Université Clermont Auvergne

Ecole Doctorale de Lettres, Sciences Humaines et Sociales (ED 370)

UFR Lettres Langues et Sciences Humaines

Laboratoire de Recherche sur le Langage (EA 999)

Thèse pour l'obtention du grade de Docteur de l'Université Clermont Auvergne

Discipline : Sciences du langage

Spécialité : Traitement automatique des langues

Recherche et développement du Logiciel Intelligent de Cartographie Inversée, pour l'aide à la compréhension de texte par un public dyslexique.

ANNEXES

Thèse présentée et soutenue publiquement le 05 octobre 2017 par

Mario LAURENT

Sous la direction de Thierry CHANIER (2012/2016)

Puis de Damien CHABANAL (2016/2017)

Membres du jury :

Georges Antoniadis, Professeur, Université Grenoble Alpes (président du jury)

Maryse Bianco, Maître de Conférences HDR, Université Grenoble Alpes (rapporteur)

Jean Écalle, Professeur, Université Lumière Lyon 2 (rapporteur)

André Tricot, Professeur, Université de Toulouse Jean Jaurès

Sommaire

Introduction

Ce volume d'annexes contient des informations complémentaires au volume principal de ma thèse, intitulée "Recherche et développement du Logiciel Intelligent de Cartographie Inversée, pour l'aide à la compréhension de texte par un public dyslexique". Ces informations sont de différents types : explications rédigées, tableaux de données brutes et code en langage Python commenté. Elles sont organisées en section suivant l'ordre d'apparition des chapitres avec lesquels elles sont en relation dans le volume principal.

Annexe A. La prise en charge, la reconnaissance et la remédiation

A - 1. Prise en charge : ce que nous dit la loi

A - 1.1. Le socle commun de connaissances et compétences :

Au niveau législatif la LOI n° 2005-380 du 23 avril 2005 d'orientation et de programme pour l'avenir de l'école nous dit :

> « Art. L. 122-1-1. - La scolarité obligatoire doit au moins garantir à chaque élève les moyens nécessaires à l'acquisition d'un socle commun constitué d'un ensemble de connaissances et de compétences qu'il est indispensable de maîtriser pour accomplir avec succès sa scolarité, poursuivre sa formation, construire son avenir personnel et professionnel et réussir sa vie en société. Ce socle comprend :
>
> - la maîtrise de la langue française ;
>
> - la maîtrise des principaux éléments de mathématiques ;
>
> - une culture humaniste et scientifique permettant le libre exercice de la citoyenneté ;
>
> - la pratique d'au moins une langue vivante étrangère ;
>
> - la maîtrise des techniques usuelles de l'information et de la communication. » (Assemblée Nationale, 2005b)

Ce socle commun de connaissances et de compétences à atteindre est un obstacle important pour les élèves dyslexiques, puisqu'ils rencontrent des difficultés spécifiques à la maîtrise de la langue écrite et aux apprentissages liés à l'usage des supports écrits.

A - 1.2. La loi sur l'égalité des chances

Des moyens spécifiques sont mis à la disposition des personnes handicapées, ceux-ci sont définis par la LOI n° 2005-102 du 11 février 2005 pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées (Assemblée Nationale, 2005a).

Dans l'article L 114, cette loi définit le handicap de la manière suivante :

> « Constitue un handicap, au sens de la présente loi, toute limitation d'activité ou restriction de participation à la vie en société subie dans son environnement par une personne en raison d'une altération substantielle, durable ou définitive d'une ou plusieurs fonctions physiques, sensorielles, mentales, cognitives ou psychiques, d'un poly-handicap ou d'un trouble de santé invalidant ».

Si la dyslexie est bien reconnue comme un handicap, il est toutefois nécessaire de faire une démarche auprès de la MDPH pour le faire reconnaître et accéder aux aménagements adaptés.

En complément de l'article L. 114-1, l'article L. 114-2 précise l'égalité des droits en rapport avec la scolarité :

> « A cette fin, l'action poursuivie vise à assurer l'accès de l'enfant, de l'adolescent ou de l'adulte handicapé aux institutions ouvertes à l'ensemble de la population et son maintien dans un cadre ordinaire de scolarité, de travail et de vie. Elle garantit l'accompagnement et le soutien des familles et des proches des personnes handicapées. »

Pour compléter cette loi, trois textes importants encadrent la scolarisation des enfants handicapés, dont font partie les enfants dyslexiques.

Tout d'abord, l'arrêté du 17 août 2006, articles 1, 2, 3, 4 et 5, publiés au journal officiel du 20 août 2006 stipule que des enseignants référents sont en charge du suivi des élèves handicapés, en étant en lien avec la famille. Ils sont parties prenantes des P.P.S. (projet personnalisé de scolarisation) mis en œuvre dans leur secteur d'intervention. Leur mission est d'appliquer les décisions de la C.D.A (commission des droits et de l'autonomie). Ils veillent à la continuité et la cohérence de la mise en œuvre du P.P.S.

