appliquer à la requête les mêmes traitements qu'ont subis les documents durant la phase d'indexation (segmentation, racinisation, suppression des mots vides, *etc.*). Finalement, le modèle de représentation vectoriel des documents est souvent utilisé (plus de détails sont fournis dans la section 4.4.1.1) pour comparer la requête avec les documents indexés par le moteur de recherche. Selon les termes de la requête et l'importance qu'ils ont dans les différents documents du corpus, le moteur de recherche renvoie une liste de résultats triés selon leur pertinence telle que perçue par le moteur de recherche.

Parce qu'il existe de nombreuses façons de calculer l'importance des termes et la similarité des documents par rapport à une requête, il est nécessaire d'être capable de comparer ces méthodes. Il existe de nombreuses métriques qui peuvent être utilisées pour l'évaluation d'un système de recherche. Les plus populaires sont la **précision** et le **rappel**, toujours utilisées aujourd'hui. La première correspond à la qualité des résultats renvoyés par le système après une requête. Si tous les documents de la réponse sont pertinents, alors la précision sera maximum. La mesure de rappel, elle, évalue la proportion de documents pertinents renvoyés par le système par rapport

à tous les documents pertinents existant. Ces deux mesures étant antagonistes, il convient d'ajuster le système pour trouver un compromis. Ces deux mesures sont souvent combinées grâce à une moyenne harmonique (le score F_1) pour simplifier la comparaison de deux systèmes.

2.3.3 Fonctions additionnelles

Décrire le fonctionnement de base de la plupart des moteurs de recherche nous a permis d'identifier les principaux processus opérés sur les données. Cependant, de nombreuses autres améliorations sont possibles. On peut par exemple mettre en place la gestion des expressions en utilisant un modèle de n-grammes. Ce dernier consiste à analyser non seulement les termes d'un document, mais également les ensembles de n termes (quand $n = 2$, on parle de bigramme). Ce genre de modèles permet par exemple de traiter la requête **"chat noir"** et de renvoyer uniquement les documents qui contiennent ces deux termes consécutivement.

Jusqu'ici, nous avons traité chaque document comme une seule entité constituée uniquement d'un corps de texte. Cependant, d'autres champs peuvent être associés à un document : le titre, l'auteur, *etc.* Les recherches textuelles sont donc possibles sur tous ces champs. De plus, il est possible d'influencer les résultats du moteur de recherche en augmentant artificiellement l'importance d'un de ces champs. Dans certains cas, cela peut par exemple être bénéfique de donner plus d'importance au titre d'un document pour faciliter son accès. Il est aussi possible d'augmenter l'importance d'un document entier. Ces deux opérations doivent être réalisées durant l'indexation des documents. Certains moteurs de recherche offrent aussi la possibilité d'augmenter l'importance d'un mot durant une requête. Cette augmentation (*boost* en anglais) est matérialisée par un nombre décimal positif. Si ce dernier est compris entre 0 et 1, l'importance sera au contraire diminuée.

Il est courant aujourd'hui pour les moteurs de recherche de conserver l'historique de recherche des utilisateurs. Cette pratique permet notamment de mettre en place des algorithmes de recommandation. Il en existe deux grandes catégories [Candillier 2009] :

— **Approches basées sur le contenu :** Ces méthodes s'intéressent à la similarité utilisateur/document pour faire des recommandations. Selon le profil d'un utilisateur (ses intérêts, ses recherches passées, *etc.*), des documents susceptibles de l'intéresser peuvent être recommandés.

— **Approches collaboratives :** Le principe de ces méthodes est d'identifier des groupes d'utilisateurs avec des besoins informationnels similaires. Ainsi, il sera possible de proposer à un utilisateur un document qu'il n'a pas encore vu mais qui a intéressé des utilisateurs similaires. Parce qu'elles ne s'intéressent pas au contenu des documents, ces approches ont l'avantage de pouvoir recommander des documents sans compréhension de leur contenu.

Dans certains cas, des approches hybrides sont envisagées. C'est par exemple le cas de Netflix[2], une plateforme de vidéos en ligne, qui fait des suggestions en se basant

2. https://www.netflix.com

à la fois sur les films qu'un utilisateur a déjà regardé et sur ceux appréciés par des utilisateurs aux goûts similaires.

Les requêtes que les utilisateurs expriment sont l'essence même des échanges avec les moteurs de recherche. Pour cette raison, la compréhension de requête est un domaine de recherche particulièrement pertinent. Pour répondre le mieux possible aux besoins informationnels d'un utilisateur, il est essentiel de comprendre son intention. Différentes stratégies sont généralement utilisées pour cela. Une analyse syntaxique (voir section 4.3.2) peut être menée si la requête est exprimée en langage naturel. Il est par exemple possible d'identifier la nature de l'information recherchée grâce à la présence d'un mot interrogatif particulier ("qui" correspond à une personne, "quand" correspond à une date, *etc.*). Dans certains cas, les requêtes émises par les utilisateurs sont modifiées par le moteur de recherche. Des synonymes de certains termes peuvent par exemple être ajoutés à la requête pour améliorer le nombre de documents susceptibles d'être pertinents (mesure de rappel). Certains moteurs de recherche sont aussi capable de détecter certaines erreurs d'orthographe et de les corriger automatiquement.

Un autre aspect important de l'interaction avec un moteur de recherche est la recherche par facette. Cette dernière permet de filtrer un grand nombre de résultats pour faciliter la navigation. Les facettes correspondent à différentes propriétés, physiques ou sémantiques, d'un document. Dans le cas d'un moteur de recherche de romans par exemple, une recherche par facette peut permettre de filtrer les résultats par genre ou par auteur. Les informations filtrées par les facettes sont souvent conservées sous forme de métadonnées. Ces dernières ont en effet un rôle majeur dans la recherche d'information, et la section suivante leur est consacrée.

2.4 Importance des métadonnées

Formellement, les métadonnées permettent de définir ou décrire différents aspects d'un document (son contenu, son origine, sa forme, *etc.*). Si le terme "métadonnée" est relativement récent (il a été utilisé pour la première fois dans [Bagley 1969]), l'idée n'est pas nouvelle. La plupart des documents, anciens ou non, sont en effet accompagnés d'informations complémentaires comme le titre ou encore l'auteur. Les métadonnées sont souvent tout aussi importantes que le contenu qu'elles décrivent. Un des plus gros créateurs de métadonnées sur le Web est l'encyclopédie Wikipedia[3] qui associe à la plupart des articles un nombre plus ou moins important de métadonnées structurées (voir figure 2.4). Le projet DBpedia[4] exploite toutes ces informations dans une base de connaissance accessible et utilisable par tous. C'est grâce à la forte structuration des métadonnées que ce projet a pu voir le jour. De telles bases de connaissance permettent par exemple de créer des systèmes où un utilisateur peut poser une question en langage naturel (voir figure 2.5).

Les domaines de l'archivage et de la bibliographie sont les premiers à avoir massivement fait usage des métadonnées. Les catalogues des librairies sont remplis de

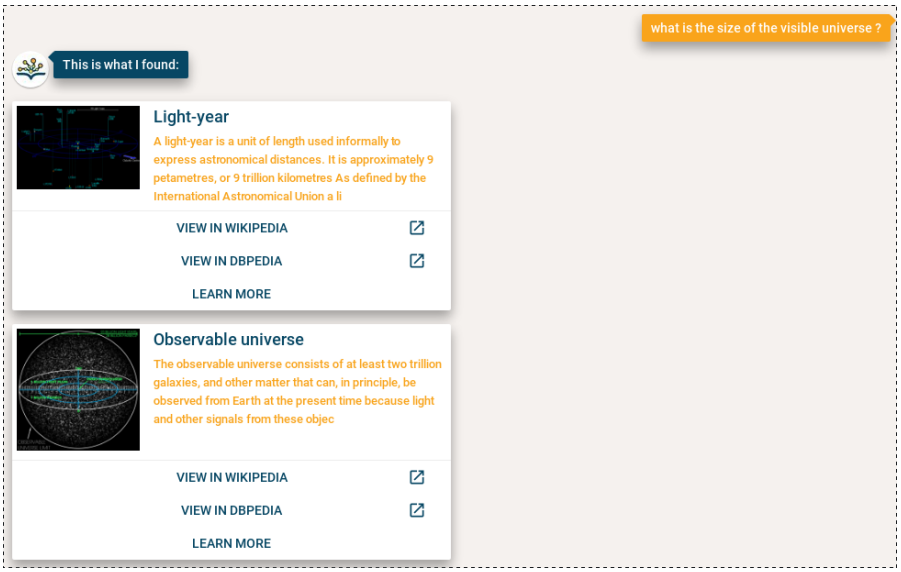
3. https://fr.wikipedia.org
4. http://wiki.dbpedia.org/

FIGURE 2.4 – Aperçu des métadonnées associées à la page Wikipedia de la ville de Paris

Administration	
Pays	France
Région	Île-de-France (préfecture)
Département	Paris (préfecture)
Arrondissement	Paris (chef-lieu)
Canton	Chef-lieu de 20 cantons (les arrondissements)
Intercommunalité	Métropole du Grand Paris
Maire Mandat	Anne Hidalgo (PS) 2014-2020
Code postal	75001 à 75020 et 75116
Code commune	75056 et de 75101 à 75120
Démographie	
Gentilé	Parisiens
Population municipale	2 220 445 hab. (2014▼)
Densité	21 067 hab./km²
Population aire urbaine	12 475 808 hab. (2014)
Géographie	
Coordonnées	48° 51′ 24″ nord, 2° 21′ 07″ est
Altitude	Min. 28 m Max. 131 m
Superficie	105,40 km²

Ces métadonnées rassemblées par les contributeurs de Wikipedia sont organisées de manière structurée. Chaque champ a ainsi une signification particulière, et deux objets similaires seront associés aux mêmes types de métadonnées. En effet, les mêmes informations sont disponibles pour d'autres villes que Paris.

FIGURE 2.5 – Agent conversationnel utilisant une base de connaissance



Cet agent est disponible à l'adresse `http://chat.dbpedia.org/`. *L'utilisateur peut y poser une question en langage naturel. Le système, après avoir compris l'intention de l'utilisateur, va chercher la réponse dans la base de connaissance DBpedia.*

métadonnées descriptives permettant de faciliter l'accès aux différentes ressources. Grâce à l'aspect hiérarchique de ces métadonnées, des groupes de documents "similaires" peuvent être simplement agrégés. C'est d'une grande importance pour les chercheurs, particulièrement en sciences sociales, qui peuvent alors rapidement identifier un corpus de recherche cohérent. C'est par exemple le cas des historiens, pour qui la mise en correspondance de différentes sources est une importante partie du travail. Il existe cependant de nombreux types et standards de métadonnées pour différentes disciplines et différents besoins (archivage, accessibilité, *etc.*).

2.4.1 Différents types de métadonnées

Les métadonnées accompagnent la majorité des documents et peuvent y associer différentes informations complémentaires. Elles ont plusieurs objectifs. D'abord, grâce à elles, la découverte d'information au sein d'un corpus peut être facilitée. Elles jouent aussi un rôle dans l'organisation des documents au sein d'un système d'information. Finalement elles apportent des informations cruciales pour l'archivage des données. Ces trois types de métadonnées, tous essentiels dans un projet qui s'inscrit dans la philosophie des Humanités Numériques, sont présentés en détail dans les sous-sections suivantes.

2.4.1.1 Métadonnées descriptives

Les métadonnées descriptives ont un double usage. Elles permettent à la fois de faciliter l'accès à l'information et d'apporter du contexte aux documents. On peut trouver dans ces métadonnées des informations comme le titre d'un document, son ou ses auteurs, sa date de création, *etc.* Les systèmes de recherche d'information permettent de sélectionner des documents selon la valeur de ces informations. Cela permet aux utilisateurs de rapidement filtrer une liste de résultats pour identifier plus rapidement un document ou obtenir une information. De plus, des métadonnées sont associées à tout type de document, peu importe leur nature. Elles peuvent ainsi permettre aux utilisateurs d'accéder à des documents hétérogènes (photos, textes, vidéos, *etc.*) grâce à une même requête. Face à l'augmentation régulière du nombre de ressources disponibles en ligne, ces métadonnées sont particulièrement importantes pour l'accessibilité de ces documents.

2.4.1.2 Métadonnées structurelles

Comme leur nom l'indique, ces métadonnées permettent de décrire la structure d'un document, ou les relations que plusieurs documents entretiennent entre eux. Leur but peut être de faciliter la navigation à travers un document ou la lecture automatique en indiquant des informations telles que l'ordonnancement des pages ou le numéro des chapitres. Par exemple, une œuvre théâtrale contient des métadonnées qui indiquent quel personnage est en train de parler. Si le système de recherche est capable de prendre en compte ce genre de métadonnées, la découverte d'information sera facilitée. Ces métadonnées peuvent aussi permettre de naviguer d'une ressource à une autre. Par exemple, une photographie pourra être associée au document duquel elle est extraite.

2.4.1.3 Métadonnées administratives

Ce dernier type de métadonnées est associé aux informations nécessaires à la gestion des documents (accès ou diffusion par exemple) ou à leur archivage. Par exemple, dans le cas de la diffusion de vidéos ou de l'affichage d'images, il est nécessaire de connaître le type des fichiers et différentes informations supplémentaires comme la résolution ou les algorithmes utilisés pour l'encodage des flux de données. Ces informations sont également essentielles à la conservation à long terme des documents. En effet, les normes de stockage des contenus numériques ne sont pas éternelles et évoluent avec le temps. Conserver ces informations permet ainsi de prévoir différentes solutions pour anticiper de possibles changements dans les normes de conservation. Une possibilité peut être de favoriser l'utilisation de formats ouverts [5] comme le PDF [6].

D'autres métadonnées peuvent être utilisées pour la gestion des droits d'accès aux différents documents. On pourra par exemple y trouver des informations de propriété intellectuelle comme le *Copyright* ou la licence *Creative Commons* qui définissent les droits de diffusion d'une ressource numérique. Il est aussi possible d'indiquer quels utilisateurs ou quels groupes d'utilisateurs ont le droit d'accéder à un document particulier au sein d'une plateforme.

2.4.2 Représentation des métadonnées

Les métadonnées peuvent souvent être exprimées de différentes manières. La première distinction importante à faire est la différence entre des métadonnées internes et externes. Comme leur nom l'indique, les métadonnées internes font intégralement partie du document qu'elles représentent. Les pages Web, par exemple, sont décrites grâce au langage HTML qui comporte différentes balises de métadonnées. La balise `<title></title>` permet de représenter le titre d'une page. Cependant, celui-ci n'est pas destiné à être affiché mais sert plutôt durant l'indexation des pages par les moteurs de recherche. De la même manière que les balises, les métadonnées techniques associées à des documents audio ou vidéo (durée, encodage, *etc.*) sont présentes dans le même conteneur que le flux de données qu'elles décrivent. Les métadonnées externes, elles, sont stockées dans des fichiers annexes. Pour associer des informations à un document particulier, il faut être capable de l'identifier. Si on peut utiliser son chemin sur le système d'information ou un identifiant provenant d'une base de données, on préfère souvent l'utilisation des **URI** [7]. Les métadonnées externes sont essentielles lorsque l'on souhaite décrire les relations qui existent entre plusieurs ressources.

Dans un souci d'interopérabilité, il est très important que les métadonnées suivent un schéma normalisé. L'utilisation d'un schéma de représentation commun

5. Un format dit "ouvert" est un format dont les spécifications techniques sont publiques. Au contraire d'un format propriétaire qui peut être amené à disparaitre, les formats ouverts sont créés dans l'objectif de durer dans le temps.

6. *Portable Document Format*

7. *Uniform Resource Identifier.* Chaîne de caractères qui permet d'identifier de manière unique une ressource. Les URL, adresses des pages Web, sont un type particulier d'URI.

permet de faciliter les échanges entre différents systèmes. Le principe de ces schémas est de définir et normaliser les informations qui peuvent être associées à une ressource particulière, peu importe sa nature (livre ou disque, analogique ou numérique). Formellement, un schéma associe à chaque terme et propriété un identifiant unique (sous la forme d'une URI), une définition et un label. Ces informations sont souvent exprimées grâce au modèle RDF [8] (*Resource Description Framework*). Il permet de représenter différentes informations liées entre elle sous la forme d'un graphe. Formellement le modèle RDF enregistre un ensemble de triplets (sujet, prédicat, objet). Le sujet est la **ressource** à décrire, le prédicat définit **l'information** que l'on veut décrire et l'objet représente la **valeur** de cette information (voir figure 2.6).

Le graphe présenté dans la figure 2.6 pourrait utiliser un schéma défini par deux classes et 5 prédicats différents (voir tableau 2.1). Les schémas de métadonnées standardisés que l'on peut trouver sont composés du même genre d'informations que ce schéma fictif (identifiant sous la forme d'une URI, label, définition, *etc.*). Différentes syntaxes peuvent être utilisées pour exprimer et automatiser l'usage de ces règles (XML, Turtle, JSON, *etc.*). Un exemple connu est le schéma Dublin Core [9]. Il est composé de termes et de propriétés permettant de décrire une ressource sur le Web. Ainsi, les termes correspondent à des classes bien particulières telles que Agent, BibliographicResource, MediaType, SizeOrDuration, Text, Image, *etc.* Toutes ces classes peuvent être reliées entre elles grâce à des propriétés définies par le schéma. D'autres schémas existent et il convient de choisir le plus adapté. On peut par exemple citer pour les photographies la norme EXIF. Elle décrit principalement différents aspects techniques liés aux réglages de l'appareil photo (résolution, ouverture, focale, compression utilisée, *etc.*) mais peut aussi contenir les coordonnées GPS de la prise de vue. Le but n'est pas ici de faire un état de tous les langages de métadonnées disponibles. Il en existe de nombreux, dédiés à chaque cas d'utilisation. Dans le cas contraire, il est possible d'en créer un, adapté à la situation. Les vocabulaires les plus usités sont présentés dans [Riley 2017].

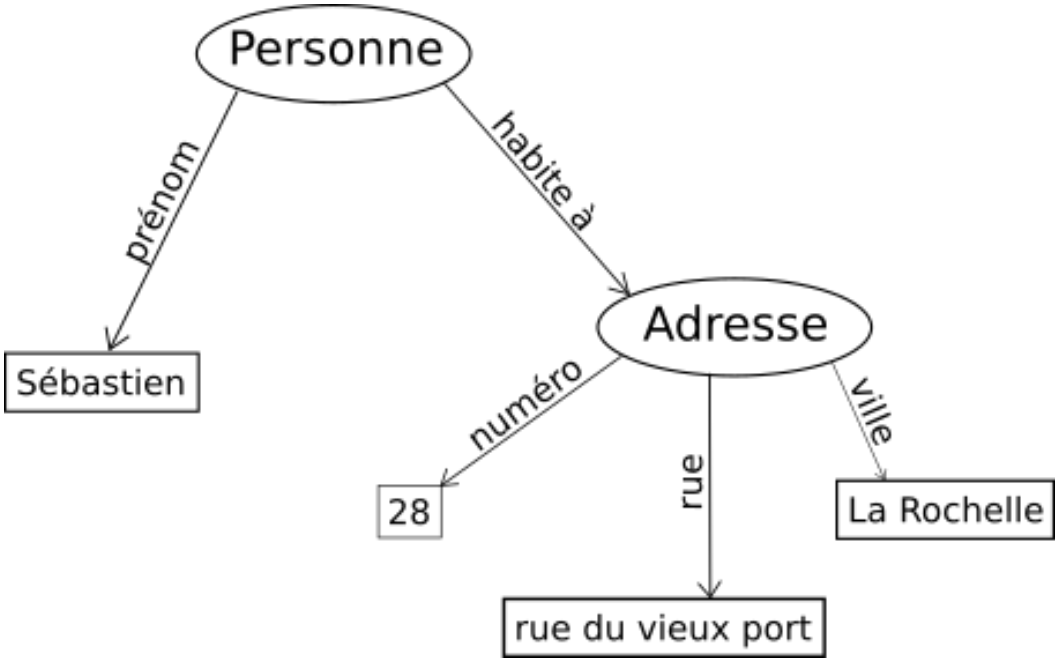
2.5 Différentes plateformes de recherche d'information

Il existe plusieurs catégories de plateformes numériques permettant de faire de la recherche d'information. Plus ou moins évoluées, ces dernières tentent toutes de répondre aux besoins des utilisateurs. Nous listons ci-dessous un certain nombre des fonctionnalités que ces plateformes peuvent implémenter, ainsi que le rôle qu'elles jouent dans la recherche d'information.

— **Qualité des métadonnées** : toutes les bibliothèques numériques fournissent un certain nombre de métadonnées pour chaque ressource (titre, auteur, date, *etc.*). Cependant, certaines métadonnées sont plus rares ou ne sont pas toujours de très bonne qualité (description très succincte ou métadonnées disponibles dans certaines langues uniquement). Du point de vue de l'accessibilité des documents, les métadonnées restent aujourd'hui un important point d'entrée vers l'information. Cette situation est sans doute amenée à changer avec l'évolution

8. https://www.w3.org/RDF
9. http://www.dublincore.org/documents/dcmi-terms/

FIGURE 2.6 – Exemple de métadonnées sous la forme d'un graphe RDF



Ce graphe associe des métadonnées à un objet de type Personne. La représentation sous forme de graphe permet de manipuler des objets composites. Dans cette figure, une Personne est associée à son prénom ainsi qu'à une adresse, elle-même associée à plusieurs propriétés (numéro, rue et ville).

TABLEAU 2.1 – Schéma fictif de métadonnées utilisé dans le graphe 2.6

	Nom	URI	Description	
Classes	Personne	http://www.example.com/classes/personne	représente un individu	
	Adresse	http://www.example.com/classes/adresse	représente une adresse postale	
	Nom	**URI**	**Sujet**	**Objet**
Prédicats	habite à	http://www.example.com/predicats/habite_a	Personne	Adresse
	prénom	http://www.example.com/predicats/prenom	Personne	chaîne de caractères
	numéro	http://www.example.com/predicats/numero	Adresse	nombre entier
	rue	http://www.example.com/predicats/rue	Adresse	chaîne de caractères
	ville	http://www.example.com/predicats/ville	Adresse	chaîne de caractères

Les propriétés du schéma sont séparées en deux catégories : les classes et les prédicats. Les URI, qui sont ici fictives, jouent le rôle d'identifiant unique pour chacune de ces propriétés.

des algorithmes sémantiques, capables d'aller chercher l'information pertinente dans des données non structurées.

— **Recherche par facettes** : les facettes permettent de rapidement filtrer une liste de résultats selon la valeur des métadonnées associées. Il existe différents types de facettes selon le type des métadonnées mais le plus courant est la liste à cocher. La plupart des bibliothèques numériques implémentent cette fonctionnalité. Cependant, la gestion du nombre de facettes affichées à l'écran n'est pas toujours bien gérée. Certaines métadonnées peuvent prendre de très nombreuses valeurs possibles (l'*organisme contributeur* associé à une ressource en est un bon exemple). Il n'est en effet pas toujours possible de modifier l'ordre d'affichage de ces facettes (que ce soit un ordre alphabétique ou fréquentiel), ce qui ne facilite pas leur utilisation.

— **Recherche dans le texte complet** : certaines bibliothèques numériques permettent de rechercher de l'information dans le texte des documents. Parfois transcrits manuellement, ces textes sont généralement extraits par des processus automatiques. Si un document de bonne qualité sera souvent correctement transcrit, le texte automatiquement extrait de documents plus complexes sera souvent de mauvaise qualité. Par exemple, la présence de taches ou de plis sur les documents scannés influence la qualité de la transcription automatique. Certaines bibliothèques présentent à l'utilisateur un taux de reconnaissance supposé du document en cours de consultation. Ce taux provient généralement directement des algorithmes d'extraction automatique. La qualité de ces processus a sans aucun doute un impact sur l'accessibilité des documents au sein d'une bibliothèque numérique. Souvent, après une requête, des extraits qui contiennent les termes de la recherche sont présentés à l'utilisateur sous les documents pertinents. C'est une fonctionnalité importante car cela permet à l'utilisateur de rapidement identifier le passage qui l'intéresse dans un document, peu importe sa taille.

— **Recherche des entités nommées** : ces dernières représentent des informations particulièrement importantes sur les différents documents. Elles permettent notamment de répondre aux questions *Qui ?*, *Quoi ?*, et *Où ?* définissant le contexte d'une situation.

— **Formulaire de recherche avancée** : ces formulaires fournissent une interface aux utilisateurs qui savent exactement ce qu'ils cherchent et qui souhaitent réaliser des requêtes plus précises. Ils ont ainsi la possibilité d'interroger la base de documents par la valeur de leurs métadonnées. Ces différents paramètres peuvent être combinés grâce aux opérateurs booléens (voir sous-section 2.3.2) pour créer une requête adaptée aux besoins de l'utilisateur. Si la plupart des bibliothèques possèdent ces fonctionnalités, leur utilisation n'est pas toujours très claire.

— **Recherche thématique** : les bibliothèques numériques rassemblent parfois certains documents au travers d'expositions ou de sélections. Ces documents sont mis en valeur et sont accompagnés de descriptions contextuelles. Souvent, ils sont accessibles grâce à une requête complexe qui contient différents mots clés et différentes valeurs possibles des métadonnées. Ce genre de fonc-

tionnalité est particulièrement intéressant pour les recherches exploratoires, où l'utilisateur n'a pas une idée très précise de ce qu'il cherche.

— **Recommandations** : les recommandations sont utilisées pour proposer de nouveaux contenus à un utilisateur après une recherche ou durant la consultation d'une ressource. Ces mécanismes sont, comme les recherches thématiques, des outils intéressants pour la découverte de nouvelles ressources.

— **Accès par API** [10] : l'automatisation des processus de recherche est essentielle pour l'analyse d'une grande masse de documents ou pour la création de nouvelles applications. Aujourd'hui, la plupart des bibliothèques numériques fournissent un accès aux fonctionnalités de recherche.

Nous présentons dans cette section une vue d'ensemble des principales plateformes existantes. Nous ferons état des différentes caractéristiques que ces outils devraient avoir, pour à la fois faciliter l'accès à l'information et fournir aux utilisateurs les informations nécessaires à une recherche critique.

2.5.1 Moteur de recherche Web

Un moteur de recherche Web peut tout à fait être considéré comme une bibliothèque numérique. Les ressources typiquement mises à la disposition des utilisateurs sont les pages de sites Web. Cela dit, depuis plusieurs années, les moteurs de recherche évoluent pour prendre en compte non seulement ces pages Web, mais aussi les différents médias qu'elles contiennent. Ainsi, les résultats de ces moteurs de recherche peuvent aujourd'hui contenir des pages Web, des documents PDF, des vidéos, des photos, *etc.* Pour pouvoir proposer ce genre de services, les moteurs de recherche doivent d'abord être capables d'indexer les ressources disponibles sur internet. Pour ce faire, ils utilisent des robots logiciels qui naviguent automatiquement de lien en lien à travers les pages internet pour découvrir de nouvelles ressources. Ce procédé permet d'agréger un très grand nombre de ressources. C'est évidemment un avantage pour les utilisateurs qui peuvent profiter de ces ressources documentaires pour répondre à leur besoin informationnel. Ce dernier est exprimé grâce à une requête qui peut prendre différentes formes. Le plus souvent, l'utilisateur exprime son besoin sous la forme de mots clés. Le moteur identifie alors les pages les plus pertinentes. Certains moteurs de recherche permettent également de formuler un besoin informationnel en langage naturel, avec une phrase grammaticalement structurée. Il est aussi possible d'utiliser une image comme requête. Le moteur de recherche renvoie alors à l'utilisateur une liste d'images similaires. Le calcul de pertinence est dans ce cas fondé sur la similarité des couleurs, des formes ou des textures.

Ces moteurs de recherche ont l'avantage d'être très généralistes et de pouvoir répondre à beaucoup de besoins différents. Cependant, la richesse des ressources disponibles est aussi un inconvénient. En effet, il n'est pas rare d'obtenir après une recherche plusieurs millions de résultats potentiellement intéressants. Les moteurs de recherche classent ces résultats selon un score de pertinence basé sur la similarité de la requête et des documents disponibles. Cependant, les utilisateurs n'ont

10. *Application Programming Interface*, interface permettant d'utiliser des fonctionnalités normalement accessibles par le Web grâce à un programme autonome.

généralement pas connaissance du détail de ces calculs. Lorsque l'objectif que souhaite atteindre un utilisateur après une recherche est un objectif personnel, l'opacité du fonctionnement des moteurs de recherche n'est pas nécessairement un problème. Cependant, lorsque ces moteurs sont utilisés à des fins plus professionnelles, il est important que l'utilisateur comprennent comment et pourquoi il a obtenu tel ou tel résultat. Dans le cas de Google, le moteur de recherche Web le plus populaire, il existe par exemple des partenariats commerciaux qui permettent à certains sites de se hisser en haut du classement des résultats d'une recherche. De la même manière, la plupart des moteurs de recherche implémentent des algorithmes qui mettent en avant les sites les plus influents (le *PageRank* par exemple, voir figure 2.2). Dans le cas d'une recherche d'information "banale" (la vérification d'un fait par exemple), ce fonctionnement n'est pas un problème. En revanche, pour des recherches plus ambigües (sur des sujets polémiques comme la politique par exemple), la mise en avant d'un point de vue particulier pose des questions de neutralité évidentes.

2.5.2 Bases de données en ligne

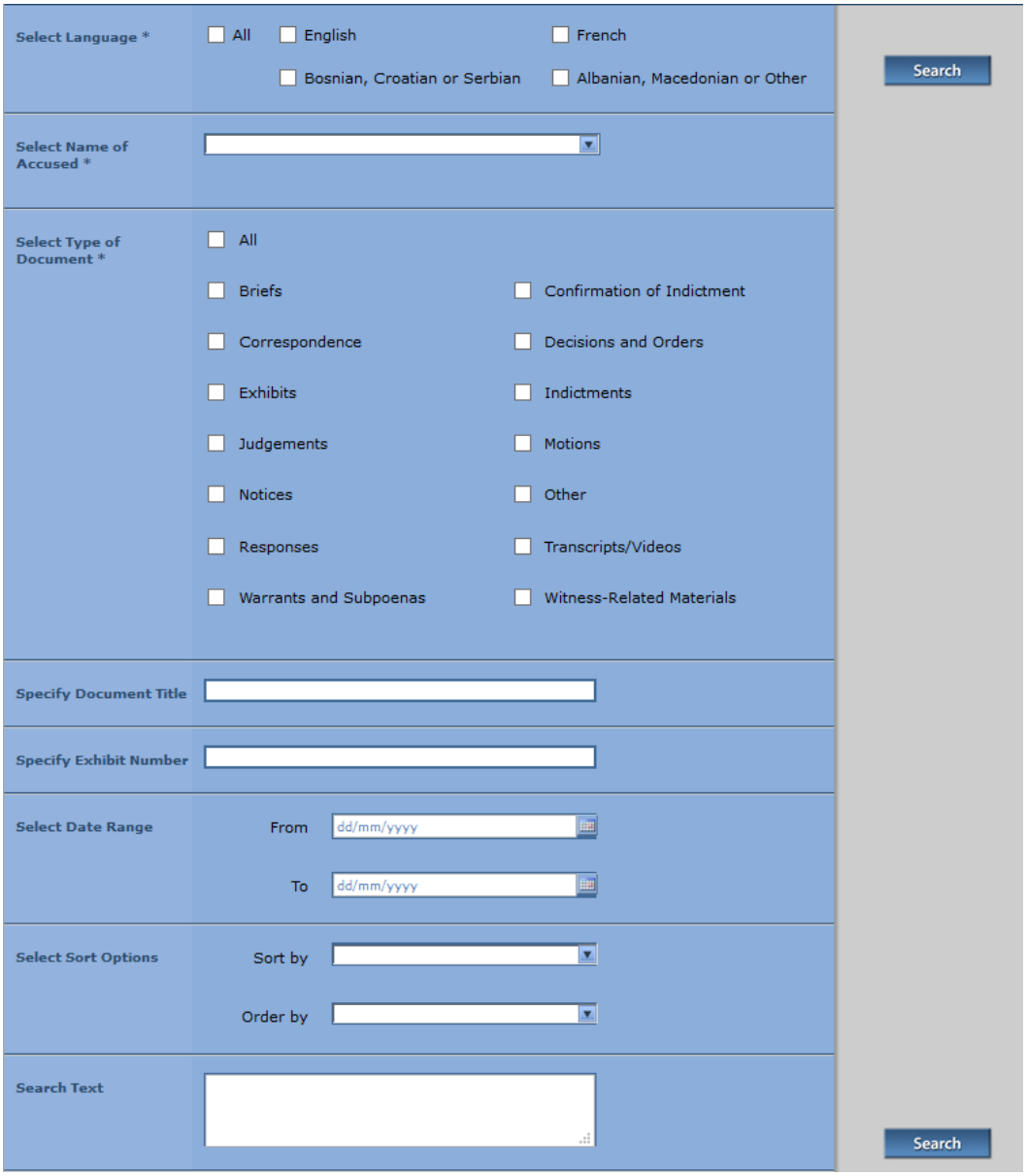
Si les sites Web sont une source d'information intéressante, ils sont particulièrement variés et il est souvent difficile d'identifier toutes les sources nécessaires à un travail de recherche. Lorsqu'un institut ou un organisme souhaite rendre disponibles des documents, la base de données en ligne est souvent la solution la plus simple à mettre en place. Il s'agit ici de proposer aux utilisateurs une interface en ligne permettant d'interroger une base de données "classique". Un exemple d'une telle interface est la base de données du Tribunal Pénal International pour l'ex-Yougoslavie [11] (TPIY) présentée dans la figure 2.7. Ce type de base de données souffre de plusieurs défauts. D'abord, il n'y a pas de documents "supports" associés aux documents de la base. Par exemple, dans le cas du TPIY, aucun texte de loi n'est disponible. Il n'y a pas non plus d'index permettant d'identifier rapidement les documents intéressants pour une recherche donnée. Il est ainsi nécessaire de déjà bien connaître la collection de documents en ligne pour être capable d'y rechercher de l'information. De plus, s'il est possible de rechercher les documents qui contiennent un mot ou une phrase particulière, cette recherche est très monolithique. Une simple faute de frappe peut ainsi mettre en péril la qualité des résultats obtenus. Finalement, les résultats après une recherche sont présentés de manière très succincte. Il y est par exemple impossible de savoir où dans les documents apparaissent les termes recherchés. Puisque les documents peuvent avoir une taille très variable (d'une à plusieurs milliers de pages dans le cas du TPIY), cela pose des problèmes d'accessibilité évidents.

2.5.3 Bibliothèques numériques

Il n'y a pas de définition unique de ce qu'est une bibliothèque numérique. Ce concept possède différentes significations selon le point de vue adopté [Borgman 1999]. Nous utiliserons ici la définition suivante : une bibliothèque numérique est un système d'information qui organise des collections de ressources numériques ainsi que différents services. Accessibles à distance, les informations mises

11. http://icr.icty.org

FIGURE 2.7 – Interface d'interrogation de la base de données du TPIY



Les différents champs que l'utilisateur peut remplir correspondent à ceux de la base de données (affaire, date, langue, ordonnancement, etc.). Il est particulièrement difficile de découvrir des documents que l'on ne connaît pas déjà.

à disposition par la plateforme peuvent être de différentes natures (ressources textuelles, audio, vidéos, *etc.*). Les différentes ressources peuvent être liées entre elles ou liées à des ressources externes fournies par un autre service. Les bibliothèques numériques ont trois objectifs principaux : organiser l'information, l'archiver et la

rendre disponible. Principalement utilisées à des fins d'éducation ou de recherche, les bibliothèques numériques permettent de répondre aux besoins d'une population d'utilisateurs.

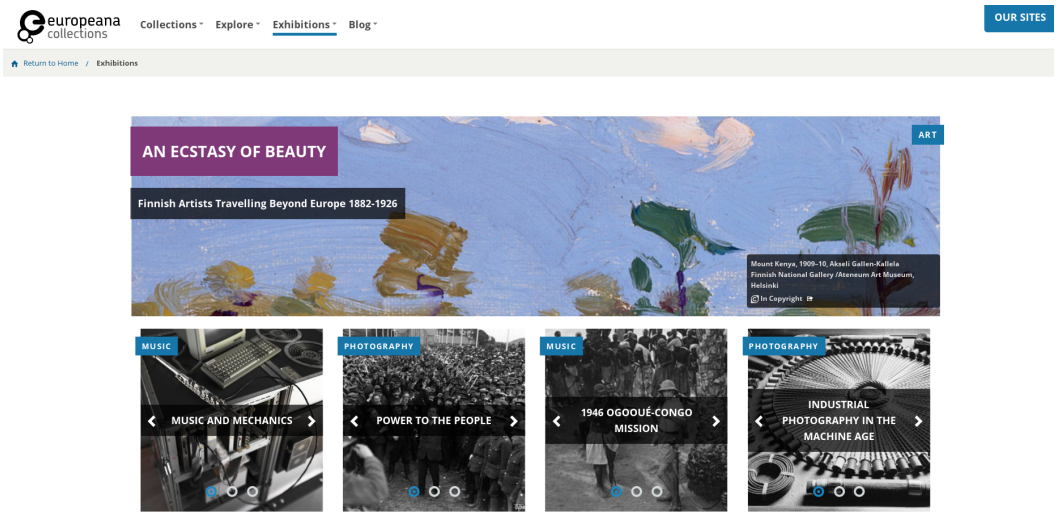
Ainsi, contrairement aux bases de données présentées dans la sous-section précédente, les bibliothèques numériques sont plus adaptées à la découverte d'information. Il n'est donc pas nécessaire d'être spécialiste pour consulter et découvrir les documents mis à disposition des utilisateurs. Une autre caractéristique des bibliothèques numériques est la multiplicité des sources. Ainsi, en plus de stocker certaines collections de documents, la plupart des bibliothèques numériques contiennent des collections distantes. C'est-à-dire qu'un utilisateur, après une recherche, se verra proposer des documents provenant non seulement de la bibliothèque numérique en question, mais aussi d'autres services (une autre bibliothèque numérique, des archives publiques, *etc.*). Les données de ces différentes sources sont agrégées grâce au "moissonnage". Les métadonnées (présentées dans la section 2.4) sont essentielles durant cette étape. En effet, les bibliothèques numériques ne stockent pas toujours l'ensemble des ressources qu'elles mettent à la disposition des utilisateurs. En revanche, elles proposent aux utilisateurs de rechercher de l'information grâce aux différentes métadonnées collectées sur les plateformes des différents partenaires. Une fois qu'un utilisateur est satisfait de sa recherche, il est redirigé vers le site permettant de consulter le document original si celui-ci n'est pas stocké dans la bibliothèque numérique. Nous présentons ici un bref échantillon de trois bibliothèques numériques. Bien que différentes entre elles, elles sont souvent riches en fonctionnalités. Nous présentons maintenant un échantillon de trois bibliothèques numériques différentes. Ces dernières sont destinées à différents utilisateurs et représentent donc un échantillon des plateformes disponibles en ligne.

2.5.3.1 Europeana

Co-financée par l'Union Européenne, cette bibliothèque numérique[12] met à la disposition du public un grand nombre d'œuvres d'Art, de livres, de musiques, *etc.* Plus de 50 millions d'objets y sont accessibles gratuitement. L'objectif principal de cette plateforme et de valoriser l'héritage culturel de l'Europe à des fins éducatives, de recherche ou pour le plaisir des utilisateurs. Europeana bénéficie de plus de 3000 partenariats avec des bibliothèques, des archives et des muséums à travers l'Europe. De plus, de nombreux projets dirigés par différentes institutions sont financés par l'Europe. De la mise à disposition de nouveaux contenus jusqu'au développement de nouvelles fonctionnalités, ces projets participent activement à l'évolution de cette plateforme (voir figure 2.8). Celle-ci fournit de nombreux moyens de filtrer les informations à travers les différentes métadonnées associées aux documents. Plusieurs critiques et remarques sont cependant émises dans [Erway 2009], et certaines d'entre elles sont toujours valides aujourd'hui. Il est tout d'abord nécessaire de noter que la fonction de classement du moteur de recherche a un impact considérable sur l'utilisabilité de cette bibliothèque numérique. En effet, avec plus de 50 millions d'objets, une requête se traduit souvent par un grand nombre de résultats renvoyés.

12. https://www.europeana.eu

FIGURE 2.8 – Expositions disponibles sur Europeana



Différents moyens peuvent être utilisés pour accéder aux documents de cette plateforme. Des expositions sont régulièrement créées par Europeana ou certaines institutions partenaires. Ces expositions virtuelles permettent aux utilisateurs de découvrir des œuvres dans un contexte particulier. Les différentes ressources mises en valeur dans l'exposition sont également consultables avec l'interface classique d'Europeana.

Europeana fonctionne de manière *top-down*. C'est-à-dire que l'accès à l'information commence avec une vue générale qui se précise au fur et à mesure des filtres utilisés. L'absence de formulaire de recherche avancée confirme ce fonctionnement. Il est possible d'ajouter des indications logiques aux requêtes grâce aux mots clés **AND**, **OR** et **NOT** (voir section 2.3.2). Il est possible d'interroger les métadonnées des documents grâce à différents préfixes lors d'une requête. Ainsi, la requête *what :peinture who :Van Gogh* renverra des documents dont les métadonnées indiquent ces informations. Il y a cependant toujours des problèmes d'accessibilité car beaucoup de documents ne contiennent pas ces métadonnées. De plus, elles sont parfois indiquées dans une langue mais pas dans une autre. Le type des documents est aussi parfois trompeur car les documents de type *Texte* sont parfois des textes, parfois des images de textes. S'il est possible de rechercher des documents selon la présence ou l'absence de texte associé, on ne peut y rechercher de l'information sans le télécharger. Finalement, il n'est pas possible d'éditer la requête courante durant une recherche d'information. Il est donc nécessaire de réécrire entièrement la requête, ce qui peut être fastidieux si elle est complexe.

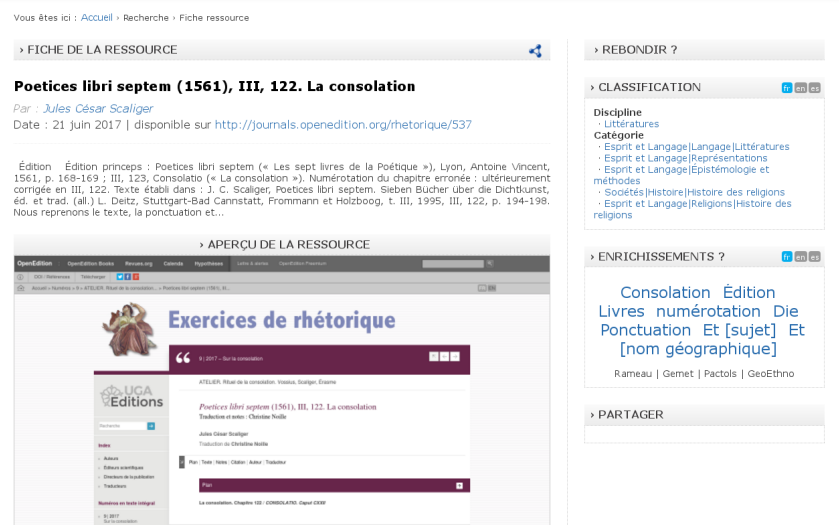
2.5.3.2 Isidore

Développée en 2009 par la Très Grande Infrastructure de Recherche (TGIR) Huma-Num, cette plateforme [13] indexe plus de 5 millions de ressources à destination

13. https://www.rechercheisidore.fr

des chercheurs en Sciences Humaines et Sociales. Comme Europeana, cette plateforme moissonne les données provenant de plus de 5000 institutions scientifiques. Les métadonnées et le texte intégral (s'il est disponible) des différents documents moissonnés sont analysés. Un algorithme basé sur une analyse morphologique [14] des différents termes permet d'identifier des équivalences avec les concepts indexés par des référentiels (principalement des thésaurus comme GeoNames ou Rameau). Ainsi, lorsqu'une telle équivalence est trouvée, la ressource est associée au concept du référentiel (voir figure 2.9).

FIGURE 2.9 – Page d'une ressource de la bibliothèque numérique Isidore



Différentes informations sont présentées à l'utilisateur. On peut retrouver des métadonnées classiques comme le titre, la date et l'auteur de la ressource. Sont également présentes les disciplines et catégories associées. On peut également observer les enrichissements automatiquement extraits ainsi que le référentiel associé.

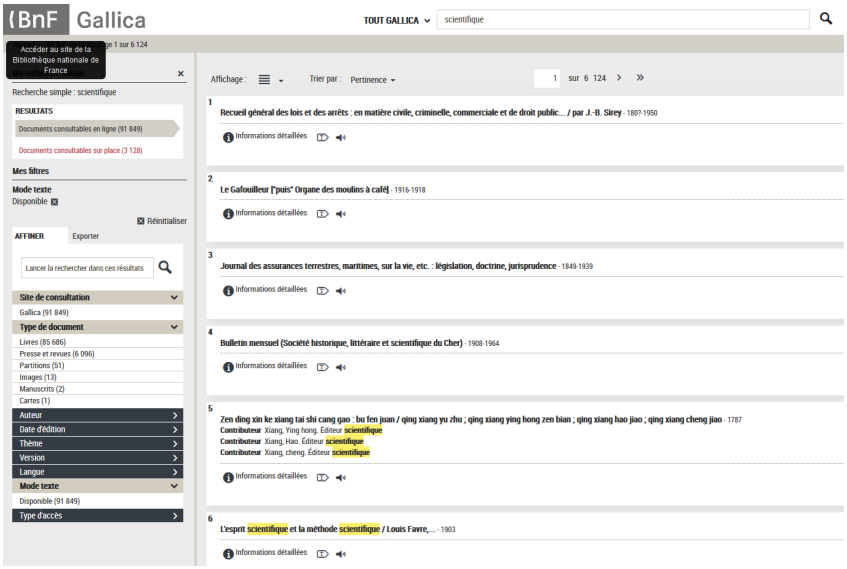
Cette fonctionnalité est cependant limitée car les algorithmes ne semblent se baser que sur l'apparition de certains termes. Les homonymies sont assez courantes et font donc apparaître des erreurs. Par exemple, un livre au sujet de Mao Zedong, fondateur de la République Populaire de Chine, est associé à l'expression "Informatique musicale" car M.A.O. est aussi un acronyme qui signifie "Musique Assistée par Ordinateur". Il est faut être vigilant lorsque l'on utilise de tels outils. S'il est possible de faire des recherches dans le texte complet des documents, l'exploitation des résultats est cependant parfois difficile. En effet, les extraits des documents qui contiennent les termes de la requête ne sont pas indiqués. L'utilisateur a donc parfois

14. La morphologie est l'étude de la forme des mots. On parle de morphologie interne lorsqu'on s'intéresse aux mots qui entretiennent certaines relations (*parleras* et *parlerons* ont tous les deux une valeur de futur par exemple). La morphologie externe se concentre sur les liens entre différents mots d'un lexique à travers l'étude de leur racine et des préfixes et affixes associés (-eur, -able, anti-, *etc.*).

besoin de parcourir le document entier pour retrouver l'information qui l'intéresse. Dans la liste des résultats, après une requête, de nombreuses valeurs différentes sont souvent disponibles pour chaque facette. Lorsque l'une d'entre elles est activée, cette information est généralement placée sur la page. Si le type des facettes activées est bien présent, leur valeur ne l'est pas. Cela ne facilite pas les recherches d'informations complexes où de nombreux filtres doivent être utilisés. Une fonctionnalité intéressante développée sur cette plateforme est la possibilité de "rebondir" après une recherche. Après une requête et l'accès à un document particulier, une fenêtre donne accès aux différentes métadonnées de ce document. L'utilisateur peut alors sélectionner les métadonnées qui l'intéressent et lancer une nouvelle recherche à partir de celles-ci.

2.5.3.3 Gallica

FIGURE 2.10 – Interface de recherche de Gallica



Gallica permet de rechercher de l'information dans un grand nombre de ressources principalement textuelles. La fonction de surbrillance permet d'identifier les mots de la requête utilisée dans les différents documents renvoyés.

Cette bibliothèque numérique [15] est maintenue par la Bibliothèque nationale de France (BnF). Mise en ligne en 1997, cette plateforme était à l'origine destinée à une utilisation hors-ligne, sur les terminaux de la bibliothèque. La démocratisation rapide du Web a cependant modifié le projet initial qui est maintenant un site accessible gratuitement sur le Web. Ce qui distingue Gallica des autres bibliothèques numériques présentées précédemment sont les différentes campagnes de numérisation mises en place. En effet, en plus d'indexer les ressources déjà numérisées par d'autres institutions, la BnF numérise activement ses différentes collections ainsi que celles

15. http://gallica.bnf.fr

Tableau 2.2 – Fonctionnalités de différentes bibliothèques numériques

	Europeana	**Isidore**	**Gallica**
Qualité des métadonnées	+	+	+
Recherche par facettes	+	+	+
Recherche dans le texte complet	-	+	+
Recherche des entités nommées	+	-	+
Formulaire de recherche avancée	-	-	+
Recherche thématique	+	-	+
Recommandation	+	+	-
Accès par API	+	+	+

de ses partenaires. Aujourd'hui, un tiers des documents numérisés chaque année provient d'autres bibliothèques que la BnF [16]. L'interface de recherche de Gallica est présentée dans la figure 2.10. Durant une recherche textuelle, les extraits des documents qui contiennent des termes de la requête sont mis en évidence dans la page des résultats. En cliquant dessus, l'utilisateur est redirigé vers la bonne page du document, où les termes sont là aussi mis en surbrillance. Lorsque le texte complet associé à un document a été extrait grâce à un processus automatique, un score de confiance est affiché.

Si ces trois bibliothèques sont différentes, elles possèdent toutes des fonctionnalités que l'on retrouve dans la plupart des plateformes de recherche en ligne. La présence ou l'absence (respectivement + et −) de certaines fonctionnalités sont présentées dans le tableau 2.2. Il convient d'adapter les outils disponibles selon le public visé par les créateurs d'une plateforme de recherche. Ainsi, la possibilité de recommander du contenu est souhaitable pour une plateforme grand public. Cependant, dans un contexte plus professionnel comme la recherche académique, les utilisateurs doivent pouvoir désactiver ces fonctions de recommandation. En effet, ces dernières entraînent des difficultés à partager la méthodologie de recherche employée et sont rarement reproductibles. Cela dit, les recommandations peuvent bien évidemment être pertinentes et il ne faut pas les rejeter systématiquement. Il faut toutefois être conscient des biais qu'elles peuvent entraîner.

2.6 Conclusion

Nous avons présenté dans ce chapitre les principales technologies numériques d'accès à l'information utilisées de nos jours. Nous avons d'abord effectué un rapide tour d'horizon de la conservation et de l'accès à l'information dans l'histoire de l'humanité. L'évolution de la technologie informatique en générale, et en particulier

16. A propos | Gallica http://gallica.bnf.fr/html/und/a-propos

de la recherche d'information, transforme les usages des utilisateurs. L'organisation, la recherche et la présentation de l'information deviennent des sujets clés car ils jouent un rôle capital dans l'interface entre le monde des humains et le monde numérique de l'information.

Les outils de recherche (utilisation des facettes, recherche dans le texte ou les métadonnées, *etc.*) sont, eux, au centre de cette interface. C'est pour cette raison que nous pensons qu'il est important de comprendre leur fonctionnement général. La notion de "pertinence" des résultats est une notion relativement abstraite. Elle est pourtant présente dans tous les moteurs de recherche qui renvoient une liste de résultats après une requête. Si les scores de pertinence sont issus de calculs complexes, il est toutefois possible d'en comprendre la logique.

Ces calculs prennent en compte les métadonnées, qui sont de nos jours aussi importantes que les données qu'elles représentent. Dates, créateurs, titres, *etc.* sont autant d'informations utiles à l'organisation et à l'accès aux documents. Les réseaux de métadonnées liées entre elles par différentes relations sont de plus en plus utilisés pour rapidement agréger des informations et en extraire de la connaissance.

Nous avons finalement présenté différents types de plateformes permettant d'accéder à de l'information sur le Web. Nous nous sommes attardés sur la description des bibliothèques numériques. En effet, souvent riches en contenu, ces plateformes rencontrent de nouvelles problématiques de recherche et de mise à disposition de l'information. Nous avons défini et identifié les principales caractéristiques d'une bibliothèque numérique, afin de comprendre leur impact sur l'utilisateur et ses recherches.

Ce chapitre établit un socle de connaissances nécessaire à la compréhension des systèmes d'information. Dans le prochain chapitre, nous discuterons des différents biais de recherche auxquels sont confrontés les utilisateurs. En effet, les technologies informatiques évoluent rapidement et sont de plus en plus populaires auprès du public. Un décalage se crée donc entre les capacités techniques des programmes et algorithmes, et ce que les utilisateurs comprennent de leur fonctionnement. Ce décalage a un impact négatif sur l'expérience des utilisateurs. Nous tenterons donc d'y remédier en identifiant les principaux biais de recherche auxquels sont confrontés les utilisateurs.

Contributions

- Nous avons décrit le fonctionnement des moteurs de recherche classiquement utilisés sur le Web, ainsi que le rôle toujours plus important des métadonnées.
- Nous avons présenté les avantages et inconvénients de plusieurs types de plateformes de recherche d'information.
- Nous avons identifié les fonctionnalités importantes des bibliothèques numériques, ainsi que leur rôle dans la recherche d'information.

Chapitre 3

Réduire le fossé entre les utilisateurs et les données

Sommaire

3.1 Introduction 35
3.2 Plateforme d'étude 37
3.2.1 Technologies utilisées 37
3.2.2 Modélisation des données 39
3.2.3 Présentation du fonctionnement 42
3.3 Ce que le système comprend des utilisateurs 44
3.3.1 Modèles de comportement pour la recherche d'information . . 44
3.3.2 Observation des utilisateurs 45
3.3.2.1 Actions à observer 46
3.3.2.2 Implémentation 47
3.3.2.3 Indicateurs comportementaux 48
3.3.3 Expérimentations 48
3.3.3.1 Contexte et contraintes de l'étude 49
3.3.3.2 Résultats 49
3.4 Ce que les utilisateurs comprennent du système 53
3.4.1 Biais méthodologiques 53
3.4.2 Fonctionnement et implémentation 55
3.5 Conclusion 59

3.1 Introduction

Les bibliothèques numériques jouent le rôle d'interface entre les utilisateurs et les documents. Il existe deux principales interactions qui régissent les processus de recherche d'information. D'un côté, les utilisateurs échangent avec la bibliothèque numérique pour exprimer leurs besoins durant toute la durée de la recherche. Dans le même temps, le système de recherche évalue les différentes actions de l'utilisateur et tente de répondre à ses besoins. Le principal problème avec ce paradigme est que le système de recherche ne parle pas le même langage que l'utilisateur et qu'il ne conserve pas les informations de la même manière. Le système est donc chargé de traduire les différentes requêtes et de mettre en évidence les documents susceptibles de satisfaire le besoin des utilisateurs. Ces différences font apparaître de nombreux

biais qui peuvent à terme affecter la qualité de la recherche d'un utilisateur. L'objectif de ce chapitre est de lister ces biais et de mettre en place différents scénarios permettant de limiter leur impact.

Nous avons tous, en tant qu'êtres humains, un schéma mental qui nous permet d'appréhender les informations que l'on rencontre. Ce schéma est susceptible d'évoluer selon la situation dans laquelle on se trouve. Les systèmes d'information ont, pour leur part, un schéma fixe déterminé par leurs concepteurs. Les moyens mis en œuvre par les utilisateurs et par le système de recherche pour appréhender de l'information ne sont pas les mêmes. Cette différence crée un fossé sémantique qui, pour être franchi, nécessite un effort d'adaptation de la part des utilisateurs. Pour améliorer l'usage que font les utilisateurs des systèmes de recherche, il est nécessaire de réduire le fossé sémantique qui existe entre eux. Nous avons pour cela deux principaux moyens d'action :

— améliorer la capacité des systèmes de recherche d'information des bibliothèques numériques à comprendre les besoins exprimés par un utilisateur. À travers ses actions dans le système, l'intention de l'utilisateur doit devenir de plus en plus claire. Satisfaire son besoin d'information devient alors plus facile ;

— former les utilisateurs aux mécanismes de base qui influencent la qualité d'une recherche. Grâce à une meilleure connaissance des différentes manipulations que le système de recherche opère pour répondre à une requête, les utilisateurs pourront adapter leur comportement afin d'exploiter au mieux les capacités du système de recherche.

Afin d'améliorer l'accessibilité des documents d'une plateforme et de faciliter la recherche d'information, il y a besoin d'une véritable coopération entre les systèmes d'information et leurs usagers, entre les algorithmes et les utilisateurs. Pour résumer, l'accessibilité des documents est influencée (positivement ou négativement) par plusieurs facteurs qui proviennent soit des systèmes d'information, soit des utilisateurs.

+ Systèmes d'information :
  - Capacité du système à comprendre les besoins de l'utilisateur
  - Outils de recherche disponibles et fonction de pertinence du moteur de recherche
  - Hétérogénéité du corpus
+ Utilisateurs :
  - Capacité de l'utilisateur à interroger le système et parcourir les résultats
  - La quantité de résultats que l'utilisateur est disposé à parcourir
  - Connaissance du corpus

Ce chapitre est organisé en trois sections principales. Avant toute chose, nous présentons dans la section suivante la plateforme expérimentale que nous avons développée et qui nous servira tout au long du chapitre. Nous nous intéressons ensuite à deux problématiques particulières. D'abord, nous étudions la capacité qu'ont les systèmes de recherche à comprendre les intentions des utilisateurs et à s'adapter en conséquence. Puisque les utilisateurs ont des besoins d'information et des pratiques de recherche différentes, nous pensons qu'il est important de les observer

afin de comprendre leur intention et leur méthodologie de recherche. Ces différentes informations peuvent par exemple être utilisées pour ajuster les paramètres de la bibliothèque numérique afin de répondre au mieux aux besoins des utilisateurs. Nous nous intéressons enfin aux utilisateurs et à l'idée qu'ils se font du fonctionnement d'un système de recherche. La représentation mentale qu'ils ont de ces systèmes est influencée par l'opacité des différents algorithmes et processus utilisés. Nous pensons qu'il est important de former les utilisateurs à comprendre le fonctionnement global des systèmes de recherche pour améliorer leur expérience.

3.2 Plateforme d'étude

Nous démontrons dans ce chapitre qu'il est possible d'améliorer les interactions entre les utilisateurs et les systèmes de recherche en ligne. Pour cela, nous avons besoin de tester différentes idées et de les expérimenter avec de vrais utilisateurs. Nous avons donc développé une bibliothèque numérique expérimentale qui facilite l'étude du comportement des utilisateurs. Semblable à la plupart des bibliothèques numériques, cette plateforme rassemble une grande partie des outils de recherche utilisés par les utilisateurs. Nous décrivons ici les moyens techniques mis en place pour son développement.

3.2.1 Technologies utilisées

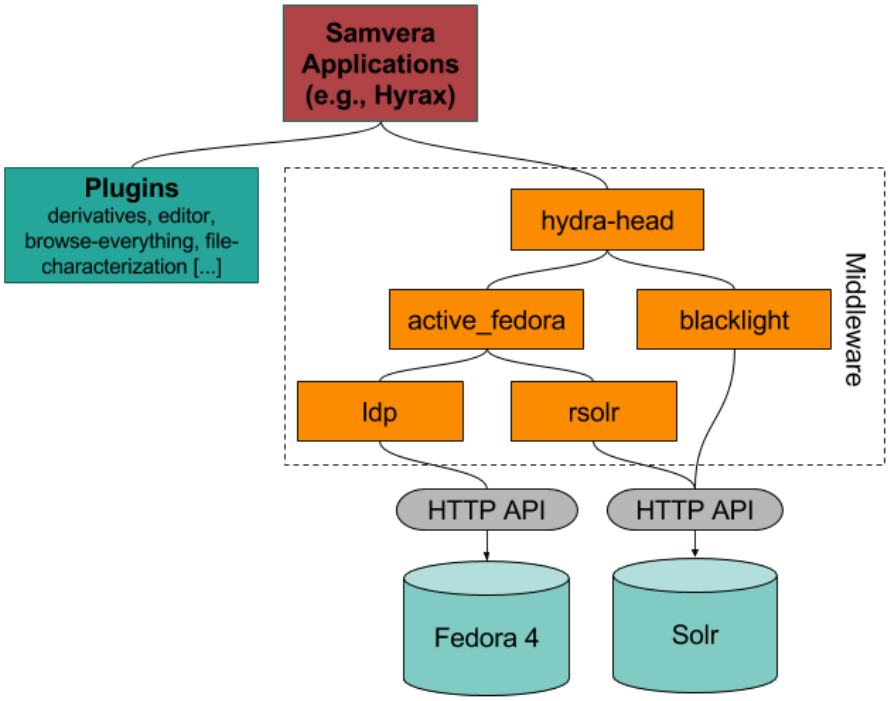
Nous voulons intégrer dans notre plateforme d'étude la majorité des caractéristiques que l'on peut trouver dans la plupart des bibliothèques numériques (voir section 2.5.3). Notre choix s'est rapidement porté sur Samvera [1], une solution open source qui se définit comme un dépôt de contenus numériques. Cette solution rassemble plusieurs technologies qui permettent de gérer la mise en production d'une bibliothèque numérique (voir figure 3.1).

Stockage des données C'est le dépôt de données open source Fedora [2] qui est chargé de pérenniser le stockage des documents et des métadonnées. Des fichiers dérivés sont souvent nécessaires pour le bon fonctionnement d'une bibliothèque numérique (miniatures d'images, vidéos à différentes résolutions, sous-titres, *etc.*). Ces fichiers dérivés sont eux aussi stockés dans Fedora. Cet outil fournit des API permettant de faciliter l'accès aux données et la dissémination des documents. L'ensemble des données et métadonnées sont accessibles par une interface Web, ou grâce à la présence d'API. Les données sont téléchargeables sous différents formats comme XML ou RDF.

Indexation et recherche Pour être exploitable, les différents documents et métadonnées stockés dans Fedora doivent être indexés. C'est le rôle de SolR [3], une

1. http://samvera.org/
2. http://fedorarepository.org/
3. http://lucene.apache.org/solr/

FIGURE 3.1 – Architecture logicielle utilisée pour le développement de notre plateforme d'étude



Les différents documents et les index sont respectivement stockés dans Fedora et SolR. Plusieurs composants intermédiaires fournissent des fonctionnalités d'accès, de recherche et une interface basique. La plateforme que nous avons développée fait usage de tous ces éléments. Diagramme réalisé par samvera.org / CC-BY

plateforme de recherche open source basée sur les technologies de recherche et d'indexation d'Apache Lucene. SolR à l'avantage d'être hautement configurable et de s'adapter à de gros volumes de trafic grâce à un mécanisme de mise en cache des requêtes et des résultats précédents. Il offre aussi de nombreuses possibilités de recherche avancée (facettes, mise en surbrillance des résultats, suggestions, *etc.*). Comme Fedora, il est accessible par le biais d'une API.

Composants intermédiaires Développés en Ruby, ils jouent différents rôles dans le fonctionnement de la plateforme :

— **ldp** est chargé de manipuler les différents documents et métadonnées stockés dans Fedora. Ce composant suit la recommandation du W3C *Linked Data Platform* qui définit un ensemble de spécifications notamment pour la manipulation de graphe RDF ;

— **rsolr** est un client qui permet de manipuler SolR à travers l'utilisation de plusieurs fonctions ;

— **active_fedora** est l'équivalent du patron de conception ActiveRecord pour

les applications Ruby on Rails (applications Web développées en Ruby). Ce dernier permet d'automatiquement faire le lien entre des objets manipulés dans des scripts ou des programmes et des tables de bases de données. ActiveFedora réalise la même chose mais pour les données stockées dans Fedora. Ce composant gère ainsi la création et l'indexation de nouveaux documents ;

— **blacklight** est un composant qui fonctionne avec SolR et qui fournit aux utilisateurs de nombreux moyens de découverte d'information au travers d'une interface Web (barre de recherche, facettes, tri et export des résultats, *etc.*) ;

— **hydra-head** permet de faire le lien entre les différents composants de l'architecture logicielle de Samvera. Ce composant possède aussi d'autres fonctionnalités nécessaires au bon fonctionnement d'une bibliothèque numérique (gestion des droits d'accès par exemple).

C'est cette pile logicielle que nous avons choisie pour répondre à notre besoin de bibliothèque numérique expérimentale. On peut résumer son fonctionnement grâce à trois éléments principaux qui interagissent ensemble :

— le **cœur du système** est chargé de l'interface entre les utilisateurs et les données, traite les différentes requêtes de l'utilisateur et lui présente les résultats ;

— dans le cas d'une recherche d'information, le système échange avec **SolR** pour récupérer les résultats ;

— quand il est nécessaire d'afficher des documents ou des métadonnées, c'est **Fedora** que le système contacte.

3.2.2 Modélisation des données

L'organisation des données au sein des systèmes d'information, en particulier les bibliothèques numériques, est un point crucial du développement. Que ce soit pour l'archivage ou la mise à disposition d'information, ces modèles supportent l'ensemble des données du système (utilisateurs, documents, métadonnées, *etc.*). Les flux de données toujours plus importants de nos jours amènent de nouvelles contraintes que ces modèles doivent pouvoir supporter :

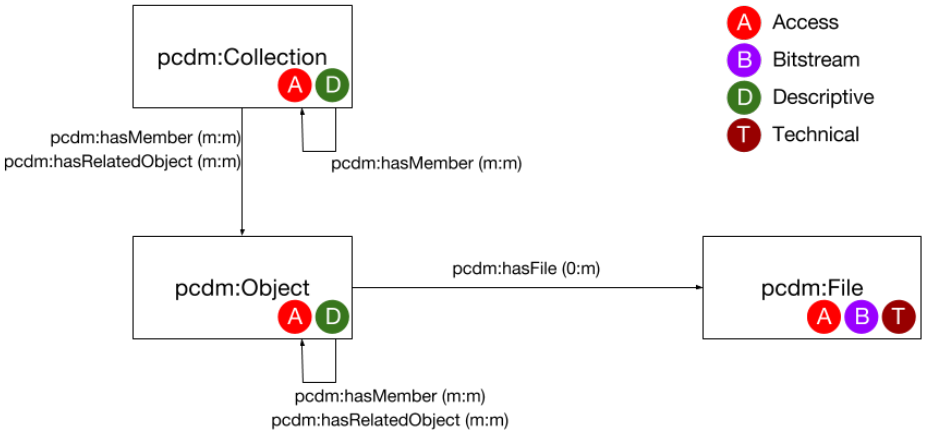
— le nombre de documents disponibles augmente régulièrement. Les modèles de données doivent pouvoir gérer cette quantité d'information tout en assurant sa disponibilité et son accessibilité ;

— dans les bibliothèques numériques, les documents sont souvent de natures très diverses : images, vidéos, documents textuels, *etc.* Les modèles doivent être suffisamment génériques pour pouvoir traiter tout type de document ;

— dans certaines bibliothèques numériques, les utilisateurs ont la possibilité d'activement contribuer aux ressources en annotant les documents (transcription de textes, identification des thèmes abordés, *etc.*). Ces données sont évolutives et les documents eux-mêmes peuvent changer. En effet, il arrive que des documents soient numérisés plusieurs fois, suivant les évolutions technologiques.

Ces contraintes impliquent qu'un bon modèle soit suffisamment généraliste pour satisfaire le plus de cas possibles, tout en étant hautement personnalisable. Dans certains cas, un modèle très simple peut suffire. Dans d'autres, il peut être nécessaire

FIGURE 3.2 – Organisation des données dans le modèle conceptuel PCDM



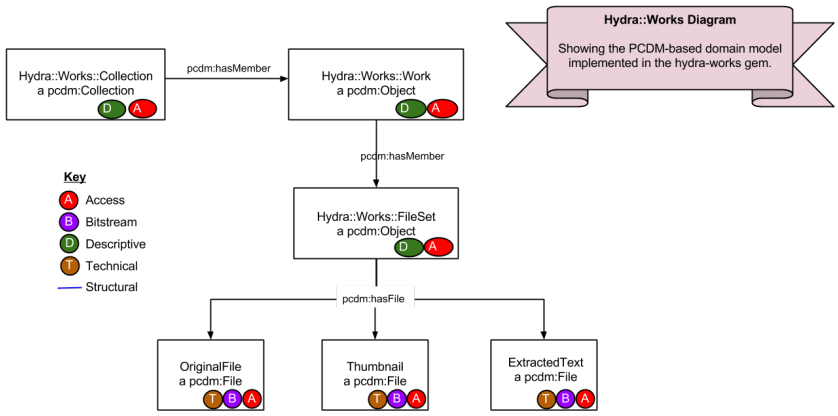
Trois classes principales peuvent être utilisées comme base pour une architecture plus complexe. Ces classes sont liées entre elles par des métadonnées qui peuvent être exprimées en RDF. Image extraite de la page github du projet ***https://github.com/duraspace/pcdm/wiki*** *.*

de représenter de complexes hiérarchies de données liées. L'architecture PCDM [4] répond exactement à ces besoins. Celle-ci est illustrée sur la figure 3.2. Ce modèle encourage l'utilisation des données liées et supporte ainsi le RDF. Trois classes principales sont définies. Tout d'abord, la classe *pcdm:Object* représente une entité de la bibliothèque numérique. Par exemple, un album de musique pourrait être considéré comme un objet. Cette classe peut elle-même contenir d'autres *pcdm:Object* grâce à la propriété *pcdm:hasMember*. Notre album de musique contient plusieurs pistes, pouvant chacune être représentée par un *pcdm:Object*. Il est aussi possible de définir une notion d'ordonnancement grâce à d'autres propriétés. Cette caractéristique serait applicable à notre exemple d'album musical. La classe *pcdm:Collection* est semblable à la classe *pcdm:Object*. Elle ajoute cependant un niveau d'abstraction en permettant de non seulement rassembler des objets mais également des groupes d'objets. Elle pourrait par exemple permettre d'organiser plusieurs albums musicaux par auteur ou par genre. Les classes *pcdm:Object* et *pcdm:Collection* sont toutes les deux associées à des métadonnées descriptives et d'accès. Finalement, la classe *pcdm:File* permet de représenter les fichiers numériques associés aux différents objets de la bibliothèque. Accompagné de métadonnées techniques, chaque *pcdm:File* doit être contenu dans exactement un *pcdm:Object*.

La solution que nous avons choisie implémente cette architecture, mais en précise certains aspects. Une ressource numérique est souvent constituée de plusieurs fichiers dérivés. Si on reprend notre exemple d'album musical, il peut être intéressant d'associer à chaque chanson une transcription. Pour pouvoir associer un auteur à cette

4. *Portland Common Data Model*, https://pcdm.org/

FIGURE 3.3 – Organisation des données dans le modèle conceptuel Hydra Works



Le modèle Hydra Works étend certaines classes du modèle PCDM pour permettre une meilleur gestion des contenus composites tels que des documents composés de plusieurs pages par exemple.

transcription, le modèle PCDM doit être légèrement complété. L'implémentation que nous avons choisie est *Hydra Works*. Développée en Ruby, elle est compatible avec Fedora, le dépôt de données que nous utilisons. Son schéma représentatif apparaît dans la figure 3.3. Dans celui-ci, deux nouveaux concepts sont ajoutés : *Work* et *FileSet*. Ces derniers étendent les fonctionnalités de la classe *pcdm:Object* du modèle PCDM. L'exemple donné dans la figure est celui d'un document transcrit. Il est en effet intéressant de rassembler sous une même identité :

— le document original (un document PDF par exemple) ;

— une ou plusieurs miniatures destinées à l'affichage ;

— la transcription du document.

Nous avons finalement choisi d'utiliser le format RDF pour la modélisation des données qui seront intégrées dans la plateforme lors des expérimentations. Les vocabulaires utilisables sont nombreux, mais nous avons uniquement intégré les plus populaires :

— DublinCore pour toutes les métadonnées générales (titre, auteur, *etc.*) ;

— MODS [5] qui permet de représenter des métadonnées plus précises (date de création de la ressource, date de mise en ligne, informations structurelles du contenu du document, *etc.*) ;

— FOAF [6] qui décrit des personnes.

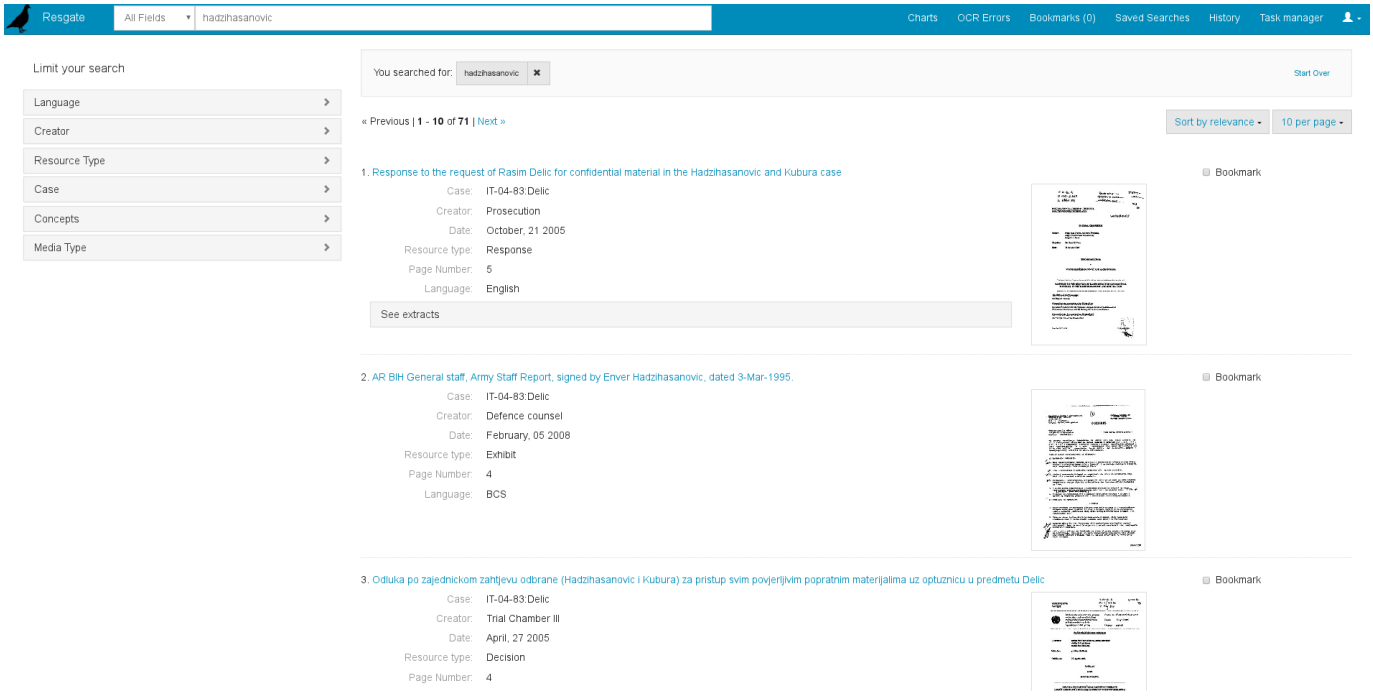
5. *Metadata Object Description Schema* – http://www.loc.gov/standards/mods/
6. *Friend Of A Friend* – http://xmlns.com/foaf/spec/

Des vocabulaires additionnels sont certainement nécessaires dans un projet à destination du grand public. Cependant, dans notre contexte expérimental, nous avons estimé que ces trois vocabulaires étaient amplement suffisants.

3.2.3 Présentation du fonctionnement

L'objectif de la plateforme que nous avons développée est de servir de support pour expérimenter différents aspects des interactions entre utilisateurs et bibliothèques numériques. Pour que ces expériences soient le plus réaliste possible, cette plateforme doit posséder des caractéristiques qui se retrouvent classiquement dans la plupart des bibliothèques numériques.

FIGURE 3.4 – Résultats affichés par le système après une requête

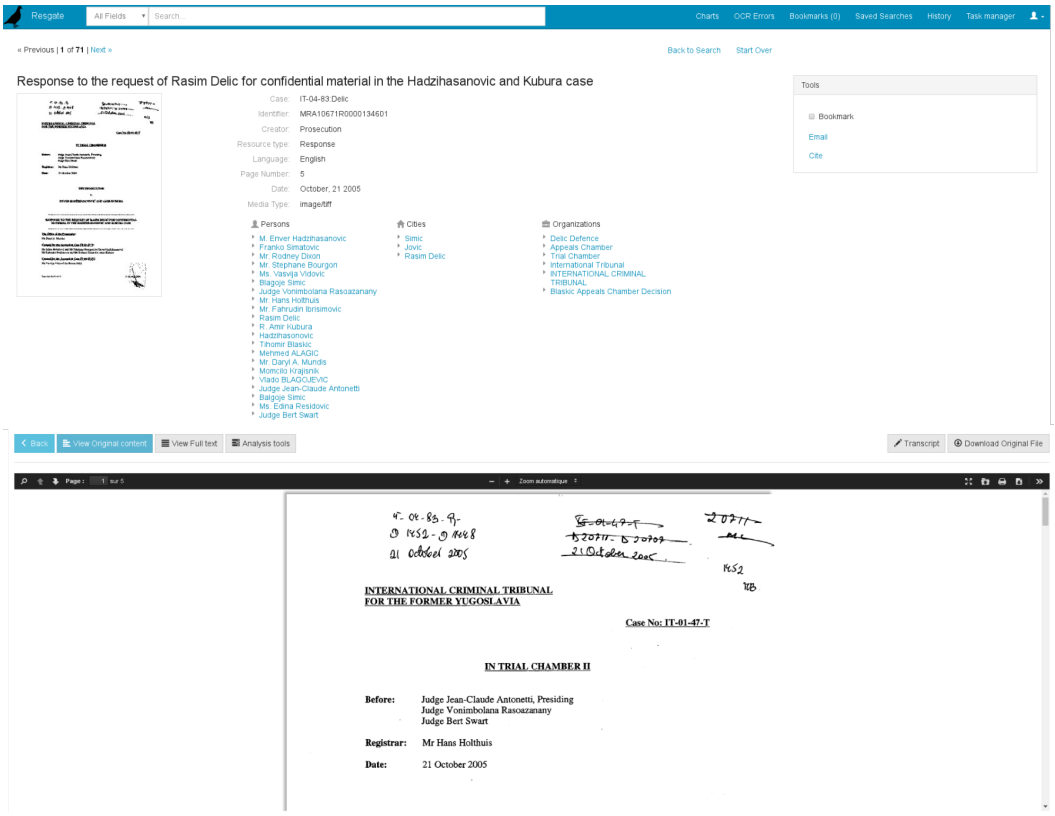


Les facettes permettant de filtrer les résultats sont sur la gauche de l'interface. Une miniature et les métadonnées principales sont présentées pour chaque résultat. Lorsque les mots clés de la requête ont été identifiés dans le texte complet du document, l'utilisateur peut visionner les extraits associés.

Le premier élément important de notre plateforme est la présence d'une barre de recherche, que l'utilisateur pourra utiliser pour saisir une requête textuelle. À côté de cette barre se trouve une liste déroulante qui permet de préciser sur quels champs cette requête sera exécutée. Par défaut, tous les champs sont sélectionnés, mais il est possible de contraindre la recherche aux titres, aux auteurs, au texte complet, *etc.* La page qui présente les résultats d'une requête est la SERP (*Search Engine Result Page*). Un exemple est donné dans la figure 3.4. Les résultats d'une requête

peuvent être filtrés grâce à l'utilisation de facettes sur les différentes métadonnées des documents. Les facettes peuvent aussi être utilisées directement, sans préciser de requête textuelle. Cette fonctionnalité facilite la découverte d'information lorsque l'utilisateur n'a pas une idée précise de ce qu'il recherche. Dans la SERP, l'utilisateur peut choisir de modifier l'ordre des résultats affichés. Par défaut, ces derniers sont triés selon leur pertinence par rapport à la requête de l'utilisateur (pertinence calculée par le système selon la configuration de SolR). Celui-ci peut toutefois modifier cet ordre (pour trier les résultats par auteur notamment) et choisir le nombre d'éléments affichés.

FIGURE 3.5 – Page de visualisation d'une ressource



La page d'une ressource permet de consulter les différentes métadonnées associées. Le document original est accessible, ainsi que le texte automatiquement extrait s'il est disponible.

L'utilisateur peut accéder au détail d'un contenu particulier en cliquant sur l'un des résultats de la SERP. Dans ce cas, la page de la ressource est affichée. Elle contient l'ensemble des métadonnées associées et permet à l'utilisateur de consulter les différents fichiers qui composent la ressource (document original, transcription automatique ou manuelle, *etc.*). Cette page est présentée dans la figure 3.5. Elle peut être configurée pour ajouter des fonctionnalités comme la recommandation d'autres documents ou la liste des entités nommées découvertes dans le texte du document ou

des métadonnées. L'accès au document original de la ressource est proposé depuis cette page. Le système choisit automatiquement le type de visionneuse à utiliser selon le document (PDF, audio, vidéos, *etc.*).

Finalement, la plateforme permet aux utilisateurs de mettre en favoris les documents qui les intéressent et d'avoir accès aux précédentes recherches qu'ils ont effectuées. Le principal aspect de cette plateforme qu'il faut retenir est sa capacité d'évolution. Selon les expériences qui doivent être menées, des fonctionnalités peuvent être ajoutées ou supprimées. Dans la section suivante, nous discuterons de la nécessité d'observer le comportement des utilisateurs. Notre méthodologie d'observation sera illustrée par une expérience réalisée avec des étudiants de Master en Histoire.

3.3 Ce que le système comprend des utilisateurs

Pour améliorer l'expérience des utilisateurs avec un outil de recherche d'information comme une bibliothèque numérique, il est important de comprendre les différentes interactions qui existent entre ces deux acteurs. Nous avons tous, en tant qu'humains, des idées préconçues du monde qui nous entoure. Chaque utilisateur d'une bibliothèque numérique a ainsi sa propre vision des données et de leur organisation. Les interfaces de recherche, au contraire, sont souvent statiques et ne proposent qu'une seule vision des données : celle des concepteurs de la plateforme. Le problème est donc de réussir à concilier les attentes des utilisateurs et les possibilités offertes par les bibliothèques numériques.

3.3.1 Modèles de comportement pour la recherche d'information

Les différents comportements adoptés par les utilisateurs durant une recherche d'information sont étudiés depuis plusieurs années. La recherche dans ce domaine s'est longtemps concentrée sur l'analyse des besoins informationnels des utilisateurs. Cependant, dans [Wilson 1981], l'auteur atteste que ce "besoin" n'est pas directement observable et qu'il est donc difficile de l'analyser. Selon lui, il convient plutôt de se concentrer sur le comportement et les actions des utilisateurs qui sont, eux, observables. Plusieurs modèles ont été développés pour tenter de décrire les différentes étapes qui régissent le comportement d'une personne engagée dans une tâche de recherche d'information. Ces modèles détaillent plus ou moins les différents comportements et actions des utilisateurs.

Modèle de Wilson Développé dans [Wilson 1981], ce modèle est né du besoin de rapprocher le monde des utilisateurs et des systèmes informatiques. C'est le premier à s'intéresser aux raisons qui poussent les utilisateurs à agir d'une certaine manière lorsqu'ils sont face à un besoin informationnel. Wilson étudie les raisons qui poussent un utilisateur à s'engager dans une recherche d'information et le rôle qu'y joue son environnement (l'usage de technologies, sa vision du monde, *etc.*).

Modèle de Ellis Ce modèle présenté dans [Ellis 1997] définit un ensemble d'activités qui peuvent être entreprises pour réaliser une tâche de recherche d'information. Ainsi, une tâche de recherche commence par l'identification de sources

initiales. Plusieurs stratégies sont alors envisageables. L'approche descendante est de sélectionner de larges domaines d'intérêt et de préciser la recherche au fur et à mesure. Au contraire, un utilisateur avec une approche ascendante effectuera d'abord une recherche précise avant de généraliser si les résultats obtenus ne lui conviennent pas. Les sources sont ensuite triées pour ne conserver que les plus pertinentes. Finalement, les informations nécessaires sont extraites des différentes sources sélectionnées.

Modèle du butinage d'information Ce modèle développé dans [Pirolli 1999] se base sur des observations éthologiques. Selon les auteurs, le comportement d'un utilisateur à la recherche d'information est très similaire à celui d'un animal à la recherche de nourriture. De la même manière qu'un animal identifiera différentes pistes de recherche grâce à des odeurs par exemple, un utilisateur engagé dans une tâche de recherche d'information utilisera différents indices visuels (mots clés, images, *etc.*) pour trouver ce qui l'intéresse.

Principe du moindre effort L'article de Donald Case, présenté dans [Fisher 2005], âffirme que ce principe joue un rôle dans les tâches de recherche d'information. Selon lui, plusieurs expériences empiriques ont déjà démontré la réalité de ce principe dans le comportement des utilisateurs, en particulier sur les plateformes numériques. En effet, l'acteur d'une recherche d'information fait toujours un compromis entre le temps qu'il est prêt à investir dans sa recherche et la qualité attendue des résultats.

Modèle de Marchionini Développé dans [Marchionini 2006], ce modèle simple définit deux principaux types de tâches qu'un utilisateur peut souhaiter réaliser lors d'une recherche d'information. Les tâches de recherche "simple" (*lookup* en anglais) et les tâches exploratoires. Les premières correspondent à des besoins précis comme trouver la date de naissance d'une personnalité, vérifier l'existence d'un document, *etc.* Les deuxièmes sont plus complexes et nécessitent souvent plus d'investissement de la part des utilisateurs (préparation d'une présentation orale, acquisition de connaissances dans un domaine particulier, *etc.*).

Tous ces modèles décrivent différents comportements que peuvent adopter des utilisateurs durant une tâche de recherche d'information. Ces modèles peuvent nous aider à comprendre certains aspects du comportement des utilisateurs en observant les actions qu'ils réalisent sur l'interface graphique. Mieux comprendre leur comportement peut nous permettre d'identifier les avantages et les inconvénients d'une bibliothèque numérique. L'objectif est, à terme, de développer une plateforme capable de comprendre les besoins informationnels d'un utilisateur et d'y répondre précisément. L'exploitation des différents modèles présentés nécessite cependant de disposer de données d'observation.

3.3.2 Observation des utilisateurs

L'objectif de la plateforme que nous avons développée est de servir de support expérimental pour comprendre les interactions entre utilisateurs et systèmes de recherche. Lorsque l'on souhaite analyser le comportement d'un utilisateur qui navigue dans une bibliothèque numérique, il est nécessaire de définir précisément les

différents événements à enregistrer.

3.3.2.1 Actions à observer

Les différents événements que nous souhaitons observer concernent différents aspects de la plateforme. Nous avons défini trois types d'actions différents.

Expression du besoin Enregistrer la manière dont l'utilisateur exprime son besoin d'information est crucial pour comprendre et tenter d'expliquer les actions subséquentes.

- *Requête textuelle (query)* : composée de mots clés ou exprimée en langage naturel, cette requête est un des points d'entrée vers les documents de la plateforme.
- *Facettes (facets)* : elles peuvent être utilisées après avoir exprimé une requête textuelle pour filtrer les résultats, ou directement pour entamer une recherche descendante (voir le Modèle de Ellis présenté dans la sous-section précédente).

Exploitation des résultats de recherche Une fois la requête traitée par le système, une liste de résultats est présentée à l'utilisateur. Les différentes actions qu'il est susceptible de réaliser sont les suivantes :

- *Liste des résultats (displayed)* : lorsque le besoin de l'utilisateur a été interprété par le système, ce dernier renvoie une liste de résultats qu'il considère pertinents ;
- *Ressources visibles (visible)* : très souvent, le système renvoie bien plus de résultats que ne peut en afficher l'écran. Nous tenons donc à enregistrer les documents effectivement affichés sur l'écran de l'utilisateur. Certains d'entre eux ne peuvent l'être que si l'utilisateur fait défiler la page ;
- *Accès à une ressources de la liste (accessed)* : nous gardons évidemment une trace des ressources qui ont intéressé l'utilisateur et qui s'est matérialisée par un accès à la page détaillée de la ressource ;
- *Accès à une ressource suggérée (recommended)* : il est possible de configurer la plateforme pour qu'elle recommande de nouveaux documents susceptibles d'intéresser l'utilisateur. Lorsque ce dernier accède à un document de cette manière, nous l'enregistrons ;
- *Sauvegarde d'une recherche (saved search)* : notre plateforme, comme la plupart des bibliothèques numériques offre la possibilité de sauvegarder les paramètres d'une recherche. Si l'utilisateur choisit de le faire, nous en gardons une trace.

Exploitation des ressources Une fois que l'utilisateur accède à une ressource, plusieurs choix s'offrent à lui. Nous gardons une trace des différentes actions qu'il peut réaliser :

— *Mise en favoris (bookmarked)* : l'utilisateur a la possibilité de mettre en favoris certaines ressources qui l'intéressent particulièrement. Comme les autres observations, cette trace est contextualisée dans une recherche et une session ;

— *Consultation de l'original (original)* : l'utilisateur a la possibilité, à partir de la page d'une ressource, d'accéder au document original ;

— *Consultation de la transcription (fulltext)* : certains documents ont pu bénéficier d'une extraction de texte manuelle ou automatique. Lorsque ce texte est disponible, l'utilisateur peut le consulter ;

— *Téléchargement de la ressource (downloaded)* : dans certains cas, les utilisateurs souhaitent pouvoir travailler sur certains documents hors-ligne. Ils ont la possibilité de les télécharger. De la même manière que l'utilisation des favoris, cette action est enregistrée.

3.3.2.2 Implémentation

Notre première intuition pour observer les actions des utilisateurs a été d'analyser les logs[7] générés par la plateforme. Malgré le détail des informations présentes, ces traces sont très peu contextualisées, ce qui rend leur exploitation difficile. Nous avons tenté d'y rajouter des informations pertinentes comme le type d'action effectuée, mais la reconstruction des différents événements réalisés par chaque utilisateur est trop complexe avec ces données brutes.

Plutôt que d'exploiter ces traces générées par la plateforme, nous avons choisi d'implémenter les différentes observations des utilisateurs comme une fonctionnalité à part entière. En effet, nous pensons que le contexte de ces événements est particulièrement important. L'accès à une ressource est par exemple conditionné par la recherche effectuée par l'utilisateur. C'est pour cette raison que nous avons défini un modèle de représentation qui associe les actions d'un utilisateur à une **session** et une **recherche**. La session démarre lorsque l'utilisateur se connecte sur la plateforme et se termine soit au moment de la déconnexion, soit après un certain temps d'inactivité. La session est composée d'une ou plusieurs recherches. Chacune d'entre elle démarre avec l'expression d'un besoin par le biais d'une recherche textuelle ou de l'activation de facettes. Une recherche peut évoluer grâce à l'utilisation de nouveaux filtres. Elle se termine lorsqu'un nouveau besoin est exprimé ou lorsque la session se clôture.

Nous avons donc modifié le modèle de données utilisé pour y rajouter des classes représentant la notion de recherche et de session. Ces informations sont automatiquement stockées en base de données lorsque l'utilisateur réalise une des actions observées. D'un point de vue technique, la plupart des actions sont enregistrées côté serveur. Seule l'analyse des *Ressources visibles (visible)* nécessite l'exécution de code sur le client. En effet, la visibilité des différentes ressources de la page de résultats évolue lorsque l'utilisateur fait défiler cette page. Les données de défilement ne sont

7. La plupart des systèmes informatiques enregistrent les différents événements qu'ils doivent traiter. Cette trace est généralement conservée dans des fichiers textuels. Les logs sont généralement utilisés pour analyser les performances d'un système ou identifier la présence d'erreurs.

pas connues du serveur et elles doivent donc être transmises (grâce à l'utilisation du Javascript [8]). Finalement, ces informations sont horodatées et enregistrées en base de données. Chaque événement est associé à un utilisateur, une recherche et une session.

3.3.2.3 Indicateurs comportementaux

Représenter l'ensemble des observations dans notre modèle (sessions et recherches) permet de simplement calculer différents indicateurs du comportement des utilisateurs. Il est par exemple possible de calculer l'ensemble des indicateurs définis dans [Liu 2016]. Parmi ceux-là, on peut par exemple citer :

— la longueur des requêtes textuelles (en mots) ;

— le nombre de documents auxquels l'utilisateur s'est intéressé ;

— le temps passé à investiguer les pages de résultats ;

— le temps passé à investiguer chaque document.

Ces indicateurs peuvent être utilisés directement, mais il est aussi possible d'en calculer une moyenne pour chaque session ou chaque recherche par exemple. Cela permet de caractériser le comportement de l'utilisateur à différents niveaux de granularité.

Nous avons aussi défini des indicateurs plus sémantiques qui permettent d'étudier l'évolution du processus cognitif d'un utilisateur durant une tâche de recherche d'information. Ils font usage de Wordnet [9] (présenté en détail dans la section 4.3.2), un thésaurus disponible en ligne qui rassemble des informations sémantiques sur les mots du langage naturel. La notion d'hyperonyme y est par exemple définie. L'hyperonyme d'un terme est une catégorie plus générique. Par exemple, "animal" est un hyperonyme de "chat". Ces informations nous permettent de calculer la distance moyenne des mots d'une requête par rapport à la racine des hyperonymes, l'entité la plus générique (*entity* dans l'arbre WordNet).