Ensuite, la circulaire n°2006-119 du 31 juillet 2006 insiste sur la nécessité d'un partenariat autour de la scolarisation des élèves handicapés. Partenariat entre : les inspections académiques, les directions départementales des affaires sanitaires et sociales et les MDPH. La coopération entre milieu scolaire ordinaire et secteur sanitaire ou médico-social est aussi nécessaire.

Les partenariats existants, sont :

- Le P.P.S. (Projet Personnalisé de Scolarisation), qui relève de la prise en charge du handicap (CDA, MDPH). Il s'agit d'une adaptation globale et concertée du parcours scolaire (aide d'une tierce personne, usage d'un logiciel) ;
- Le P.I.S. (Projet Individuel de Scolarisation), qui permet de cibler les besoins d'un élève présentant un trouble du langage écrit ou oral et d'adapter la pédagogie à l'enfant. Il est définit par le chef d'établissement, le médecin, l'orthophoniste, les parents et l'enfant ;

Ces deux premières mesures sont à renouveler chaque année.

- Le P.A.I. (projet d'aide individualisé), qui concerne l'aménagement du rythme scolaire et l'adaptation pédagogique ;
- Le P.P.R.E. (Programme Personnalisé de Réussite Éducative), qui permet de coordonner les actions des différents intervenants dans le but d'aider l'élève à surmonter ses difficultés. C'est une action de courte durée, permettant de mieux identifier l'action à mener pour améliorer le parcours d'un élève dyslexique.

En dernier lieu, le Décret n° 2005-1617 du 21 décembre 2005 régit les aménagements possibles lors des examens et des concours, pour l'enseignement scolaire comme pour l'enseignement supérieur.

Une circulaire d'application précise les conditions de ces aménagements (Circulaire N°2006-215 DU 26-12-2006 - BO n° 1 du 4 janvier 2007). Les aménagements possibles, dépendent du degré de sévérité reconnu du handicap, ils peuvent être :

- Obtention d'un tiers temps supplémentaire aux épreuves orales et écrites ;
- Utilisation d'un ordinateur et d'un logiciel à commande vocale ;
- Aide d'une personne, chargée de lire les énoncés à l'élève dyslexique ;
- Aide d'une personne, chargé de lire les énoncés mais aussi d'écrire sous la dictée du candidat.

A - 2. Les méthodes alternatives

J'ai présenté plusieurs méthodes alternatives dans mon ouvrage de thèse principal. J'ajoute ici des compléments concernant les notions de bases qui permettront de mieux appréhender ces méthodes. Ces compléments sont destinés à être lus plutôt en amont des explications sur les méthodes alternatives présentées dans le document principal et n'apportent aucune précision concernant la dyslexie et les interactions entre le son, la proprioception et les capacités de lecture.

Qu'est-ce qu'un son ? Quel est le rôle de l'oreille dans l'audition et dans l'équilibration. Qu'est-ce que la proprioception et comment s'orienter ?

A - 2.1. Méthode Davis

L'ensemble des explications suivantes sur le son et le fonctionnement de l'oreille ont été recueillies dans les documents ci-après : Dubus (2003), Marieb et Hoehn (2010), Netter (2009) et Coulon (2002).

Annexe A

Le son est porté par une onde sonore. Cette onde est qualifiée par plusieurs variables sur différentes dimensions :

- Une fréquence qui varie de 20 hz (pour les infra sons ou sons graves) jusqu'à 20000 hz (pour les ultra-sons ou son aigus). Une bonne acuité auditive permet de percevoir les fréquences allant de 1000 à 4000 hz. (le nombre de hertz correspondant au nombre d'événements sonores par seconde) ;

- Une amplitude, qui dépend de la puissance du son (une conversation usuelle entre deux personnes produit environ 50 décibels). Le décibel est une unité logarithmique et est exprimé en quantité d'énergie par unité de temps ;

- Le timbre, qui correspond à la capacité de l'oreille à discerner les uns des autres des objets, ou des instruments, émettant une même note. L'oreille humaine est capable de discriminer environ 400 000 timbres différents.

La voix humaine se qualifie également par sa tessiture, son articulation, son intonation, son accent et sa prosodie.