3.3.3 Expérimentations

Pour montrer l'intérêt d'utiliser notre plateforme à des fins expérimentales, nous allons reproduire les expériences réalisées dans [Athukorala 2016]. Ces derniers ont étudié l'intérêt de certains indicateurs pour identifier le type de tâche dans lequel un utilisateur est engagé. Ils se sont intéressés au modèle comportemental de Marchionini (voir section 3.3.1) qui définit deux principaux types de tâches : les tâches exploratoires (*exploratory*) et les tâches de recherche simples (*lookup*). Les auteurs ont étudié le comportement d'utilisateurs issus du domaine de la recherche en Informatique au sein d'une plateforme de recherche académique semblable à Google Scholar [10].

8. Ce langage de programmation permet d'exécuter du code sur l'ordinateur d'un utilisateur selon le contenu des pages Web qu'il visite.

9. https://wordnet.princeton.edu/

10. https://scholar.google.fr/

3.3.3.1 Contexte et contraintes de l'étude

Nous avons choisi de faire nos tests avec un public habitué aux bibliothèques numériques. Ainsi, 40 étudiants en début de spécialisation sur l'étude du patrimoine culturel ont accepté de participer. Ces derniers ne peuvent pas être considérés comme des experts du domaine, mais ils ont un bagage de connaissances intermédiaire. Ils sont de plus, comme beaucoup de personnes aujourd'hui, habitués à utiliser des outils de recherche en ligne. Avec l'aide de leur enseignant, nous avons défini un corpus contenant 240 documents en français et en anglais sur le domaine du patrimoine. Les ressources de ce corpus sont diverses : articles scientifiques, ressources iconographiques, documents primaires [11], *etc.* Un processus de reconnaissance automatique des caractères (OCR) a été exécuté sur les documents pour lesquels c'était pertinent. Ces documents sont ceux qui contiennent de l'information textuelle sous la forme d'une image (un document scanné par exemple). Le texte ainsi extrait est exploitable par les utilisateurs au travers du moteur de recherche de la plateforme.

Nous avons défini quatre tâches différentes que les étudiants devront réaliser en se servant des informations présentes dans les documents du corpus. Trois d'entre-elles tombent dans la catégorie des tâches de recherche simples (*lookup*), la dernière est une tâche exploratoire (*exploratory*).

— *T1 (lookup)* : Identifier la date d'un événement particulier ;
— *T2 (lookup)* : Identifier le lieu d'un événement particulier ;
— *T3 (lookup)* : Identifier un document particulier à l'aide d'indications ;
— *T4 (exploratory)* : Identifier et définir une problématique de recherche traitée par une partie des documents du corpus.

Au début de l'expérience, les participants ont reçu une rapide formation sur la plateforme. Pour limiter l'impact des biais expérimentaux, tous les étudiants ont travaillé avec les mêmes navigateurs et les mêmes écrans. Nous avons ajouté au sein de la plateforme une interface de gestion des tâches, permettant aux étudiants de choisir la tâche à réaliser et de la valider lorsqu'ils estiment y avoir répondu. Cela nous permet d'étudier les différents indicateurs dans un contexte précis.

3.3.3.2 Résultats

Tout comme dans [Athukorala 2016], notre objectif est d'étudier l'utilité de différents indicateurs pour discriminer le type de tâche que réalisent des utilisateurs. Les tâches que nous allons analyser sont similaires à la notion de session que nous avons déjà définie. En effet, pour réaliser une tâche, les utilisateurs seront susceptibles de faire plusieurs recherches différentes. Nous analysons l'évolution des indicateurs sur deux périodes de temps différentes : la première recherche de chaque tâche et la tâche complète. Cette distinction est importante car plus vite le système identifie le type de tâche d'un utilisateur, plus vite il sera capable de s'adapter.

Le tableau 3.1 présente la moyenne de différents indicateurs pour l'ensemble des utilisateurs :

11. Un document primaire est un document original, qui contient directement l'information, au contraire des documents secondaires comme les catalogues.

1. la longueur de la requête correspond au nombre moyen de mots des différentes requêtes textuelles effectuées par les utilisateurs ;
2. la durée de la première recherche ou de la tâche exprimée en secondes ;
3. la proportion visible moyenne des pages de résultats rencontrées. Cet indicateur est calculé à partir des événements de défilement effectués par les utilisateurs, ainsi que la résolution de l'écran et le niveau de zoom adopté ;
4. nombre moyen de documents des pages de résultats sur lesquels les utilisateurs ont cliqué ;
5. position moyenne de ces documents dans les pages de résultats ;
6. nombre de documents originaux consultés (accessibles depuis la page de chaque ressource) ;
7. temps moyen passé à examiner les documents. Cela correspond au temps passé hors des formulaires de recherche et de la page de résultats.

Tableau 3.1 – Valeurs des indicateurs pour les quatre tâches de recherche d'information

		Lookup			**Exploratoire**
		T1	*T2*	*T3*	*T4*
Longueur de la requête	f_1	4.72	3.72	5.32	2.88
(en nombre de mots)	f'_1	3.61	3.57	5.16	2.57
Durée	f_2	264.49	139.87	153.68	408.60
(en secondes)	f'_2	427.88	317.94	172.63	1464.27
Proportion de la page vue	f_3	43.79	30.54	13.64	66.89
(en pourcentage)	f'_3	37.96	28.78	10.59	61.59
Nombre d'éléments cliqués	f_4	0.89	1.04	0.93	1.14
depuis la page de résultats	f'_4	1.68	2.18	1.07	5.04
Position moyenne	f_5	1.24	1.44	1.18	2.61
de ces éléments	f'_5	1.33	1.73	1.21	3.70
Nombre de documents	f_6	0.61	0.96	0.96	0.89
originaux consultés	f'_6	1.21	2.07	1.07	5.04
Temps passé sur	f_7	33.28	71.76	56.35	156.71
les documents (en secondes)	f'_7	79.57	143.04	61.68	485.50

Les indicateurs sont calculés sur deux périodes temporelles différentes : la première recherche (f_n) et la tâche entière (f'_n). Les valeurs des indicateurs sont des moyennes calculées sur tous les participants de l'expérience.

Finalement, nous avons effectué une analyse statistique de ces résultats pour estimer les corrélations qui existent entre les différents indicateurs et le type des

TABLEAU 3.2 – Différences statistiques entre deux types de tâches de recherche d'information

		Première recherche (f_n)			Tâche entière (f'_n)		
		T4			*T4*		
		T1	*T2*	*T3*	*T1*	*T2*	*T3*
Longueur de la requête	*p*	***	***	***	*	***	***
	Z	-4.28	-3.33	-4.16	-2.56	-4.05	-4.08
Durée	*p*	0.1	***	***	*	***	***
	Z	1.61	-3.75	-3.78	-4.33	-4.55	-4.53
Proportion de la page vue	*p*	0.06	**	**	**	0.18	*
	Z	-1.87	-2.97	-3.74	-2.64	-1.34	-2.09
Nombre d'éléments cliqués depuis la page de résultats	*p*	0.29	0.62	0.24	**	***	***
	Z	-1.05	-0.49	-1.15	-3.21	-3.42	4.20
Position moyenne de ces éléments	*p*	*	*	0.05	***	***	***
	Z	-2.25	-2.31	-1.96	-3.39	-3.92	-3.51
Nombre de documents originaux consultés	*p*	0.09	0.65	0.62	**	**	***
	Z	-1.66	-0.45	-0.49	-3.63	-3.21	4.04
Temps passé sur les documents	*p*	*	0.13	0.16	***	0.28	***
	Z	-2.46	-1.51	-1.40	-3.39	-1.07	-4.60

*Ce tableau présente les différences statistiques qui existent entre trois tâches de recherche simple (T1, T2 et T3) et une tâche exploratoire (T4). Les scores Z sont calculés par le test de Wilcoxon. Les valeurs p correspondent à une probabilité que notre hypothèse nulle soit vraie. Nous étudions la pertinence des indicateurs sur deux fenêtres temporelles différentes : la première recherche et la tâche complète. Les valeurs p avec des * sont statistiquement significatives. Nous avons une étoile (*) pour* $p < 0.05$*, deux étoiles (**) pour* $p < 0.01$ *et trois étoiles (***) pour* $p < 0.001$.

tâches. Parce que les données que nous avons obtenues ne suivent pas une distribution normale, nous avons utilisé le test signé de Wilcoxon [Wilcoxon 1945]. Ce dernier va nous permettre d'identifier les indicateurs capable de discriminer le type de tâche en cours. L'hypothèse nulle H_0 que nous souhaitons réfuter est que la valeur des indicateurs décrits dans le tableau 3.1 n'est pas significativement différente selon les deux types de tâche étudiés. Les résultats de ce test sont présentés en détail dans le tableau 3.2. Les probabilités p de réfuter H_0 pour chaque indicateur sont calculées à partir des valeurs Z du test signé de Wilcoxon.

Nous pouvons en conclure que la plupart des indicateurs étudiés sont fondamentalement différents selon le type de tâche (recherche simple ou tâche exploratoire) :

1. la longueur des requêtes textuelles est une information très pertinente, dès le début de la recherche. Les requêtes semblent plus longues (et donc plus

précises) dans les tâches de recherche simple ;

2. la durée observée est une caractéristique déterminante sur l'ensemble de la tâche, mais elle l'est également durant la première recherche. En effet, durant une tâche exploratoire, les utilisateurs n'ont pas d'idée précise de ce qu'ils cherchent, ce qui prend plus de temps ;
3. la proportion visible des pages de résultats est presque toujours discriminante. En effet, lorsqu'un utilisateur est engagé dans une tâche de recherche simple, il s'attend à trouver la réponse à ses interrogations dans les premiers résultats présentés par le système. Si ce n'est pas le cas, il précise sa recherche ;
4. les utilisateurs consultent plus de ressources durant une tâche exploratoire. Ils cliquent donc sur plus d'éléments de la page de résultats ;
5. de la même manière que les utilisateurs s'intéressent à plus de ressources dans la page de résultats, ils sont nécessairement obligés de défiler plus loin dans cette page ;
6. les utilisateurs consultent plus facilement les documents originaux durant les tâches de recherche exploratoires. Cela peut s'expliquer par le fait qu'ils espèrent y trouver plus d'informations que ne présentent déjà les métadonnées de la ressource ;
7. le temps moyen passé à investiguer les documents n'est pas discriminant durant la première recherche de la tâche. Cela peut s'expliquer par le fait que la première recherche des utilisateurs est souvent trop vague et ils ne vont pas jusqu'à visionner les documents. Ils préfèrent préciser ou généraliser leur requête selon les cas.

Nos résultats sont cohérents avec l'étude réalisée dans [Athukorala 2016]. Nous nous plaçons cependant dans un contexte plus large. En effet, notre plateforme est une bibliothèque numérique capable de stocker non seulement des articles scientifiques, mais aussi des ressources plus hétérogènes. De plus, bien que les deux publics testés aient des habitudes de recherche très différentes (des chercheurs en informatique dans l'étude d'Athukorala *et. al.*, des étudiants en Histoire dans notre étude), les résultats comportementaux obtenus sont très similaires. Cela semble suggérer que la prise en main d'outils de recherche n'est pas un problème spécifique au domaine, mais bien un problème commun à tous les utilisateurs d'outils de recherche en ligne.

L'étude du comportement des utilisateurs dans des tâches de recherche d'information peut avoir d'autres applications. Nous pensons par exemple à la validation d'interfaces graphiques ou de systèmes de recommandation. En effet, si le système est capable de recommander l'information pertinente que l'utilisateur cherche, le comportement de celui-ci va être modifié. Il arrêtera sans doute sa tâche courante avant d'en commencer une nouvelle. Nous avons aussi observé que certains indicateurs sont fortement impactés lorsqu'un utilisateur manipule la plateforme pour la première fois. Ces informations pourraient servir à évaluer la difficulté de prise en main d'une interface particulière.

Finalement, nous avons mis en évidence le fait que les différentes caractéristiques permettent de discriminer le type de tâche en cours très rapidement. Les observations

réalisées au début de la tâche peuvent souvent être généralisées à l'ensemble de la tâche. De plus, bien que les participants soient pour la plupart familiers avec ces outils de recherche, nous pouvons mettre en évidence une courbe d'apprentissage (ou de prise en main). Elle est particulièrement visible avec les indicateurs de durée (f_2 et f'_2) et de proportion de la page vue (f_3 et f'_3). Cela peut s'expliquer par un facteur de découverte de la plateforme : même si les participants ont suivi une courte formation, ils utilisent la plateforme pour la première fois et prennent le temps de manipuler et d'observer ses caractéristiques. Nous avons étudié dans cette section différents moyens d'observer et de comprendre le comportement des utilisateurs dans le but d'améliorer les systèmes de recherche. Nous nous intéressons maintenant au point de vue des utilisateurs de ces plateformes.

3.4 Ce que les utilisateurs comprennent du système

Les outils de recherche en ligne jouent le rôle d'interface entre les utilisateurs et les données. Ils ont un rôle particulièrement important. En effet, ils sont d'abord chargés de comprendre les besoins informationnels des utilisateurs. À travers l'analyse des requêtes exprimées, les systèmes de recherche identifient les documents pertinents capables de répondre au besoin des utilisateurs. Il existe cependant de nombreux biais dans les interactions entre utilisateurs et système de recherche. Alors que certains sont intrinsèques aux systèmes d'information, d'autres concernent plutôt les compétences des utilisateurs.

3.4.1 Biais méthodologiques

Pour répondre à leurs besoins informationnels, les utilisateurs sont obligés d'interagir avec des systèmes automatisés (comme les bibliothèques numériques par exemple). Pour cela, ils expriment leur besoin qui est ensuite analysé par le système. Parce que les utilisateurs et les systèmes de recherche n'ont pas la même conception de l'information et du savoir, une étape de *traduction* des besoins est nécessaire. Malheureusement, les représentations conceptuelles humaines sont assez éloignées de celles des systèmes informatiques. Cette *traduction* est donc souvent approximative et peut générer des erreurs de compréhension. Améliorer cette *traduction* est un problème très complexe et même si la technologie évolue vite, nous sommes encore loin de l'interface homme-machine parfaite. Un autre levier d'action disponible est celui de la formation.

Même si nos systèmes de recherche ne sont pas parfaits, ils sont les meilleurs outils que nous avons. Il est donc important d'avoir une compréhension au moins basique de leur fonctionnement. Cette idée est détaillée dans [Schmidt 2016], où l'auteur explique que "l'important n'est pas de comprendre les algorithmes mais les transformations qu'ils génèrent". Une métaphore souvent utilisée pour décrire cette situation est la métaphore de la boîte noire (voir figure 3.6). Il est important d'avoir un regard critique sur les outils utilisés, et de ne pas faire confiance aveuglément aux réponses automatiques d'un système.

Plusieurs points nécessitent l'attention des utilisateurs. Un exemple simple est

FIGURE 3.6 – Schéma d'une boîte noire

Les boîtes noires décrivent les systèmes dont seules les entrées et les sorties sont connues. Le fonctionnement interne est complètement masqué. Les utilisateurs d'outils de recherche se retrouvent souvent face à des systèmes conçus ainsi. Il est difficile pour eux de comprendre pourquoi telle ou telle réponse a été générée.

celui du traitement que les données subissent durant leur indexation par un moteur de recherche (voir section 2.3). Les textes subissent différentes transformations pour limiter l'impact des variations grammaticales. Ainsi, les termes "université" et "universel" ont une chance d'être tous les deux transformés en "univers". Cela pose de sérieux problèmes de compréhension au regard du besoin des utilisateurs. Cependant, être conscient de ces processus permet de mettre en perspective les résultats des moteurs de recherche. Un utilisateur familier avec les traitements opérés par le système sera en mesure d'adapter son comportement et d'exploiter au mieux les capacités de recherche à sa disposition. Au contraire, un utilisateur peu expérimenté aura du mal à rebondir après une recherche infructueuse. Il pourra vite arriver à la conclusion que l'information qu'il cherche n'existe pas, alors qu'elle est simplement difficilement accessible.

Il nous semble également important d'insister sur d'autres mécanismes de recherche, notamment le classement des résultats présentés à l'utilisateur après une recherche. En effet, la très grande majorité des outils de recherche d'information présente les résultats sous forme d'une liste triée par ordre de *pertinence*. Cette notion de *pertinence* varie selon les paramètres du système mais reste floue. Elle est en effet rarement explicitée par les créateurs de ces plateformes. Elle a pourtant un impact considérable sur les utilisateurs. De nombreuses observations empiriques montrent en effet que ces derniers ont tendance à faire aveuglément confiance aux outils de recherche, et ne consultent généralement que les documents qui apparaissent au sommet de la liste de résultats, c'est-à-dire ceux qui sont considérés comme les plus *pertinents* par le système. Cependant, cette présentation sous forme de liste écrase totalement les différences de pertinence entre les documents. Il est souvent impossible de savoir *à quel point* un document est plus pertinent qu'un autre par rapport à une requête donnée.

La qualité des documents indexés peut aussi générer de nombreux biais de recherche. En effet, le texte de certains documents doit être extrait automatiquement avant l'étape d'indexation. Ces extractions automatiques peuvent générer de nombreuses erreurs susceptibles de limiter l'accessibilité des documents de la plateforme. En effet, une recherche n'aboutira pas aux mêmes résultats selon la qualité des textes indexés. Encore une fois, c'est le manque de transparence des plateformes qui pose

le plus de problème. Les utilisateurs n'ont que très rarement d'indications sur la qualité des textes indexés (sous la forme d'un pourcentage de reconnaissance par exemple). Ils doivent pourtant prendre conscience des problèmes d'accessibilité associés à l'extraction automatique de texte.

Pour résumer, le manque de transparence des outils, la qualité des données et le manque d'expérience des utilisateurs créent de nombreux biais méthodologiques. Une autre manière d'exprimer cette idée est d'affirmer que les paramètres des systèmes ont beaucoup d'impact sur l'accessibilité de l'information, mais les comportements des utilisateurs également. Si certaines caractéristiques des plateformes existantes devraient évoluer, il est nécessaire de former les utilisateurs. Pour cela, ces derniers doivent réaliser l'impact que peuvent avoir ces différents biais sur le déroulement d'une tâche de recherche d'information. Si la plateforme que nous avons présentée dans la section 3.2 peut être utilisée pour observer le comportement des utilisateurs, elle peut aussi être utilisée comme ressource pédagogique en tant que telle.

3.4.2 Fonctionnement et implémentation

Même si le fonctionnement des systèmes de recherche est parfois imprévisible, les utilisateurs n'adoptent pas toujours le comportement le plus optimal pour satisfaire leur besoin informationnel. Nous pensons qu'il est important que les utilisateurs soient conscients des différents biais susceptibles d'affecter la qualité de leurs recherches. La plateforme que nous avons développée a pour l'instant été utilisée dans un contexte expérimental. Nous pouvons cependant aussi nous placer dans un contexte pédagogique. Nous avons ainsi ajouté des fonctionnalités à notre plateforme pour prendre en compte ces nouvelles contraintes.

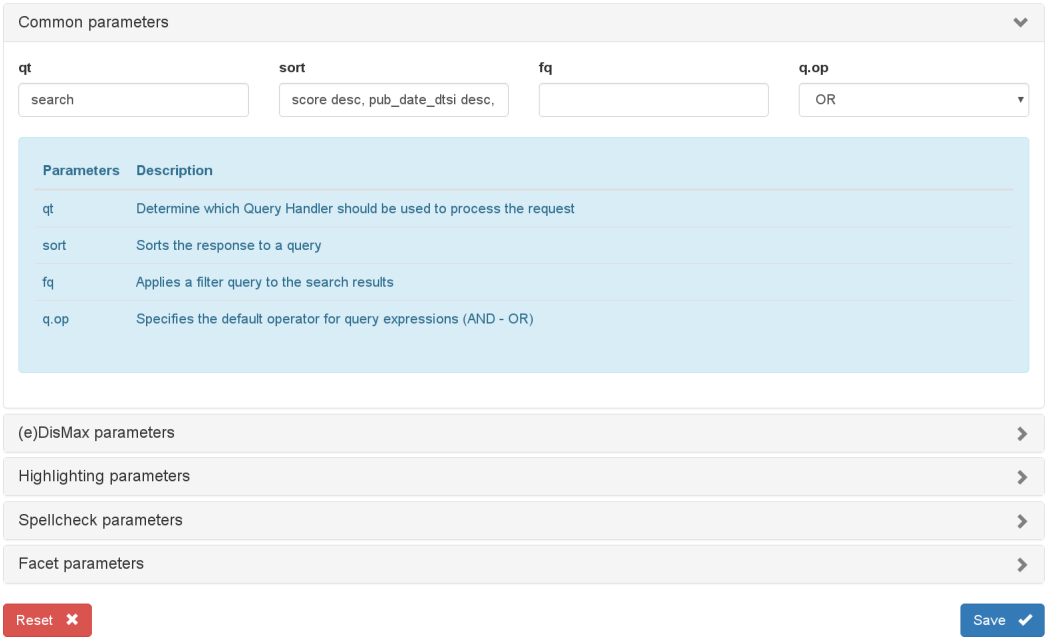
Gestion des rôles Nous nous plaçons dans le contexte d'une séance de formation, par exemple sous la forme de travaux pratiques. Un formateur mène la séance devant des étudiants. Chacun d'entre eux a accès à la plateforme dans laquelle ont préalablement été indexés des documents. Nous voulons permettre au formateur de modifier dynamiquement, au cours de la séance, la valeur de certains paramètres du moteur de recherche. Ces paramètres ont un impact non négligeable sur l'accessibilité des documents. Les étudiants peuvent alors visualiser directement comment ces paramètres modifient la liste de résultats pour une même requête. Seul le formateur doit avoir accès à ces paramètres. Ces fonctionnalités ont été mises en place et doivent être gérées par l'administrateur de la plateforme durant l'inscription.

Contexte de recherche Comme nous venons de le préciser, le formateur d'une session a la possibilité de modifier les paramètres du système. Ces paramètres sont nombreux et affectent le fonctionnement du moteur de recherche et les résultats renvoyés. L'interface de modification du formateur est présentée dans la figure 3.7. Il a la possibilité de paramétrer de nombreux aspects du moteur de recherche, notamment ceux présentés dans la section 2.3 :

— il peut contrôler la présence ou l'absence de certaines facettes ;

— il peut choisir la qualité du corpus sur lequel les recherches seront effectuées ;

FIGURE 3.7 – Interface de modifications des paramètres du moteur de recherche



Les différents paramètres modifiables sont rassemblés dans plusieurs catégories. La signification de chaque paramètre est présentée au formateur. On peut observer sur la figure les paramètres les plus communs. Il est aussi possible de modifier le calcul de pertinence dans la catégorie (e)DisMax, les paramètres de surbrillance, de correction automatique et de gestion des facettes.

— il peut activer ou désactiver la correction automatique ;

— il peut activer ou désactiver la fonction de surbrillance. Celle-ci permet de mettre en évidence les mots de la requête dans les documents renvoyés par le système ;

— les mots vides (*stopwords*) peu porteurs de sens comme les pronoms ou les mots les plus courants du langage ne sont pas toujours pris en compte par les moteurs de recherche. Le formateur peut choisir de les considérer ou non ;

— il peut activer ou non la racinisation des mots ;

— il peut contrôler l'opérateur booléen (**OR** ou **AND**) par défaut utilisé dans les requêtes. À titre d'exemple, la requête "chat noir" peut être interprétée comme l'ensemble des documents qui contiennent le mot "chat" **ET** le mot "noir", ou comme l'ensemble des documents qui contiennent l'un **OU** l'autre de ces deux mots ;

— il peut contrôler les coefficients associés aux *boosts*. Ces paramètres permettent par exemple de favoriser un document qui contient un terme de la requête dans son titre plutôt qu'un autre dans son corps de texte ;

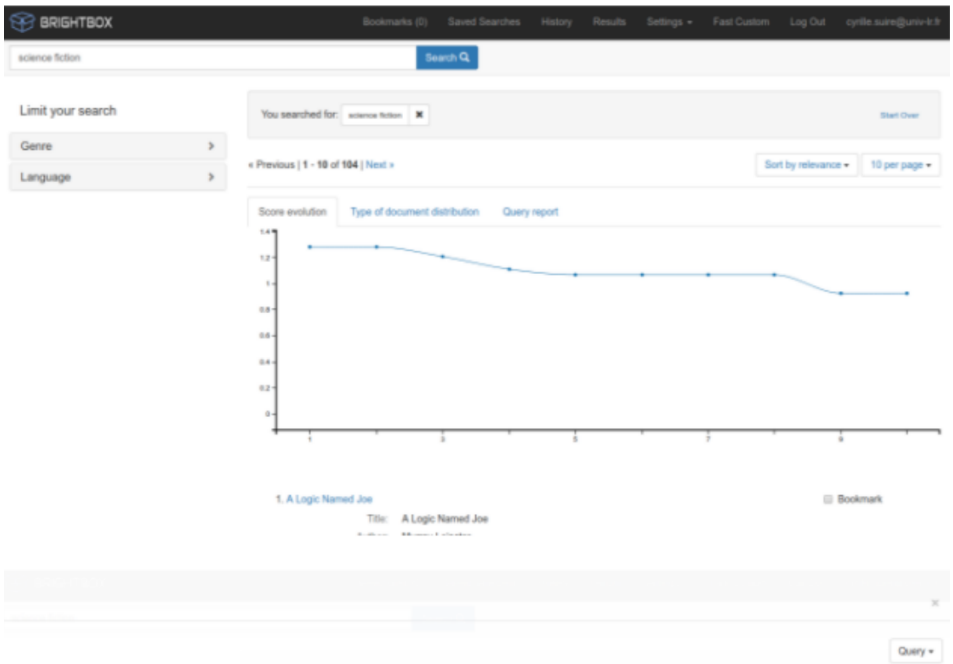
— il peut modifier l'écart maximum à considérer entre les mots. Ce paramètre, le

phrase slop, permet de définir à quel point deux termes d'une requête peuvent être éloignés dans un document pour que celui-ci reste pertinent ;
— il peut contrôler le nombre de mots de la requête qui doivent nécessairement apparaître dans un document pour qu'il soit considéré comme pertinent.

L'ensemble des paramètres utilisés à un moment donné sont considérés comme le contexte de recherche courant. Pour faciliter leur réutilisation, le formateur a la possibilité de sauvegarder ces paramètres pour pouvoir les restaurer en un clic durant une prochaine séance.

Indicateurs et visualisations Pour supporter le discours du formateur, nous avons ajouté différentes visualisations qui permettent de mettre en évidence des informations auparavant cachées. Par exemple, dans la liste de résultats renvoyés par le système après une recherche, nous avons ajouté une courbe qui présente la pertinence de chaque document. Il est ainsi possible d'identifier le degré de pertinence que le système a accordé aux documents (voir figure 3.8). Nous mettons aussi à la disposition des utilisateurs un résumé du calcul de pertinence effectué pour chaque document.

FIGURE 3.8 – Courbe de pertinence dans la page des résultats

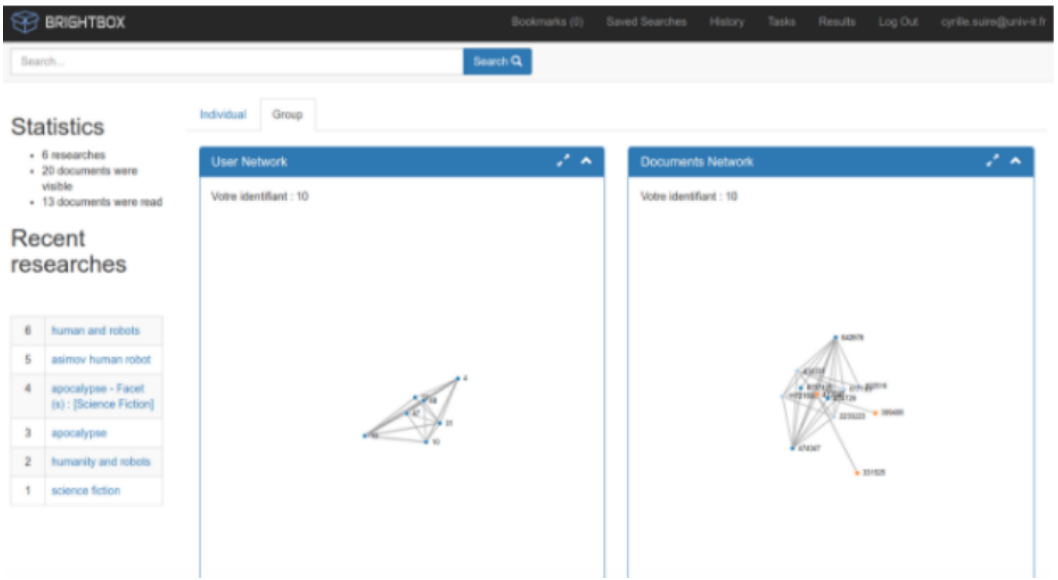


Cette courbe permet de rapidement identifier la pertinence relative des résultats présentés par le moteur de recherche. L'utilisateur peut grâce à elle décider si certains documents valent la peine d'être observés. Cette décision dépendra de la pente de la courbe. Une courbe lente comme sur cette figure indique que les documents sont globalement tous aussi pertinents les uns que les autres. Au contraire, une pente plus forte indiquera que le ou les premiers documents sont significativement plus pertinents.

FIGURE 3.9 – Interface de visualisation des indicateurs personnels

Les indicateurs présentés dans la section 3.3.2 sont présentés de manière visuelle. On peut observer leur évolution, requête après requête.

FIGURE 3.10 – Interface de visualisation des indicateurs collectifs



Les graphes présentés ici permettent aux étudiants de visualiser l'utilisation collective qu'ils ont fait du corpus. Cela permet d'identifier des groupes d'utilisateurs similaires ou des comportements particuliers.

Quand nous utilisons la plateforme à des fins d'observation, nous calculons les différents indicateurs *a posteriori*. Dans le contexte pédagogique, ces indicateurs doivent être disponibles pendant la séance afin de supporter les différentes discussions. Du point de vue de l'implémentation, nous avons déporté le calcul de tous ces indicateurs dans une API à part entière. Celle-ci peut être utilisée après chaque tâche. Une interface de visualisation de ces indicateurs est proposée aux utilisateurs (voir figure 3.9). Ces derniers peuvent choisir de visualiser des statistiques individuelles ou les statistiques de leur groupe. Concernant les statistiques collectives, nous avons ajouté la possibilité d'observer l'utilisation qui est faite du corpus durant une tâche ou une séance. Il s'agit en fait de rassembler les informations de consultation des documents par utilisateurs. Ces données peuvent finalement être organisées sous forme d'un graphe dans lequel les sommets sont des utilisateurs. Deux utilisateurs sont liés entre eux s'ils ont consulté un même document pendant la tâche ou la séance. L'ensemble des informations récoltées permet de construire un réseau qui met en évidence les utilisateurs qui ont eu une utilisation similaire du corpus (voir figure 3.10).

3.5 Conclusion

Nous avons présenté dans cette section différentes actions qui peuvent être entreprises pour diminuer le fossé qui sépare les utilisateurs des données. Si cette problématique est très générale et concerne la plupart des utilisateurs d'outils numériques, nous nous sommes concentrés sur le cas des bibliothèques numériques. Nous avons présenté la plateforme expérimentale que nous avons développée. Cette dernière peut être utilisée à des fins de recherche pour observer le comportement des utilisateurs engagés dans des tâches de recherche d'information. L'étude de leur comportement permet plusieurs choses : identification des principaux biais méthodologiques rencontrés, validation d'interfaces graphiques ou de nouvelles méthodes de recherche, *etc.* Nous avons réalisé une expérience avec un groupe d'étudiants et les résultats de précédentes recherches ont été validés dans un contexte plus large. Ces travaux ont fait l'objet d'une publication [Suire 2016].

Nous avons aussi fait usage de la plateforme dans un contexte pédagogique. Nous pensons en effet que la formation des utilisateurs a autant d'impact sur la qualité de leurs recherches que l'amélioration des systèmes existants. L'outil que nous avons développé peut être utilisé comme support pédagogique pour mener une séance de formation sur la recherche d'information de nos jours. En effet, les différents outils de recherche utilisés créent de nombreux biais méthodologiques dont les utilisateurs doivent être conscients. Notre plateforme a l'ambition de réduire l'effet de boîte noire souvent présent avec ces outils numériques. Ces travaux ont été publiés dans [Suire 2017].

Nous nous sommes intéressés, dans ce chapitre en particulier et dans cette première partie en général, aux interactions qui existent entre les utilisateurs et les outils de recherche d'information qu'ils manipulent. Nous avons aussi montré que l'organisation des données d'un système est particulièrement importante et que les métadonnées doivent être précisément décrites. Pour limiter les problèmes d'acces-

sibilité à l'information, plusieurs stratégies sont envisageables. Nous avons étudié dans cette première partie l'impact que peut avoir le comportement des utilisateurs sur différentes tâches de recherche d'information. Cela dit, même un utilisateur expérimenté dans l'usage des outils de recherche peut rencontrer des problèmes d'accessibilité. En effet, les utilisateurs n'ont aucun contrôle sur la manière dont sont intégrés les documents au sein d'une plateforme de recherche. Certains aspects de cette intégration peuvent en effet générer du bruit informationnel susceptible de diminuer l'accessibilité des documents. C'est pour cette raison que nous avons choisi d'étudier dans la partie suivante les aspects plus techniques d'analyse et d'extraction d'information.

Contributions

- Nous avons développé une plateforme d'observation expérimentale semblable à la majorité des bibliothèques numériques accessibles sur le Web.
- Nous avons montré que certains indicateurs comportementaux peuvent être utilisés pour prédire le type de tâche dans lequel un utilisateur est engagé.
- Nous avons fait état des biais méthodologiques que peuvent rencontrer les utilisateurs, et nous avons décrit comment notre plateforme expérimentale peut être utilisée à des fins pédagogiques.

Deuxième partie

Données et Algorithmes

Chapitre 4

Analyse de documents et extraction de connaissances

Sommaire

4.1 Introduction 64
4.2 Différents types d'hétérogénéités 64
4.2.1 Hétérogénéité des médiums 64
4.2.2 Hétérogénéité des types de fichiers 65
4.2.3 Hétérogénéité des contenus 65
4.2.4 Hétérogénéité de la qualité 66
4.3 Analyse et enrichissement 66
4.3.1 Extraction de texte 66
4.3.2 Compréhension de la structure 67
4.3.3 Compréhension sémantique 68
4.3.3.1 Information non structurée et connaissance 69
4.3.3.2 Entités nommées 72
4.3.4 Lecture distante 73
4.3.4.1 Analyse de réseaux 73
4.3.4.2 Modèles thématiques 75
4.4 Méthodes de modélisation et d'analyse du langage 76
4.4.1 Représentation vectorielle des éléments textuels 76
4.4.1.1 Sacs de mots 76
4.4.1.2 N-grammes 77
4.4.2 Word embedding 79
4.4.2.1 Continuous Bag of Words 80
4.4.2.2 Skip Gram 83
4.4.2.3 Optimisations algorithmiques 84
4.4.2.4 Améliorations 85
4.5 Conclusion 87

4.1 Introduction

Après avoir analysé l'usage qui est fait des bibliothèques numériques dans la partie précédente, nous présentons dans ce chapitre différents outils d'analyse qui peuvent être utilisés par un chercheur lorsqu'il est confronté à de grandes masses de données. Ce chapitre peut être vu comme une liste de bonnes pratiques à adopter ainsi qu'une présentation de la "boîte à outils" d'un chercheur des Humanités Numériques. Nous y détaillerons les différentes étapes qu'il peut rencontrer lorsqu'il tente de répondre à une question de recherche associée à un corpus de documents. Numérisation, extraction d'information et enrichissement sont autant d'étapes nécessaires avant de pouvoir analyser de manière pertinente une masse de données. Benjamin Schmidt, maître de conférences en Histoire à l'université *Northeastern* affirme dans [Schmidt 2016] que les chercheurs en Humanités Numériques n'ont pas besoin de comprendre le fonctionnement intrinsèque des différents algorithmes qu'ils utilisent. En revanche, il est essentiel d'apprécier les différentes transformations qu'ils opèrent sur les données. Pour le citer : "ce qu'un algorithme fait est à la fois différent et plus important que la manière dont il le fait".

La section suivante présente des notions essentielles à connaître pour mener à bien un projet de numérisation d'un corpus documentaire. Nous présentons ensuite dans la section 4.3 les différents traitements à opérer sur des documents numérisés avant de pouvoir en analyser le contenu. De l'extraction de texte à la lecture distante, les différents algorithmes et méthodologies présentés permettront à un chercheur novice d'identifier les outils dont il pourra avoir besoin. Finalement, la section 4.4 présente dans le détail différents moyens de représenter l'information textuelle.

4.2 Différents types d'hétérogénéités

Avant d'être capable d'analyser les informations présentes dans les documents d'un corpus, il faut distinguer les différences qui existent entre les documents pour adapter les processus de numérisation, d'extraction d'information et d'indexation. En effet, différents outils devront être utilisés pour le traitement d'une photo, d'un texte ou d'une vidéo. Il y a cependant d'autres niveaux d'hétérogénéité qu'il convient de comprendre.

4.2.1 Hétérogénéité des médiums

Lors d'une campagne de numérisation, le premier choix critique est celui des documents qu'on veut numériser. Par "médiums", nous entendons ici support physique de l'information. Il en existe de nombreux types : livres, manuscrits, bandes magnétiques, plaques de verre, photographies, *etc.* Selon l'importance, l'âge ou la qualité de ces sources, différentes méthodes de numérisation doivent être mises en place. Certains scanners possèdent par exemple un mécanisme qui permet de numériser plusieurs pages automatiquement. Il est cependant très déconseillé de les utiliser pour la numérisation de documents fragiles, qui pourraient être abimés par ce processus. Ces derniers doivent donc être scannés un par un, augmentant ainsi le

temps de traitement. Il est important d'adapter les méthodes de numérisation aux documents à traiter.

4.2.2 Hétérogénéité des types de fichiers

Un autre point important lors de la numérisation d'un média est le choix de la norme numérique pour stocker l'information. Lorsque l'on s'intéresse à un type de média particulier, il est toujours nécessaire de faire la différence entre le contenu et le contenant. En effet, il existe de nombreuses normes de codage et de stockage de l'information. Sans être exhaustif, on peut citer les formats les plus courants pour différents types d'information à stocker.

— *Vidéo* : .mp4, .wmv, .mov, .avi, .mkv, ...

— *Audio* : .mp3, .wma, .wav, ...

— *Texte* : .odt, .doc, .docx, .txt, .rtf, ...

— *Image* : .png, .bmp, .jpg, .tif, ...

— *Autre* : .pdf, .ppt, .odp, ...

Il faut être attentif aux normes choisies car elles peuvent à terme avoir un impact très important sur l'accessibilité et la préservation de l'information. En effet, certaines normes évoluent ou disparaissent. Les logiciels utilisés pour la lecture de ces fichiers ne sont pas toujours rétro-compatibles et il est parfois très compliqué d'accéder à l'information stockée [1]. Selon la BnF (Bibliothèque nationale de France), un format de fichier est dit viable si :

— le format est largement utilisé et plusieurs logiciels sont capables de le lire

— le format doit être "ouvert" (au contraire d'un format propriétaire) et bien décrit

— si une compression est utilisée (avec ou sans perte), le format de compression doit lui aussi être correctement décrit

— il n'y a pas de mécanismes empêchant la copie (DRM).

Durant un processus de numérisation, le choix du format de conservation des documents est une étape essentielle qui doit être effectuée avec attention.

4.2.3 Hétérogénéité des contenus

Au sein d'un corpus, même parmi les documents de même type ou de même format, il y a une certaine forme d'hétérogénéité. En effet, si deux documents scannés sont traités de manière similaire par le système, leur contenu peut être fondamentalement différent et ainsi nécessiter un traitement particulier. Dans un corpus administratif par exemple, il faudra discriminer les factures des bons de commande car ces deux types de document ne contiennent pas le même type d'information. De la même manière, on ne va pas opérer les mêmes traitements sur un livre scanné,

1. Un collègue en a récemment fait les frais avec un corpus stocké à l'intérieur d'un fichier exécutable. A priori sans difficulté, l'extraction des documents a dû être effectuée sous une machine virtuelle Windows 98, pas si simple à mettre en place.

une page de journal ou un manuscrit. Des processus particuliers dépendant du type d'information à extraire (tableaux, mise en page, images, *etc.*) doivent donc être utilisés. Cette hétérogénéité des contenus peut rendre les processus d'extraction et de mise à disposition des informations très complexes. Il est cependant essentiel d'adapter les algorithmes au contenu des documents.

4.2.4 Hétérogénéité de la qualité

Un dernier point à noter concernant l'hétérogénéité des documents au sein d'un corpus est l'hétérogénéité de la qualité. En effet, les différents processus d'extraction de l'information sont souvent dépendants de la qualité du document. Les documents nativement textuels ne sont pas trop concernés car ils sont la traduction directe de l'idée de l'auteur. En revanche, les images, vidéos et enregistrements audio sont rarement parfaits. Leur qualité peut notamment varier selon le matériel utilisé pour la capture (appareil photo, microphone, caméra, *etc.*) ou selon la qualité du document original dans le cas du scanner. Les différents processus de transcription automatique (OCR ou SpeechToText par exemple) sont susceptibles de faire des erreurs qui auront à terme un impact sur la facilité d'accès ou l'analyse de ces documents. Trop souvent, l'utilisateur final n'a que peu d'information sur la qualité intrinsèque des documents d'un corpus. Il convient donc d'être attentif à cela lors de l'étude d'un corpus hétérogène.

4.3 Analyse et enrichissement

Une fois que les différents documents d'un corpus ont été numérisés, il est nécessaire de mettre en place différentes méthodes d'extraction d'information et d'analyse, destinées à faciliter l'accès et la recherche d'information au sein de ce corpus. Une première étape est la transcription des documents.

4.3.1 Extraction de texte

La majorité des algorithmes permettant l'accès et la recherche d'information sont basés sur le texte. Ainsi, il est souvent nécessaire de transformer l'information du document original en texte. Dans le cas de documents audio (un enregistrement ou la piste son d'une vidéo), il s'agit de transformer le signal audio en texte. Si le nombre de documents à traiter est raisonnable, ce traitement peut être confié à un ou plusieurs opérateurs humains, chargés de transcrire les paroles en texte. Cependant, si un grand nombre de documents doivent être traités, il est nécessaire d'automatiser ce processus. Il existe de nombreux outils capable de reconnaître la parole. Évidemment, le choix de l'outil est dépendant des documents ainsi que du budget disponible. La langue, le nombre de locuteurs ou la présence de bruits ambiants sont autant de contraintes qui vont conditionner le choix de tel ou tel algorithme. La capacité de ces algorithmes à correctement reconnaître la parole est évaluée par le taux d'erreurs sur les mots (*Word Error Rate*). Ce taux évolue généralement entre 10% et 15% [Solera-Ureña 2005]. Depuis l'avènement de l'apprentissage profond ces dernières années, les taux de reconnaissance se rapprochent cependant des 5% ([Xiong 2017] par exemple).

Dans le cas des photographies ou des documents scannés, l'objectif de l'extraction de texte est de transformer l'information représentée par les pixels de l'image en texte brut. Ce processus d'extraction est la reconnaissance optique de caractères (OCR en anglais). La reconnaissance des caractères est un problème difficile à résoudre. Il y a en effet un fossé sémantique important entre la représentation des mots par des pixels et le sens même de ces mots. Comme dans le cas de la reconnaissance de la parole, la qualité de la reconnaissance est principalement liée à la qualité des documents. Il convient également de rappeler que ces algorithmes sont surtout efficaces sur les textes imprimés. En effet, les trop grandes variations des textes manuscrits rendent la reconnaissance de caractères très difficile. Les principales étapes d'un algorithme de reconnaissance des caractères sont les suivantes :

1. **Pré-traitements de l'image** : selon la qualité de l'image originale, différents traitements doivent être opérés pour faciliter la reconnaissance des caractères. Redressement de l'image, binarisation (passage en noir et blanc), accentuation des contours, suppression des tâches, *etc.* sont autant d'étapes qui vont considérablement améliorer la qualité de la reconnaissance.
2. **Segmentation** : on précède souvent la reconnaissance d'une étape de segmentation qui permet d'identifier les zones de l'image à reconnaître (lignes de texte et caractères). Ainsi, on évite aux algorithmes de reconnaissance d'essayer d'identifier du texte là où il n'y en a pas.
3. **Reconnaissance** : étape clé du processus d'OCR, la reconnaissance des caractères permet d'identifier les caractères et les mots représentés par les pixels d'une image.
4. **Post-traitements** : l'étape précédente est susceptible de faire des erreurs. On fait alors souvent usage de modèles de langage, de méthodes statistiques ou de dictionnaires pour corriger ces erreurs (la méthode présentée dans le chapitre suivant en est un exemple).

Les capacités de ces algorithmes sont principalement évaluées grâce à deux mesures : le taux d'erreur sur les caractères (CER) et le taux d'erreur sur les mots (WER). Dans le cas de documents de bonne qualité, le CER est souvent assez faible (typiquement entre 1% et 2%). Cependant, si les documents sont de moins bonne qualité, ces taux peuvent monter jusqu'à 10% et plus [Holley 2009].

Les textes produits par ces algorithmes sont rarement parfaits. Cependant, ils servent bien souvent de point d'entrée aux utilisateurs. Les textes automatiquement extraits peuvent notamment servir à indexer les documents dans un moteur de recherche. Beaucoup d'autres algorithmes peuvent exploiter ces textes. Nous en présenterons quelques uns dans le reste de ce chapitre. La qualité relative de ces textes aura bien évidemment un impact, parfois difficilement mesurable, sur ces algorithmes. C'est un détail dont les utilisateurs doivent être conscients pour adopter une vision critique de la recherche d'information.

4.3.2 Compréhension de la structure

Pour pouvoir exploiter ces textes à des fins de recherche d'information, il est souvent utile d'en analyser la structure. Beaucoup d'algorithmes d'analyse de textes

ont besoin d'informations complémentaires sur les textes, telles que la délimitation des phrases, la valeur grammaticale des mots, *etc.* Nous présentons ici différents traitements souvent opérés sur les textes.

Segmentation des mots et des phrases Une première étape nécessaire est d'identifier et de segmenter les mots contenus dans un bloc de texte. Cette étape est triviale pour les langages latins ou anglo-saxons qui utilisent l'espace comme caractère de séparation. En revanche, pour des langues telles que le mandarin ou le japonais, cette séparation n'est pas si claire et de nombreux travaux se sont penchés sur la question ([Zheng 2013] ou [Peng 2004] par exemple). Après la segmentation des mots, il est nécessaire d'identifier les phrases d'un texte. Si les règles grammaticales apprises à l'école élémentaire sont simples (une phrase commence par une majuscule et se termine par un point), la réalité est plus compliquée. En effet, les marques de ponctuation peuvent être ambigües. Un point peut aussi bien dénoter une fin de phrase qu'une abréviation ou un séparateur décimal. S'il existe des algorithmes de plus en plus efficaces, ce problème n'est pas encore réglé [Read 2012].

Étiquetage morpho-syntaxique Cette étape correspond à l'identification de la nature grammaticale des mots d'un texte (adjectif, nom propre, verbe, *etc.*). Autrefois annotés à la main, il existe aujourd'hui des algorithmes capables d'associer à chaque mot leur nature. Les premiers travaux faisaient usage des modèles de Markov cachés [DeRose 1988]. Ces derniers permettent de prédire la séquence d'étiquettes morpho-syntaxiques la plus probable pour une phrase donnée. Depuis quelques années, comme dans beaucoup de sous-domaines de l'informatique, les algorithmes d'apprentissage automatique tels que les réseaux de neurones ou les machines à état de support (*SVM* en anglais) sont régulièrement utilisés [Wang 2015].

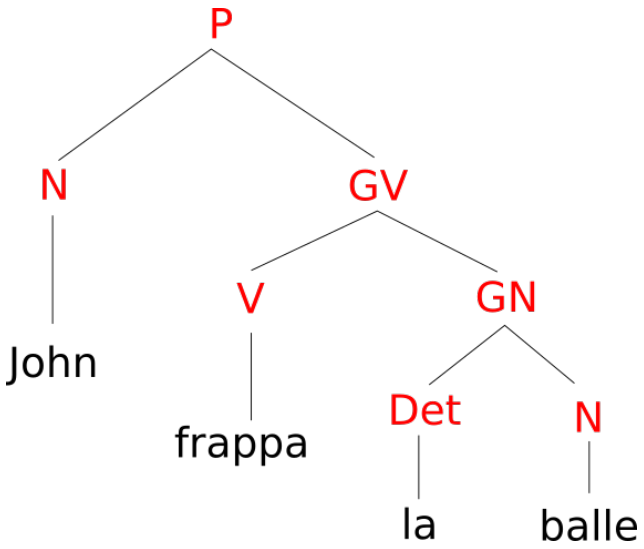
Analyse syntaxique Une fois que les mots d'un texte sont associés à leur nature grammaticale, il devient possible de construire un arbre syntaxique qui obéit aux règles de grammaire d'une langue donnée (voir figure 4.1). Ce dernier permet d'identifier les composants d'une phrase tels que les groupes verbaux ou les groupes nominaux. Ces derniers permettent de faciliter la compréhension sémantique d'une phrase en mettant en évidence le sujet qui fait une action, l'action effectuée, *etc.*

Si ces différentes étapes ne sont pas toujours directement utiles aux utilisateurs, elles permettent cependant d'augmenter le taux de réussite de certains algorithmes de plus haut niveau comme la détection d'entité nommées ou de coréférences (voir plus bas). Il est bon de noter que lorsque le texte a fait l'objet d'une extraction automatique (par un processus d'OCR par exemple), les potentielles erreurs peuvent avoir un impact sur la qualité des différents traitements décrits ci-dessus.

4.3.3 Compréhension sémantique

Les outils d'analyse de la structure d'un texte présentés dans la section précédente ne sont pas suffisants pour faire de la recherche d'information. Ces derniers

FIGURE 4.1 – Exemple d'un arbre syntaxique



L'arbre syntaxique permet de représenter la structure d'une phrase P. Il permet d'identifier la catégorie morphosyntaxique de chaque mot (nom, déterminant, verbe, etc.) ainsi que les différents groupes (groupe nominal et groupe verbal par exemple).

sont toutefois essentiels pour faciliter l'analyse sémantique d'un texte. Une fois que des ensembles de mots cohérents ont été identifiés dans un texte, il devient par exemple possible de les lier à des bases de connaissances (lexique, thésaurus ou ontologie) pour enrichir l'information.

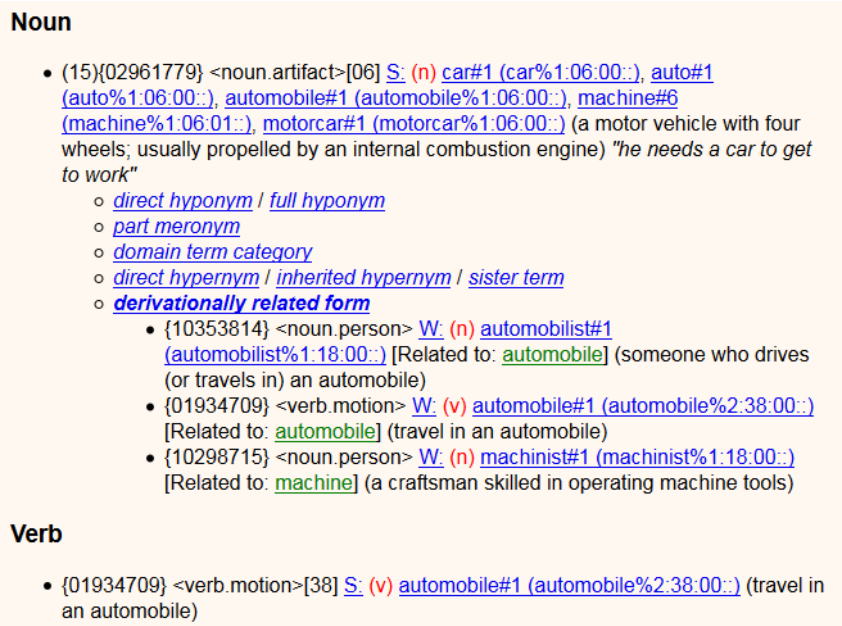
4.3.3.1 Information non structurée et connaissance

L'information contenue dans un corpus est très souvent non structurée. Cela signifie qu'elle n'est pas organisée selon un modèle prédéfini. Si l'être humain est naturellement capable de lier cette information avec ses propres connaissances, ce n'est pas le cas des ordinateurs. Ces derniers surpassent certes l'humain en terme de puissance de calcul brute, mais ils n'ont aucune capacité d'abstraction. Il convient donc de les assister dans cette tâche. Il existe trois principales stratégies pour faciliter la représentation des connaissances. Chacune d'entre elles a une utilité bien particulière.

Thésaurus Une première solution est d'identifier les termes courant d'un domaine donné permettant de caractériser des concepts. Le langage naturel que les humains utilisent est non structuré. Les thésaurus permettent en quelque sorte de faire le lien entre humain et machine. Il s'agit par exemple "d'expliquer" à un ordinateur les liens qui existent entre différents termes d'un même concept. Par exemple, dans le domaine du transport, il peut être intéressant de savoir qu'une **voiture** et un **vélo** font tous les deux partie du concept plus large de **véhicule**. On peut aussi définir des notions d'équivalence. Ainsi, **automobile** est un terme équivalent à **voiture**. Cette

information peut être utile à différents niveaux, de l'indexation à la recherche d'information. Il existe de nombreux thésaurus pour différents domaines d'application, ainsi que pour le domaine général dont *WordNet* [Fellbaum 1998] est un exemple. Leur principal inconvénient vient de leur construction qui doit être manuelle. Ainsi, les thésaurus sont incapables de s'adapter au langage propre d'un utilisateur qui doit alors suivre les règles et le vocabulaire définis par les créateurs. Un exemple est donné dans la figure 4.2.

FIGURE 4.2 – Exemple d'entrée du thésaurus WordNet

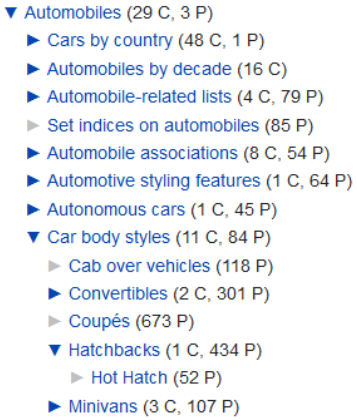


On peut ici observer l'entrée "automobile" dans Wordnet. Il est facile d'accéder aux différents termes associés (synonymes, hyperonymes, termes dérivés, etc.).

Taxonomies Elles contiennent des hiérarchies de catégories et permettent essentiellement de classifier des documents ou des ressources. Par exemple, les articles de Wikipedia sont classés dans des catégories construites manuellement. Les taxonomies permettent ainsi de faciliter le classement et peuvent servir à implémenter une recherche par facettes. Dans le cas de Wikipedia, il ne semble pas exister de règles strictes pour la création et le nommage de nouvelles catégories (voir figure 4.3), ce qui peut rendre leur utilisation assez peu intuitive.

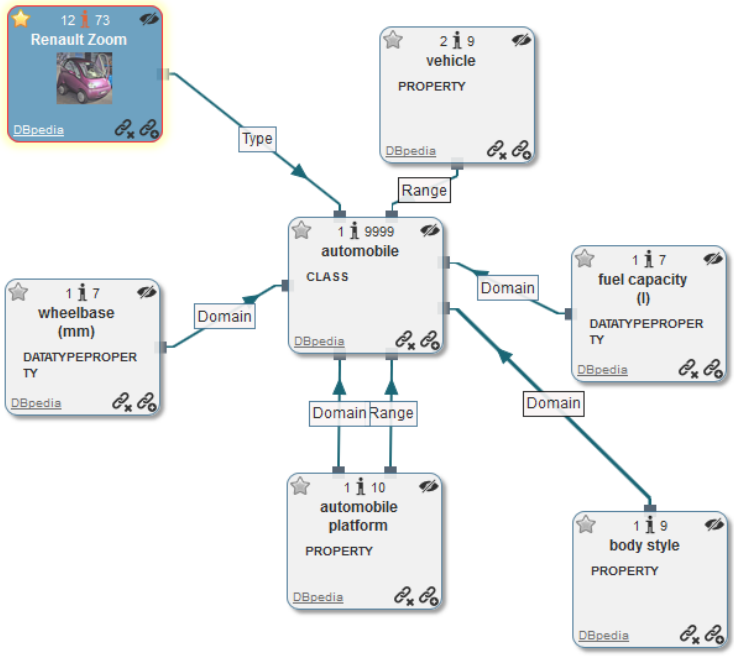
Ontologies Ces dernières permettent de structurer l'information en définissant de manière formelle les différentes relations qui peuvent exister entre les termes et les concepts d'un domaine donné. Elles permettent de structurer précisément l'information. Si on garde notre exemple, une ontologie spécifique au domaine du transport pourrait ainsi décrire la classe "véhicule", associée à des propriété comme le "milieu

FIGURE 4.3 – Aperçu des catégories Wikipedia



On observe ici une partie de la taxonomie utilisée pour classer les articles Wikipedia concernant les "automobiles". Ces catégories sont créées manuellement par les contributeurs.

FIGURE 4.4 – Aperçu d'une ontologie sur les automobiles



On peut ici voir la classe "automobile" associée à différentes propriétés possibles. La Renault Zoom est une instance de cette classe et possède donc certaines de ses propriétés. Cette visualisation sous forme de graphe utilise les données de DBPedia et est disponible à cette adresse : `http://lodmilla.sztaki.hu/lodmilla/`

de déplacement" (eau, air, terre, *etc.*). La classe "véhicule terrestre" est alors définie comme un type particulier de véhicule, associé à la propriété "milieu de déplacement = terre". Le principal intérêt des ontologies est leur capacité à inférer de nouvelles informations. Par exemple, une classe "voiture" est un véhicule qui se déplace sur terre, on peut donc déduire que c'est une instance de "véhicule terrestre". À une plus grande échelle, ce type de raisonnement permet de faciliter l'extraction de connaissance. Il existe de nombreuses ontologies dans différents domaines. Le principal défi à relever est l'alignement d'ontologies. C'est-à-dire être capable de fusionner deux ontologies différentes pour les exploiter et agrandir le champ des prédictions possibles. Les ontologies se présentent sous la forme de graphe (un exemple est donné dans la figure 4.4). Certaines ontologies assez génériques sont très populaires. DublinCore [2] permet par exemple de décrire des ressources documentaires (auteur, date, type de données, *etc.*). FOAF [3] (*Friend Of A Friend*) permet de décrire des personnes et les relations qu'elles entretiennent entre elles. Si aucune ontologie existante ne permet de représenter certains concepts, on peut alors en créer une à l'aide de langages de description tels que OWL ou RDF.

4.3.3.2 Entités nommées

On peut attribuer à certains mots ou expression une valeur sémantique particulièrement importante. Les déterminants sont par exemple beaucoup moins porteurs de sens qu'un nom ou qu'un verbe. C'est également le cas des entités nommées. Ces dernières sont des termes (ou des ensembles de termes) se référant souvent à un nom propre, une description définie ou un élément factuel (date, quantité, *etc.*). Ainsi, les expressions "Washington", "Google", "le premier roi de France" ou "jeudi 15 juin" peuvent être considérées comme des entités nommées. Si l'importance informationnelle des entités nommées est connue depuis longtemps, c'est particulièrement depuis les années 1990 (époque de la popularisation de l'informatique) que ces expressions font l'objet de nombreuses recherches dans le domaine du traitement de l'information. Le principal domaine d'intérêt est leur reconnaissance dans du texte non structuré (*Named Entity Recognition* en anglais). Il existe deux principales stratégies pour cela. La première est l'application d'un ensemble de règles (présence de majuscules, valeur grammaticale des mots du contexte, *etc.*). Ces règles peuvent être déduites de textes annotés ou peuvent être définies manuellement pour s'adapter à un corpus particulier. Cette stratégie est presque toujours combinée à une approche statistique. C'est notamment le cas dans les travaux pionniers de [Mikheev 1999].

Depuis quelques années, des approches par apprentissage automatique sont aussi utilisées. Dans ce cas, il s'agit d'entraîner un système à reconnaître les entités nommées selon différents exemples annotés. L'apprentissage des règles est dans ce cas automatique. Les CRF (*Conditionnal Random Field*) sont très souvent utilisés [Finkel 2005], mais les réseaux de neurones sont de plus en plus populaires [Dernoncourt 2017]. Finalement, ces algorithmes permettent non seulement de détecter la position de ces entités nommées dans un texte, mais sont aussi capables de

2. http://dublincore.org/
3. http://xmlns.com/foaf/spec/

leur attribuer un type (Personne, Lieu, Date, Organisation, *etc.*). D'autres travaux s'intéressent aux entités nommées. Lorsque deux entités identiques ne représentent pas la même chose (deux homonymes par exemple), il est nécessaire de lever toute ambiguïté. Le principe est de lier chacune de ces entités à une base de connaissance externe, capable de les différencier. Ainsi, l'entité "washington" peut faire référence à la ville (Washington D.C.) ou au président (George Washington). Très souvent, c'est l'encyclopédie Wikipedia qui est utilisée. Cependant, dans le cas de corpus très spécifiques, il peut être nécessaire de préalablement créer une base de connaissance (par exemple grâce à une ontologie, voir sous-section précédente). Une autre tâche intéressante est la résolution des coréférences. Il s'agit dans ce cas d'identifier dans un texte les expressions qui font référence à une même entité. Ce traitement est essentiel pour la compréhension automatique de textes en langue naturelle [4] et peut également bénéficier à d'autres algorithmes d'analyse.

L'identification et l'analyse des entités nommées dans un corpus de documents permet d'en avoir une vue d'ensemble, sans avoir à parcourir chaque document individuellement. Ce genre d'analyse haut-niveau est un outil très intéressant lorsque l'on est confronté à une grande masse de documents.

4.3.4 Lecture distante

Ce concept développé par Franco Moretti s'oppose à la lecture attentive qui est l'étude précise d'un texte ou d'un passage de texte. Dans la lecture distante, l'objectif est d'analyser de grandes masses de textes pour en extraire de l'information et de la connaissance, qui permettent à terme d'énoncer des vérités générales sur un corpus particulier. Selon Moretti, cette pratique est essentielle, étant donné le nombre croissant de documents disponibles. Une phrase choc qu'il a énoncée résume bien cette pensée : "Pour comprendre la littérature, nous devons arrêter de lire des livres" [Moretti 2013]. Cette phrase volontairement caricaturale a le mérite d'introduire l'utilité de nombreuses méthodes d'analyse de textes.

4.3.4.1 Analyse de réseaux

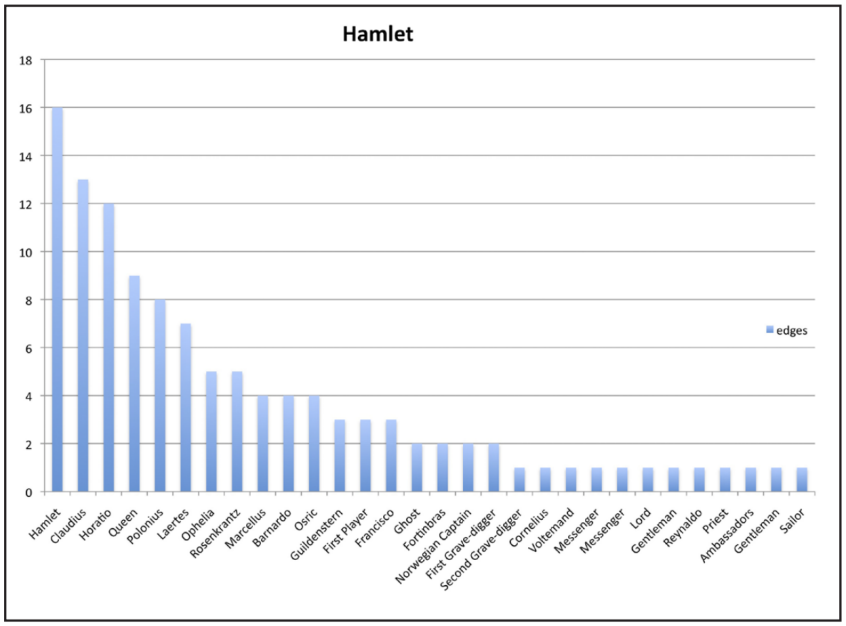
L'une de ces méthodes est l'utilisation de réseaux (aussi appelés graphes). Ces derniers permettent de représenter les liens qui peuvent exister entre différents concepts. La plupart (si ce n'est la totalité) des données peuvent être représentées sous la forme d'un graphe. Formellement, un graphe est composé d'un ensemble de nœuds (aussi appelés sommets) représentant différents concepts. Ces derniers peuvent être liés entre eux. Ces liens sont orientés ou non, selon la symétrie des propriétés qu'ils représentent. Par exemple, dans un graphe représentant des auteurs et des livres, un lien représentant la propriété *"a été écrit par"* aura pour origine un nœud représentant un livre et ira vers un nœud représentant un auteur. Au contraire, un lien représentant la distance entre deux villes, par exemple, sera non orienté. Les liens peuvent également être pondérés selon le type de données manipulées. Les graphes sont par exemple très populaires pour l'analyse de données

4. Les langages naturels sont à opposer aux langages formels tels que le langage informatique.

provenant de *Twitter*[5]. On peut les utiliser pour représenter les relations qu'entretiennent les utilisateurs entre eux (nombre de *retweets*, mentions *j'aime*, *followers*, *etc.*).

Dans son pamphlet n°2 [Moretti 2011], Moretti nous donne un exemple d'utilisation de graphes dans les Humanités Numériques. Il y analyse les relations qui existent entre les différents personnages de la pièce "*Hamlet*" de Shakespeare. Ces derniers sont représentés par les nœuds d'un graphe non orienté. Deux personnages sont reliés entre eux s'ils se sont adressé la parole à un moment de la pièce. Selon Moretti, cette visualisation permet, entre autres, d'aplatir le temps en "rendant le passé aussi visible que le futur". Si cette représentation permet de rapidement visualiser certains aspects de la pièce, elle est bien sûr simpliste. Moretti affirme que cette représentation n'est rien d'autre qu'un modèle approximatif. Cependant, il est tout de même possible d'en retirer de la connaissance. Par exemple, en prenant pour acquis que le nombre de liens associés à un personnage dans le graphe traduit son importance, Moretti fait une remarque intéressante. L'histogramme du nombre de liens par personnage (voir figure 4.5) traduit le fait que leur importance dans la pièce n'est pas binaire. Cela signifie qu'on ne devrait pas pouvoir résumer les choses en parlant de personnage principal ou de personnage secondaire. Toutes les variations "d'importance" sont représentées.

FIGURE 4.5 – Importance des personnages dans la pièce *Hamlet* de Shakespeare.



Cet histogramme présente le nombre de liens associés à chaque personnage de la pièce. Les données semble suivre une loi de puissance, traduisant une absence de centralité. Cette image est extraite du pamphlet de Moretti [Moretti 2011]

5. https://twitter.com

Les graphes peuvent être représentés de nombreuses façons. En effet, la position des différents nœuds d'un graphe n'influe pas sur sa topologie (sa forme). Ainsi, d'un point de vue mathématique, les différentes visualisations d'un même graphe sont équivalentes. Cependant, la manière de présenter ces données à un impact sur le message que l'on veut faire passer. De nombreux algorithmes de visualisation existent. Il convient donc de les explorer pour trouver celui le plus à même de servir un propos. Plusieurs logiciels sont en mesure d'assister l'utilisateur. On peut par exemple citer Gephi [6] ou Cytoscape [7] mais il en existe beaucoup d'autres.

4.3.4.2 Modèles thématiques

Une autre manière d'analyser de grandes quantités de textes est d'identifier automatiquement les principaux concepts évoqués dans chaque document d'un corpus. Il existe pour cela plusieurs méthodes. Un premier algorithme est l'analyse sémantique latente (ou LSA pour *Latent Semantic Analysis*) [K. Landauer 1997]. Ce dernier construit une matrice qui décrit la fréquence des termes du vocabulaire dans chacun des documents du corpus. Les lignes de cette matrice correspondent aux termes tandis que les colonnes correspondent aux documents. L'idée derrière cet algorithme est de factoriser cette matrice pour diminuer sa taille et atténuer le bruit statistique. C'est la technique de la décomposition en valeurs singulières qui est utilisée. Elle permet en quelque sorte de compresser la matrice pour en extraire des informations "générales" sur le corpus. Les différentes matrices obtenues après l'application de cette méthode sont représentatives du comportement des données au sein de la matrice originale. Leur analyse permet d'identifier différents concepts exprimés dans le corpus. Ces concepts permettent de relier termes et documents. Il devient alors possible d'identifier des documents similaires selon les concepts qu'ils expriment, indépendamment des termes qui les composent. Il y a cependant plusieurs limitations à l'utilisation de cet algorithme, notamment la prise en compte de la polysémie (un même terme qui peut avoir plusieurs sens). En effet, le vecteur d'un terme donné correspond finalement à la moyenne de tous ses sens exprimés dans le corpus. Ce problème est cependant limité dans le cas de corpus relativement homogènes car un mot a souvent un sens prédominant dans un domaine donné. De plus, parce que ce modèle est basé sur les sacs de mots, l'ordre des mots n'est toujours pas considéré.

Une évolution probabiliste de cet algorithme existe : l'analyse sémantique latente probabiliste (ou PLSA pour *Probabilistic Latent Semantic Analysis*) [Hofmann 1999]. Cet algorithme considère trois types de variables : les documents, les termes et les thèmes. Si les documents et les termes du vocabulaire sont définis intrinsèquement par le corpus, le nombre de thèmes est un paramètre du modèle qui doit être fixé manuellement. Cet algorithme tente de modéliser les thèmes du corpus selon la distribution des termes du vocabulaire dans les différents documents.

L'héritier direct de PLSA est l'algorithme d'allocation de Dirichlet latente (ou LDA pour *Latent Dirichlet Allocation*) [Blei 2003]. C'est en quelque sorte une version

6. https://gephi.org/
7. http://www.cytoscape.org/

bayésienne de PLSA. Il a été montré que cet algorithme donne de bons résultats sur de petits corpus, dans la mesure où les méthodes bayésiennes évitent le sur-apprentissage des données et sont capables de généraliser. Cet algorithme tente d'inférer des poids pour chaque mot et chaque thème, correspondant respectivement à la probabilité qu'un mot appartienne à un thème donné, et qu'un thème apparaisse dans un document donné.

Nous avons présenté dans cette section quelques algorithmes populaires pour l'analyse de corpus textuels. La prochaine section est dédiée à la description d'autres méthodes permettant de modéliser le langage. Ces dernières offrent de nouvelles perspectives d'analyse et seront utilisées dans le chapitre suivant.

4.4 Méthodes de modélisation et d'analyse du langage

Le domaine de la science informatique qui s'intéresse au traitement de la langue est le Traitement Automatique du Langage Naturel (TALN ou NLP en anglais pour *Natural Language Processing*). Les différentes méthodes et outils qui en sont issus permettent la compréhension, l'analyse, l'extraction d'information, *etc.* Au début de son histoire, dans les années 1950, ce domaine utilisait essentiellement des approches basées sur les grammaires formelles décrites par Noam Chomsky [Chomsky 1956]. Ainsi, un ensemble de règles sont définies pour décrire un langage particulier. Selon la complexité du langage ou de ce qu'on veut en comprendre, ces règles peuvent être nombreuses et alambiquées. Ce n'est qu'à la fin des années 1980 qu'une révolution arrive avec l'apparition d'algorithmes d'apprentissage automatique pour le traitement de la langue [Johnson 2009]. Les premiers travaux se sont notamment intéressés à la génération automatique de règles à partir d'arbres de décisions [Adamo 1980]. Puis, le domaine s'est de plus en plus intéressé aux modèles statistiques, plus souples qu'un ensemble figé de règles.

4.4.1 Représentation vectorielle des éléments textuels

4.4.1.1 Sacs de mots

Classiquement dans les algorithmes de TALN, les mots (ou n-grammes, voir section 4.4.1.2) sont représentés de manière atomique, sans contexte. C'est notamment le cas de la représentation par sacs de mots [8] qui permet de décrire un document textuel pour différentes applications de recherche d'information. Ainsi, un document sera représenté par le multi-ensemble [9] des mots qui le composent, sans considération d'ordre ou de grammaire. Seule l'information de multiplicité (c'est-à-dire la fréquence des mots) est conservée. Imaginons un corpus composé de deux documents textuels d_1 et d_2 :

8. Une des premières apparition de ce terme est attribuée à Zellig Harris, professeur de Noam Chomsky, dans [S. Harris 1954]. Il est intéressant de noter que dans sa publication, Harris déclare que le langage est bien plus complexe qu'un simple sac de mots. Ces modèles sont toutefois capables de résoudre certaines tâches de recherche d'information (typiquement la recherche par mots clés).

9. Autrement appelé "sac". C'est une extension mathématique des ensembles classiques, où chaque élément peut apparaître plusieurs fois.

d_1 **:** Quel magnifique hôtel

d_2 **:** Ce motel est délabré

Dans le contexte clos de ce corpus, le monde est défini par un vocabulaire V :

$$V = \{\text{ce}, \text{est}, \text{délabré}, \text{hôtel}, \text{magnifique}, \text{motel}, \text{quel}\} \tag{4.1}$$

Chaque document peut alors être représenté par un vecteur faisant état de la fréquence des mots du vocabulaire dans ce document. Ainsi, on obtient les représentations vectorielles des documents d_1 et d_2 :

$$\begin{aligned} \overrightarrow{d_1} &= [0, 0, 0, 1, 1, 0, 1] \\ \overrightarrow{d_2} &= [1, 1, 1, 0, 0, 1, 0] \end{aligned} \tag{4.2}$$

La dimension de ces vecteurs est égale à la taille du vocabulaire V. De la même manière, on est capable de représenter chaque mot du vocabulaire par un vecteur. Dans ce cas, ces vecteurs sont également de taille $|V|$ et sont remplis de zéros sauf à l'indice du mot en question où la valeur est 1. On appelle cela l'encodage *one-hot*. Par exemple :

$$\begin{aligned} \overrightarrow{\text{hôtel}} &= [0, 0, 0, 1, 0, 0, 0] \\ \overrightarrow{\text{motel}} &= [0, 0, 0, 0, 0, 1, 0] \end{aligned} \tag{4.3}$$

Cette définition est mathématiquement cohérente puisque le vecteur représentatif d'un document est alors égal à la somme des vecteurs représentatifs des mots qui le composent.

Si des modèles similaires sont encore utilisés aujourd'hui (notamment pour l'indexation de documents dans les moteurs de recherche), ils ont deux principaux inconvénients. Le premier est l'absence d'encodage sémantique dans les représentations vectorielles des mots. En effet, si on observe les vecteurs définis dans l'équation 4.3, ils sont très différents alors que les sens de ces deux mots ne sont pas si éloignés. Le deuxième problème est l'accroissement rapide de la taille du vocabulaire selon les documents du corpus à analyser. On devient alors potentiellement sujet à la malédiction de la dimensionnalité [10]. Pour régler ces problèmes, nous devons être capable d'obtenir, pour chaque mot du corpus, un vecteur de faible dimension. De plus, les vecteurs de deux mots de sens proches devront être similaires.

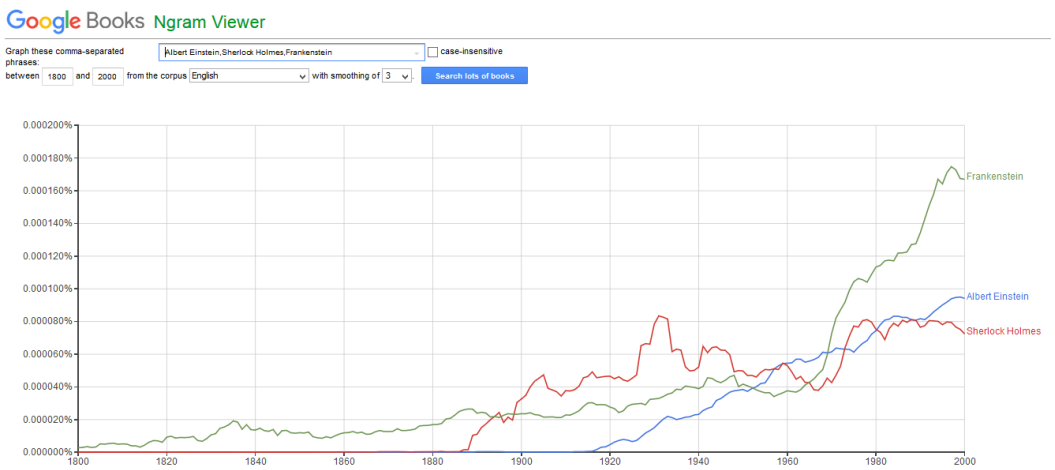
4.4.1.2 N-grammes

Plutôt que de ne s'intéresser qu'aux mots comme éléments atomiques du langage, il est parfois nécessaire d'en combiner certains. Par exemple les phrases "le ballon rouge" et "le ballon bleu" comportent toutes les deux le mot "ballon" mais décrivent clairement deux objets différents. Dans le domaine de la recherche d'information, s'intéresser aux mots sans considérer leur contexte local est problématique.

10. La paternité de ce terme est attribuée dans les années soixante à Richard E. Bellman, un mathématicien qui a décrit l'impact d'un trop grand nombre de dimensions sur l'analyse statistique de données.

En effet, le sens d'un mot peut beaucoup varier selon les autres mots qui l'entourent. L'utilisation des n-grammes permet ainsi d'éviter certaines ambiguïtés sémantiques. L'analyse des n-grammes d'un corpus permet également de résoudre des problèmes tels que l'amélioration de l'OCR ou de la reconnaissance de la parole. On peut par exemple déduire la probabilité qu'a un mot de se trouver à un endroit donné, ce qui peut confirmer ou infirmer les prédictions des algorithmes.

FIGURE 4.6 – Outil de visualisation des n-grammes dans une collection de documents numérisés



Cet exemple montre l'évolution de la fréquence des termes "Frankenstein", "Albert Einstein" et "Sherlock Holmes" au cours du temps. Un grand nombre de livres numérisés sont analysés.

Plus formellement, les modèles statistiques basés sur les n-grams s'intéressent à la probabilité que n éléments donnés se succèdent. Ces éléments peuvent avoir différents niveaux de granularité. On peut ainsi s'intéresser aussi bien aux mots qu'aux caractères. Si s'intéresser aux mots permet d'éviter certaines ambiguïtés sémantiques, l'analyse des caractères peut aider à automatiquement identifier le langage source d'un texte. En effet, le bigramme *th* est plus susceptible d'apparaître dans un texte anglais qu'un texte français. C'est l'analyse des fréquences des bigrammes, trigrammess, *etc.*, qui permet d'extraire la loi de probabilité associée au corpus d'apprentissage. Ces modélisations sont généralisées grâce à l'utilisation de modèles de Markov cachés, dont les fondements mathématiques ont été définis par Baum *et. al.* dans [Baum 1966]. Formellement, un modèle basé sur les n-grammes permet de prédire le mot ou le caractère (selon la granularité choisie) x_i selon les éléments précédents $x_{i-(n-1)}, \ldots, x_{i-1}$, c'est-à-dire $P(x_i|x_{i-(n-1)}, \ldots, x_{i-1})$.

Quoi qu'il en soit, la prise en compte des n-grammes est essentielle dans la plupart des langues naturelles. Pour information, Google a mis en ligne un outil [11]

11. https://books.google.com/ngrams

pour visualiser la popularité de différents n-grammes au sein d'un large corpus de livres numérisés (voir figure 4.6).

4.4.2 Word embedding

Si les représentations et méthodes d'analyse décrites dans la section précédente permettent de représenter un document comme un ensemble de mots ou d'expressions, le contexte n'est jamais pris en compte. En effet, une idée popularisée par John Rupert Firth, un linguiste anglais du XX$^{\text{ème}}$ siècle, est que les mots sont caractérisés par leur entourage. La citation originale est : "You shall know a word by the company it keeps". Une autre façon d'exprimer cette idée est de dire que des mots qui apparaissent dans un contexte similaire ont sans doute un sens proche. C'est ce qu'on appelle l'**hypothèse distributionnelle**.

Les modèles de type *Word Embedding*[12] représentent un ensemble d'algorithmes permettant de modéliser le langage et d'apprendre des descripteurs contextuels caractéristiques de mots ou d'expressions. Le principe de ces techniques est d'apprendre automatiquement pour chaque mot du vocabulaire un vecteur représentatif de faible dimension (typiquement entre 100 et 1000, bien moins que les vecteurs extraits par les modèles de type Sacs de mots, voir section 4.4.1.1). Selon l'hypothèse distributionnelle, deux mots différents mais avec un contexte similaire auront certainement un sens très proche. Cela se traduit techniquement par une similarité des vecteurs représentatifs calculés. Ainsi, ces techniques d'apprentissage permettent notamment de répondre à des questions sémantiques ou syntaxiques telles que :

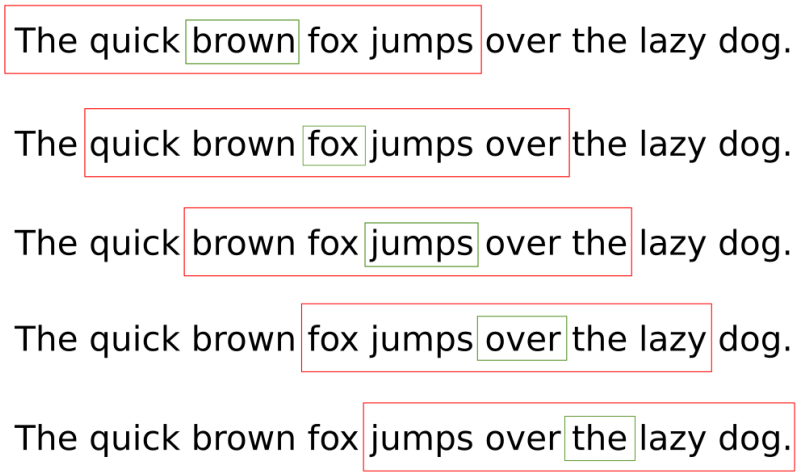
— "homme" est au "roi" ce que la "femme" est à ... *"reine"*

— "voiture" est à "voitures" ce que "cheval" est à ... *"chevaux"*

Il existe deux principales méthodes pour apprendre automatiquement de tels vecteurs : la réduction de dimensionnalité dans les matrices de co-occurrences ou l'apprentissage par réseaux de neurones. Ces deux méthodes obtiennent des résultats similaires sur différentes tâches de TALN. Seul le processus de construction des vecteurs diffère. Une implémentation de la première méthode nommée GloVe (*Global Vectors for Word Representation*) est présentée dans [Pennington 2014]. Dans celle-ci, des statistiques sur les co-occurrences entre termes et contextes sont rassemblées dans une matrice. Ces contextes sont constitués de plusieurs mots. La nature combinatoire de leur contenu entraîne un très grand nombre de contextes. La matrice résultante est factorisée pour extraire des vecteurs de plus faible dimension pour chacun des mots du corpus. Parce que c'est celle que nous avons utilisée pour nos travaux, nous présenterons plus en détail la deuxième méthode, WORD2VEC [Mikolov 2013a], qui utilise un réseau de neurones pour l'apprentissage des vecteurs. En effet, si GloVe est plus facilement parallélisable, l'apprentissage des vecteurs est cependant assez coûteux, surtout sur un ordinateur personnel. Au contraire, WORD2VEC peut être exploité par un utilisateur lambda sur des corpus de taille raisonnable. Cette contrainte est particulièrement importante pour nous car nous souhaitons que les contributions développées dans le chapitre suivant soient

12. Le terme francisé est "plongement lexical" mais il est très peu utilisé.

FIGURE 4.7 – Construction des exemples d'apprentissage



Construction des exemples $(w, context)$ *pour entraîner le réseau de neurones, où* $context$ *est une liste de mots de taille* $2 \times window\_size$*. Chacun de ces mots peut être associé à un poids selon sa distance avec le mot central. Ici la fenêtre glissante à une taille de* 2*.*

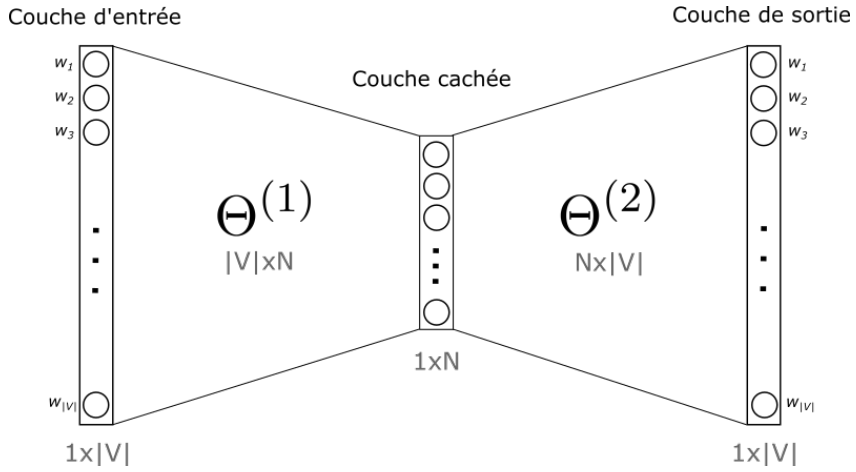
facilement utilisables sans moyen technique particulier (comme un serveur de calcul par exemple).

WORD2VEC est constitué d'un réseau de neurones. Le fonctionnement général d'un réseau de neurones est décrit en Annexe A, et nous présentons ici l'architecture particulière de WORD2VEC. Classiquement, après avoir été entraînés, les réseaux de neurones permettent de faire des prédictions selon des données en entrée. Ici, l'objectif est simplement d'apprendre les poids qui le composent et qui formeront les vecteurs caractéristiques de chaque mot. Cette tâche "factice" n'est ainsi qu'un moyen d'arriver à nos fins : l'obtention de vecteurs denses capables de capturer de la sémantique. L'idée est donc d'entraîner un réseau de neurones à prédire un mot à partir de son contexte (ou l'inverse). Comme tout réseau de neurones, la phase d'apprentissage des poids nécessite des exemples d'entraînement. Ces derniers sont constitués à partir du corpus grâce à une fenêtre glissante dynamique de taille $window\_size$. Les exemples d'entraînement prendront la forme d'une liste de mots de contexte $context$ et d'un mot central w. Le processus est répété pour chaque phrase, paragraphe ou document du corpus. L'exemple du traitement d'une phrase est présenté dans la figure 4.7. WORD2VEC est capable d'utiliser deux architectures de réseau différentes pour générer les vecteurs représentatifs des mots.

4.4.2.1 Continuous Bag of Words

La première architecture est celle des sacs de mots continus (ou CBOW). L'objectif d'apprentissage dans ce cas (c'est-à-dire la tâche factice que l'on veut résoudre)

FIGURE 4.8 – Réseau de neurones utilisé pour l'architecture CBOW avec un seul mot de contexte.



Les dimensions des couches et des matrices de poids sont indiquées en gris. $|V|$ correspond à la taille du vocabulaire et N à la taille des vecteurs représentatifs créés.

est la prédiction d'un mot connaissant son contexte. Soit un vocabulaire V de taille $|V|$ et N le nombre de *features* à extraire pour chaque mot (c'est-à-dire la taille du vecteur représentatif). Le réseau de neurones est initialisé avec une couche d'entrée de taille $|V|$, une couche cachée de taille N et une couche de sortie de taille $|V|$. Imaginons pour le moment que nous ne nous intéressions qu'à **un seul mot de contexte**. L'aspect du réseau de neurones dans ce cas est présenté dans la figure 4.8. L'objectif d'apprentissage devient alors de prédire un mot selon celui qui le précède. Deux vecteurs à N dimensions sont initialisés pour chaque mot w du vocabulaire : un vecteur d'entrée $\overrightarrow{w_{in}}$ et un vecteur de sortie $\overrightarrow{w_{out}}$. Classiquement, N prend une valeur entre 100 et 1000. Les vecteurs d'entrée constituent la matrice des poids entre la première couche et la couche cachée tandis que les vecteurs de sortie constituent la matrice des poids entre la couche cachée et la couche de sortie. En réutilisant les notations adoptées dans l'Annexe A :

$$\Theta^{(1)} = \begin{pmatrix} \overrightarrow{w_{1\text{ in}}} \\ \overrightarrow{w_{2\text{ in}}} \\ \vdots \\ \overrightarrow{w_{|\text{V}|\text{ in}}} \end{pmatrix} \qquad \Theta^{(2)} = \begin{pmatrix} \overrightarrow{w_{1\text{ out}}} \\ \overrightarrow{w_{2\text{ out}}} \\ \vdots \\ \overrightarrow{w_{|\text{V}|\text{ out}}} \end{pmatrix} \tag{4.4}$$

Lors de l'entraînement du réseau, chaque exemple est traité comme suit :

1. le mot de contexte est placé dans la couche d'entrée sous la forme d'un vecteur *one-hot* $a^{(1)}$(voir section 4.4.1.1)
2. les valeurs de la couche cachée sont calculées en multipliant la couche d'entrée et la première matrice de poids ($z^{(2)} = \Theta^{(1)} a^{(1)}$). Puisque nous utilisons l'encodage *one-hot*, cela revient à sélectionner le vecteur $\overrightarrow{w_{in}}$ associé au mot

de contexte. La fonction d'activation f_1 des neurones de la couche cachée est linéaire et transmet donc à la couche suivante le vecteur $\overrightarrow{w_{in}}$ ($a^{(2)} = f_1(z^{(2)})$).

3. On calcule l'entrée de la couche de sortie contenant un score u_j pour chaque mot w_j du vocabulaire V, obtenu en multipliant la couche cachée par le vecteur $\overrightarrow{w_{out}^{(j)}}$ ($z^{(3)} = \Theta^{(2)} a^{(2)}$).
4. On calcule la sortie du réseau qui contient les probabilités $p(w_{out}|w_{in})$ pour chaque mot du vocabulaire d'apparaître après le mot de contexte présenté au réseau. Ceci est réalisé grâce à la fonction d'activation f_2 softmax, un classifieur log-linéaire qui calcule un distribution multinomiale ($a^{(3)} = f_2(z^{(3)})$).
5. La phase d'apprentissage et de mises à jour des poids est effectuée grâce à l'algorithme de rétropropagation de gradient présenté en Annexe A.

L'objectif est de maximiser la probabilité d'observation du mot w_{out} dans le contexte de w_{in}, c'est-à-dire $p(w_{out}|w_{in})$. La fonction de perte utilisée est un cas particulier de la mesure d'entropie croisée et est définie ainsi :

$$J(\Theta) = -log(p(w_{out}|w_{in})) \tag{4.5}$$

Cette fonction permet de fortement pénaliser le classifieur lorsqu'il attribue une faible probabilité au mot correct. On poursuit ensuite l'algorithme de rétropropagation de gradient afin de mettre à jour les différents poids, et donc les différents vecteurs représentatifs des mots du vocabulaire.

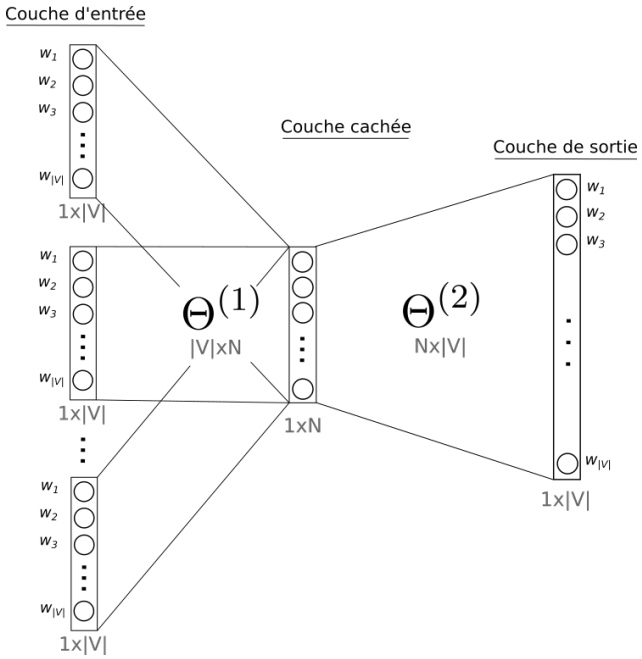
Concrètement, le vecteur du mot contextuel est éloigné ou rapproché des vecteurs de sortie selon les prédictions du système. Si on a surestimé une prédiction (c'est-à-dire qu'on a prédit à tort qu'un mot devrait apparaître avec contexte donné), le vecteur du mot contextuel est légèrement éloigné du vecteur de sortie du mot prédit. Inversement, si la prédiction est sous-estimée et qu'une trop faible probabilité est affectée au mot de sortie correct, le vecteur du mot contextuel va se rapprocher du vecteur du mot prédit. Les positions successives du vecteur du mot contextuel sont donc dépendantes des erreurs de prédiction sur tous les autres mots du vocabulaire en sortie du système. Au cours de l'entraînement, les modifications des poids induites par chaque exemple vont se cumuler. On peut visualiser ces mouvements en imaginant que les vecteurs sont liés par des ressorts. À chaque exemple présenté au réseau, ces ressorts se tendent ou se détendent, jusqu'à ce que le système atteigne un équilibre [Rong 2014].

Nous venons donc de traiter le cas de l'architecture CBOW avec un seul mot de contexte. Cependant, il n'est pas plus difficile d'en considérer plusieurs. L'aspect du réseau de neurones lorsqu'on utilise l'architecture CBOW avec plusieurs mots de contexte est présenté dans la figure 4.9. En effet, plutôt que de simplement copier le vecteur d'entrée dans la couche cachée du réseau, on utilise la moyenne des vecteurs de contexte. Pour un contexte de C mots :

$$z^{(2)} = \frac{1}{C} \sum_{i=1}^{C} \overrightarrow{w_{\text{i in}}} \tag{4.6}$$

Le fonctionnement est ensuite le même : la fonction de perte est très similaire, seule la valeur de la couche cachée a changé.

FIGURE 4.9 – Aspect du réseau de neurones lorsque plusieurs mots de contexte sont pris en compte.



Dans ce cas, l'entrée de la couche cachée correspond à la moyenne des vecteurs représentatifs des mots de contexte (voir équation 4.6). Ce vecteur moyen est ensuite utilisé de la même manière que lorsqu'on s'intéresse à un seul mot de contexte.

4.4.2.2 Skip Gram

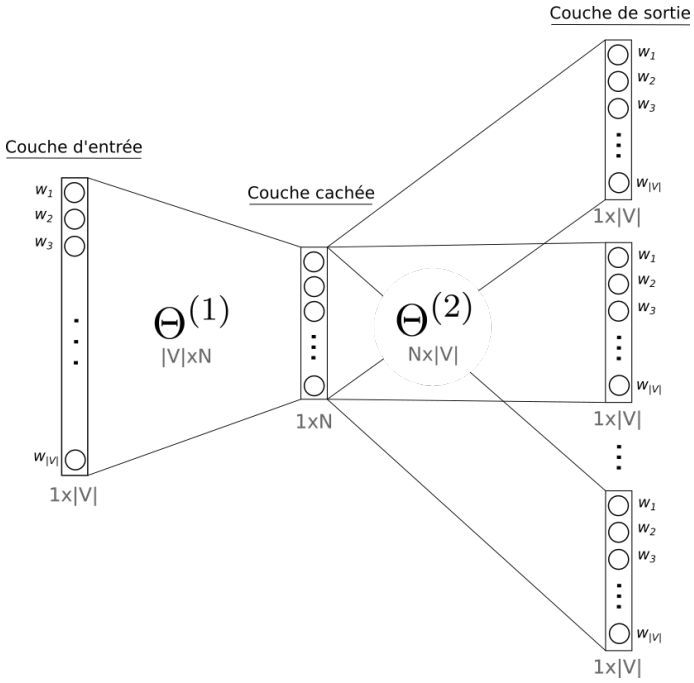
À la différence du modèle CBOW présenté dans la section précédente, l'objectif d'apprentissage lorsqu'on utilise l'architecture Skip-Gram est la prédiction du contexte d'un mot donné (voir figure 4.10). La couche cachée est calculée de la même manière qu'avec le modèle CBOW lorsqu'il n'y a qu'un seul mot de contexte : on copie le vecteur d'entrée $\overrightarrow{w_{\text{in}}}$ du mot vers la couche cachée. Enfin, plutôt que de calculer une seule distribution multinomiale comme dans CBOW, on en calcule une pour chacun des mots du contexte attendu.

L'algorithme de rétro-propagation de gradient est ensuite assez semblable. La fonction de perte est modifiée pour prendre en compte toutes les sorties du système, c'est-à-dire les probabilités assignées aux différents mots contextuels.

$$E = -log(p(w_{1\ \text{out}}, w_{2\ \text{out}}, \cdots, w_{\text{C out}}|w_{\text{in}})) \tag{4.7}$$

Les équations de dérivation sont elles aussi similaires à celles utilisées pour l'architecture CBOW, à la différence près que la fonction de perte va sommer les prédictions d'erreurs sur tous les mots du contexte.

FIGURE 4.10 – Aspect du réseau de neurones lorsque l'architecture Skip-Gram est utilisée.



Ici, il s'agit de calculer une distribution multinomiale pour chacun des mots de contexte.

4.4.2.3 Optimisations algorithmiques

Pour chacun des modèles décrits précédemment, chaque mot du vocabulaire possède deux vecteurs : un vecteur d'entrée et un vecteur de sortie. L'apprentissage (à travers la mise à jour des poids) est facile pour le vecteur d'entrée mais très difficile pour celui de sortie. En effet, pour chaque exemple d'apprentissage fourni au système, il faut itérer sur chacun des mots du vocabulaire pour calculer l'entrée de la couche de sortie. Puisque ce calcul est infaisable en pratique, il est nécessaire de l'optimiser. Deux approches différentes sont envisagées : *hierarchical softmax* et *negative sampling*.

Hierarchical Softmax Développée dans [Mnih 2009], c'est une optimisation pour le calcul du softmax en sortie du réseau. Les différents mots du vocabulaire V en sortie du réseau deviennent les feuilles d'un arbre binaire et chaque nœud interne possède exactement deux nœuds fils. Pour chaque feuille, il existe un unique chemin vers la racine. Le nombre de nœuds interne est égal à $|V| - 1$. Dans ce cas, les différents mots du vocabulaire V ne possèdent plus de vecteur de sortie $\overrightarrow{w_{out}}$. En revanche, chacun des nœuds internes de l'arbre (y compris la racine) est associé à un vecteur. La probabilité qu'un mot soit effectivement le mot de sortie est alors définie comme la probabilité d'une marche aléatoire depuis la racine jusqu'à la feuille.

Les probabilités d'aller à gauche ou à droite dépendent de la couche cachée et du vecteur du nœud courant. Finalement, on peut prouver que la somme des probabilités des feuilles est égale à 1, ce qui fait de Hierarchical Softmax une distribution multinomiale. La mise à jour des différents poids du réseau est ensuite similaire à la méthode décrite précédemment. Si le nombre de vecteurs de sortie présents dans le réseau est quasiment le même ($|V|-1$ au lieu de $|V|$), la complexité des calculs pour chaque mot de contexte de chaque exemple d'apprentissage est réduite de $O(|V|)$ à $O(log(|V|))$.

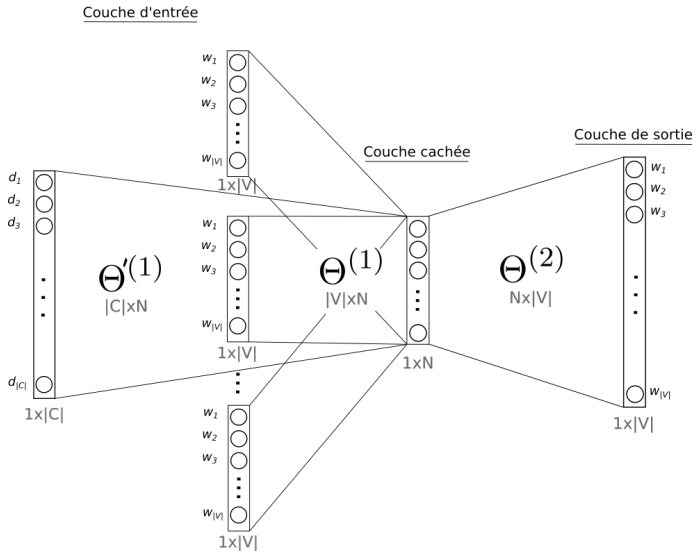
Negative Sampling Cette technique d'optimisation est développée dans [Mikolov 2013b]. Plutôt que de mettre à jour chacun des vecteurs de sortie, l'idée est ici de n'en mettre à jour qu'une partie. On met évidemment à jour le vecteur de sortie du mot prédit, mais également un échantillon de vecteurs de mots choisis au hasard selon une distribution des mots du corpus, élevée à la puissance 3/4 [Mikolov 2013b]. Il en résulte que les mots peu fréquents sont *boostés* pour être choisis plus souvent. La fonction d'erreur dépend de tous les vecteurs de l'échantillon. Ensuite on calcule les dérivées successives de la fonction d'erreur selon l'entrée de la couche de sortie et selon le vecteur de sortie du mot. On peut alors appliquer la fonction de mise à jour aux vecteurs de cet échantillon plutôt qu'à tous les vecteurs lorsqu'on n'utilise pas le *negative sampling*.

4.4.2.4 Améliorations

Les travaux présentés dans [Le 2014] introduisent deux méthodes pour inférer les vecteurs représentatifs d'un document. Ici on appelle document tout groupe de mots (phrase, paragraphe, livre, *etc.*). Les deux méthodes présentées sont similaires à celles utilisées dans WORD2VEC. Le premier modèle est appelé *Distributed Memory Model of Paragraph Vectors* (PV-DM). Il est assez similaire au modèle CBOW présenté dans la section 4.4.2.1 (voir figure 4.11). En plus de la matrice qui contient les vecteurs représentatifs des mots, une nouvelle matrice est ajoutée. Cette dernière est en charge des vecteurs représentatifs des documents. Pour chaque exemple d'entraînement, la couche cachée est calculée en faisant la moyenne des vecteurs de contexte et du vecteur du document. La seconde méthode, *Distributed Bag of Words version of Paragraph Vector* (PV-DBOW) est très semblable à l'architecture Skip-Gram présentée dans la section 4.4.2.2. Cependant, plutôt que d'utiliser un mot du vocabulaire en entrée, c'est le document lui-même qui est utilisé (voir figure 4.12).

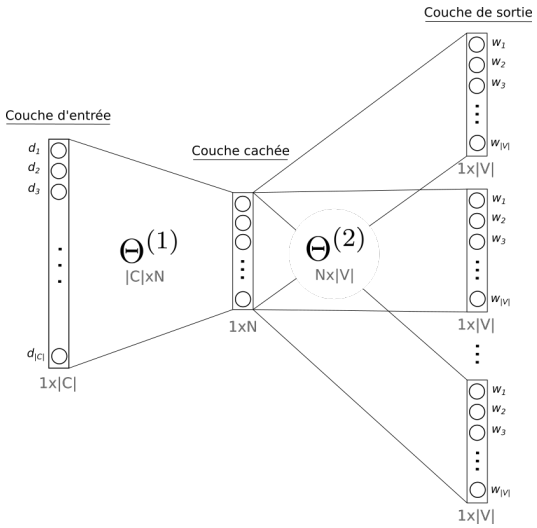
Le principal intérêt d'une telle méthode est de pouvoir résumer le contenu sémantique d'une phrase à un simple vecteur. La qualité de ces vecteurs a été évaluée dans [Lau 2016] pour deux tâches distinctes : la détection de doublons dans les messages d'un forum et le calcul de similarité sémantique entre des paires de phrases. Dans les deux cas, DOC2VEC est comparé à un modèle basé sur WORD2VEC et à un autre basé sur les n-grammes. Les résultats de DOC2VEC sont meilleurs dans la majorité des configurations possibles.

FIGURE 4.11 – Aspect du réseau de neurones lorsque l'architecture PV-DM est utilisée.



Ce modèle est semblable à l'architecture Skip-Gram présentée précédemment. Une matrice de poids supplémentaire ($\Theta'^{(1)}$) est utilisée pour stocker les vecteurs représentatifs des documents.

FIGURE 4.12 – Aspect du réseau de neurones lorsque l'architecture PV-DBOW est utilisée.



À la différence du modèle Skip-Gram, la matrice d'entrée contient les vecteurs représentatifs des documents.

4.5 Conclusion

Nous avons présenté dans ce chapitre les différentes étapes nécessaires à l'analyse d'un corpus de documents, de l'exploitation des données textuelles bas niveau jusqu'à l'extraction d'informations importantes comme les entités nommées. Le principal intérêt de ce chapitre est d'offrir aux chercheurs (particulièrement ceux des Humanités Numériques) un aperçu des différents points sur lesquels il est nécessaire de se concentrer durant les différentes étapes d'un travail de recherche sur un corpus documentaire. De plus, nous souhaitons apporter quelques connaissances nécessaires pour bénéficier d'une vision critique des différents outils présentés dans le chapitre. Certains outils peuvent paraître "magiques" et il est important de démystifier leur usage. Il ne s'agit pas de comprendre les différentes formules mathématiques qui régissent son fonctionnement mais bien d'identifier les différentes transformations opérées sur les données. Par exemple, la plupart des algorithmes d'analyse travaillent sur la base d'un texte extrait automatiquement. Ce processus d'extraction n'étant pas parfait, les différentes erreurs produites auront nécessairement un impact sur la qualité des analyses subséquentes. Il est important d'être conscient de ce biais qui peut potentiellement mettre à mal une théorie.

Le chapitre suivant va présenter dans le détail l'effet que peut avoir ce bruit informationnel sur la détection des entités nommées. Ces dernières étant particulièrement populaires, l'impact du bruit n'en est que plus fort. Pour cette raison, nous présenterons une méthode permettant de limiter l'incidence du bruit sur l'analyse d'un corpus.

Contributions

- Nous avons décrit les différentes transformations que peuvent subir les données durant un projet de numérisation, de l'extraction de texte brut à une description sémantique plus riche.
- Nous avons fait état des modèles mathématiques associés à la représentation machine de données textuelles et à l'extraction d'information à partir d'un corpus de documents.
- Nous avons finalement décrit en détail le fonctionnement de l'algorithme WORD2VEC, utilisé dans le prochain chapitre.

Chapitre 5

Correction automatique des entités nommées

Sommaire

5.1 Introduction . . . 90
5.2 Travaux précédents . . . 91
5.3 Évaluation préalable . . . 92
5.3.1 Création d'une vérité terrain . . . 92
5.3.2 Entités nommées et bruit OCR . . . 94
5.4 Présentation de la méthode . . . 96
5.4.1 Entraînement du modèle contextuel de langage . . . 96
5.4.2 Définition d'une bi-similarité lexicographique . . . 98
5.4.2.1 Similarité contextuelle . . . 98
5.4.2.2 Similarité d'édition . . . 99
5.4.2.3 Combinaison des deux similarités . . . 99
5.4.3 Construction du graphe des entités nommées similaires . . . 101
5.4.4 Clustering . . . 102
5.5 Évaluation . . . 103
5.5.1 Évaluation du clustering . . . 103
5.5.1.1 Mesure MUC . . . 103
5.5.1.2 Mesure B^3 . . . 104
5.5.1.3 Mesure CEAF . . . 105
5.5.1.4 Mesure combinée . . . 106
5.5.2 Évaluation de la correction . . . 108
5.5.2.1 Sélection d'un représentant . . . 108
5.5.2.2 Correction du corpus . . . 108
5.5.3 Évaluation de l'accessibilité des documents . . . 109
5.5.3.1 Mesure d'accessibilité . . . 110
5.5.3.2 Résultats et discussion . . . 112
5.6 Conclusion . . . 115

5.1 Introduction

Une caractéristique essentielle d'un système de recherche d'information est l'accessibilité des documents pour les utilisateurs. En effet, le nombre de documents à disposition d'un utilisateur importe peu s'il n'existe pas de moyen d'accéder aux informations qu'ils contiennent. La numérisation d'un corpus n'est qu'une étape préalable avant la mise à disposition des documents et d'outils nécessaires à leur interrogation. Comme nous l'avons expliqué dans la section 3.4.1, la qualité des documents indexés dans un moteur de recherche a beaucoup d'impact sur l'accessibilité de ces documents. Cette variation de la qualité peut être due à un mauvais paramétrage de l'outil de capture (scanner, caméra, *etc.*) ou simplement à la nature même du document (un manuscrit ancien et abimé par exemple). De plus, les différents processus d'extraction d'information utilisés avant l'indexation des documents sont imparfaits. Les algorithmes d'OCR sont par exemple susceptibles de mal interpréter certains caractères. La qualité du texte généré dépend dépend des algorithmes utilisés ainsi que de la qualité de l'image originale :

— le texte est quasiment parfait,

— le texte contient des erreurs mineures,

— le texte est difficilement déchiffrable,

— le texte est complètement illisible.

La qualité de ce texte extrait automatiquement a relativement peu d'impact sur la lecture d'un document par un utilisateur. En effet, les utilisateurs s'intéressent plus souvent au document original (une image d'un document scanné par exemple) qu'au texte issu de l'OCR. Cependant, c'est ce texte qui est utilisé par le moteur de recherche lors de l'indexation. Ainsi, si certains mots ont été mal reconnus par le processus d'OCR, ils seront indexés avec leurs erreurs. Cela pose un sérieux problème durant la phase d'interrogation du moteur de recherche.

À cause du nombre grandissant d'outils de consultation de documents, les utilisateurs ont souvent le sentiment d'avoir accès à l'intégralité de l'information. Un utilisateur, peu conscient de l'impact que peut avoir l'OCR sur la recherche d'information, pourra facilement imaginer que sa recherche a échoué car aucun document ne la satisfaisait. Comme nous l'avons présenté dans la première partie de ce manuscrit, il est possible (et nécessaire) de former les utilisateurs à comprendre et appréhender les systèmes de recherche d'information avec plus d'esprit critique. Nous pensons cependant que ce premier levier d'action n'est pas suffisant. Selon nous, il est également important de s'assurer de l'accessibilité des documents d'un point de vue plus technique. Pour cette raison, nous allons présenter dans ce chapitre une méthode que nous avons développée pour corriger automatiquement les textes extraits par un processus d'OCR. Nous présenterons dans la section suivante différents travaux qui se sont intéressés à la correction de textes bruités générés par un processus OCR. Nous ferons également une analyse de l'impact du bruit OCR sur la reconnaissance d'entités nommées par différentes méthodes. Puis, nous exposerons la chaîne de traitement générale que nous avons développée ainsi que le détail de notre méthode. Finalement, nous procéderons à une évaluation sous différents points de vue de notre méthode.

5.2 Travaux précédents

La plupart des approches qui s'intéressent à la correction automatique de textes produits par l'OCR utilisent des connaissances extérieures, comme un dictionnaire de termes correctement orthographiés par exemple. C'est notamment le cas dans les travaux d'Evershed *et. al.* Ces derniers ont développé dans [Evershed 2014] une chaîne de traitement où deux procédés s'enchaînent. Le premier est chargé de détecter la position des erreurs à partir d'un modèle de langage entraîné sur un corpus sans erreur. La seconde étape sélectionne des candidats de correction à partir du modèle de langage et d'un modèle d'erreur. Finalement, le meilleur candidat est celui qui maximise le produit des probabilités des deux modèles selon le théorème de Bayes. Les travaux de Thompson *et. al.* [Thompson 2015] sont un autre exemple d'utilisation de connaissances extérieures. Ces derniers se sont concentrés sur l'amélioration des capacités d'un logiciel de correction orthographique grâce à l'utilisation d'un dictionnaire. Leur méthode est cependant difficilement généralisable car les mots du dictionnaire sont dépendants du corpus que l'on tente de corriger. Dans leur cas, le dictionnaire qu'ils utilisent est adapté à un corpus médical. D'autres méthodes préfère analyser la fréquence d'erreurs OCR typiques pour ensuite les détecter et les corriger. C'est par exemple le cas des travaux développés par Afli *et. al.* dans [Afli 2016], où les auteurs ont utilisé des méthodes de traduction automatique qui mettent à profit un modèle d'erreurs OCR entraîné sur un corpus annoté. Les approches probabilistes sont également populaires. Dans celles-ci, on utilise la probabilité qu'a un mot de se trouver dans un contexte donné. C'est par exemple ce qui a été réalisé par Mei *et. al.* dans [Mei 2016]. Les auteurs ont défini un ensemble de descripteurs pour classifier les erreurs OCR à partir de la fréquence d'un mot ou de la probabilité associée à son contexte. Leur approche nécessite cependant l'utilisation de connaissances extérieures telles qu'un lexique de mots et de n-grams, ainsi qu'un corpus annoté pour récolter des statistiques sur les erreurs OCR. De manière similaire, les travaux de Kissos *et. al.* dans [Kissos 2016] s'intéressent à la construction d'un modèle de langage à différents niveaux de granularité (mots et caractères). Ce modèle est construit à partir d'un corpus extérieur ainsi qu'une liste fréquentielle de termes. Pour chaque mot du corpus à corriger, des candidats de correction sont sélectionnés. Le résultat correct est finalement choisi grâce à un processus d'apprentissage automatique.

La principale faiblesse des travaux présentés ci-dessus est l'utilisation de connaissances extérieures (corpus ou dictionnaires par exemple) pour apprendre des modèles de langage ou valider l'existence de certains mots. La création de dictionnaires ou l'annotation de corpus sont des tâches qui nécessitent beaucoup de temps et de ressources. De plus, l'utilisation de connaissances extérieures peut vite nuire à la généricité d'une méthode. Pour ces raisons, nous pensons qu'il est nécessaire d'envisager une approche non-supervisée de correction automatique des erreurs OCR. En revanche, nous sommes, nous aussi, convaincus de l'importance de prendre en compte le contexte des mots pour détecter et corriger ces erreurs. Cependant, les précédents travaux utilisent souvent un contexte relativement limité. Dans [Mei 2016] et [Evershed 2014] par exemple, les contextes ont une taille de 5 (5 mots avant et 5 mots après le mot d'intérêt). Nous pensons qu'augmenter la taille de la fenêtre

contextuelle a du sens lorsque nous travaillons avec un corpus bruité. Si les précédents travaux tentent de corriger l'intégralité des erreurs OCR, nous avons fait le choix de nous concentrer sur un type de termes particulier : les entités nommées. En effet, comme nous l'avons déjà expliqué dans la sous-section 4.3.3.2, les entités ont un intérêt particulier pour les utilisateurs. De ce fait, elles servent souvent de point d'entrée lors des tâches de recherche d'information [Gooding 2014]. La méthode que nous présenterons dans la section 5.4 se place après de l'étape de détection des entités nommées. Nous allons donc avant tout étudier l'impact du bruit OCR sur la détection des entités nommées.

5.3 Évaluation préalable

Avant de nous consacrer à la correction des entités nommées dans un corpus bruité, nous devons nous assurer de leur détection malgré le bruit OCR. Pour évaluer cet aspect, nous avons testé différents outils disponibles sur deux versions d'un même corpus. La première contient les textes originaux, sans erreur, tandis que la deuxième a été artificiellement bruitée. Nous présentons dans la sous-section suivante la méthodologie utilisée pour la création de la vérité terrain. Puis, nous évaluerons l'impact de l'OCR sur différents outils de reconnaissance des entités nommées.

5.3.1 Création d'une vérité terrain

À notre connaissance, il n'existe pas de corpus d'évaluation qui s'intéresse en même temps aux entités nommées et à la correction post-OCR. On trouve dans la littérature des corpus où les entités nommées sont bien annotées, mais le texte n'est pas bruité. Inversement, il existe des corpus où un texte produit par un processus OCR est aligné avec le texte original, mais les entités nommées ne sont pas annotées. Nous avons donc construit notre propre corpus contenant des entités nommées en contexte bruité. Nous sommes partis d'un corpus existant spécialisé sur les entités nommées. Nous avons choisi le corpus utilisé lors des 3$^{\text{ème}}$ et 4$^{\text{ème}}$ Message Understanding Conference (MUC). Il est constitué de 1297 actualités sur des attaques terroristes en Amérique latine. 1194 entités nommées différentes sont annotées (certaines sont présentes plusieurs fois). Nous construisons la vérité terrain comme suit :

— les textes sont transformés en image

— les images sont dégradées avec une légère rotation, un flou gaussien et quelques tâches. Ce sont des dégradations communes lors de l'utilisation d'un scanner [Farahmand 2013]

— un système OCR est utilisé pour extraire le texte (nous avons utilisé Finereader [1]).

L'exemple du traitement d'un document est présenté dans la figure 5.1. Pour évaluer la qualité d'un texte ayant subi un processus d'OCR, une des mesures les plus populaires [Holley 2009] est le Character Error Rate (CER) qui calcule la proportion d'erreurs commises sur les **caractères** par rapport au texte original. Les erreurs sont de trois types :

1. https://www.abbyy.com/finereader/

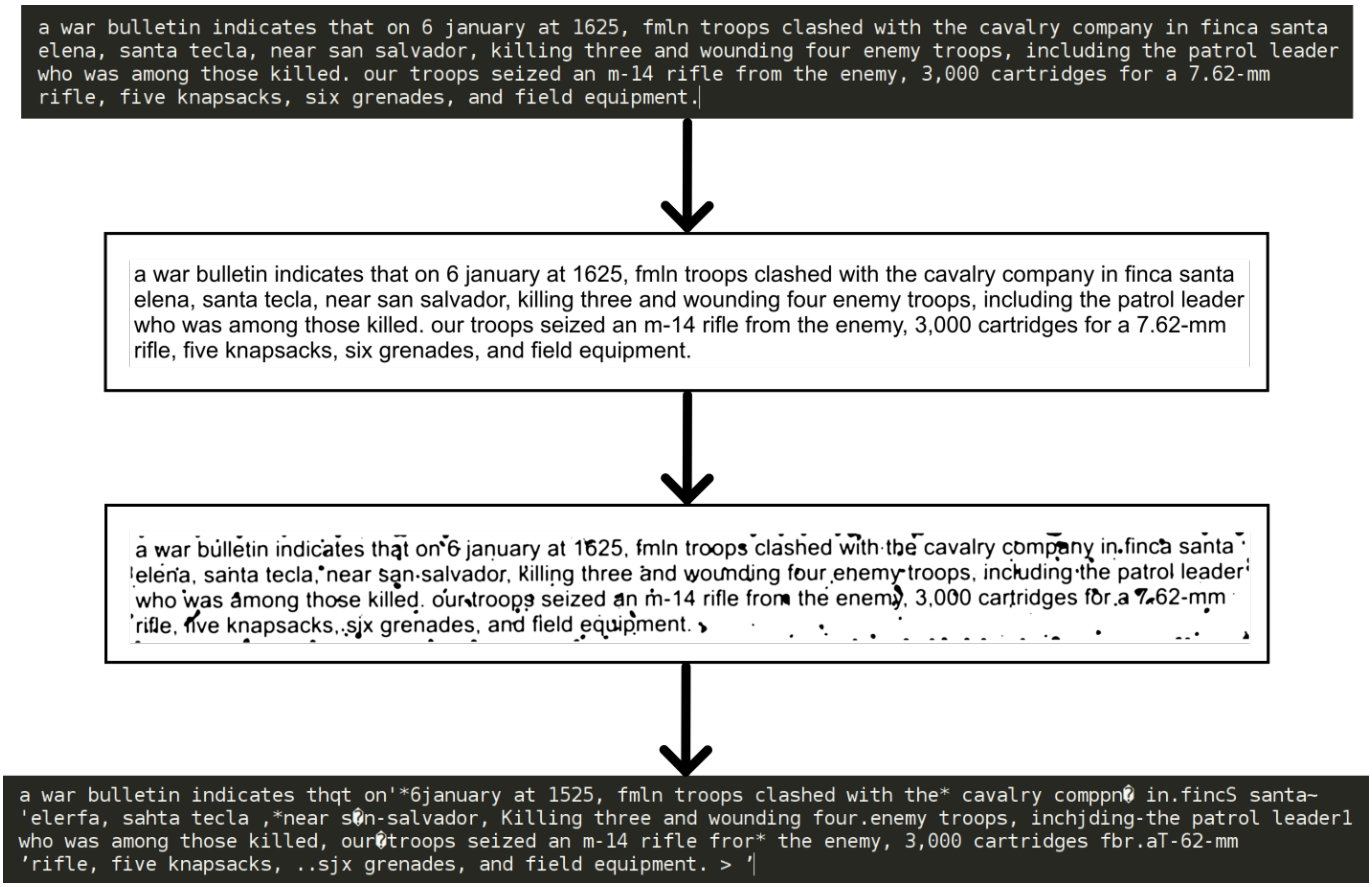
— **la substitution** où un caractère a été inversé avec un autre

— **la suppression** où un caractère n'a tout simplement pas été reconnu par l'OCR

— **l'insertion** où un caractère supplémentaire a été incorrectement ajouté.

Ici, selon les documents, le CER varie entre 2.8% et 23.4%. Le taux d'erreur moyen sur les caractères est de 6.6%, un taux relativement classique selon [Holley 2009]. Cependant, le bruit distribué sur les documents du corpus est homogène. Cela se traduit par un fort taux d'erreur sur les mots. Le Word Error Rate (WER) est une autre mesure utilisée pour traduire la qualité d'un corpus [Lund 2013]. Contrairement au CER, cette mesure calcule le pourcentage de **mots** correctement reconnus. Deux mots sont différents s'ils diffèrent ne serait-ce que d'un caractère. Dans notre cas, le WER s'élève à environ 47%, ce qui signifie que près d'un mot sur deux comporte au moins une erreur.

FIGURE 5.1 – Exemple de création d'un document bruité artificiellement



Le texte original est d'abord transformé en image. Puis, différents types de bruit sont ajoutés pour simuler un document de mauvaise qualité : bruit gaussien, rotation, ajout de tâches. Finalement, cette image est traitée par un moteur OCR.

FIGURE 5.2 – Alignement du texte d'un document original et de sa version bruitée

```
OCR:    a war bulletin indicates thq@t on'*@6@january at 15@25, fmln troops clashed with the* cavalry compp@
GT :    a war bulletin indicates th@at on@@ 6 january at 1@625, fmln troops clashed with the@ cavalry comp@a

OCR:    n@@ in.@fincS@ santa~ 'elerf@a, sah@ta tecla ,*@near s@@n-@salvador, K@illing three and wounding fou
GT :    n@y in@ finc@a santa@ @ele@@na, sa@nta tecla@,@ near s@an@ salvador, @killing three and wounding fou

OCR:    r.@enemy troops, inchj@@ding-@the patrol leader1 who was among those killed,@ our@@troops seized an
GT :    r@ enemy troops, inc@@luding@ the patrol leader@ who was among those killed@. our@ troops seized an

OCR:    m-14 rifle fror*@ the enemy, 3,000 cartridges fb@r.@aT-@@@62-mm 'rifle, five knapsacks, ..sj@x grena
GT :    m-14 rifle fro@@m the enemy, 3,000 cartridges f@or@ a@@ 7.62-mm @rifle, five knapsacks, @@s@ix grena

OCR:    des, and field equipment. > '
GT :    des, and field equipment. @@@
```

Cet alignement fait état des différentes erreurs commises par le processus d'OCR. Les différences entre les deux textes sont indiquées par la présence du caractère "@".

FIGURE 5.3 – Versions bruitées de quelques entités nommées identifiées dans le corpus

```
"regalado":          "paraguay":          "panama city":          "itagui":
    "regaiado"           "parqguay"           "panama* r@ity"         "tagui"
    "regalSdo.'"         "araguay"            "panama pity"           "itagfti'"
    "regafado"                                "panama city,"
```

La vérité terrain doit associer à chaque entité nommée originale un ensemble de mots ou d'expressions produit par le processus OCR. Nous avons donc aligné le texte original de chaque document avec le texte reconnu par l'OCR. L'outil RETAS développé par Yalniz *et. al.* dans [Yalniz 2011] permet de réaliser cette mise en correspondance. L'alignement des versions originale et bruitée d'un document est présenté dans la figure 5.2. Connaissant la position des entités nommées dans le texte original, il est trivial d'identifier les entités nommées reconnues par l'OCR. Au total, 3623 entités nommées dégradées ont été identifiées, chacune étant associée à sa version originale. Des exemples d'entités nommées bruitées sont présentés dans la figure 5.3.

5.3.2 Entités nommées et bruit OCR

De nombreux outils servent à la détection et à la classification des entités nommées. Comme nous l'avons dit dans la sous-section 4.3.3.2, il existe deux grandes stratégies pour la détection des entités nommées dans un texte : l'utilisation de règles ou l'apprentissage automatique. En présence de bruit OCR, les méthodes qui utilisent la première stratégie sont très désavantagées. En effet, elles n'ont pas la souplesse suffisante pour passer outre les dégradations générées par l'extraction du texte. En revanche, certaines méthodes y sont beaucoup moins sensibles. Nous allons ici analyser l'impact que peut avoir le bruit OCR sur la détection des entités

nommées. Pour cela, trois modèles différents seront testés :

— IBM Watson [2] : anciennement AlchemyAPI, ce service permet d'annoter automatiquement des textes pour en extraire les entités nommées. Parce que c'est un service propriétaire, nous avons peu d'informations sur le corpus d'apprentissage et la méthode utilisée. Nous savons simplement qu'elle fait usage du *Deep Learning*.

— StanfordNER [Finkel 2005] : cet outil fait usage des CRF (*Conditionnal Random Fields*). Le modèle a été entraîné sur plusieurs corpus publics (MUC-6, MUC-7, ACE-2002, *etc.*).

— NeuroNER [Dernoncourt 2017] : ce programme fait usage des réseaux de neurones pour la détection et la reconnaissance des entités nommées. Le modèle par défaut a été entraîné sur le corpus CoNLL-2003. Nous avons également entrainé un autre modèle sur ce même corpus, sans prendre en compte la casse.

La vérité terrain que nous avons construite dans la section précédente nous sert de base pour l'évaluation. Nous testons les capacités de différents systèmes à identifier les entités nommées dans le corpus original et le corpus bruité. Les différents résultats sont présentés dans le tableau 5.1.

Tableau 5.1 – Impact de l'OCR sur la détection des entités nommées.

Modèle	Watson			Stanford NER			NeuroNER					
Corpus d'apprentissage	Corpus propriétaire			MUC-6, MUC-7, ACE-2002, autre			CoNLL-2003			CoNLL-2003 (caseless)		
Corpus cible	original	bruité	Δ	original	bruité	Δ	original	bruité	Δ	original	bruité	Δ
Rappel	1.66	2.50	0.84	55.72	37.62	-18.10	1.07	8.89	7.82	67.36	58.58	-8.78
Précision	3.03	3.44	0.41	37.07	33.89	-3.78	4.83	10.12	5.29	30.97	17.47	-13.50
F1	1.83	2.57	0.74	41.24	32.44	-8.80	1.62	8.28	6.66	40.04	25.48	-14.56

Différents modèles sont ici testés pour détecter les entités nommées d'un corpus. Les scores (en pourcentage) sont exprimés pour le corpus original et sa version bruitée générée par un processus d'OCR. La valeur Δ correspond à la dégradation ou l'amélioration des capacités de reconnaissance du système testé sur le corpus.

Les résultats obtenus par l'outil propriétaire Watson ou le modèle NeuroNER entraîné sur le corpus CoNLL-2003 sont faibles. Étonnamment, ces deux méthodes obtiennent un meilleur score lorsque le corpus est bruité. Cela dit, ces scores sont presque tous en dessous de 10%. Il est difficile de trouver une explication pour les faibles résultats de Watson car aussi bien le corpus d'apprentissage que la méthode utilisée sont inconnus. En ce qui concerne les scores obtenus par NeuroNER (entraîné avec le corpus CoNNL-2003 original), ils s'expliquent par le fait que le corpus de test contient des caractères avec une casse unique. En revanche, lorsque l'on entraîne ce

2. https://www.ibm.com/watson/services/natural-language-understanding/

même modèle avec un corpus sans casse, les résultats sont bien meilleurs. Comme on peut le voir avec ce dernier modèle ou le modèle développé par Stanford, le bruit dans un texte a un impact non négligeable sur la détection d'entités nommées. Cela dit, le corpus de test étant particulièrement bruité (avec un WER de près de 50%), les scores de détection d'entités nommées restent très honorables.

Si les moteurs indexent les textes de manière assez poussée (voir section 2.3), ils n'opèrent pas de traitement particulier pour certaines erreurs de reconnaissance. Ce constat est d'autant plus vrai sur les entités nommées qui nécessitent souvent un traitement spécifique (pas de *stemming*, présence dans un dictionnaire ou une base de connaissance, *etc.*). Pour ne rien arranger, il se trouve que les entités nommées sont le premier points d'entrée des utilisateurs dans un système de recherche [Gefen 2015]. Pour cette raison, on peut leur attribuer une plus grande valeur sémantique que la plupart des autres mots. Pour améliorer la qualité des recherches des utilisateurs dans un système, il est nécessaire de s'assurer de la qualité de ces termes particuliers que sont les entités nommées. Pour cette raison, nous avons développé une méthode de correction automatique des entités nommées dans un corpus bruité.

5.4 Présentation de la méthode

Cette section présente une méthode pour automatiquement corriger les erreurs de reconnaissance induites par le processus d'OCR sur les entités nommées. À cause des erreurs produites, certaines entités nommées identiques dans le document original se retrouvent modifiées. L'objectif de la méthode que nous avons développée est d'identifier les entités nommées similaires malgré le bruit présent dans le texte. La chaîne de traitement globale est présentée dans la figure 5.4, et chacune de ses étapes seront détaillées dans ce chapitre. Intuitivement, dans le contexte de notre problématique, deux entités nommées sont dites similaires si :

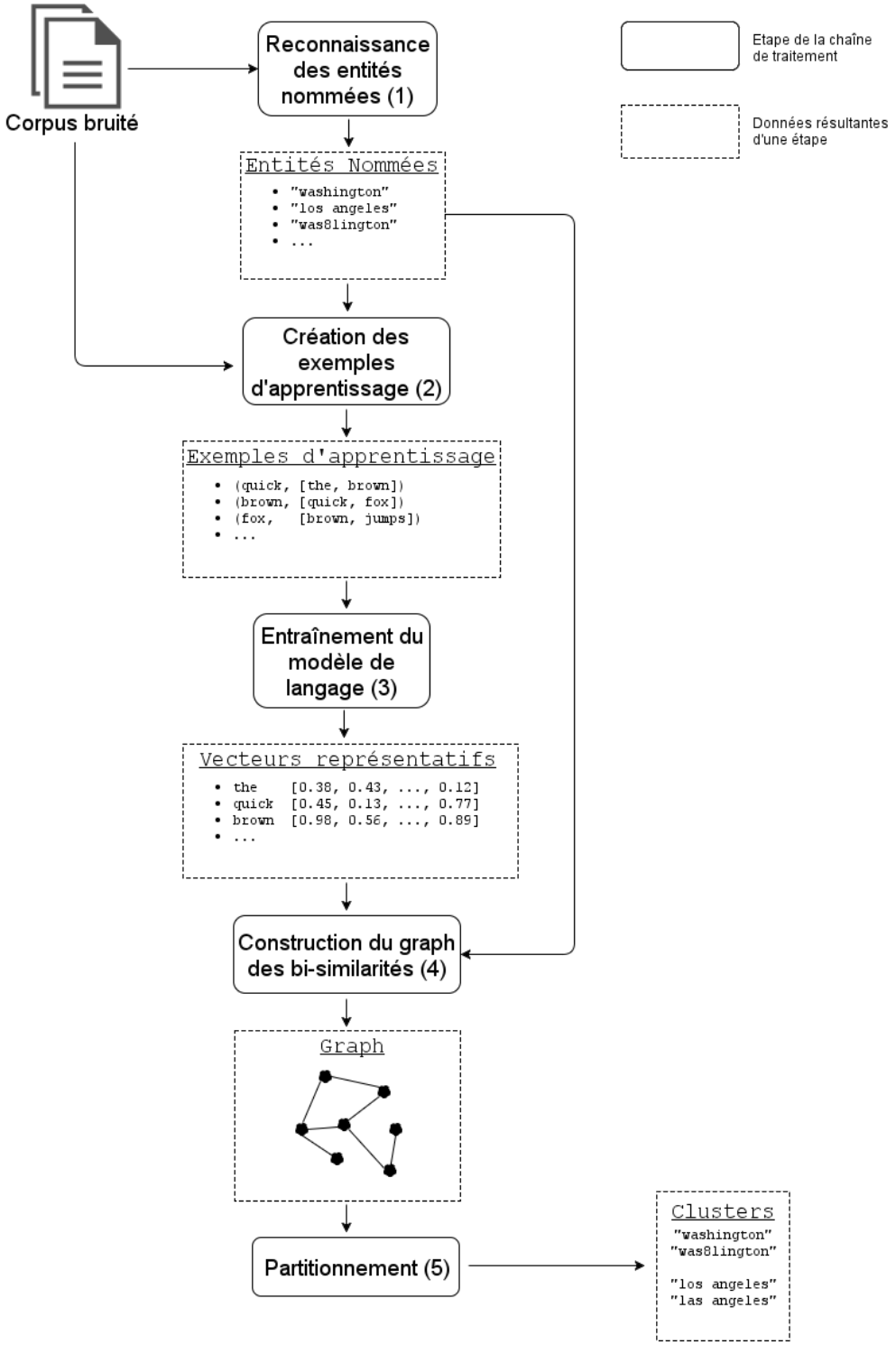
1. elles apparaissent dans un contexte similaire (les mêmes mots les entourent)
2. elles sont à peu près composées des mêmes lettres aux mêmes endroits

La première condition résulte de **l'hypothèse distributionnelle** (voir section 4.3.4). La deuxième se base sur l'hypothèse que la majorité des erreurs OCR ne modifient qu'une faible part des caractères de chaque mot. L'ordre de ces conditions nous semble important car deux entités qui désignent la même chose apparaissent certainement dans un contexte commun. Elles sont ensuite plus ou moins différentes selon les lettres qui les composent. Au contraire, deux entités qui se ressemblent mais dont les contextes n'ont rien de commun ont peu de chance de représenter un même concept.

5.4.1 Entraînement du modèle contextuel de langage

Imaginons un corpus de textes issus d'un processus d'OCR. Nous partons du principe que les entités nommées ont été automatiquement détectées (grâce à l'utilisation d'un des modèles présentés dans la section précédente par exemple). Nous nous situons donc à l'étape (2) de la chaîne de traitement présentée dans la figure 5.4.

FIGURE 5.4 – Chaîne de traitement pour corriger les entités nommées dans un corpus bruité



Les boîtes en trait plein représentent chacune une étape du traitement, tandis que les boîtes en trait pointillé sont les données en sortie des différentes étapes.

Les outils récents comme WORD2VEC ou GLOVE (voir section 4.4.2) sont capables d'apprendre de manière non supervisée des caractéristiques représentatives d'un contexte donné. WORD2VEC est un modèle prédictif qui utilise un réseau de neurones pour apprendre les vecteurs représentatifs de chaque mot. GLOVE est un modèle statistique fondé sur l'exploitation d'une matrice de co-occurrence. Les vecteurs représentatifs des mots sont extraits après qu'un algorithme de réduction de dimensionnalité a été appliqué sur la matrice. Bien que ces deux modèles soit fondamentalement différents, ils représentent de manière assez similaire le contexte des mots. Nous avons choisi d'utiliser WORD2VEC mais nous aurions pu atteindre les mêmes résultats avec GLOVE.

Considérons le mot "Washington" qui apparaît à différents endroits d'un corpus. Il y a de grandes chances que les contextes autour des différentes occurrences du mot soit semblables. Par exemple, on y trouvera souvent les mots "D.C.", "capitol" ou "white house". Dans le cas d'un corpus traité par un système OCR, il est très probable que certaines occurrences du mot "Washington" aient été mal reconnues, par exemple "Washingtan" ou "Was8lngton". Bien que ces mots soient différents, leurs contextes respectifs possèderont toujours des similarités. Ce sont ces similarités qui sont capturées par le vecteur représentatif des mots.

Nous avons configuré le modèle WORD2VEC pour utiliser l'architecture *Skip-Gram* (voir section 4.4.2.2). En effet, Mikolov *et. al.* prétendent que cette architecture fonctionne bien avec relativement peu d'exemples d'entraînement et est capable de correctement représenter le contexte des mots plus rares. Nous avons utilisé pour nos expériences les paramètres par défaut : la taille des vecteurs est fixée à 300, le negative sampling à 10 et le modèle est entrainé sur 30 itérations. Pour s'assurer de capturer les similarités des contextes de certaines entités malgré le bruit, nous avons fixé la taille du contexte d'apprentissage à 15 (la valeur par défaut est de 5). Cela nous permet d'augmenter la chance de trouver, malgré le bruit, des mots de contexte similaire entre deux entités nommées. Certaines entités nommées sont cependant composées de plusieurs mots. Nous avons donc préalablement fusionné ces entités en remplaçant les espaces par des tirets-bas "_" pour qu'elle soient considérées comme un seul objet par le modèle. Une fois l'entraînement terminé, chaque entité nommée est associée à un vecteur représentatif. Nous nous situons alors à la fin de l'étape (3) présentée dans la figure 5.4.

5.4.2 Définition d'une bi-similarité lexicographique

Notre objectif est maintenant de déterminer un moyen de calculer la similarité entre entités nommées selon deux axes : une similarité contextuelle et une similarité d'édition.

5.4.2.1 Similarité contextuelle

Soit $\mathbb{E}$ l'ensemble des entités nommées reconnues. Grâce au modèle de langage, nous sommes capables de calculer la similarité contextuelle sim_{ctx} entre deux entités nommées e_i et e_j en comparant leurs vecteurs représentatifs $\vec{e_i}$ et $\vec{e_j}$. La distance

cosinus est très souvent utilisée pour comparer deux vecteurs [Mikolov 2013a]. Elle peut varier de -1 (les vecteurs sont totalement différents) à 1 (les vecteurs sont exactement égaux). Elle est définie comme suit.

$$\begin{aligned} cosine &: \mathbb{E} \times \mathbb{E} \to [-1, 1] \\ (e_i, e_j) &\mapsto cosine(e_i, e_j) = \frac{\vec{e_i}.\vec{e_j}}{\|\vec{e_i}\| \times \|\vec{e_j}\|} \end{aligned} \tag{5.1}$$

Il est nécessaire de normaliser cette similarité entre 0 et 1 pour se conformer aux propriétés des opérateurs de comparaison définis ci-après. La similarité contextuelle sim_{ctx} entre deux entités nommées devient donc :

$$\begin{aligned} sim_{ctx} &: \mathbb{E} \times \mathbb{E} \to [0, 1] \\ (e_i, e_j) &\mapsto sim_{ctx}(e_i, e_j) = \frac{\vec{e_i} \cdot \vec{e_j} + \|\vec{e_i}\| \times \|\vec{e_j}\|}{2 \times \|\vec{e_i}\| \times \|\vec{e_j}\|} \end{aligned} \tag{5.2}$$

5.4.2.2 Similarité d'édition

Il se trouve que deux mots dérivés d'une même entité nommée ont plus en commun que leur seul contexte. En effet, selon la qualité de la reconnaissance effectuée par le processus d'OCR, ces deux mots seront plus ou moins semblables et possèderont typiquement une grande proportion de caractères en commun. La distance d'édition qui les sépare sera donc faible. Nous définissons donc une similarité d'édition sim_{edit} fondée sur la distance de Levenshtein [Levenshtein 1966] qui compte le nombre de suppression, d'ajouts et de substitutions entre deux chaînes de caractères.

$$\begin{aligned} sim_{edit} &: \mathbb{E} \times \mathbb{E} \to [0, 1] \\ (e_i, e_j) &\mapsto sim_{edit}(e_i, e_j) = 1 - \frac{Levenshtein(e_i, e_j)}{max(length(e_i), length(e_j))} \end{aligned} \tag{5.3}$$

5.4.2.3 Combinaison des deux similarités

La combinaison de ces deux similarités donne naissance à notre mesure de bi-similarité notée $\mathbb{S}$, renvoyant un couple de valeurs et définie comme suit.

$$\begin{aligned} \mathbb{S} &: \mathbb{E} \times \mathbb{E} \to [0, 1] \times [0, 1] \\ (e_i, e_j) &\mapsto \mathbb{S}(e_i, e_j) = (sim_{ctx}(e_i, e_j), sim_{edit}(e_i, e_j)) \end{aligned} \tag{5.4}$$

Parce que nous cherchons avant tout à identifier des termes qui parlent "de la même chose", il nous semble important de privilégier la similarité contextuelle par rapport à la similarité d'édition. En effet, deux termes semblables mais qui n'apparaissent pas dans le même contexte ont peut de chance de référencer la même entité. Pour définir un opérateur ordonné capable de cela, nous devons être en mesure d'évaluer des extrema locaux. C'est-à-dire que nous devons définir un moyen de comparer deux paires de mesures pour identifier la "plus grande" ou la "plus petite".

Nous définissons donc les fonctions lexicographiques lex_{min} et lex_{max}.

$$\begin{array}{rcl} ([0,1]\times[0,1])^2 & \rightarrow & [0,1]\times[0,1] \\ ((x,y),(x',y')) & \mapsto & lex_{min}((x,y),(y',y')) = \\ & & \begin{cases} (x,y) & if\ x < x' \vee (x = x' \wedge y < y') \\ (x',y') & otherwise. \end{cases} \end{array} \tag{5.5}$$

$$\begin{array}{rcl} ([0,1]\times[0,1])^2 & \rightarrow & [0,1]\times[0,1] \\ ((x,y),(x',y')) & \mapsto & lex_{max}((x,y),(x',y')) = \\ & & \begin{cases} (x,y) & if\ x > x' \vee (x = x' \wedge y > y') \\ (x',y') & otherwise. \end{cases} \end{array} \tag{5.6}$$

Les deux opérateurs lex_{min} et lex_{max} bénéficient de propriétés algébriques qui les rendent appropriés à la comparaison. Ces opérateurs sont tous les deux commutatifs, associatifs et monotones. L'élément neutre de lex_{min} (respectivement lex_{max}) est $(1,1)$ (respectivement $(0,0)$) et l'élément absorbant est $(0,0)$ (respectivement $(1,1)$). Les preuves de ces différentes propriétés sont présentées dans l'Annexe B.1.

Nous avons fait le choix conceptuel de considérer en priorité la similarité contextuelle par rapport à la similarité d'édition. En effet deux entités nommées mal reconnues mais issues d'un même mot apparaitront dans un contexte similaire, même si elles diffèrent de plusieurs caractères. C'est la raison pour laquelle nous avons défini des opérateurs d'ordre lexicographique dans les équations (5.5) et (5.6).

Nous aurions aussi pu combiner ces deux types de similarité grâce à une moyenne pondérée. Cependant, les résultats produits auraient été incompatibles avec nos objectifs. Le plus gros problème, ici, serait le choix des poids associés aux deux similarités. La similarité d'édition est stable et associera toujours la même valeur à deux même mots. En revanche, la similarité contextuelle dépend avant tout du modèle de langage utilisé. Selon les paramètres du modèle et le corpus utilisé pendant l'apprentissage (sa taille, le type de documents qu'il contient, *etc.*), les vecteurs associés à chaque mot peuvent varier. Pour cette raison, il serait difficile de choisir des poids par défaut qui auraient un sens. Il faudrait au contraire adapter ces poids selon le corpus étudié, ce qui nuirait à la généricité de notre méthode.

Grâce aux fonctions lex_{min} et lex_{max} décrites précédemment, nous pouvons définir deux opérateurs capables de comparer les similarités entre deux paires d'entités nommées : $\preceq_{lex}$ pour *"plus petit ou égal à"* et son dual $\succeq_{lex}$ pour *"plus grand ou égal à"*. Soit e_i, e_j, e'_i et e'_j des entités nommées dans $\mathbb{E}$.

• On dit que la bi-similarité du couple (e_i, e_j) est plus petite ou égale à la bi-similarité du couple (e'_i, e'_j), c'est-à-dire $\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)$ si et seulement si $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$. Formellement :

$$\begin{array}{r} \forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j) \Leftrightarrow \\ lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j) \end{array} \tag{5.7}$$

• On dit que la bi-similarité du couple (e_i, e_j) est plus grande ou égale à la bi-similarité du couple (e'_i, e'_j), c'est-à-dire $\mathbb{S}(e_i, e_j) \succeq_{lex} \mathbb{S}(e'_i, e'_j)$ si et seulement si $lex_{max}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$. Formellement :

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \mathbb{S}(e_i, e_j) \succeq_{lex} \mathbb{S}(e'_i, e'_j) \Leftrightarrow lex_{max}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j) \tag{5.8}$$

Les opérateurs $\succeq_{lex}$ et $\preceq_{lex}$ sont tous les deux antisymétriques et transitifs. Ils représentent chacun un ordre total sur $\mathbb{E}$. Les preuves de ces propriétés sont présentées dans l'Annexe B.2.

5.4.3 Construction du graphe des entités nommées similaires

Nous nous situons maintenant à l'étape (4) présentée dans la figure 5.4. Le graphe G_n^t que nous construisons est un graphe pondéré non-orienté. Il connecte des entités nommées (les sommets V du graphe) selon l'opérateur de bi-similarité $\mathbb{S}$. L'ensemble L contient les liens existant entre les sommets du graphe. Deux sommets du graphe seront connectés si et seulement si les entités qu'ils représentent sont contextuellement similaires et si leur similarité d'édition (voir section 5.4.2.2) est au dessus d'un seuil t. La méthodologie utilisée pour construire ce graphe est présentée dans l'algorithme 1.

Algorithme 1 Construction du graphe

Précondition : $\mathbb{E}$: Ensemble d'entités nommées, n : nombre d'entités contextuellement similaires à prendre en compte, t : seuil pour la création de liens.

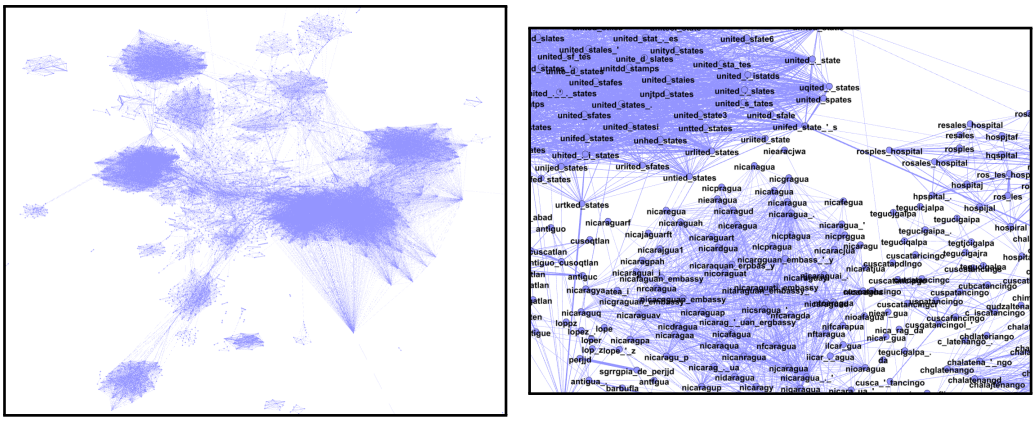
Postcondition : $G_n^t(V, L, W)$: graphe des entités nommées tel que V est l'ensemble des sommets, L est l'ensemble des liens et W est une fonction qui associe un poids à chaque lien.

```
1: V ← 𝔼
2: pour tout e_i ∈ 𝔼 faire
3:    Sim ← ∅
4:    pour tout e_j ∈ 𝔼 − {e_i} faire
5:       Sim ← Sim ∪ 𝕊(e_i, e_j)
6:    LexSort(Sim, ≻_lex)
7:    pour tout (x, y) ∈ Top_n(Sim, lex_max) faire
8:       si y ≥ t alors
9:          L ← L ∪ Edge(e_i, e_j)
10:         W(e_i, e_j) ← y
```

Pour chaque entité nommée $e_i \in \mathbb{E}$, on sélectionne le sous-ensemble $E_{e_i}^n \subseteq \mathbb{E}$ des n entités nommées contextuellement les plus similaires à e_i selon l'opérateur lex_{max}. Cet ensemble est par la suite appelé Top_n. Un lien pondéré est créé entre deux sommets e_i et e_j, avec $e_j \in E_{e_i}^n$, si $sim_{edit}(e_i, e_j) \geq t$, ou t est un seuil fixé expérimentalement. sim_{edit} est aussi utilisé comme poids du lien dans le graphe.

Ce poids jouera un rôle important dans l'étape de clustering détaillée dans la sous-section 5.4.4. Les paramètres n et t permettent de limiter la taille du graphe et seront étudiés lors de la phase d'évaluation dans la section 5.5. Le graph associé à notre vérité terrain est présenté dans la figure 5.5.

FIGURE 5.5 – Graphe des entités nommées similaires.



La vignette de gauche représente une vision globale du graphe des entités nommées similaires. Sa structure est formée d'îlots de différentes tailles. La vignette de droite présente un détail de ce graphe. On peut y notamment y distinguer les variations des entités "United States" ou encore "Nicaragua".

5.4.4 Clustering

Une fois le graphe construit, nous appliquons un algorithme de clustering pour rassembler les sommets représentant la même entité nommée. Le choix de l'algorithme est basé sur différentes contraintes :

— le nombre de clusters est inconnu *a priori*. Dans un monde parfait, nous pourrions extraire autant de clusters que d'entités nommées présentes dans le corpus original. Chacune de ces entités possédant un nombre variable de représentants bruités.

— un même mot doit pouvoir appartenir à plusieurs clusters. En effet, un mot généré par un processus d'OCR peut tout à fait être dérivé de deux mots originaux différents. Par exemple, le mot "alite" peut avoir été reconnu à partir des mots "alive" ou "alike" et doit donc être présent dans les deux clusters.

Ces hypothèses nous ont conduit à choisir l'algorithme suivant : weighted Clique Percolation Method (CPMw) [Palla 2005]. La version non pondérée de cet algorithme permet de calculer des communautés au sein d'un graphe en calculant des k-cliques, qui sont des sous-graphes complets possédant k sommets. On dit que deux k-cliques sont adjacentes si et seulement si elles partagent $k-1$ sommets. Les k-cliques adjacentes sont rassemblées pour former les communautés. La version pondérée de cet algorithme de clustering rajoute simplement une condition sur la validité d'une k-clique : la moyenne géométrique des poids de cette k-clique doit être supérieure

à un seuil pour que cette k-clique soit considérée comme valide. Nous avons utilisé la version "classique" de cet algorithme. Certaines zones du graphe étant fortement connectées, la complexité des calculs devient vite un problème. L'algorithme s'adapte en limitant le temps passé à traiter chaque sommet du graphe. Il serait intéressant de tester les améliorations apportées par une version "distribuée" de cet algorithme.

5.5 Évaluation

Nous présentons dans cette section la méthodologie que nous avons utilisée pour évaluer notre méthode. Notre objectif principal est d'identifier et de rassembler, au sein d'un corpus bruité, les entités nommées représentant la même chose. Nous utiliserons le même corpus d'évaluation que précédemment. Sa construction est décrite dans la section 5.3.1.

5.5.1 Évaluation du clustering

L'objectif de notre méthode est d'identifier parmi des entités nommées bruitées, celles qui se réfèrent à la même chose. Par exemple, les mots "Washingtan" et "Was8lington" correspondent tous les deux à la même entité nommée : "Washington". Ce problème est semblable à une tâche classique du traitement naturel du langage : la résolution de coréférences. C'est-à-dire identifier les pronoms ou expressions qui se réfèrent à la même entité nommée (ex : **Le chat** est gros, **il** mange trop !). Dans les deux cas, on dispose d'un ensemble de mots ou d'expressions — les entités nommées bruitées dans notre cas, des entités et des pronoms dans le cas de la résolution de coréférences. L'objectif est de trouver une partition de cet ensemble rassemblant les éléments qui se réfèrent à la même entité. Cette tâche est évaluée en comparant la sortie du système avec une vérité terrain. Il existe différentes mesures pour évaluer la qualité d'un clustering. Maidasani *et. al.* ont comparé dans [Maidasani 2012] différentes mesures régulièrement utilisées. Les plus communes sont présentées ici dans l'ordre de leur découverte.

5.5.1.1 Mesure MUC

Cette mesure, créée en 1995 dans [Vilain 1995], a été introduite lors de la 6$^{\text{ème}}$ Message Understanding Conference d'où elle tire son nom. Les clusters sont ici considérés comme un ensemble de liens entre entités. Si deux entités sont liées, elles appartiennent au même cluster. La mesure MUC calcule le nombre de modifications nécessaires (ajout ou suppression de liens) pour rendre la sortie du système identique à la vérité terrain. Le calcul de cette mesure se base sur la fonction $partition(c, S)$ (voir équation 5.9). Cette dernière calcule et retourne à partir d'un cluster c et d'un ensemble de clusters S les clusters dans S qui intersectent c.

$$partition(c, S) = \{s \mid s \in S \wedge s \cap c \neq 0\} \tag{5.9}$$

Soient T l'ensemble des clusters de la vérité terrain et R l'ensemble des clusters retourné par le système. Le calcul de la précision MUC se base sur le nombre de liens manquants dans les clusters du système par rapport à ceux de la vérité terrain.

Formellement, cette mesure de précision est calculée comme une somme sur chaque cluster retourné par le système (voir équation 5.10).

$$MUC_{precision}(T,R) = \sum_{r \in R} \frac{|r| - |partition(r,T)|}{|r| - 1} \tag{5.10}$$

De la même manière, le rappel MUC est calculé comme le nombre de liens manquants dans les clusters de la vérité terrain par rapport aux clusters du système. La définition de cette mesure est donnée dans l'équation 5.11.

$$MUC_{rappel}(T,R) = \sum_{t \in T} \frac{|t| - |partition(t,R)|}{|t| - 1} \tag{5.11}$$

Finalement, la mesure finale est la moyenne harmonique de la précision et du rappel (voir équation 5.12).

$$MUC_{F_1} = 2 \times \frac{MUC_{precision} \times MUC_{rappel}}{MUC_{precision} + MUC_{rappel}} \tag{5.12}$$

Le principal problème de cette mesure est qu'elle ne considère pas le nombre d'entités au sein d'un cluster. Reprenons l'exemple fourni dans [Bagga 1998], avec une vérité terrain et deux résultats à évaluer.

— Vérité terrain : $\{e_1, e_2, e_3, e_4, e_5\}$ $\{e_6, e_7\}$ $\{e_8, e_9, e_{10}, e_{11}, e_{12}\}$
— Résultat n°1 : $\{e_1, e_2, e_3, e_4, e_5\}$ $\{e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\}$
— Résultat n°2 : $\{e_1, e_2, e_3, e_4, e_5, e_8, e_9, e_{10}, e_{11}, e_{12}\}$ $\{e_6, e_7\}$

Dans le premier résultat, on remarque que les entités e_6 à e_{12} ont été identifiées à tort comme équivalente. Finalement, 7 entités ne sont pas correctement rassemblées. Dans le deuxième résultat, ce sont 10 entités qui sont incorrectement regroupées. Or, ces deux résultats obtiennent le même score quand ils sont comparés à la vérité terrain. Il serait donc intéressant d'utiliser une autre mesure pour contrôler la qualité d'une partition de l'ensemble des entités.

5.5.1.2 Mesure B^3

Cette mesure est très populaire dans le champ de recherche du traitement automatique du langage naturel. Bagga *et. al.* ont introduit cette mesure en 1998 dans [Bagga 1998]. B^3 tente de corriger certains défauts inhérents à MUC. Au lieu de se concentrer sur l'absence ou la présence de liens entre entités, B^3 préfère s'intéresser à la présence ou l'absence d'une entité par rapport aux autres entités présentes dans le cluster. Ainsi, un score de précision et de rappel est calculé pour chaque entité. Ces scores sont ensuite combinés pour produire un résultat final. Soient n le nombre total d'entités dans la vérité terrain et une fonction $results(t_i)$ qui prend une entité t_i en entrée et qui retourne l'ensemble des clusters du système qui la contiennent. La définition de la précision B^3 et du rappel B^3 sont respectivement définis dans les équations 5.13 et 5.14.

$$B^3_{precision}(T,R) = \frac{1}{n} \sum_{t \in T} \sum_{t_i \in t} \sum_{r \in results(t_i)} \frac{|r \cap t|}{|r|} \tag{5.13}$$

$$B^3_{rappel}(T,R) = \frac{1}{n}\sum_{t\in T}\sum_{t_i\in t}\sum_{r\in results(t_i)}\frac{|r\cap t|}{|t|} \tag{5.14}$$

Comme pour la mesure MUC définie précédemment, la mesure finale est la moyenne harmonique de la précision et du rappel (voir équation 5.15).

$$B^3_{F_1} = 2\times\frac{B^3_{precision}\times B^3_{rappel}}{B^3_{precision}+B^3_{rappel}} \tag{5.15}$$

Ainsi, un score (précision et rappel) est calculé pour chaque entité. Ces scores sont ensuite combinés pour obtenir une mesure finale. Le principal reproche fait à cette mesure est qu'elle ne permet pas d'évaluer le résultat d'un système s'il contient des entités qui ne sont pas présentes dans la vérité terrain (elles sont appelées *"twin-less mentions"*). Différentes variations de cette mesure ont été créées pour parer ce problème. Cependant, ce cas ne nous concerne pas car les entités du corpus sont considérées comme un point d'entrée de notre méthode et sont donc connues.

5.5.1.3 Mesure CEAF

Si la mesure B^3 améliore certains défauts de MUC, elle n'en est pas exempte pour autant. Ainsi, Luo *et. al.* ont critiqué cette dernière car elle utilise les clusters plusieurs fois lors du calcul de la précision et du rappel. Ils ont donc défini une nouvelle mesure dans [Luo 2005] appelée Constrained Entity Alignment F-measure. La première étape pour calculer cette mesure est de trouver l'alignement optimal des clusters de la vérité terrain avec les clusters calculés par le système à évaluer. Cet alignement est réalisé en maximisant la similarité entre les clusters, avec la contrainte d'une relation bijective entre les clusters de la vérité terrain et ceux du système. Pour cela, on définit la fonction $m(r)$ qui prend en entrée un cluster r et renvoie le cluster dans T qui assure le meilleur alignement. Il est bon de noter que tous les clusters du système ne possède pas nécessairement d'alignement optimal. Dans ce cas, la fonction $m(r)$ retourne l'ensemble vide.

En plus de cet alignement optimal, il faut être capable de calculer la similarité entre deux clusters. Les auteurs ont défini pour cela quatre mesures différentes définies par les équations 5.16, 5.17, 5.18 et 5.19.

$$\phi_1(t,r) = \begin{cases} 1, & \text{si } r = t \\ 0, & \text{sinon} \end{cases} \tag{5.16}$$

$$\phi_2(t,r) = \begin{cases} 1, & \text{si } r\cap t \neq 0 \\ 0, & \text{sinon} \end{cases} \tag{5.17}$$

$$\phi_3(t,r) = |r\cap t| \tag{5.18}$$

$$\phi_4(t,r) = 2\times\frac{|r\cap t|}{|r|+|t|} \tag{5.19}$$

Les deux premières mesures, ϕ_1 et ϕ_2, sont très simples mais ne possèdent pas une granularité suffisante pour être vraiment intéressantes. C'est pour cette raison que

ϕ_3 et ϕ_4 sont plus souvent utilisées. Elles comptabilisent le nombre d'éléments en commun dans les deux clusters. Ces deux mesures sont respectivement une mesure absolue et une mesure relative à la taille des clusters. Finalement, les mesures de précision et de rappel sont définies par les équations 5.20 et 5.21. La mesure finale est comme précédemment calculée comme étant la moyenne harmonique de la précision et du rappel (voir 5.22).

$$CEAF\phi_{i_{precision}}(T,R) = \frac{\arg\max_m \sum_{r\in R} \phi_i(r, m(r))}{\sum_{r\in R} \phi_i(r,r)} \tag{5.20}$$

$$CEAF\phi_{i_{rappel}}(T,R) = \frac{\arg\max_m \sum_{r\in R} \phi_i(r, m(r))}{\sum_{t\in T} \phi_i(t,t)} \tag{5.21}$$

$$CEAF\phi_{i_{F_1}}(T,R) = 2 \times \frac{CEAF\phi_{i_{precision}} \times CEAF\phi_{i_{rappel}}}{CEAF\phi_{i_{precision}} + CEAF\phi_{i_{rappel}}} \tag{5.22}$$

Avec de telles mesures, un système qui partitionnerait trop l'ensemble de départ sera pénalisé sur la précision. Inversement, un système qui ne partitionnerait pas assez sera pénalisé sur le rappel. Différentes variantes de cette mesure ont été définies par Luo *et. al.*. Les deux principales sont $CEAF$-ϕ_3 et $CEAF$-ϕ_4. La première est une mesure absolue qui se concentre sur la qualité des éléments à l'intérieur des clusters, c'est-à-dire la quantité d'éléments correctement rattachés à leur représentant. La deuxième est une mesure plus relative qui préfère mettre en avant les clusters dans leur ensemble en comparant leur nombre avec le nombre original d'entités.

5.5.1.4 Mesure combinée

Les mesures décrites précédemment ne sont pas parfaites et capturent chacune un aspect du problème. Pour cette raison, il est intéressant de les combiner. C'est ce qui a été fait lors d'une compétition sur la résolution de coréférences organisée en 2011 pour la conférence CoNLL (Conference on Computational Natural Language Learning). La mesure finale utilisée pour comparer les différents travaux est la moyenne des trois mesures précédentes [3]. Finalement, nous avons appliqué ces différentes mesures sur les clusters obtenus par notre méthode pour différents paramètres n et t. Les résultats sont présentés dans le tableau 5.2.

Comme nous l'avons déjà expliqué, chacune des mesures utilisées s'intéressent à un aspect particulier du problème de l'évaluation du clustering. Ainsi, pour les mêmes paramètres n et t, les scores de précision et de rappel de chaque mesure diffèrent. Nous allons donc nous concentrer sur la mesure finale **CoNLL** qui est la moyenne des autres mesures du tableau. La qualité des clusters par rapport à une vérité terrain est donc évaluée grâce à la précision moyenne et au rappel moyen. La précision correspond en quelque sorte à la qualité des éléments au sein d'un cluster. Le rappel permet d'évaluer si tous les éléments qui devaient être rassemblés le sont effectivement. Le meilleur score est obtenu lorsque $n = 200$ et $t = 0.85$.

3. http://conll.cemantix.org/2011/faq.html

TABLEAU 5.2 – Èvaluation par différentes mesures du clustering des entités nommées et de la qualité de la correction du corpus bruité

n	100						200						300						400						500					
t	.65	.70	.75	.80	.85	.90	.65	.70	.75	.80	.85	.90	.65	.70	.75	.80	.85	.90	.65	.70	.75	.80	.85	.90	.65	.70	.75	.80	.85	.90
MUC																														
Précision	81.03	86.64	86.93	**88.09**	86.49	79.93	80.00	84.52	85.68	87.50	85.96	79.48	78.88	84.01	85.17	87.60	86.10	79.37	80.24	83.89	85.04	87.79	86.13	79.10	80.72	83.91	85.06	87.71	86.10	78.96
Rappel	12.51	19.11	20.73	20.86	29.54	18.08	10.04	16.68	20.12	22.29	**32.26**	20.53	10.32	16.87	18.69	22.81	24.90	21.50	9.84	16.92	18.56	23.36	25.12	21.96	9.93	16.87	18.78	22.73	24.65	22.11
F_1	21.68	31.31	33.48	33.74	44.04	29.49	17.84	27.86	32.59	35.53	**46.91**	32.64	18.26	28.10	30.66	36.20	38.62	33.83	17.53	28.16	30.47	36.90	38.89	34.38	17.68	28.10	30.77	36.10	38.33	34.55
B^3																														
Précision	82.05	85.74	86.61	**88.91**	78.05	83.22	82.96	85.40	86.56	87.66	75.24	79.72	81.36	83.87	85.40	87.59	84.98	78.48	83.22	83.52	85.16	87.40	84.97	77.92	83.54	83.42	84.99	87.25	84.77	77.71
Rappel	11.49	15.14	16.03	15.74	21.37	15.33	9.69	14.44	16.19	16.75	**22.88**	16.49	9.84	14.56	15.56	17.02	17.49	17.10	9.48	14.62	15.54	17.31	17.65	17.37	9.47	14.60	15.69	17.27	17.57	17.39
F_1	20.16	25.74	27.05	26.75	33.55	25.89	17.35	27.70	27.28	28.12	**35.09**	27.32	17.56	24.82	26.32	28.50	29.00	28.08	17.02	24.88	26.29	28.90	29.22	28.40	17.01	24.84	26.49	28.83	29.10	28.41
CEAF $-\phi_3$																														
Précision	48.03	43.97	39.17	29.09	24.33	22.48	49.03	45.57	41.47	31.62	21.65	20.68	48.35	44.70	39.37	30.88	26.85	20.15	**49.28**	44.66	39.35	30.79	26.13	19.43	48.92	44.70	39.42	30.42	25.69	19.32
Rappel	14.06	20.00	22.22	25.56	**28.76**	22.98	12.31	18.11	21.59	24.97	28.69	23.29	12.40	17.89	20.06	25.09	26.52	23.61	12.28	17.98	19.91	25.26	26.67	23.69	12.24	17.93	19.99	24.79	26.16	23.74
F_1	21.76	27.50	28.36	27.22	26.36	22.73	19.68	25.92	**28.40**	27.90	24.68	21.91	19.74	25.55	26.58	27.69	26.68	21.74	19.66	25.64	26.44	27.75	26.39	21.35	19.58	25.59	26.52	27.32	25.92	21.30
CEAF $-\phi_4$																														
Précision	31.31	27.02	24.98	20.56	18.07	15.58	33.26	29.30	26.85	21.66	18.43	15.86	33.44	29.20	26.88	21.64	18.30	15.77	34.14	29.55	27.17	21.79	18.15	15.57	**34.33**	29.84	27.40	21.80	18.19	15.60
Rappel	21.98	27.33	30.95	38.04	41.38	**42.79**	19.55	25.50	29.21	36.61	40.30	41.75	19.25	24.84	28.48	35.85	39.36	40.94	19.12	24.90	28.33	35.60	39.21	40.57	18.95	24.78	28.13	35.39	39.05	40.47
F_1	25.83	27.17	27.64	26.69	25.15	22.84	24.63	27.27	**27.98**	27.22	25.29	22.99	24.44	26.84	27.66	26.99	24.98	22.77	24.51	27.03	27.74	27.03	24.81	22.51	24.42	27.07	27.76	26.98	24.82	22.52
CoNLL																														
Précision	60.60	60.84	59.42	56.66	51.73	50.30	61.31	61.20	60.14	57.11	50.32	48.93	60.51	60.44	59.20	56.93	54.06	48.44	61.72	60.40	59.18	56.94	53.84	48.00	**61.88**	60.47	59.22	56.79	53.69	47.90
Rappel	15.01	20.39	22.48	25.05	30.26	24.79	12.90	18.68	21.78	25.15	**31.03**	25.51	12.95	18.54	20.70	25.19	27.07	25.79	12.68	18.60	20.58	25.38	27.16	25.90	12.65	18.54	18.15	25.04	26.86	25.93
F_1	22.36	27.93	29.13	28.60	32.27	25.24	19.87	27.19	29.06	29.69	**32.99**	26.21	20.00	26.33	27.80	29.84	29.82	26.60	19.68	26.43	27.73	30.14	29.83	26.66	19.67	26.40	27.88	29.81	29.54	26.69
Correction																														
WER	**45.05**	45.14	45.34	45.26	47.14	46.72	45.06	45.08	45.38	45.22	47.93	47.56	45.12	45.15	45.46	45.32	45.33	47.60	45.12	45.16	45.46	45.36	45.36	50.12	45.48	45.24	45.46	45.37	45.39	50.12
Différence	**1.45**	1.36	1.16	1.24	-0.64	-0.21	1.44	1.42	1.13	1.28	-1.43	-1.05	1.39	1.35	1.04	1.19	1.17	-1.10	1.39	1.35	1.05	1.14	1.15	-3.61	1.02	1.27	1.05	1.14	1.11	-3.61
Amélioration	**51.11**	47.96	40.87	43.72	-22.39	-7.55	50.83	50.11	39.66	45.13	-50.21	-37.09	48.82	47.59	36.74	41.78	41.29	-38.68	48.83	47.45	36.83	40.24	40.40	-127.28	36.02	44.64	36.82	40.11	39.12	-127.34

*La première partie de ce tableau présente les résultats du clustering des entités similaires pour différentes valeurs des paramètres n et t. L'évaluation est faite grâce aux différentes mesures présentées dans la section 5.5.1. La mesure **CoNLL** représente la moyenne des autres mesures pour la précision, le rappel et le score F_1. La deuxième partie du tableau fait état de la qualité de la correction à partir des clusters créés. Le Word Error Rate du texte bruité non-corrigé est de 46.51%. On présente pour chaque couple (n, t) la différence absolue dans la qualité de correction et le pourcentage d'amélioration que cela représente. Les meilleures valeurs pour chaque méthode d'évaluation sont présentées en gras.*

5.5.2 Évaluation de la correction

Grâce aux clusters calculés avec notre méthode, nous avons à notre disposition des ensembles de mots ou d'expressions représentant dans une certaine mesure la même entité nommée (malgré les erreurs d'OCR). Nous pouvons utiliser ces clusters pour corriger automatiquement les entités nommées présentes dans le corpus. La correction des entités nommées du corpus grâce aux clusters calculés se fait en deux étapes : la sélection de l'entité originale et le remplacement des autres éléments du cluster dans le corpus.

5.5.2.1 Sélection d'un représentant

Avant de pouvoir corriger les entités nommées du corpus, il est nécessaire d'identifier dans chaque cluster l'entité originale, non bruitée. Cette entité est alors utilisée comme remplacement pour tous les autres éléments du cluster.

Approche naïve : Il est intuitif de penser que l'entité originale sera plus présente au sein du corpus que les entités dégradées. En effet, selon la qualité du document, l'OCR a un certain pourcentage de chance de mal reconnaître un mot. Si plusieurs occurrences de ce mot sont présentes dans le corpus original, la plupart seront cependant correctement reconnues. Ainsi, l'entité originale d'un cluster donné est sélectionnée comme étant celle qui apparaît le plus dans le corpus. C'est cette méthode que nous avons utilisée et qui sera évaluée.

Approche générative : Dans le cas où le cluster ne contient pas d'entité particulièrement plus présente que les autres dans le corpus, il faut trouver un autre moyen de sélectionner l'entité de référence. Si le cluster comporte assez d'éléments, il est possible de générer l'entité "valide" à partir des membres du cluster. Ce problème peut se traduire ainsi : il s'agit de trouver un mot qui minimise sa distance d'édition avec les autres mots du clusters. Formellement, un cluster $\mathbb{C}$ est un ensemble composé de n mots : $word_1$, $word_2$, ..., $word_n$. L'objectif est de trouver un mot m qui minimise la somme des distances d'édition de ce mot avec les autres éléments du cluster.

$$\arg\min_{m} \sum_{i=1}^{n} sim_{edit}(m, word_i) \tag{5.23}$$

5.5.2.2 Correction du corpus

Les deux mesures les plus utilisées pour évaluer la qualité d'un texte extrait d'un processus OCR sont le Character Error Rate (CER) et le Word Error Rate (WER). Ces deux mesures sont très similaires à la distance de Levenshtein. En effet, le CER compte et normalise le nombre de suppressions, insertions et substitutions de caractères nécessaire pour passer du texte original au texte issu du processus OCR.

Le WER effectue les mêmes calculs mais à un niveau de granularité plus large : au niveau des mots. Ces deux mesures permettent de se rendre compte de la proportion des mots (ou des caractères) valides dans un texte bruité.

Une fois que l'entité de référence de chaque cluster est identifiée, on peut entamer la correction du corpus. On remplace l'ensemble des éléments du cluster dans le corpus par l'entité de référence. Nous avons utilisé pour nos expériences l'approche naïve présentée dans la sous-section précédente. Imaginons un cluster dont l'entité de référence est soulignée : (“Washington”, “Washingtan”, “Was8lngton”). Ici, on remplacera dans le corpus toutes les occurrences de “Washingtan” ou “Was8lngton” par “Washington”.

Le corpus bruité obtenu après le processus OCR possède un WER de 46.51%. Cela signifie que près de la moitié des mots n'ont pas été correctement reconnus. Comme nous l'avons déjà dit, ce taux d'erreurs a un impact significatif sur la qualité des recherches. L'objectif est donc de le diminuer le plus possible grâce à l'étape de correction. Les différents résultats sont présentés dans la deuxième partie du tableau 5.2. Les valeurs des paramètres n et t qui ont entraîné la meilleur correction ne sont pas celles qui ont obtenu les meilleurs scores à l'étape de clustering. Cela peut s'expliquer par la méthode de correction assez brutale et l'aspect approximatif du contenu des clusters (voir contraintes de calculs présentées dans la section 5.4.4). Beaucoup de mots sont remplacés à tort, ce qui explique les scores de correction parfois très bas.

Nous avons effectué une analyse statistique de ces résultats pour identifier quelle mesure de clustering est la plus corrélée au WER. Les données que nous utilisons sont ordinales et nous nous intéressons aux relations qu'entretiennent les différentes mesures de clustering avec le WER. Ainsi, nous avons utilisé la corrélation de Spearman. Cette dernière permet d'évaluer la probabilité p que deux variables aient une relation monotone (si l'une augmente, alors l'autre aussi). Après une analyse des résultats, il se trouve que le WER et les mesures de précision pour $\mathbf{CEAF} - \phi_{\mathbf{3}}$ et $\mathbf{CEAF} - \phi_{\mathbf{4}}$ sont très corrélées (respectivement $p < 1 \times 10^{-5}$ et $p = 1.8 \times 10^{-5}$). Ces mesures semblent donc pouvoir servir d'indicateur pour prédire la qualité de la correction selon celle du clustering.

5.5.3 Évaluation de l'accessibilité des documents

Lorsqu'on recherche de l'information dans un corpus bruité, on est souvent confronté à des problèmes d'accessibilité. Par exemple, un utilisateur s'intéresse aux documents qui contiennent le terme “washington”. Il tape donc cette requête dans le moteur de recherche ce terme et un certain nombre de documents sont renvoyés. Or, il existe certainement des documents, mal reconnus par le processus OCR, qui contiennent des variations du terme “washington”. À cause de leur qualité, ces documents ne seront jamais accessibles en utilisant cette requête. Cette notion d'accessibilité est critique pour un système de recherche. Idéalement, tous les documents devraient être aussi accessibles les uns que les autres. Cependant, les

documents eux-mêmes et les nombreux biais induits par les utilisateurs et le système (requêtes, algorithmes de pertinence, tendance à limiter l'exploration des résultats, *etc.*) participent à l'inégalité de l'accessibilité du corpus. Pour évaluer cette inégalité d'accessibilité des documents d'un corpus au sein d'une plateforme de recherche, Azzopardi *et. al.* ont défini une mesure décrite dans [Azzopardi 2008].

5.5.3.1 Mesure d'accessibilité

Un système de recherche d'information contient un corpus D de documents d_i indexés. Ce système peut être interrogé en exprimant une requête q. Est alors retournée une liste R_q de documents issus de D, triés selon leur pertinence par rapport à la requête q. Intuitivement, l'accessibilité d'un document du corpus correspond aux documents présents et à la fonction de pertinence du système. Imaginons l'ensemble Q de toutes les requêtes possibles. Chacune des requêtes q est associée à un poids o_q qui traduit sa popularité. Azzopardi *et. al.* estiment que le score d'accessibilité d'un document devrait être grand si :

— un grand nombre de requêtes dans Q permettent d'y accéder, ou peu de requêtes mais avec une popularité o_q élevée

— le rang des documents retournés après une requête est le plus faible possible, 1 étant la meilleure valeur.

À partir des observations précédentes, Azzopardi *et. al.* ont défini la mesure d'accessibilité définie dans l'équation 5.24. Dans celle-ci, $f(k_{dq}, c)$ est une fonction de coût où k_{dq} est le rang du document retourné après une requête q et c est le rang maximum qu'un utilisateur est susceptible d'observer dans une liste de résultats. Cette fonction renvoie 1 si $k_{dq} \leq c$ et 0 autrement. Les auteurs précisent dans leur article que cette fonction peut être modifiée pour refléter de manière plus fine la probabilité qu'un utilisateur observe un document dans la liste des résultats de recherche. Ce score est un score cumulatif proportionnel au nombre de fois où un document d_i donné apparaît dans la liste de résultats, avec un rang inférieur à la valeur c.

$$r(d) = \sum_{q \in Q} o_q \times f(k_{dq}, c) \tag{5.24}$$

Après avoir calculé le score d'accessibilité de tous les documents d'un corpus, il serait intéressant de pouvoir exprimer l'accessibilité générale du corpus par rapport à un système donné. Azzopardi *et. al.* utilisent pour cela des courbes de Lorenz. Ces courbes permettent de mettre en évidence les inégalités dans une population selon un paramètre donné. Elles sont notamment utilisées dans les sciences économiques pour visualiser les écarts de richesse entre individus. Ici, nous visualisons les différences d'accessibilité dans un corpus de documents. La construction de cette courbe se fait en deux étapes :

— les documents sont d'abord classés dans l'ordre ascendant selon leur accessibilité calculée dans l'équation 5.24

— on trace ensuite une distribution cumulative de l'accessibilité.

On s'attend à une distribution linéaire si tous les documents sont également accessibles. Plus la distribution que l'on obtient en est éloignée, moins le corpus est globalement accessible.

Une mesure qui résume cet écart par rapport à une distribution linéaire est introduite dans [Gastwirth 1972]. C'est le coefficient de Gini G décrit dans l'équation 5.25. Si la mesure G est égale à 0, alors il n'y a pas de biais et tous les documents sont également accessibles. Au contraire, si elle est égale à 1, cela signifie qu'un seul document est accessible et que tous les autres ne le sont pas.

$$G = \frac{\sum_{i=1}^{N}(2 \times i - N - 1) \times r(d_i)}{N \sum_{j=1}^{N} r(d_j)} \tag{5.25}$$

Cette mesure correspond en fait à l'aire entre la courbe de Lorenz et la ligne d'égalité. Plus ces courbes sont éloignées l'une de l'autre, plus le coefficient de Gini sera proche de 1. Cependant, cette mesure seule ne permet pas de décrire où se concentrent les inégalités au sein d'un corpus. Par exemple, deux courbes peuvent avoir le même coefficient de Gini tout en exprimant des inégalités différentes. En effet, dans un cas la majorité des documents peut être très peu accessible alors que dans un autre, une petite partie des documents peut concentrer l'accessibilité totale du corpus. Cette caractéristique se traduit graphiquement par l'emplacement du point de la courbe de Lorenz où la tangente a la même pente que la ligne d'égalité, c'est-à-dire 1. Un exemple visuel est donné dans la figure 5.6.

Une mesure intéressante pour décrire numériquement cet aspect de la courbe de Lorenz est le coefficient d'asymétrie de Lorenz, S, développé par Damgaard *et. al.* dans [Damgaard 2000]. Ce coefficient est défini ainsi :

$$S = F(\mu) + L(\mu) \tag{5.26}$$

μ est l'accessibilité moyenne du corpus, F est la fonction de répartition des documents ordonnés selon leur accessibilité et L est la fonction de répartition de l'accessibilité. Le point $(F(\mu), L(\mu))$ correspond au point de la courbe de Lorenz où la pente de la tangente est 1 c'est-à-dire le point où cette tangente est parallèle à la ligne d'égalité. Les détails des calculs sont présentés dans les équations suivantes :

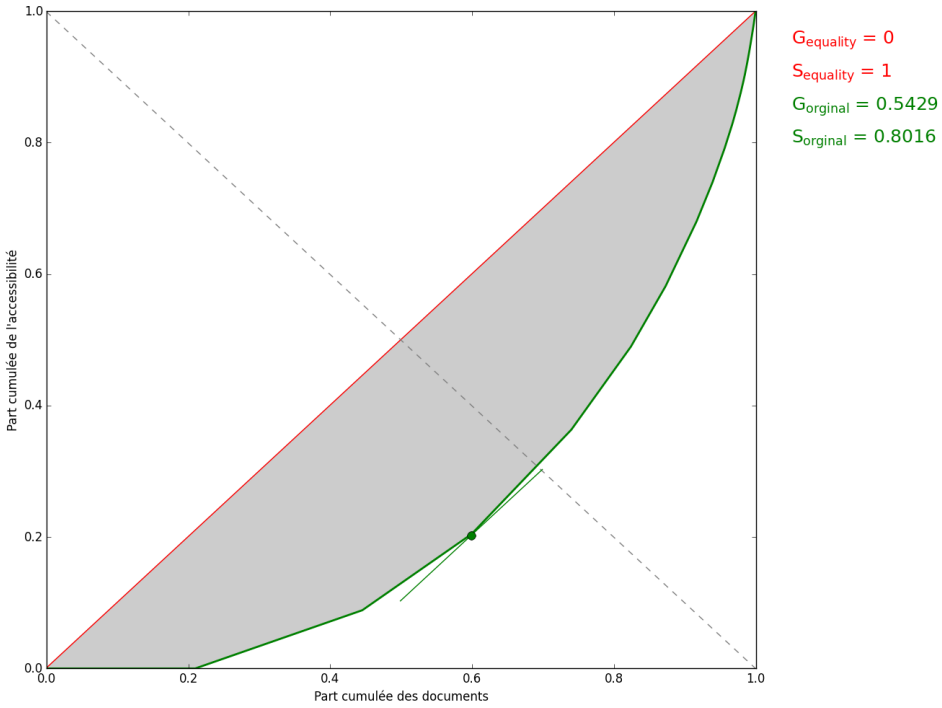
$$\delta = \frac{\mu - r(d_m)}{r(d_{m+1}) - r(d_m)} \tag{5.27}$$

$$F(\mu) = \frac{m + \delta}{n} \tag{5.28}$$

$$L(\mu) = \frac{\sum_{i=1}^{m} r(d_i) + \delta r(d_{m+1})}{\sum_{j=1}^{n} r(d_j)} \tag{5.29}$$

m correspond au nombre de documents qui ont une accessibilité inférieure à μ. Finalement, si $S = 1$, alors le point $(F(\mu), L(\mu))$ se situe sur la ligne de symétrie et

FIGURE 5.6 – Exemple d'une courbe de Lorenz



La courbe rouge représente la ligne d'égalité. C'est le cas parfait où tous les documents sont aussi accessibles les uns que les autres. La courbe verte représente l'accessibilité réelle des documents au sein d'une plateforme. L'espace gris entre les deux courbes est la représentation visuelle du coefficient de Gini. Le coefficient d'asymétrie représenté par le point de la courbe où la tangente est parallèle à la ligne d'égalité est mis en évidence.

la courbe de Lorenz est elle aussi symétrique. Les inégalités sont alors réparties sur toute la population. Si $S < 1$, alors le point $(F(\mu), L(\mu))$ est sous la ligne de symétrie et les inégalités touchent particulièrement les documents les moins accessibles. Au contraire, si $S > 1$, alors ce sont les documents les plus accessibles qui sont principalement touchés par ces inégalités.

5.5.3.2 Résultats et discussion

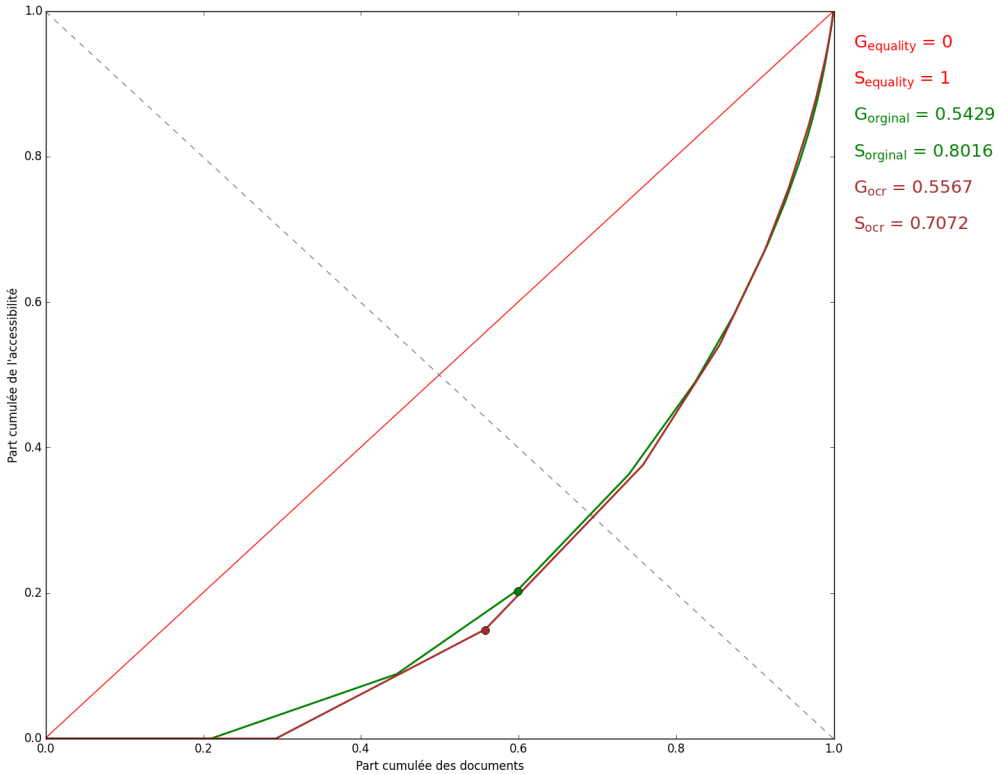
Nous allons tester l'accessibilité de plusieurs corpus grâce aux courbes de Lorenz et aux mesures associées. Nous avons expliqué dans la sous-section précédente que l'accessibilité des documents était évaluée par rapport à l'ensemble Q des requêtes possibles. Bien évidemment cet ensemble est infini et ne peut donc pas être utilisé. On utilise classiquement un ensemble de requêtes exprimées par de véritables utilisateurs au sein d'une plateforme. Nous n'avons cependant pas accès à de telles données. Nous avons donc défini notre ensemble Q comme la liste des entités nommées du corpus non bruité de notre vérité terrain (voir section 5.3.1). Nous avons

en effet déjà expliqué que la majorité des requêtes effectuées par les utilisateurs contiennent des entités nommées. L'ensemble Q choisi est donc cohérent avec cette affirmation. Toujours faute de données réelles, la popularité O_q de chaque requête est fixée à 1.

Nous avons indexé dans le moteur de recherche SolR les deux corpus créés dans la section 5.3.1 : le corpus original et le corpus artificiellement bruité. Notre objectif est d'évaluer l'impact de notre méthode sur l'accessibilité générale du corpus bruité. Les utilisateurs s'intéressent rarement aux résultats au-delà de la première page de résultats (voir section 3.3.2). Pour cette raison, le paramètre c de la formule d'accessibilité (voir équation 5.24) est fixé à $c = 10$, ce qui correspond à une page de résultats avec les paramètres par défaut de SolR.

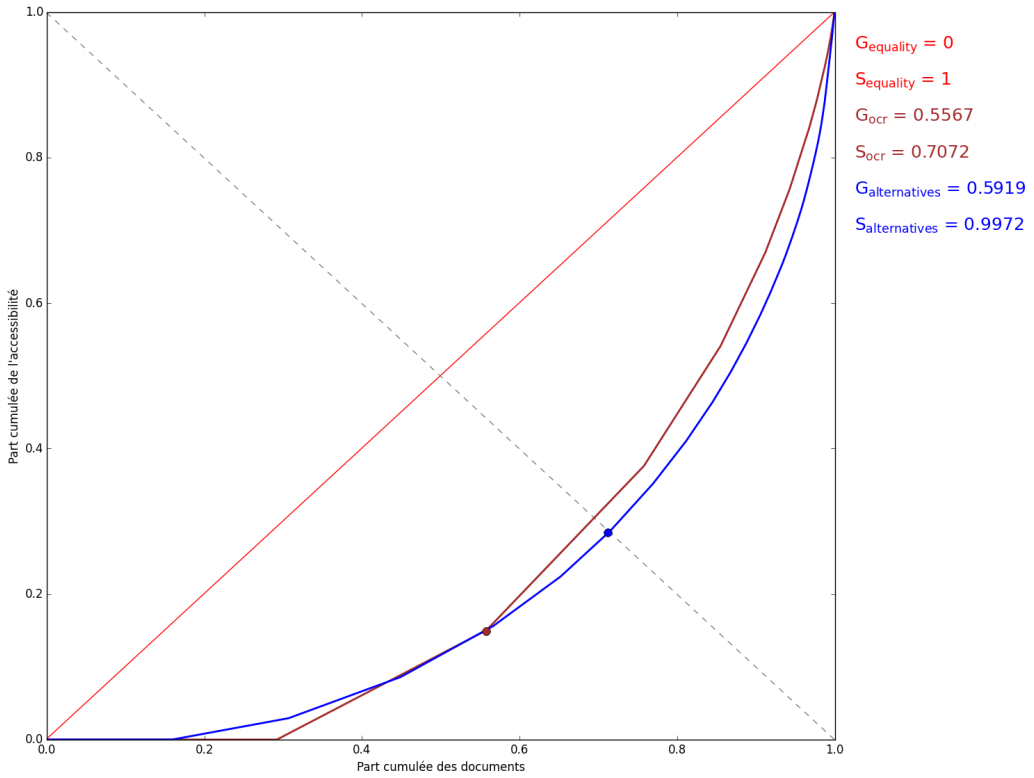
Notre première observation porte sur l'impact du bruit OCR sur la répartition de l'accessibilité au sein du corpus. Nous avons donc calculé l'accessibilité des documents des deux corpus pour construire les courbes de Lorenz correspondantes. Elles sont toutes les deux présentées dans la figure 5.7. On peut d'abord observer que le coefficient de Gini a augmenté pour le corpus bruité. Cela signifie que les différences d'accessibilité des documents se sont globalement creusées après le processus OCR.

FIGURE 5.7 – Comparaison de la distribution de l'accessibilité des documents après un processus OCR



Ensuite, on observe que le coefficient d'asymétrie S a diminué. Cela se traduit par le fait que ce sont les documents déjà peu accessibles qui ont le plus pâti du processus OCR. Cela peut s'expliquer par le fait que ces documents n'étaient atteignables que par quelques requêtes dans le corpus original. Les erreurs générées par le processus OCR ont alors certainement rendu ces requêtes caduques. Au contraire, les documents accessibles par de nombreuses requêtes différentes dans le corpus original ont été moins touchés.

FIGURE 5.8 – Comparaison de la distribution de l'accessibilité des documents ayant subi un processus OCR après application de notre méthode



Nous avons ensuite testé l'impact de notre méthode sur l'accessibilité du corpus bruité. Pour ce faire, nous avons modifié l'ensemble des requêtes Q pour qu'il intègre le contenu des clusters calculés par notre méthode (voir section 5.4.4). Le moteur de recherche SolR dans lequel les documents sont indexés permet en effet d'évaluer des requêtes booléennes. Imaginons que notre méthode a identifié les variantes d'une entité nommée. Par exemple, nous savons que les mots "washingtan" et "was8lington" se rapportent à l'entité originale "washington". Dans ce cas, la requête originale 'washington" est modifiée pour devenir "washington OR washingtan OR was8lington". Ainsi, le moteur de recherche renverra les documents qui contiennent **au moins** l'un des mots précédents. Chaque requête de l'ensemble Q est donc modifiée pour

prendre en compte les variantes des entités nommées découvertes automatiquement par notre méthode. Nous avons finalement calculé les courbes de Lorenz du corpus bruité, pour deux versions de l'ensemble des requêtes Q. Elles sont présentées dans la figure 5.8. Il y a deux observations importantes à faire. D'abord, le coefficient de Gini G a augmenté après l'application de notre méthode (G est passé de 0.56 à 0.59). Cela signifie qu'il y a alors plus d'inégalité dans l'accessibilité du corpus. En effet, l'accessibilité absolue des documents a augmenté de 57% en moyenne. Cela signifie que les écarts d'accessibilité se sont creusés. Ainsi, le score $r(d)$ des documents les moins accessibles a **un peu** augmenté, tandis que le score des documents déjà très accessibles a **beaucoup** augmenté. Ensuite, on peut voir que l'application de notre méthode a très fortement amélioré la symétrie générale de la courbe (S est passé de 0.71 à 0.99). Cela signifie qu'il y a eu un décalage dans la distribution des inégalités. Cette quasi-symétrie de la nouvelle courbe de Lorenz nous assure que ce sont les documents auparavant peu accessibles qui ont le plus profité de notre méthode. L'impact de notre méthode est donc le lissage des inégalités sur tous les documents du corpus, au détriment d'une augmentation globale des inégalités. En effet, plus de 45% des documents auparavant complètement inaccessibles (avec un score $r(d) = 0$) le sont devenus.

5.6 Conclusion

Nous avons présenté dans ce chapitre l'impact que peut avoir un processus OCR sur l'accessibilité des documents au sein d'un corpus. Si ce dernier permet de faciliter l'indexation de documents textuels numérisés, il a un impact non négligeable sur les pratiques des utilisateurs. En effet, l'OCR est une boîte noire qui fonctionne plus ou moins bien selon la qualité des documents traités. La qualité de ce processus d'extraction d'information est rarement prise en compte lors de l'indexation des documents. Ainsi, puisqu'il est difficile d'identifier les erreurs de reconnaissance, tous les mots sont indexés de la même manière.

Cela impacte fortement les pratiques des utilisateurs qui n'ont pas toujours conscience des processus sous-jacents à la recherche d'information. En effet, si une requête exprimée par un utilisateur ne renvoie à aucun résultat, au moins trois cas doivent être envisagés :

1. l'information existe mais la requête a été mal exprimée
2. l'information existe mais n'est pas correctement indexée dans le moteur
3. l'information n'existe pas

Nous nous sommes concentrés dans ce chapitre sur la problématique levée par le deuxième cas. Nous avons donc présenté une méthode pour limiter l'effet d'un processus OCR sur la qualité de l'indexation de documents. Nous nous sommes concentrés sur les entités nommées, très souvent utilisées dans des requêtes. En analysant leur contexte et les caractères qui les composent, nous sommes capables d'identifier des entités nommées identiques aux erreurs de reconnaissance près ("washington" et

“was8lington” par exemple). Grâce à cela, nous sommes en mesure d’aider l’utilisateur à améliorer sa requête en prenant en compte certaines erreurs qui découlent du processus OCR. Les contributions du chapitre sont résumées dans la liste suivante.

1. Création d’un corpus d’évaluation pour la recherche sur les entités nommées en contexte bruité (voir section 5.3.1).
2. Analyse de l’impact de l’OCR sur la détection des entités nommées (voir section 5.3.2).
3. Développement d’une méthode d’identification automatique des entités nommées similaires malgré la présence de bruit OCR (voir section 5.4).
4. Évaluation de cette méthode sous trois aspects différents (voir section 5.5) :
   (a) Évaluation de la qualité des clusters
   (b) Évaluation de la qualité de la correction
   (c) Évaluation de l’accessibilité du corpus

La vérité terrain, la méthodologie d’identification des entités nommées similaires ainsi que son évaluation ont fait le sujet de publications dans des conférences internationales [Jean-Caurant 2017a][Jean-Caurant 2017b].

Outre l’aide concrète que la méthode développée peut apporter aux utilisateurs, elle a également l’avantage de mettre en évidence l’impact que peut avoir un processus OCR sur la recherche d’information. Il serait intéressant de mettre en œuvre un tel outil à l’intérieur de la plateforme pédagogique que nous avons développée (voir chapitre 3).

Contributions

- Nous avons développé une vérité terrain, qui combine les problématiques associées à l’OCR et à la détection des entités nommées.
- Nous avons développé une chaîne de traitement pour identifier et corriger les entités nommées similaires extraites d’un texte généré par un processus OCR.
- Cette chaîne a été évaluée grâce à différentes mesures, chacune livrant un point de vue différent du problème.

Chapitre 6

Conclusion générale

Rappel de la problématique et bilan

Que ce soit pour des besoins personnels ou professionnels, nous avons besoin de pouvoir accéder aux informations disponibles en ligne. Ces informations n'ont jamais été si nombreuses qu'aujourd'hui. Beaucoup de besoins informationnels peuvent donc en théorie être résolus. Cependant, en pratique, plusieurs facteurs affectent l'accessibilité de l'information. Or, une information inaccessible ne peut être exploitée. Nous avons choisi d'étudier ce problème à travers les interfaces qui existent entre les utilisateurs et les données disponibles en ligne.

Nous nous sommes d'abord intéressés aux interactions entre les utilisateurs et les système de recherche, en particulier les bibliothèques numériques. Nous avons développé une plateforme d'expérimentation qui nous a permis d'observer le comportement des utilisateurs engagés dans une tâche de recherche d'information. Ces observations nous permettent d'affirmer que les utilisateurs sont sujets à différents biais méthodologiques qui peuvent nuire à la qualité des recherches effectuées. En particulier, ces derniers font trop souvent aveuglément confiance aux moteurs de recherche. En ne consultant que les premiers résultats renvoyés, les utilisateurs se privent d'informations potentiellement pertinentes. Ce constat nous a conduit à adapter le fonctionnement de notre plateforme à des fins pédagogiques. En effet, nous pensons que la formation des utilisateurs aux outils numériques est essentielle de nos jours. Comprendre le fonctionnement de base des outils couramment utilisés permet d'entreprendre une démarche de recherche critique qui ne peut qu'améliorer les résultats obtenus. Cela dit, même un utilisateur parfaitement expert dans les démarches de recherche d'information n'est pas assuré d'accéder à toutes les informations susceptibles de l'intéresser. En effet, ce dernier ne maîtrise pas la manière dont les documents sont intégrés dans les plateformes de recherche. Très souvent, ces derniers subissent différentes transformations avant d'être mis à la disposition des utilisateurs.

C'est pour cette raison que nous avons décidé d'étudier la manière dont les données sont intégrées dans les systèmes de recherche. De nombreux facteurs entrent en jeu et sont susceptibles d'affecter l'accessibilité de l'information. Nous avons choisi de nous concentrer sur la problématique de la qualité des données. En effet, beaucoup de documents sont sujets à des processus automatiques d'extraction d'information tels que l'OCR. Selon la qualité des documents originaux, les transformations sont plus ou moins exploitables. Les erreurs de reconnaissance commises par ces algo-

rithmes limitent la portée des recherches effectuées par les utilisateurs. De par leur contenu informationnel riche, les entités nommées sont très souvent utilisées dans les recherches textuelles. Lorsque les erreurs de reconnaissance affectent ces expressions en particulier, il devient difficile d'accéder à l'information pertinente. Nous avons donc choisi de développer une chaîne de traitement capable de limiter l'impact de ces erreurs sur l'accessibilité de l'information. Grâce à l'utilisation d'un modèle de langage et de mesures de similarité, nous sommes capable d'identifier les entités identiques malgré la présence d'erreurs de reconnaissance.

Résumé des contributions

- Nous avons développé une méthode permettant de limiter l'impact des erreurs d'OCR sur l'exploitation des entités nommées dans une tâche de recherche d'information. Cette méthode peut être appliquée dans un contexte de représentation distribuée de mots [Jean-Caurant 2017a], ou dans un contexte de recherche d'information par requête textuelle [Jean-Caurant 2017b].
- Nous avons développé une plateforme d'observation du comportement des utilisateurs dans un système de recherche d'information semblable à la majorité des bibliothèques numériques. Celle-ci nous a permis d'identifier un ensemble d'indicateurs particulièrement représentatifs du comportement d'un utilisateur [Suire 2016].
- Nous avons modifier l'usage premier de la plateforme pour s'en servir à des fins pédagogiques. Une première séance de formation a été réalisée et a fait l'objet d'une communication [Suire 2017].

Limites et perspectives

Si les travaux conduits dans cette thèse peuvent permettre aux utilisateurs de mieux appréhender les tâches de recherche d'information, de nombreuses pistes nécessitent encore d'être explorées. L'observation du comportement des utilisateurs grâce à la plateforme que nous avons développée permet d'identifier différents facteurs susceptibles d'affecter les résultats et le bilan d'une recherche d'information. Nous nous sommes concentrés sur la problématique de la qualité des données mises à la disposition des utilisateurs. La chaîne de traitement développée dans le chapitre 5 a été évaluée techniquement. Cependant, il sera important de la valider dans un contexte plus pratique pour comprendre son impact sur le comportement des utilisateurs. Cela pourra par exemple être réalisé grâce à la plateforme d'évaluation développée dans le chapitre 3. Si la méthode que nous avons développée permet bien de limiter l'impact que peut avoir la qualité des documents sur leur accessibilité, elle peut cependant être améliorée. Nous pouvons en effet étendre l'analyse non seulement aux entités nommées, mais aussi aux autres formes de mots et d'expressions. Ainsi, les problèmes d'accessibilité dûs aux erreurs de reconnaissance n'en seront que plus limités. Outre la correction d'erreurs de reconnaissance, la chaîne de traitement développée semble compatible avec l'identification des variations orthographiques. Que ce soit par exemple l'étude des variations dans un contexte historique [Jaffré 2010] ou encore l'identification automatique des erreurs commises par des élèves dyslexiques.

Outre la qualité des données, de nombreux autres facteurs entrent en jeu lorsque l'on étudie l'accessibilité de l'information au sein d'une plateforme. Nous n'avons par exemple pas eu le temps d'étudier l'impact de l'aspect des plateformes de recherche. Par exemple, dans [Jahanghiri 2016], les auteurs s'intéressent aux liens qui existent entre ergonomie et accessibilité de l'information. Comme eux, nous pensons que cette piste est particulièrement intéressante. La plateforme d'observation que nous avons développée pourrait être adaptée pour étudier l'impact de différents changements de l'interface graphique sur l'accessibilité des documents.

Finalement, nous sommes persuadés qu'il est nécessaire d'étudier les interfaces qui existent entre les utilisateurs et les données (utilisateur/système et système/données) de manière parallèle. En effet, les différents algorithmes de recherche d'information sont trop rarement mis en perspective par rapport à l'usage qu'en font les utilisateurs. Nous pensons qu'il faut apporter un point de vue critique aux technologies de recherche massivement utilisées de nos jours, et parfois limiter leur pouvoir mystificateur.

Annexe A

Fonctionnement général d'un réseau de neurones

A.1 Introduction

Nous présentons dans cette annexe une technologie d'apprentissage automatique très populaire depuis quelques années : les réseaux de neurones. Cette première section introductive va rapidement présenter l'origine et les caractéristiques des réseaux de neurones ainsi que la description d'un neurone simple. Nous présenterons ensuite de manière plus générale les **réseaux** de neurones et les algorithmes de prédiction et d'apprentissage associés. Finalement, nous ferons état de différents conseils pour le bon fonctionnement d'un réseau avant de conclure.

A.1.1 Présentation générale

Un réseau de neurones est une architecture particulière et un ensemble d'algorithmes utilisés pour effectuer des tâches d'apprentissage automatique. Il existe différents types de réseau mais l'exemple le plus classique est le "perceptron multicouches" inventé par Frank Rosenblatt [Rosenblatt 1958]. La popularité de ces méthodes vient également de l'algorithme d'apprentissage dit de "rétropropagation de gradient". Plusieurs chercheurs ont participé à son élaboration. On peut par exemple citer Paul Werbos [Werbos 1974] qui a le premier mentionné l'idée d'utiliser cet algorithme pour les réseaux de neurones, ou encore David Rumelhart [Rumelhart 1986] qui a expérimentalement montré l'intérêt de ces méthodes.

Il existe deux grands types de tâches qu'un tel réseau peut résoudre. Il peut d'abord s'agir de prédire un label particulier à partir de données en entrée (il s'agit alors d'une tâche de classification), ou bien de prédire une valeur numérique sur un intervalle continu (c'est alors une tâche de régression). Les réseaux de neurones font partie des algorithmes d'apprentissage supervisé. Il est donc nécessaire de les entraîner en leur présentant des exemples d'entraînement constitués des données en entrée et de la sortie attendue. On peut par exemple imaginer vouloir prédire le type d'un animal, **chien** ou **chat**, à partir de différentes caractéristiques. Imaginons-en trois :

— x_1 le nombre d'heures de sommeil par jour

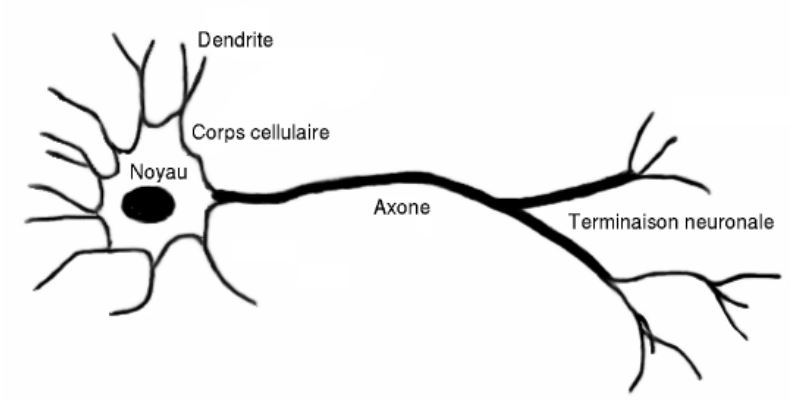
— x_2 la hauteur au garrot

— x_3 la taille du territoire

Selon la valeur de ces caractéristiques, le réseau devra prédire la probabilité que l'animal décrit soit un chien ou un chat. Cet exemple servira de fil rouge tout au long du chapitre.

A.1.2 Analogie entre neurone biologique et neurone formel

FIGURE A.1 – Schéma d'un neurone biologique



Le fonctionnement d'un neurone formel est très inspiré de celui d'un neurone biologique. Un neurone biologique (voir figure A.1) est composé principalement de quatre éléments :

— les dendrites qui supportent les informations qui entrent dans le neurone

— le corps cellulaire et le noyau qui sont chargés d'activer ou non la sortie du neurone en fonction des signaux en entrée

— l'axone qui permet au neurone de transmettre une information

— les synapses (non présentes sur le schéma) qui sont les zones d'interaction entre neurones, au niveau des dendrites et des terminaisons neuronales de l'axone.

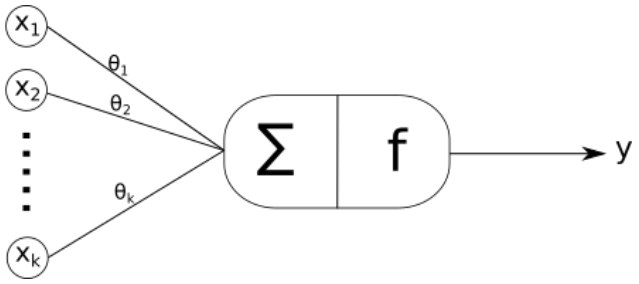
Chacun de ces éléments a un équivalent dans le neurone formel (présenté dans la figure A.2). Ainsi, le neurone formel possède différents signaux d'entrée x_1, x_2, ..., x_k semblables aux dendrites du neurone biologique. Ces signaux sont combinés grâce à une somme pondérée u avec un ensemble de poids synaptiques θ_1, θ_2, ..., θ_k chargés de contrôler leur importance (voir équation A.1).

$$u = \sum_{i=1}^{n} \theta_i x_i \tag{A.1}$$

Cette somme est ensuite envoyée dans une fonction d'activation chargée de contraindre sa valeur dans un intervalle donné (par exemple $[0; 1]$). Cette nouvelle valeur représente l'activation du neurone. Elle est alors transmise à la couche de neurones suivante (rôle de l'axone dans le neurone biologique). Lorsqu'on parle d'apprentissage, il s'agit en fait de trouver les valeurs des différents poids w_i pour que les prédictions du réseau soient le plus proche possible des valeurs attendues. Ces

poids représentent en quelque sorte la mémoire du réseau concernant les informations qu'il a déjà observées. Finalement, toutes ces similitudes sont résumées dans le tableau A.1.

FIGURE A.2 – Schéma d'un neurone formel.



La somme pondérée des entrées et des poids est calculée avant d'être traitée par la fonction d'activation qui renvoie la valeur de sortie du neurone.

TABLEAU A.1 – Récapitulatif des similarités entre neurone biologique et neurone formel.

Neurone biologique	**Neurone formel**
Dendrites	Signaux d'entrée x_1, x_2, ..., x_k
Corps cellulaire et noyau	Somme et fonction d'activation
Axone	Sortie y du neurone
Synapse	Poids synaptiques θ_1, θ_2, ..., θ_k

A.2 Architecture en réseau et apprentissage

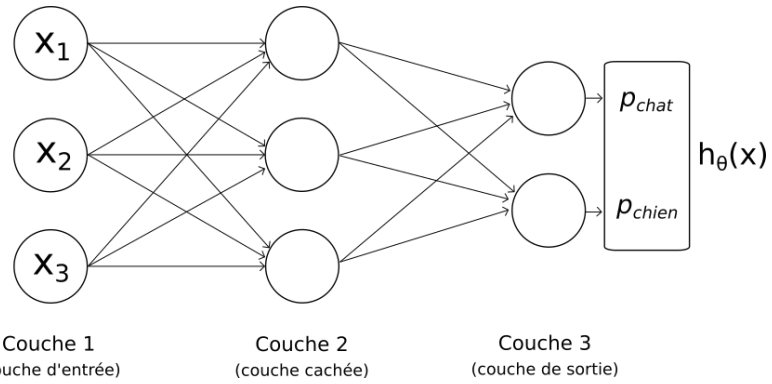
Un réseau de neurones est un graphe rassemblant différents neurones liés entre eux. Comme nous l'avons déjà indiqué dans l'introduction, l'exemple le plus classique est le perceptron multicouches. Un tel réseau est composé de différentes variables en entrée, d'une ou plusieurs couches cachées de neurones et d'une couche de sortie. Chaque couche contient des neurones non reliés entre eux. Cependant, les différentes couches sont liées entre elles. Pour reprendre notre exemple de classification chien/chat, le réseau associé est présenté dans la figure A.3. La sous-section suivante présente le fonctionnement détaillé de certaines parties du réseau tandis que la suivante présente les deux principaux algorithmes associés aux réseaux de neurones.

A.2.1 Pré-requis

A.2.1.1 Fonctions d'activation

Les valeurs calculées à partir des données en entrée et des poids synaptiques du réseau (u dans l'équation A.1) jouent le rôle de signaux qui passent entre les diffé-

FIGURE A.3 – Exemple d'un réseau de neurones avec trois couches.



Ce réseau est capable de prédire deux valeurs à partir de trois données en entrée. Dans notre exemple, x_1, x_2, et x_3 sont décrits dans la sous-section A.1.1. Chaque flèche est associée à un poids particulier. Durant l'apprentissage, ces poids vont évoluer pour permettre au réseau de correctement faire des prédictions. La sortie finale du réseau est le vecteur $h_\Theta(x)$, composé des deux probabilités p_{chat} et p_{chien} prédites. Les valeurs de sortie des deux derniers neurones doivent donc sommer à 1.

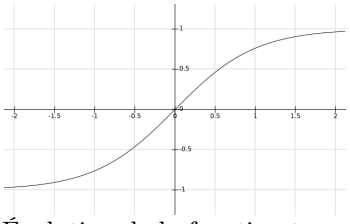
rentes couches du réseau. Cependant, ces signaux ne sont pas transmis tels quels. Ils sont transformés par ce qu'on appelle une fonction d'activation. Beaucoup de fonctions d'activation différentes peuvent être utilisées à différents endroits du réseau. Le choix le plus déterminant est celui de la fonction d'activation de la couche de sortie du réseau car c'est elle qui conditionnera le domaine de définition du résultat. Dans le cas de la résolution de problèmes de régression, il est intéressant d'obtenir comme valeur de sortie un réel borné. Pour cela on peut par exemple utiliser la fonction tangente hyperbolique *tanh* à valeur dans $[-1; 1]$ (voir figure A.4a) ou la fonction sigmoïde *sigmo* à valeur dans $[0; 1]$ (voir figure A.4b).

Lorsqu'on tente de résoudre un problème de classification binaire, on s'attend à obtenir une sortie dans $\{0, 1\}$. Pour cela, on utilise souvent la fonction généralisée de Heavyside (voir figure A.4c) dont la sortie est conditionnée par le paramètre θ.

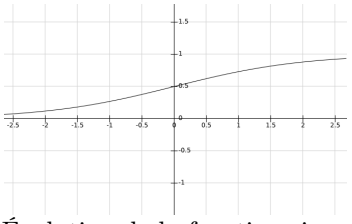
Finalement, lorsqu'on s'intéresse à un problème de classification générale à K classes, on peut utiliser la fonction $softmax$ (voir équation A.2) qui est une généralisation de la fonction sigmoïde. Elle prend donc en entrée un vecteur $\vec{x}$ constitué de n valeurs réelles arbitraires et calcule en sortie un vecteur $\sigma(\vec{x})$ de n nombres réels positifs dans $[0; 1]$ dont la somme est évaluée à 1. Ces caractéristiques en font une fonction particulièrement utile lorsque l'on veut représenter une loi de probabilité. Dans le cas de notre exemple de classification chien/chat, c'est cette fonction que nous utiliserons comme fonction d'activation dans la couche de sortie (voir figure A.3).

$$\sigma(\vec{x})_i = \frac{e^{\vec{x}_i}}{\sum_{k=1}^{n} e^{\vec{x}_k}} \text{ pour tout } i \in \{1, \cdots, n\} \tag{A.2}$$

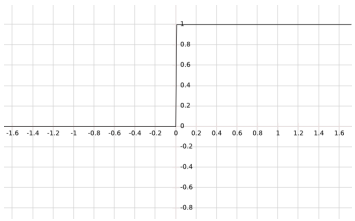
FIGURE A.4 – Graphiques et équations de trois fonctions d'activation souvent utilisées dans divers réseaux de neurones.



(a) Évolution de la fonction tangente hyperbolique définie comme : $tanh(x) = 1 - \frac{2}{e^{2x}+1}$



(b) Évolution de la fonction sigmoïde définie comme : $sigmo(x) = \frac{1}{1+e^{-x}}$



(c) Évolution de la fonction heavyside, pour $\theta = 0$, définie comme : $f(u) = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$

A.2.1.2 Entraînement et fonctions de coût

Pour l'apprentissage des poids du réseau, nous avons à notre disposition un ensemble de m exemples $\left\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\right\}$, où chaque $x^{(i)}$ est le vecteur des données en entrée et chaque $y^{(i)}$ est le résultat attendu. Pour chaque exemple i, on demande au réseau de prédire la sortie $h_\Theta(x^{(i)})$. On compare alors cette prédiction avec la sortie attendue $y^{(i)}$. Si on reprend notre exemple de classification chien/chat, un tel exemple d'entraînement (x, y) pourrait ressembler à cela :

$$x = \begin{pmatrix} 18 \\ 28 \\ 280000 \end{pmatrix} \qquad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{A.3}$$

Il existe plusieurs stratégies pour évaluer l'erreur commise par le réseau après une prédiction. Elles passent toutes par l'utilisation d'une fonction de coût dont les paramètres sont la sortie prédite $h_\Theta(x)$ et la sortie attendue y. Une valeur réelle est alors calculée. Plus cette valeur est grande, plus le réseau de neurones s'est trompé dans sa prédiction. Cela implique que les poids qui le composent devront être plus largement modifiés pour se rapprocher du résultat attendu. Deux modes d'apprentissage peuvent être utilisés. Le mode **batch** somme les erreurs produites

par les m exemples d'entraînement avant de modifier les poids du réseau. Le mode **online** effectue une modification des poids après chaque exemple.

Quoi qu'il en soit, différentes fonctions de coût peuvent être utilisées. Selon la tâche que tente de résoudre le réseau de neurones, il faudra utiliser une fonction de coût particulière. Par exemple, dans le cas de la résolution de problèmes de régression, une fonction adaptée est l'erreur quadratique moyenne (aussi appelée coût quadratique) définie dans l'équation A.4.

$$J_\Theta(h_\Theta(x), y) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})^2 \tag{A.4}$$

Pour les problèmes de classification binaire, le réseau prédit une valeur réelle dans l'intervalle $[0; 1]$ et la sortie attendue est un entier parmi $0, 1$. Ainsi, une fonction couramment utilisée est la fonction de perte d'entropie croisée (*cross-entropy loss*) définie dans l'équation A.5.

$$J_\Theta(h_\Theta(x), y) = -\frac{1}{m}\sum_{i=1}^{n} y^{(i)} ln(h_\Theta(x^{(i)})) + (1 - y^{(i)}) ln(1 - h_\Theta(x^{(i)})) \tag{A.5}$$

Nous allons finalement définir la fonction de coût qui généralise la fonction précédente à tous les réseaux de neurones de type perceptron multicouche (voir équation A.6). Soient :

— L le nombre de couches dans le réseau

— S_l le nombre de neurones dans la couche l

— K le nombre de neurones dans la couche de sortie

Alors,

$$J_\Theta(h_\Theta(x), y) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\Big(h_\Theta(x^{(i)})_k\Big) + \Big(1 - y_k^{(i)}\Big) log\Big(1 - h_\Theta(x^{(i)})_k\Big)\right] \tag{A.6}$$

Cette fonction a l'avantage d'être convexe et donc de posséder exactement un minimum global. De plus, elle est facilement dérivable et de nombreuses optimisations algorithmiques existent pour les calculs nécessaires à l'entraînement du réseau. C'est typiquement la fonction que nous pouvons utiliser pour notre exemple de classification chien/chat ($K = 2$).

A.2.2 Algorithmes

Dans cette section, nous présentons les différents calculs utilisés pour entraîner un réseau de neurones et faire des prédictions. Pour des raisons d'optimisation, ces réseaux sont modélisés par des matrices et les calculs utilisent l'algèbre linéaire. Ainsi, les données en entrée sont représentées par un vecteur et les poids entre

deux couches successives par une matrice. Les exemples de calculs sont tous basés sur le réseau présenté dans la figure A.3. Dans ce cas, il faut fournir au réseau un ensemble d'exemples d'entraînement sous la forme (x, y), où x représente l'ensemble des données en entrée et y est la sortie attendue. x est un vecteur de dimension 3×1 et y est un vecteur de dimension 2×1 (voir équation A.3).

A.2.2.1 Prédire un résultat : Forward Propagation

Que ce soit pour l'apprentissage ou la mise en production, un réseau de neurones doit être capable de prédire un résultat à partir de données en entrée. Pour ce faire, ces dernières sont propagées à travers chaque couche du réseau. Nous rappelons que chaque flèche de la figure A.3 correspond à un poids numérique particulier. Soient :

— $\Theta^{(j)}$ la matrice des poids entre la couche j et la couche $j+1$

— $z_i^{(j)}$ la somme pondérée en entrée du neurone i de la couche j

— $a_i^{(j)}$ l'activation du neurone i de la couche j

La première couche (la couche d'entrée) est particulière car elle ne fait que transmettre l'information brute. Il n'y a donc pas de calculs particulier et nous obtenons directement l'activation de la première couche à partir des données en entrée.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = a^{(1)} \tag{A.7}$$

L'activation de chaque neurone i de la couche cachée est calculée comme suit. D'abord, on s'occupe de la somme pondérée $z_i^{(2)}$.

$$\begin{aligned} z_i^{(2)} &= \Theta_i^{(1)} x \\ &= \Theta_{i,1}^{(1)} x_1 + \Theta_{i,2}^{(1)} x_2 + \Theta_{i,3}^{(1)} x_3 \end{aligned} \tag{A.8}$$

Une fois cette somme calculée, on obtient l'activation de chaque neurone i de la couche cachée grâce à la fonction d'activation choisie. Dans notre exemple de classification chien/chat, plusieurs choix sont possibles. Le meilleur dépend essentiellement des données disponibles. Nous choisissons pour l'exemple la fonction sigmoïde (voir figure A.4b).

$$a_i^{(2)} = sigmo(z_i^{(2)}) \tag{A.9}$$

Une fois que le calcul de l'activation de tous les neurones de la couche cachée est terminé, on réitère les étapes précédentes pour la couche de sortie qui contient dans notre exemple deux neurones. D'abord, on calcule les sommes pondérées $z^{(3)}$. Cependant, la fonction d'activation utilisée est maintenant la fonction $softmax$ définie dans l'équation A.2. La sortie du réseau devient donc :

$$h_\Theta(x) = a^{(3)} = \sigma(z^{(3)}) \tag{A.10}$$

Pour résumer, dans un réseau à L couches, où f est une fonction d'activation, les équations pour calculer la sortie de la $n^{ème}$ couche sont de la forme suivante :

$$\begin{aligned} z^{(n)} &= \Theta^{(n-1)} a^{(n-1)} \\ a^{(n)} &= f(z^{(n)}) \end{aligned} \tag{A.11}$$

Le résultat de la prédiction correspond à l'activation de la couche de sortie du réseau. L'équation finale est donc la suivante :

$$h_\Theta(x) = a^{(L)} \tag{A.12}$$

A.2.2.2 Mettre à jour les poids : Backpropagation

Comme nous l'avons déjà dit, l'entraînement du réseau correspond à la mise à jour des différents poids. L'objectif est de minimiser la fonction de coût J_Θ. L'idée générale est de trouver comment modifier les poids entre deux couches pour minimiser cette fonction. En effet, la sortie du réseau dépend de la somme pondérée précédente, qui elle-même dépend de la sortie de la couche cachée précédente, *etc.* Pour cela, il est nécessaire de calculer J_Θ ainsi que les dérivées partielles $\frac{dJ_\Theta}{d\Theta^l_{ij}}$ selon chacun des poids. Les couches sont alors traitées les unes après les autres, en commençant par la dernière. L'erreur commise par le réseau est alors propagée vers les premières couches. On calcule à chaque fois les dérivées partielles pour savoir comment modifier les différents poids du réseau et diminuer l'erreur commise. Ainsi, si un poids est particulièrement responsable de l'erreur dans la couche suivante, il sera modifié plus agressivement qu'un autre.

Si on observe la sous-section précédente, nous pouvons remarquer que les calculs sont majoritairement des compositions de fonctions. Nous allons donc tirer avantage du théorème de dérivation des fonctions composées (*chain rule*). Il stipule que si la valeur d'une variable y dépend de la valeur d'une variable u, et que cette dernière dépend de la valeur d'une troisième variable x, on peut calculer l'évolution de y selon x grâce à la formule suivante :

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \times \frac{\partial u}{\partial x} \tag{A.13}$$

Nous nous plaçons maintenant dans le cadre du réseau défini dans la figure A.3. Ce dernier est défini ainsi :

- la couche 1 (couche d'entrée) est constituée de trois unités
- la couche 2 (couche cachée) est constituée de trois unités
- la couche 3 (couche de sortie) est constituée de deux unités
- $\Theta^{(1)}$, de dimension 3×3, est la matrice des poids entres les couches 1 et 2
- $\Theta^{(2)}$, de dimension 3×2, est la matrice des poids entres les couches 2 et 3

Voici la procédure à suivre pour modifier les poids du réseau afin d'améliorer la qualité des prédictions. Imaginons qu'un exemple d'entraînement (x, y) a été fourni

au réseau. L'étape de prédiction vient de se terminer et le réseau a produit une valeur $h_\Theta(x)$. La première chose à faire est de calculer l'erreur E commise par le réseau de neurones avec une fonction de coût (voir section A.2.1.2). Cette erreur est la somme des erreurs commises par chacun des neurones de la couche de sortie. Ainsi,

$$E = E_1 + E_2 \tag{A.14}$$

où E_1 et E_2 représentent l'erreur respectivement commise par les deux neurones de la couche de sortie.

En reprenant les équations de la sous-section précédente, nous devons découvrir comment modifier les poids $\Theta^{(2)}$ pour diminuer l'erreur E. Nous nous trouvons exactement dans la situation où nous pouvons utiliser le théorème de dérivations de fonctions composées. En effet, l'erreur E dépend de la sortie $h_\Theta(x)$ (ou $a^{(3)}$) du réseau, qui elle-même dépend de la somme pondérée $z^{(3)}$ à l'entrée des neurones de la couche de sortie. Finalement, cette somme dépend directement des poids $\Theta(2)$. Ainsi,

$$\frac{\partial E}{\partial \Theta^{(2)}} = \frac{\partial E}{\partial h_\Theta(x)} \times \frac{\partial h_\Theta(x)}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial \Theta^{(2)}} \tag{A.15}$$

Les valeurs obtenues pour chaque poids correspondent à leur impact sur l'erreur finale commise par le réseau. Chaque poids $\theta_i^{(2)}$ de la matrice $\Theta^{(2)}$ est alors modifié selon l'équation suivante :

$$\theta_i^{(2)} = \theta_i^{(2)} - \alpha \frac{\partial E}{\partial \theta_i^{(2)}} \tag{A.16}$$

La valeur α représente le taux d'apprentissage. Elle est typiquement assez faible (inférieure à 1) pour éviter à l'algorithme de dépasser la solution idéale. De la même manière, nous allons maintenant nous occuper de la mise à jour des poids de la matrice $\Theta^{(1)}$. Nous devons pour cela calculer l'impact de ces poids sur l'erreur finale commise par le réseau. Nous cherchons donc :

$$\frac{\partial E}{\partial \Theta^{(1)}} = \frac{\partial E}{\partial a^{(2)}} \times \frac{\partial a^{(2)}}{\partial z^{(2)}} \times \frac{\partial z^{(2)}}{\partial \Theta^{(1)}} \tag{A.17}$$

Le terme le plus problématique est ici $\frac{\partial E}{\partial a^{(2)}}$. En effet, la sortie d'un seul neurone de la couche cachée conditionne les valeurs calculées par les deux neurones de la couche de sortie. L'équation finale devient donc :

$$\frac{\partial E}{\partial \Theta^{(1)}} = \left(\frac{\partial E_1}{\partial a^{(2)}} + \frac{\partial E_2}{\partial a^{(2)}} \right) \times \frac{\partial a^{(2)}}{\partial z^{(2)}} \times \frac{\partial z^{(2)}}{\partial \Theta^{(1)}} \tag{A.18}$$

Les poids sont finalement mis à jour de la même manière que précédemment. Idéalement et une fois que le réseau a été nourri d'assez d'exemples, les poids devraient converger pour permettre au réseau de faire des prédictions correctes lorsqu'on lui présentera à nouveau des données.

A.3 Discussion

Nous avons présenté dans cette annexe le fonctionnement des réseaux de neurones de type perceptron multicouches. Il reste encore cependant de nombreux détails à développer. Par exemple, pour des raisons de simplicité, nous avons omis de parler des unités de biais présentes dans toutes les couches sauf celle de sortie. Ces unités particulières ne sont dépendantes d'aucune autre donnée. Leur rôle est simplement d'émettre une constante (typiquement 1 ou -1) vers la couche suivante. Comme les autres neurones, un poids synaptique est associé à ce biais. L'importance de ce biais dans la résolution de problèmes a été démontré par Bernard Widrow [Widrow 1960]. Sans la présence de cette unité, certains problèmes deviennent impossible à résoudre.

Il faut également être très attentif aux données que l'on fournit au réseau. Nous n'en avons pas parlé jusque là mais il est important de préparer les données avant l'apprentissage. La normalisation est notamment une étape essentielle pour un entraînement efficace du réseau. Selon les performances de ce réseau, il convient aussi de se demander si on dispose d'assez de données, ou si l'architecture du réseau est correcte (nombre de couches cachées, nombre de neurones pour chaque couche, *etc*). Le terme "deep learning" (ou apprentissage profond) est utilisé lorsque le réseau comporte plus qu'une seule couche cachée. Quoi qu'il en soit, ces paramètres auront un impact sur la qualité des prédictions du réseau. Ce dernier est également sensible au sur-apprentissage. La conséquence est que le réseau est incapable d'effectuer une prédiction correcte lorsque les données en entrée sont différentes des données d'entraînement. Pour éviter cela, on introduit dans la fonction de coût J_Θ ce qu'on appelle un terme de régularisation. La nouvelle fonction de coût devient alors :

$$J_\Theta^{new} = J_\Theta^{old} + \frac{\lambda}{2m} \sum_\theta \theta^2 \tag{A.19}$$

Le fait d'ajouter la somme des poids du réseau à la fonction de coût permet de forcer le réseau à les minimiser. La valeur λ joue le rôle de levier entre la minimisation du coût ou des poids. Cette technique permet pour résumer d'empêcher au réseau d'apprendre une fonction trop compliquée et peu généralisable.

Il y donc beaucoup de détails auxquels il faut faire attention lorsqu'on s'intéresse aux réseaux de neurones. Et les résultats obtenus sont souvent très dépendants de l'étape de conception du réseau. Pour cette raison, il est important de tester ses performances de manière cohérente et réfléchie. C'est souvent fait en séparant les données d'entraînement en trois échantillons :

— un échantillon d'apprentissage qui est utilisé pour l'entraînement du réseau, c'est-à-dire la mise à jour des poids,

— un échantillon de validation qui permet d'optimiser la topologie du réseau (nombre de neurones dans les couches cachées par exemple),

— un échantillon de test qui permet d'évaluer les performances d'un réseau entraîné.

A.4 Conclusion

Nous avons donc présenté dans cette annexe le fonctionnement général du perceptron multicouches, un type particulier de réseau de neurones. Ces derniers ont gagné en popularité ces dernières années, notamment grâce à l'évolution des technologies matérielles, qui permettent des calculs toujours plus rapides. Les réseaux de neurones sont extrêmement polyvalents et peuvent être utilisés dans de très nombreux cas : génération de texte, reconnaissance d'image, intelligence artificielle, *etc.*

Annexe B

Preuves des propriétés des opérateurs lexicographiques

B.1 Propriétés des fonctions lex_{min} et lex_{max}

La fonction lex_{min} est définie comme suit :

$$\begin{aligned} lex_{min} : ([0,1]\times[0,1])^2 &\rightarrow [0,1]\times[0,1] \\ ((x,y),(a,b)) &\mapsto lex_{min}((x,y),(x',y')) = \begin{cases}(x,y) & if\ x<x' \vee (x=x' \wedge y<y')\\ (x',y') & sinon\end{cases}\end{aligned}$$

Sa fonction duale est lex_{max}. Elle est définie comme suit :

$$\begin{aligned} lex_{max} : ([0,1]\times[0,1])^2 &\rightarrow [0,1]\times[0,1] \\ ((x,y),(a,b)) &\mapsto lex_{maw}((x,y),(x',y')) = \begin{cases}(x,y) & if\ x>x' \vee (x=x' \wedge y>y')\\ (x',y') & sinon\end{cases}\end{aligned}$$

Propriété 1. Les opérateurs lex_{min} et lex_{max} sont tous les deux commutatifs, associatifs et monotones. L'élément neutre de lex_{max} (respectivement lex_{max}) est $(1,1)$ (respectivement $(0,0)$) et son élément absorbant est $(0,0)$ (respectivement $(1,1)$).

Preuves des propriétés de lex_{min}

Commutativité

Nous allons maintenant montrer que $lex_{min}((x,y),(x',y')) = lex_{min}((x',y'),(x,y))$. Les différents cas possibles sont présentés dans le tableau B.1.

Tableau B.1 – Cas possibles pour l'opérateur lex_{min}.

	$x<x'$	$(x=x')\wedge(y<y')$	$x>x'$	$(x=x')\wedge(y\geq y')$
$lex_{min}((x,y),(x',y'))$	(x,y)	(x,y)	(x',y')	(x',y')
$lex_{min}((x',y'),(x,y))$	(x,y)	(x,y)	(x',y')	(x',y')

Dans tous les cas, $lex_{min}((x,y),(x',y')) = lex_{min}((x',y'),(x,y))$; ainsi, lex_{min} est commutatif.

Associativité

Soient $(x,y),(x',y'),(x'',y'')$ des couples dans $([0,1]\times[0,1])^2$, nous allons montrer que les formules A et B sont égales.

$$A = lex_{min}((x,y), lex_{min}((x',y'),(x'',y''))) = \begin{cases} lex_{min}((x,y),(x',y')) & if\ x' < x'' \vee (x' = x'' \wedge y' < y'') \\ lex_{min}((x,y),(x'',y'')) & sinon. \end{cases}$$

$$B = lex_{min}(lex_{min}((x,y),(x',y')),(x'',y'')) = \begin{cases} lex_{min}((x,y),(x'',y'')) & if\ x < x' \vee (x = x' \wedge y < y') \\ lex_{min}((x',y'),(x'',y'')) & sinon. \end{cases}$$

La preuve est fondée sur les cas suivants.

1. **cas 1 :** $x' < x'' \vee (x' = x'' \wedge y' < y'')$

   $$A = lex_{min}((x,y),(x',y')) = \begin{cases} (x,y) & if\ x < x' \vee (x = x' \wedge y < y') \\ (x',y') & sinon. \end{cases},$$

   (a) **cas 1-1 :** $x < x' \vee (x = x' \wedge y < y')$

   Nous obtenons :

   $A = (x,y)$

   $B = lex_{min}((x,y),(x'',y''))$

   Le tableau B.3 résume les valeurs possibles de B suivant les contraintes des cas 1 et 1-1.

   Tableau B.3 – Valeurs de B selon les cas 1 et 1-1.

	$x < x'$	$x = x' \wedge y < y'$
$x' < x''$	$\Rightarrow x < x'' \Rightarrow B = (x,y)$	$\Rightarrow x < x'' \Rightarrow B = (x,y)$
$x' = x'' \wedge y' < y''$	$\Rightarrow x < x'' \Rightarrow B = (x,y)$	$\Rightarrow (x = x'' \wedge y < y'') \Rightarrow B = (x,y)$

   Comme indiqué dans le tableau B.3 : $A = B$.

   (b) **cas 1-2 :** la partie “sinon” du cas 1-1

   Nous obtenons :

   $A = (x',y')$,

   $B = lex_{min}((x',y'),(x'',y'')) = (x',y')$

   Ce résultat est imposé par la condition du cas 1. Ainsi, nous obtenons $A = B$.

2. **cas 2 :** la partie “sinon” du cas 1

   Cette condition est exprimée comme suit :

$x' \geq x'' \wedge (x' \neq x'' \vee y' \geq y'') \Leftrightarrow (x' \geq x'' \wedge x' \neq x'') \vee (x' \geq x'' \wedge y' \geq y'') \Leftrightarrow (x' > x'') \vee (x' = x'' \wedge y' \geq y'')$

La valeur de A est :

$$A = lex_{min}((x,y),(x'',y'')) = \begin{cases} (x,y) & if\ x < x'' \vee (x = x'' \wedge y < y'') \\ (x'',y'') & sinon. \end{cases}$$

(a) **cas 2-1 :** $x < x'' \vee (x = x'' \wedge y < y'')$

$A = (x,y)$, Le tableau B.5 résume les valeurs possibles de B suivant les contraintes des cas 2 et 2-1.

TABLEAU B.5 – Valeurs de B selon les cas 2 et 2-1.

	$x < x''$	$x = x'' \wedge y < y''$
$x' > x''$	$\Rightarrow x < x' \Rightarrow B =$ $lex_{min}((x,y),(x'',y''))= (x,y)$ car $x < x''$	$\Rightarrow x < x' \Rightarrow B =$ $lex_{min}((x,y),(x'',y''))= (x,y)$ car $x = x'' \wedge y < y''$
$x' = x'' \wedge y' \geq y''$	$\Rightarrow x < x' \Rightarrow B =$ $lex_{min}((x,y),(x'',y''))= (x,y)$ car $x < x''$	$\Rightarrow x = x' \wedge y < y' \Rightarrow B =$ $lex_{min}((x,y),(x'',y''))= (x,y)$ car $x = x'' \wedge y < y''$

Nous obtenons alors $A = B$.

(b) **cas 2-2 :** la partie "sinon" du cas 2-1

Cette condition est exprimée comme suit : $x > x'' \vee (x = x'' \wedge y \geq y'')$. Nous obtenons $A = (x'',y'')$. Le tableau B.7 résume les valeurs possibles de B suivant les contraintes des cas 2 et 2-2.

TABLEAU B.7 – Valeurs de B selon les cas 2 et 2-2.

	$x > x''$	$x = x'' \wedge y \geq y''$
$x' > x''$	$\Rightarrow B = (x'',y'')$ dans tous les cas car $x'' < x \wedge x'' < x$	$\Rightarrow x < x' \Rightarrow B =$ $lex_{min}((x,y),(x'',y''))= (x'',y'')$ car $x = x'' \wedge y'' \leq y$
$x' = x'' \wedge y' \geq y''$	$\Rightarrow x > x' \Rightarrow B =$ $lex_{min}((x',y'),(x'',y''))= (x'',y'')$ car $x' = x'' \wedge y'' \leq y'$	$x = x' = x''$ et $(y'' \leq y \wedge y'' \leq y')$ $\Rightarrow B = (x'',y'')$ dans tous les cas.

Nous obtenons également $A = B$.

Élément neutre

Nous allons montrer que le couple $(1,1)$ est l'élément neutre de lex_{min}.

$lex_{min}((x,y),(1,1)) = (x,y)$ puisque $\forall x,y : x \leq 1 \wedge y \leq 1$, nous avons :

si $x < 1$ alors, à partir de la définition de lex_{min}, nous obtenons $lex_{min}((x,y),(1,1)) = (x,y)$,

sinon nous avons $x = 1$ et comme $y \leq 1$ alors $lex_{min}((x,y),(1,1)) = (x,y)$.

Élément absorbant

Nous allons montrer que le couple $(0,0)$ est l'élément absorbant de lex_{min}.

$lex_{min}((x,y),(0,0) = (0,0)$ puisque $\forall x,y : x \geq 0 \wedge y \geq 0$, nous avons :

si $x > 0$ alors, à partir de la définition de lex_{min}, nous obtenons $lex_{min}((x,y),(0,0)) = (0,0)$,

sinon nous avons $x = 0$ et comme $y \geq 0$ alors $lex_{min}((x,y),(0,0)) = (0,0)$.

Monotonie

Nous allons démontrer l'aspect monotone de cet opérateur. C'est-à-dire :

$$\left.\begin{array}{c}(x,y) \geq (a,b)\\(x',y') \geq (a',b')\end{array}\right\} \Rightarrow lex_{min}((x,y),(x',y')) \geq lex_{min}((a,b),(a',b')).$$

Supposons que $(x,y) \geq (a,b)$ et $(x',y') \geq (a',b')$, soient A, B deux couples définis comme suit :

$$A = lex_{min}((x,y),(x',y')) = \begin{cases}(x,y) & if\ x < x' \vee (x = x' \wedge y < y')\\(x',y') & sinon.\end{cases}$$

$$B = lex_{min}((a,b),(a',b')) = \begin{cases}(a,b) & if\ a < a' \vee (a = a' \wedge b < b')\\(a',y') & sinon.\end{cases}$$

Nous allons démontrer que $A \geq B$ dans tous les cas contraints par les conditions initiales. Nous résumons dans le tableau B.9 tous les cas possibles pour A et B.

Tableau B.9 – Comparaison entre A et B.

	(x,y)	(x',y')
(a,b)	comme $(x,y) \geq (a,b)$ alors $A \geq B$	voir cas (1)
(a',b')	voir cas (2)	comme $(x',y') \geq (a',b')$ alors $A \geq B$

Les cas 1 et 2 sont basés sur les équivalences suivantes :

1. $(x,y) \geq (a,b) \Leftrightarrow (x > a) \vee (x = a \wedge y \geq b)$
2. $(x',y') \geq (a',b') \Leftrightarrow (x' > a') \vee (x' = a' \wedge y' \geq b')$

— Cas (1) : $A = (x',y')$ et $B = (a,b)$

Comme $B = (a, b)$ alors $a < a' \vee (a = a' \wedge b < b')$. Nous nous basons sur l'équivalence $(x', y') \geq (a', b') \Leftrightarrow (x' > a') \vee (x' = a' \wedge y' \geq b')$ pour prouver que $(x', y') \geq (a, b)$. Le tableau B.11 résume les cas possibles.

Tableau B.11 – Comparaison de A et B dans le cas (1).

	$x' > a'$	$x' = a' \wedge y' \geq b'$
$a < a'$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$
$a = a' \wedge b < b'$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$	$\Rightarrow x' = a \wedge y' \geq b' > b \Rightarrow$ $(x', y') \geq (a, b), \Rightarrow A \geq B$

Nous obtenons en effet $A \geq B$.

— Cas (2) : $A = (x, y)$ et $B = (a', b')$
as $B = (a', b')$ then $a > a' \vee (a = a' \wedge b \geq b')$, nous nous basons sur l'équivalence $(x, y) \geq (a, b) \Leftrightarrow (x > a) \vee (x = a \wedge y \geq b)$ pour prouver que $(x, y) \geq (a', b')$. Le tableau B.13 résume les cas possibles.

Tableau B.13 – Comparaison de A et B pour le cas (2).

	$x > a$	$x = a \wedge y \geq b$
$a > a'$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$
$a = a' \wedge b \geq b'$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$	$\Rightarrow x = a' \wedge y \geq b \geq b' \Rightarrow$ $(x, y) \geq (a', b'), \Rightarrow A \geq B$

Nous obtenons aussi $A \geq B$.

Preuves des propriétés de lex_{max}

Commutativité

Nous allons maintenant montrer que $lex_{max}((x, y), (x', y')) = lex_{max}((x', y'), (x, y))$. Les différents cas possibles sont présentés dans le tableau B.15.

Tableau B.15 – Cas possibles pour l'opérateur lex_{max}.

	$x > x'$	$(x = x') \wedge (y > y')$	$x < x'$	$(x = x') \wedge (y \leq y')$
$lex_{max}((x, y), (x', y'))$	(x, y)	(x, y)	(x', y')	(x', y')
$lex_{max}((x', y'), (x, y))$	(x, y)	(x, y)	(x', y')	(x', y')

Dans tous les cas, nous avons $lex_{max}((x,y),(x',y')) = lex_{max}((x',y'),(x,y))$; ainsi, lex_{max} est commutatif.

Associativité

Soient $(x,y),(x',y'),(x'',y'')$ des couples dans $([0,1]\times[0,1])^2$, nous allons montrer que les formules A et B sont égales.

$$A = lex_{max}((x,y), lex_{max}((x',y'),(x'',y''))) = \begin{cases} lex_{max}((x,y),(x',y')) & if\ x' > x'' \vee (x' = x'' \wedge y' > y'') \\ lex_{max}((x,y),(x'',y'')) & sinon. \end{cases}$$

$$B = lex_{max}(lex_{max}((x,y),(x',y')),(x'',y'')) = \begin{cases} lex_{max}((x,y),(x'',y'')) & if\ x > x' \vee (x = x' \wedge y > y') \\ lex_{max}((x',y'),(x'',y'')) & sinon. \end{cases}$$

La preuve est fondée sur les cas suivants.

1. **cas 1 :** $x' > x'' \vee (x' = x'' \wedge y' > y'')$

   $$A = lex_{max}((x,y),(x',y')) = \begin{cases} (x,y) & if\ x > x' \vee (x = x' \wedge y > y') \\ (x',y') & sinon. \end{cases},$$

   (a) **cas 1-1 :** $x > x' \vee (x = x' \wedge y > y')$

   Nous obtenons : $A = (x,y)$ et $B = lex_{max}((x,y),(x'',y''))$.

   Le tableau B.17 résume les valeurs possibles de B suivant les contraintes des cas 1 et 1-1 :

   Tableau B.17 – Valeurs de B selon les cas 1 et 1-1.

	$x > x'$	$x = x' \wedge y > y'$
$x' > x''$	$\Rightarrow x > x'' \Rightarrow B = (x,y)$	$\Rightarrow x > x'' \Rightarrow B = (x,y)$
$x' = x'' \wedge y' > y''$	$\Rightarrow x > x'' \Rightarrow B = (x,y)$	$\Rightarrow (x = x'' \wedge y > y'') \Rightarrow B = (x,y)$

   Comme indiqué dans le tableau B.17, $A = B$.

   (b) **case 1-2 :** la partie "sinon" du cas 1-1

   Ici, nous obtenons : $A = (x',y')$, $B = lex_{min}((x',y'),(x'',y'')) = (x',y')$, résultat imposé par la condition du cas 1. Ainsi, nous obtenons $A = B$.

2. **case 2 :** la partie "sinon" du cas 1

   Cette condition est exprimée comme suit : $x' \leq x'' \wedge (x' \neq x'' \vee y' \leq y'') \Leftrightarrow (x' \leq x'' \wedge x' \neq x'') \vee (x' \leq x'' \wedge y' \leq y'') \Leftrightarrow (x' < x'') \vee (x' = x'' \wedge y' \leq y'')$.

   La valeur de A est : $A = lex_{max}((x,y),(x'',y'')) = \begin{cases} (x,y) & if\ x > x'' \vee (x = x'' \wedge y > y'') \\ (x'',y'') & sinon. \end{cases}$

(a) **case 2-1 :** $x > x'' \vee (x = x'' \wedge y > y'')$
$A = (x, y)$,
Le tableau B.19 résume les valeurs possibles de B suivant les contraintes des cas 2 et 2-1.

TABLEAU B.19 – Valeurs de B selon les cas 2 et 2-1.

	$x > x''$	$x = x'' \wedge y > y''$
$x' < x''$	$\Rightarrow x > x' \Rightarrow B =$ $lex_{max}((x, y), (x'', y''))= (x, y)$ car $x > x''$	$\Rightarrow x > x' \Rightarrow B =$ $lex_{max}((x, y), (x'', y''))= (x, y)$ car $x = x'' \wedge y > y''$
$x' = x'' \wedge y' \leq y''$	$\Rightarrow x > x' \Rightarrow B =$ $lex_{max}((x, y), (x'', y''))= (x, y)$ car $x > x''$	$\Rightarrow x = x' \wedge y > y' \Rightarrow B =$ $lex_{max}((x, y), (x'', y''))= (x, y)$ car $x = x'' \wedge y > y''$

Nous obtenons $A = B$.

(b) **case 2-2 :** La partie "sinon" du cas 2-1
Cette condition est exprimée comme suit : $x < x'' \vee (x = x'' \wedge y \leq y'')$.
Ainsi, nous obtenons $A = (x'', y'')$.
Le tableau B.21 résume les valeurs possibles de B suivant les contraintes des cas 2 et 2-1.

TABLEAU B.21 – Valeurs de B selon les cas 2 et 2-2.

	$x < x''$	$x = x'' \wedge y \leq y''$
$x' < x''$	$\Rightarrow B = (x'', y'')$ dans tous les cas car $x'' > x \wedge x'' > x$	$\Rightarrow x > x' \Rightarrow B =$ $lex_{max}((x, y), (x'', y''))= (x'', y'')$ car $x = x'' \wedge y'' \geq y$
$x' = x'' \wedge y' \leq y''$	$\Rightarrow x < x' \Rightarrow B =$ $lex_{max}((x', y'), (x'', y''))= (x'', y'')$ car $x' = x'' \wedge y'' \geq y'$	$x = x' = x''$ and $(y'' \geq y \wedge y'' \geq y') \Rightarrow B = (x'', y'')$ dans tous les cas

Nous obtenons finalement $A = B$.

Élément neutre

Nous allons montrer que le couple $(0, 0)$ est l'élément neutre de lex_{max}.

$lex_{max}((x, y), (0, 0)) = (x, y)$ puisque :

Comme $x \in [0, 1]$ et $y \in [0, 1]$ alors $\forall x, x \geq 0$ et deux cas sont à distinguer :

— Cas 1 : $x > 0$ alors par définition, $lex_{max}((x, y), (0, 0)) = (x, y)$.

— Cas 2 : $x = 0$ et $y \geq 0$ alors $lex_{max}((x, y), (0, 0)) = (x, y)$.

Élément absorbant

Nous allons montrer que le couple $(1,1)$ est l'élément neutre de lex_{max}.

$lex_{max}((x,y),(1,1)) = (1,1)$ puisque :

Comme $x \in [0,1]$ et $y \in [0,1]$ alors $\forall x, x \leq 1$ et deux cas sont à distinguer :

— Cas 1 : $x < 1$ alors par définition, $lex_{max}((x,y),(1,1)) = (1,1)$.

— Cas 2 : $x = 1$ et $y \leq 1$ alors $lex_{max}((x,y),(1,1)) = (1,1)$.

Monotonie

Nous allons montrer l'aspect monotone de cet opérateur. C'est-à-dire :

$$\left.\begin{array}{c}(x,y) \geq (a,b)\\(x',y') \geq (a',b')\end{array}\right\} \Rightarrow lex_{max}((x,y),(x',y')) \geq lex_{max}((a,b),(a',b')).$$

Supposons que $(x,y) \geq (a,b)$ et $(x',y') \geq (a',b')$, soient A, B deux couples définis comme suit :

$$A = lex_{max}((x,y),(x',y')) = \begin{cases}(x,y) & if\, x > x' \vee (x = x' \wedge y > y')\\(x',y') & sinon.\end{cases}$$

$$B = lex_{max}((a,b),(a',b')) = \begin{cases}(a,b) & if\, a > a' \vee (a = a' \wedge b > b')\\(a',y') & sinon.\end{cases}$$

Nous allons démontrer que $A \geq B$ dans tous les cas possibles. Nous résumons dans le tableau B.23 les cas possibles pour A et B :

Tableau B.23 – Comparaison de A et B.

	(x,y)	(x',y')
(a,b)	comme $(x,y) \geq (a,b)$ alors $A \geq B$	voir cas (1)
(a',b')	voir cas (2)	comme $(x',y') \geq (a',b')$ alors $A \geq B$

Les cas 1 et 2 sont basés sur les équivalences suivantes :

1. $(x,y) \geq (a,b) \Leftrightarrow (x > a) \vee (x = a \wedge y \geq b)$
2. $(x',y') \geq (a',b') \Leftrightarrow (x' > a') \vee (x' = a' \wedge y' \geq b')$

— Cas (1) : $A = (x',y')$ et $B = (a,b)$

Comme $A = (x',y')$ alors $x < x' \vee (x = x' \wedge y \leq y')$. Nous nous basons sur l'équivalence $(x,y) \geq (a,b) \Leftrightarrow (x > a) \vee (x = a \wedge y \geq b)$ pour montrer que $(x',y') \geq (a,b)$. Le tableau B.25 résume les cas possibles.

TABLEAU B.25 – Comparaison de A et B dans le cas (1).

	$x > a$	$x = a \wedge y \geq b$
$x < x'$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$
$x = x' \wedge y \leq y'$	$\Rightarrow x' > a \Rightarrow (x', y') \geq (a, b)$, $\Rightarrow A \geq B$	$\Rightarrow x' = a \wedge y' \geq y \geq b \Rightarrow$ $(x', y') \geq (a, b), \Rightarrow A \geq B$

Nous obtenons en effet $A \geq B$.

— Cas (2) : $A = (x, y)$ et $B = (a', b')$
Comme $A = (x, y)$ alors $x > x' \vee (x = x' \wedge y > y')$. Nous nous basons sur l'équivalence $(x', y') \geq (a', b') \Leftrightarrow (x' > a') \vee (x' = a' \wedge y' \geq b')$ pour montrer que $(x, y) \geq (a', b')$. Le tableau B.27 résume les cas possibles.

TABLEAU B.27 – Comparaison de A et B dans le cas (2).

	$x > x'$	$x = x' \wedge y > y'$
$x' > a'$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$
$x' = a' \wedge y' \geq b'$	$\Rightarrow x > a' \Rightarrow (x, y) \geq (a', b')$, $\Rightarrow A \geq B$	$\Rightarrow x = a' \wedge y > y' \geq b' \Rightarrow$ $(x, y) \geq (a', b'), \Rightarrow A \geq B$

Nous obtenons aussi $A \geq B$.

B.2 Opérateurs de comparaison $\preceq_{lex}$ et $\succeq_{lex}$

Définition. [Les opérateurs $\preceq_{lex}$ et $\succeq_{lex}$]. Soient e_i, e_j, e'_i et e'_j des entités nommées dans $\mathbb{E}$.

• la similarité du couple (e_i, e_j) est plus petite ou égale à la similarité du couple (e'_i, e'_j), dénoté par $(e_i, e_j) \preceq_{lex} (e'_i, e'_j)$, si et seulement si $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$. Formellement :

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : (e_i, e_j) \preceq_{lex} (e'_i, e'_j) \Leftrightarrow lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j) \tag{B.1}$$

• la similarité du couple (e_i, e_j) est plus grande ou égale à la similarité du couple (e'_i, e'_j), dénoté par $(e_i, e_j) \succeq_{lex} (e'_i, e'_j)$, si et seulement si $lex_{max}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$. Formellement :

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : (e_i, e_j) \succeq_{lex} (e'_i, e'_j) \Leftrightarrow lex_{max}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j) \tag{B.2}$$

Théorème 1. [Ordre total]. Les opérateurs $\preceq_{lex}$ et $\succeq_{lex}$ sont tous les deux des ordres totaux.

Preuve 2. Les preuves que $\preceq_{lex}$ et $\succeq_{lex}$ sont antisymétriques, transitifs et totaux.

Preuve que $\preceq_{lex}$ est un ordre total

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j) \Leftrightarrow lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$$

Antisymétrie

($\preceq_{lex}$ est antisymétrique) $\Leftrightarrow$

$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \Big(\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)\Big) \wedge \Big(\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e_i, e_j)\Big) \Rightarrow \mathbb{S}(e_i, e_j) = \mathbb{S}(e'_i, e'_j)$

Soient $(e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2$ deux couples d'entités nommées. Nous avons alors :

— $\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j) \Rightarrow lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$

— $\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e_i, e_j) \Rightarrow lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) = \mathbb{S}(e'_i, e'_j)$

Comme lex_{min} est commutatif, alors :

$lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) \Rightarrow \mathbb{S}(e_i, e_j) = \mathbb{S}(e'_i, e'_j)$. Ainsi, $\preceq_{lex}$ est antisymétrique.

Transitivité

($\preceq_{lex}$ est transitive) $\Leftrightarrow$

$\forall (e_i, e_j), (e'_i, e'_j), (e''_i, e''_j) \in \mathbb{E}^2 : \Big(\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)\Big) \wedge \Big(\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e''_i, e''_j)\Big) \Rightarrow \Big(\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e''_i, e''_j)\Big)$

Soient $(e_i, e_j), (e'_i, e'_j), (e''_i, e''_j) \in \mathbb{E}^2$ trois couples d'entités nommées. Nous avons alors :

1. $\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j) \Rightarrow lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$
2. $\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e''_i, e''_j) \Rightarrow lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e''_i, e''_j)\big) = \mathbb{S}(e'_i, e'_j)$

On dénote $\mathbb{S}(e_i, e_j)$, $\mathbb{S}(e'_i, e'_j)$ et $\mathbb{S}(e''_i, e''_j)$ par (x, y), (x', y') et (x'', y'') respectivement.

— Depuis 1.) nous avons $(x < x') \vee (x = x' \wedge y < y')$

— Depuis 2.) nous avons $(x' < x'') \vee (x' = x'' \wedge y' < y'')$

Il y a quatre cas possibles pour calculer $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big)$:

— Cas 1 : $(x < x') \wedge (x' < x'')$ qui implique que $x < x''$, ainsi : $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big) = (x, y) = \mathbb{S}(e_i, e_j)$.

— Cas 2 : $(x < x') \wedge (x' = x'' \wedge y' < y'')$, qui implique que $x < x''$, ainsi : $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big) = (x, y) = \mathbb{S}(e_i, e_j)$.

— Cas 3 : $(x = x' \wedge y \leq y') \wedge (x' < x'')$, qui implique aussi que $x < x''$, ainsi : $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big) = (x, y) = \mathbb{S}(e_i, e_j)$.

— Cas 4 : $(x = x' \wedge y < y') \wedge (x' = x'' \wedge y' < y'')$, qui implique que $(x = x'') \wedge (y < y'')$, ainsi : $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big) = (x, y) = \mathbb{S}(e_i, e_j)$.

Finallement, nous obtenons dans tous les cas possibles : $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e''_i, e''_j)\big) = \mathbb{S}(e_i, e_j)$. Alors, l'opérateur $\preceq_{lex}$ est transitif.

Totalité

($\preceq_{lex}$ est total) $\Leftrightarrow$

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \Big(\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)\Big) \vee \Big(\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e_i, e_j)\Big)$$

Soient $(e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2$ deux couples d'entités nommées.

On dénote $\mathbb{S}(e_i, e_j)$ et $\mathbb{S}(e'_i, e'_j)$ par (x, y) et (x', y') respectivement.

Cela implique que :

1. $\mathbb{S}(e_i, e_j) \quad \preceq_{lex} \quad \mathbb{S}(e'_i, e'_j) \quad \Leftrightarrow \quad lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) \quad =$
$\begin{cases} \mathbb{S}(e_i, e_j) & if (x < x') \vee (x = x' \wedge y < y') \\ \mathbb{S}(e'_i, e'_j) & sinon \end{cases}$
2. $\mathbb{S}(e'_i, e'_j) \quad \preceq_{lex} \quad \mathbb{S}(e_i, e_j) \quad \Leftrightarrow \quad lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) \quad =$
$\begin{cases} \mathbb{S}(e'_i, e'_j) & if (x' < x) \vee (x' = x \wedge y' < y) \\ \mathbb{S}(e_i, e_j) & sinon \end{cases}$

Comme $(x, y), (x', y') \in [0, 1]^2$ alors $\forall x, x' \in [0, 1] : (x < x') \vee (x' < x) \vee (x = x')$.

— Si $x < x'$ alors $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$ et $\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)$

— Si $x' < x$ alors $lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) = \mathbb{S}(e'_i, e'_j)$ et $\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e_i, e_j)$

— Si $x = x'$ alors nous appliquons le même raisonnement sur $y, y' \in [0, 1]$ avec l'opérateur $<$. Ainsi, $\forall y, y' \in [0, 1] : (y < y') \vee (y' < y) \vee (y' = y)$.

  — Si $(y < y')$ ou $(y' < y)$, nous avons soit :
  $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$ ou $lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) = \mathbb{S}(e'_i, e'_j)$ respectivement.

  — Si $y = y'$, alors $\mathbb{S}(e_i, e_j) = \mathbb{S}(e'_i, e'_j)$ et $lex_{min}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e'_i, e'_j)$ et
  $lex_{min}\big(\mathbb{S}(e'_i, e'_j), \mathbb{S}(e_i, e_j)\big) = \mathbb{S}(e_i, e_j)$

Finalement, dans tous les cas, $\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \Big(\mathbb{S}(e_i, e_j) \preceq_{lex} \mathbb{S}(e'_i, e'_j)\Big) \vee \Big(\mathbb{S}(e'_i, e'_j) \preceq_{lex} \mathbb{S}(e_i, e_j)\Big)$ et $\preceq_{lex}$ est total.

Preuve que $\succeq_{lex}$ est un ordre total

$$\forall (e_i, e_j), (e'_i, e'_j) \in \mathbb{E}^2 : \mathbb{S}(e_i, e_j) \succeq_{lex} \mathbb{S}(e'_i, e'_j) \Leftrightarrow lex_{max}\big(\mathbb{S}(e_i, e_j), \mathbb{S}(e'_i, e'_j)\big) = \mathbb{S}(e_i, e_j)$$

Les preuves des propriétés sont similaires à celles détaillées pour l'opérateur $\preceq_{lex}$.

Bibliographie

[Adamo 1980] J.M. Adamo. *Fuzzy decision trees.* Fuzzy Sets and Systems, vol. 4, no. 3, pages 207 – 219, 1980. (Cité en page 76.)

[Afli 2016] Haithem Afli, Zhengwei Qiu, Andy Way et Páraic Sheridan. *Using SMT for OCR error correction of historical texts.* In Proceedings of LREC-2016, Portoro ?, Slovenia (2016, to appear), 2016. (Cité en page 91.)

[Athukorala 2016] Kumaripaba Athukorala, Dorota Głowacka, Giulio Jacucci, Antti Oulasvirta et Jilles Vreeken. *Is exploratory search different ? A comparison of information search behavior for exploratory and lookup tasks.* Journal of the Association for Information Science and Technology, vol. 67, no. 11, pages 2635–2651, 2016. (Cité en pages 48, 49 et 52.)

[Azzopardi 2008] Leif Azzopardi et Vishwa Vinay. *Retrievability : an evaluation measure for higher order information access tasks.* In Proceedings of the 17th ACM conference on Information and knowledge management, pages 561–570. ACM, 2008. (Cité en page 110.)

[Bagga 1998] Amit Bagga et Breck Baldwin. *Algorithms for scoring coreference chains.* In The first international conference on language resources and evaluation workshop on linguistics coreference, volume 1, pages 563–566, 1998. (Cité en page 104.)

[Bagley 1969] Philip R. Bagley. Extension of programming language concepts. National Bureau of Standards, Institute for Applied Technology, 1969. (Cité en page 18.)

[Baum 1966] Leonard E. Baum et Ted Petrie. *Statistical Inference for Probabilistic Functions of Finite State Markov Chains.* The Annals of Mathematical Statistics, vol. 37, no. 6, pages 1554–1563, 12 1966. (Cité en page 78.)

[Blei 2003] David M Blei, Andrew Y Ng et Michael I Jordan. *Latent dirichlet allocation.* Journal of machine Learning research, vol. 3, no. Jan, pages 993–1022, 2003. (Cité en page 75.)

[Blum 1991] Rudolf Blum. Kallimachos : the alexandrian library and the origins of bibliography. Univ of Wisconsin Press, 1991. (Cité en page 10.)

[Borgman 1999] Christine L. Borgman. *What are digital libraries ? Competing visions.* Inf. Process. Manage., vol. 35, no. 3, pages 227–243, 1999. (Cité en page 26.)

[Brooking 1999] Annie Brooking. Corporate memory : Strategies for knowledge management. Cengage Learning EMEA, 1999. (Cité en page 1.)

[Bush 1945] Vannevar Bush*et al. As we may think.* The atlantic monthly, vol. 176, no. 1, pages 101–108, 1945. (Cité en page 11.)

[Candillier 2009] Laurent Candillier, Kris Jack, Françoise Fessant et Frank Meyer. *State-of-the-art recommender systems.* Collaborative and Social Information Retrieval and AccessTechniques for Improved User Modeling, 2009. (Cité en page 17.)

[Chomsky 1956] N. Chomsky. *Three models for the description of language.* IRE Transactions on Information Theory, vol. 2, no. 3, pages 113–124, September 1956. (Cité en page 76.)

[Dacos 2015] Marin Dacos et Pierre Mounier. *Humanités numériques.* Research report, Institut français, 2015. (Cité en page 3.)

[Damgaard 2000] Christian Damgaard et Jacob Weiner. *Describing inequality in plant size or fecundity.* Ecology, vol. 81, no. 4, pages 1139–1142, 2000. (Cité en page 111.)

[Dernoncourt 2017] Franck Dernoncourt, Ji Young Lee et Peter Szolovits. *NeuroNER : an easy-to-use program for named-entity recognition based on neural networks.* 2017. (Cité en pages 72 et 95.)

[DeRose 1988] Steven J. DeRose. *Grammatical Category Disambiguation by Statistical Optimization.* Comput. Linguist., vol. 14, no. 1, pages 31–39, 1988. (Cité en page 68.)

[Ellis 1997] David Ellis et Merete Haugan. *Modelling the information seeking patterns of engineers and research scientists in an industrial environment.* Journal of documentation, vol. 53, no. 4, pages 384–403, 1997. (Cité en page 44.)

[Erway 2009] Ricky Erway. *A view on Europeana from the US perspective.* Liber Quarterly, vol. 19, no. 2, 2009. (Cité en page 28.)

[Evershed 2014] John Evershed et Kent Fitch. *Correcting noisy OCR : context beats confusion.* pages 45–51. ACM Press, 2014. (Cité en page 91.)

[Farahmand 2013] Atena Farahmand, Hossein Sarrafzadeh et Jamshid Shanbehzadeh. *Document image noises and removal methods.* 2013. (Cité en page 92.)

[Fellbaum 1998] Christiane Fellbaum. Wordnet. Wiley Online Library, 1998. (Cité en page 70.)

[Finkel 2005] Jenny Rose Finkel, Trond Grenager et Christopher Manning. *Incorporating non-local information into information extraction systems by gibbs sampling.* In Proceedings of the 43rd annual meeting on association for computational linguistics, pages 363–370. Association for Computational Linguistics, 2005. (Cité en pages 72 et 95.)

[Finnemann 2001] Niels Ole Finnemann. The internet : A new communicational infrastruture. CFI, Center for Internetforskning, Institut for Informations- og medievidenskab, 2001. (Cité en page 9.)

[Fisher 2005] Karen E Fisher, Sanda Erdelez et Lynne McKechnie. Theories of information behavior. Information Today, Inc., 2005. (Cité en page 45.)

[Gantz 2011] John Gantz et David Reinsel. *Extracting value from chaos.* In IDC iview, volume 1142, pages 1–12, 2011. (Cité en page 1.)

[Gastwirth 1972] Joseph L Gastwirth. *The estimation of the Lorenz curve and Gini index*. The review of economics and statistics, pages 306–316, 1972. (Cité en page 111.)

[Gefen 2015] Alexandre Gefen. *Les enjeux épistémologiques des humanités numériques*. Socio, vol. 4, 2015. (Cité en page 96.)

[Gooding 2014] Paul Gooding. *Exploring Usage of Digital Newspaper Archives through Web Log Analysis : A Case Study of Welsh Newspapers Online*. Digital Humanities 2014, 2014. (Cité en page 92.)

[Hofmann 1999] Thomas Hofmann. *Probabilistic Latent Semantic Indexing*. In Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, pages 50–57. ACM, 1999. (Cité en page 75.)

[Holley 2009] Rose Holley. *How good can it get ? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs*. volume 15, 2009. (Cité en pages 67, 92 et 93.)

[Jaffré 2010] Jean-Pierre Jaffré. *De la variation en orthographe*. Ela. Études de linguistique appliquée, no. 3, pages 309–323, 2010. (Cité en page 119.)

[Jahanghiri 2016] Saeideh Jahanghiri et Yaghoub Norouzi. *Software Ergonomics of Iranian Digital Library Softwares : an Accessibility-Centered Survey*. Iranian journal of Information Processing & Management, vol. 31, no. 3, pages 823–844, 2016. (Cité en page 119.)

[Jean-Caurant 2017a] Axel Jean-Caurant, Cyrille Suire, Vincent Courboulay et Jean-Christophe Burie. *Limiter l'impact des erreurs OCR sur les représentations distribuées de mots*. In Book of Abstracts of DH, 2017. (Cité en pages 116 et 118.)

[Jean-Caurant 2017b] Axel Jean-Caurant, Nouredine Tamani, Vincent Courboulay et Jean-Christophe Burie. *Lexicographical-based Order for Post-OCR Correction of Named Entities*. In 14th International Conference on Document Analysis and Recognition ICDAR. IEEE, 2017. (Cité en pages 116 et 118.)

[Jenart 2013] Delphine Jenart. *"The Internet : A Belgian Story ?" The Mundaneum*. In Making the History of Computing Relevant : IFIP WG 9.7 International Conference, HC 2013, London, UK, June 17-18, 2013, Revised Selected Papers, volume 416, page 79. Springer, 2013. (Cité en page 11.)

[Johnson 2009] Mark Johnson. *How the Statistical Revolution Changes (Computational) Linguistics*. In Proceedings of the EACL 2009 Workshop on the Interaction Between Linguistics and Computational Linguistics : Virtuous, Vicious or Vacuous ?, ILCL '09, pages 3–11. Association for Computational Linguistics, 2009. (Cité en page 76.)

[K. Landauer 1997] Thomas K. Landauer et Susan T. Dumais. *A solution to Plato's problem : The latent semantic analysis theory of acquisition, induction, and representation of knowledge*. vol. 104, pages 211–240, 04 1997. (Cité en page 75.)

[Kissos 2016] Ido Kissos et Nachum Dershowitz. *OCR Error Correction Using Character Correction and Feature-Based Word Classification.* pages 198–203. IEEE, 2016. (Cité en page 91.)

[Kramer 1988] Samuel Noah Kramer. *The Origin and Development of the Cuneiform System of Writing.* In Thirty Nine Firsts In Recorded History, pages 381–383, 1988. (Cité en page 10.)

[Lau 2016] Jey Han Lau et Timothy Baldwin. *An empirical evaluation of doc2vec with practical insights into document embedding generation.* arXiv preprint arXiv :1607.05368, 2016. (Cité en page 85.)

[Le 2014] Quoc V. Le et Tomas Mikolov. *Distributed Representations of Sentences and Documents.* CoRR, vol. abs/1405.4053, 2014. (Cité en page 85.)

[Levenshtein 1966] Vladimir I Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals.* In Soviet physics doklady, volume 10, pages 707–710, 1966. (Cité en page 99.)

[Liu 2016] Jingjing Liu, Chang Liu et Nicholas J. Belkin. *Predicting Information Searchers' Topic Knowledge at Different Search Stages.* J. Assoc. Inf. Sci. Technol., vol. 67, no. 11, pages 2652–2666, 2016. (Cité en page 48.)

[Lovins 1968] Julie Beth Lovins. *Development of a stemming algorithm.* Mech. Translat. & Comp. Linguistics, vol. 11, no. 1-2, pages 22–31, 1968. (Cité en page 14.)

[Lund 2013] William B Lund, Douglas J Kennard et Eric K Ringger. *Combining multiple thresholding binarization values to improve OCR output.* In DRR, page 86580R, 2013. (Cité en page 93.)

[Luo 2005] Xiaoqiang Luo. *On coreference resolution performance metrics.* In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 25–32. Association for Computational Linguistics, 2005. (Cité en page 105.)

[Maidasani 2012] Hitesh Maidasani, Galileo Namata, Bert Huang et Lise Getoor. *Entity resolution evaluation measures.* 2012. (Cité en page 103.)

[Marchionini 2006] Gary Marchionini. *Exploratory search : from finding to understanding.* Communications of the ACM, vol. 49, no. 4, pages 41–46, 2006. (Cité en page 45.)

[Mei 2016] Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Moh'd et Evangelos E. Milios. *Statistical Learning for OCR Text Correction.* In arXiv preprint arXiv :1611.06950, 2016. (Cité en page 91.)

[Mikheev 1999] Andrei Mikheev, Marc Moens et Claire Grover. *Named Entity Recognition Without Gazetteers.* In Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics, pages 1–8. Association for Computational Linguistics, 1999. (Cité en page 72.)

[Mikolov 2013a] Tomas Mikolov, Kai Chen, Greg Corrado et Jeffrey Dean. *Efficient estimation of word representations in vector space.* In arXiv preprint arXiv :1301.3781, 2013. (Cité en pages 79 et 99.)

[Mikolov 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado et Jeffrey Dean. *Distributed Representations of Words and Phrases and their Compositionality.* CoRR, vol. abs/1310.4546, 2013. (Cité en page 85.)

[Mitev 1985] Nathalie N Mitev, Gillian M Venner et Stephen Walker. Designing an online public access catalogue : Okapi, a catalogue on a local area network. The British Library, 1985. (Cité en page 16.)

[Mnih 2009] Andriy Mnih et Geoffrey E Hinton. *A scalable hierarchical distributed language model.* In Advances in neural information processing systems, pages 1081–1088, 2009. (Cité en page 84.)

[Moretti 2011] F Moretti. *Network theory, plot analysis.* A Stanford Lit Lab Pamphlet, 2011. (Cité en page 74.)

[Moretti 2013] Franco Moretti. Distant reading. Verso Books, 2013. (Cité en page 73.)

[Palla 2005] Gergely Palla, Imre Derényi, Illés Farkas et Tamás Vicsek. *Uncovering the overlapping community structure of complex networks in nature and society.* arXiv preprint physics/0506133, 2005. (Cité en page 102.)

[Penalva 2002] Jean-Michel Penalva et Jackie Montmain. *Les référentiels de connaissances. Travail collaboratif et intelligence collective.* no. 48, 2002. (Cité en page 1.)

[Peng 2004] Fuchun Peng, Fangfang Feng et Andrew McCallum. *Chinese segmentation and new word detection using conditional random fields.* In Proceedings of the 20th international conference on Computational Linguistics, page 562. Association for Computational Linguistics, 2004. (Cité en page 68.)

[Pennington 2014] Jeffrey Pennington, Richard Socher et Christopher D Manning. *Glove : Global vectors for word representation.* In EMNLP, volume 14, pages 1532–1543, 2014. (Cité en page 79.)

[Pirolli 1999] Peter Pirolli et Stuart Card. *Information foraging.* Psychological review, vol. 106, no. 4, page 643, 1999. (Cité en page 45.)

[Read 2012] Jonathon Read, Rebecca Dridan, Stephan Oepen et Lars Jørgen Solberg. *Sentence boundary detection : A long solved problem ?* COLING (Posters), vol. 12, pages 985–994, 2012. (Cité en page 68.)

[Riley 2017] Jenn Riley. *Understanding Meta Data.* Primer Publication of National Information Standard Organization Baltimore, vol. 18, 2017. (Cité en page 22.)

[Robertson 1976] Stephen E Robertson et Karen Spärck Jones. *Relevance weighting of search terms.* In Journal of the American Society for Information Science, volume 27, pages 129–146, 1976. (Cité en page 16.)

[Rong 2014] Xin Rong. *word2vec Parameter Learning Explained.* CoRR, vol. abs/1411.2738, 2014. (Cité en page 82.)

[Rosenblatt 1958] Frank Rosenblatt. *The perceptron : A probabilistic model for information storage and organization in the brain.* Psychological review, vol. 65, no. 6, page 386, 1958. (Cité en page 121.)

[Rowley 2007] Jennifer Rowley. *The wisdom hierarchy : representations of the DIKW hierarchy.* Journal of Information Science, vol. 33, no. 2, pages 163–180, 2007. (Cité en page 1.)

[Rumelhart 1986] David Rumelhart, Geoffrey Hinton et Ronald Williams. *Learning representations by back-propagating errors.* Nature, pages 533–536, 1986. (Cité en page 121.)

[S. Harris 1954] Zellig S. Harris. *Distributional Structure.* vol. 10, pages 146–162, 08 1954. (Cité en page 76.)

[Salton 1973] Gerard Salton et Chung-Shu Yang. *On the specification of term values in automatic indexing.* Journal of documentation, vol. 29, no. 4, pages 351–372, 1973. (Cité en page 15.)

[Sanderson 2012] Mark Sanderson et W Bruce Croft. *The history of information retrieval research.* Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pages 1444–1451, 2012. (Cité en page 12.)

[Schmidt 2016] Benjamin M Schmidt. *Do digital humanists need to understand algorithms ?* Debates in the Digital Humanities 2016, 2016. (Cité en pages 53 et 64.)

[Silem 1997] A Silem et B Lamizet. Dictionnaire encyclopédique des sciences de l'information et de la communication. Ellipses, 1997. (Cité en page 3.)

[Solera-Ureña 2005] Rubén Solera-Ureña, Jaume Padrell, Darío Iglesias, Ascensión Gallardo-Antolín, Carmen Peláez-Moreno et Fernando Díaz-de María. *SVMs for Automatic Speech Recognition : A Survey.* pages 190–216, 2005. (Cité en page 66.)

[Suire 2016] Cyrille Suire, Axel Jean-Caurant, Vincent Courboulay, Pascal Estraillier et Jean-Christophe Burie. *User activity characterization in a cultural heritage digital library system.* In Digital Libraries (JCDL), 2016 IEEE/ACM Joint Conference on, pages 257–258. IEEE, 2016. (Cité en pages 59 et 118.)

[Suire 2017] Cyrille Suire, Axel Jean-Caurant et Charles Illouz. *Ouvrir les boîtes noires : un outil pédagogique pour une approche critique de la recherche d'information en ligne.* 2017. (Cité en pages 59 et 118.)

[Thompson 2015] Paul Thompson, John McNaught et Sophia Ananiadou. *Customised OCR correction for historical medical text.* In Digital Heritage, 2015, volume 1, pages 35–42. IEEE, 2015. (Cité en page 91.)

[Tsuchiya 1993] Shigehisa Tsuchiya. *Improving Knowledge Creation Ability through Organizational Learning.* In ISMICK 93 Proceedings, 1993. (Cité en page 1.)

[Tulving 1966] Endel Tulving et Zena Pearlstone. *Availability versus accessibility of information in memory for words.* Journal of Verbal Learning and Verbal Behavior, vol. 5, no. 4, pages 381–391, 1966. (Cité en page 2.)

[Van Rijsbergen 1980] Cornelis J Van Rijsbergen, Stephen Edward Robertson et Martin F Porter. New models in probabilistic information retrieval. British

Library Research and Development Department London, 1980. (Cité en page 14.)

[Vilain 1995] Marc Vilain, John Burger, John Aberdeen, Dennis Connolly et Lynette Hirschman. *A model-theoretic coreference scoring scheme*. In Proceedings of the 6th conference on Message understanding, pages 45–52. Association for Computational Linguistics, 1995. (Cité en page 103.)

[Wang 2015] Peilu Wang, Yao Qian, Frank K. Soong, Lei He et Hai Zhao. *Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network*. CoRR, vol. abs/1510.06168, 2015. (Cité en page 68.)

[Werbos 1974] Paul John Werbos. *Beyond regression : New tools for prediction and analysis in the behavioral sciences*. Doctoral Dissertation, Applied Mathematics, Harvard University, MA, 1974. (Cité en page 121.)

[Widrow 1960] B. Widrow. *An adaptive "ADALINE" neuron using chemical "memistors"*. In Technical Report 1553-2. 1960. (Cité en page 130.)

[Wilcoxon 1945] Frank Wilcoxon. *Individual comparisons by ranking methods*. Biometrics bulletin, vol. 1, no. 6, pages 80–83, 1945. (Cité en page 51.)

[Wilson 1981] Tom D Wilson. *On user studies and information needs*. Journal of documentation, vol. 37, no. 1, pages 3–15, 1981. (Cité en page 44.)

[Xiong 2017] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu et Geoffrey Zweig. *The Microsoft 2016 conversational speech recognition system*. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, pages 5255–5259. IEEE, 2017. (Cité en page 66.)

[Yalniz 2011] Ismet Zeki Yalniz et Raghavan Manmatha. *A fast alignment scheme for automatic ocr evaluation of books*. In Document Analysis and Recognition (ICDAR), 2011 International Conference on, pages 754–758. IEEE, 2011. (Cité en page 94.)

[Zheng 2013] Xiaoqing Zheng, Hanyang Chen et Tianyu Xu. *Deep Learning for Chinese Word Segmentation and POS Tagging*. In EMNLP, pages 647–657, 2013. (Cité en page 68.)