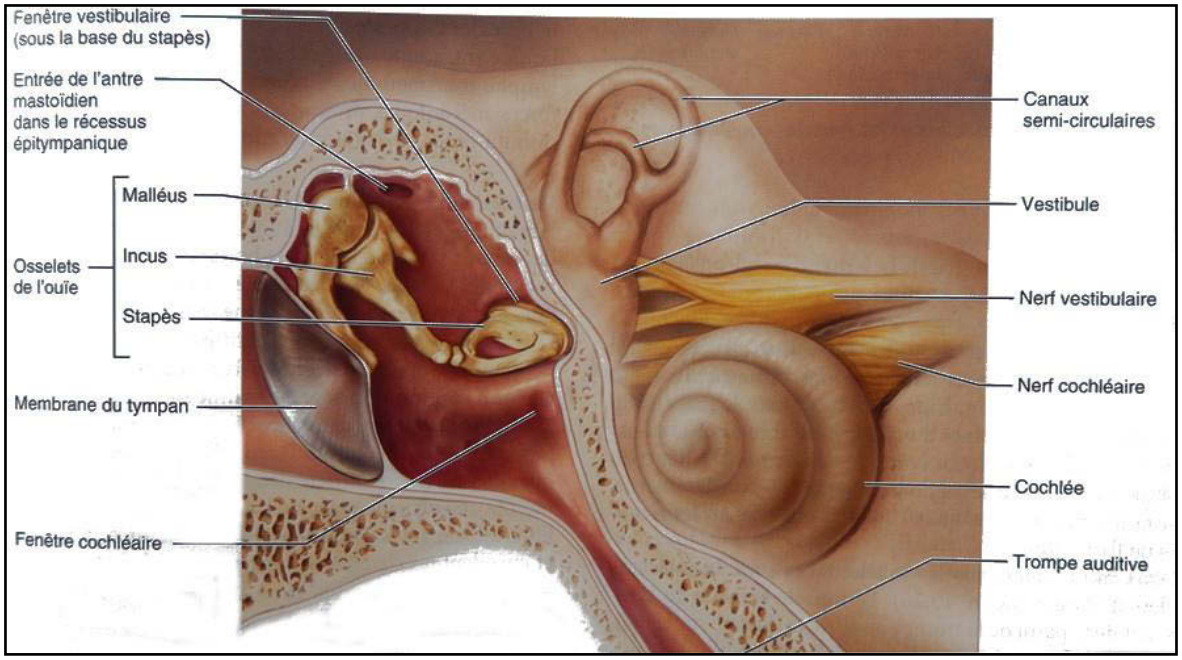


Figure A.1 Structure de l'oreille, extrait de Marieb et Hoehn (2010)

L'oreille est un organe neurosensoriel, elle est composée de trois grandes parties qui ont un rôle dans l'audition, l'équilibration et l'orientation :

- L'oreille externe, qui capte les sons grâce au pavillon, et les oriente vers l'intérieur du conduit auditif jusqu'à la membrane du tympan qui se met alors à vibrer ;

- L’oreille moyenne, qui transmet les vibrations tympaniques aux cavités de l'oreille interne, via la chaîne des osselets, depuis le marteau jusqu'à l'étrier qui est en contact avec la fenêtre cochléaire ;

- L’oreille interne, qui joue deux rôles : elle permet l'audition et elle permet l'équilibre et l'orientation.

L'équilibre est assuré plus précisément par les canaux semi-circulaires, qui sont situés à l'arrière du vestibule. A l'intérieur de ces canaux se trouvent les otolithes. Ceux-ci sont contenus dans la membrane gélatineuse qui recouvre les cellules ciliées maculaires. Les otolithes exercent une pression sur les cellules ciliées maculaires qui change en fonction du mouvement de la tête. Ceci permet de reconnaître par exemple un changement de position de la tête ou encore une accélération en ligne droite sans changement de position. En effet, les modifications de pression subies par les cellules des macules sont détectées puis transmises aux fibres nerveuses.

A - 2.2. La proprioception

Les éléments de compréhension sur le proprioception exposés ici sont issue des documents suivants : Touzalin-Chretien (2009) et Berthoz (2013).

Le sens proprioceptif est un complément aux sens extéroceptifs que sont les cinq sens communs, qui nous renseignent sur l'environnement extérieur. Le sens proprioceptif est également complémentaire aux récepteurs nociceptifs (douleur) et intéroceptif (état des organes).

La proprioception est un ensemble d'information sur notre corps, elle permet de nous renseigner sur :

- la place occupée par le corps dans l'espace ;

- le niveau de tension interne du corps ;

- l'état de mobilité du corps : en déplacement ou immobile.

Ces informations sont acquises d'une part de l'oreille interne et, d'autre part, par des capteurs situés dans les articulations et les fibres musculaires. Le traitement de ces informations est réalisé par le cerveau de façon globale. En effet, le cerveau intègre toutes les sources d'information simultanément (somesthésie). Parmi ces sources sont incluses les informations visuelles. Le traitement de ces dernières renforce et valide les informations proprioceptives captées. Ainsi, le cerveau est capable de déterminer l'état du corps, la posture et l'équilibre, mais aussi d'établir une planification des mouvements pouvant être réalisés, en fonction de la situation, à chaque instant.

Annexe A

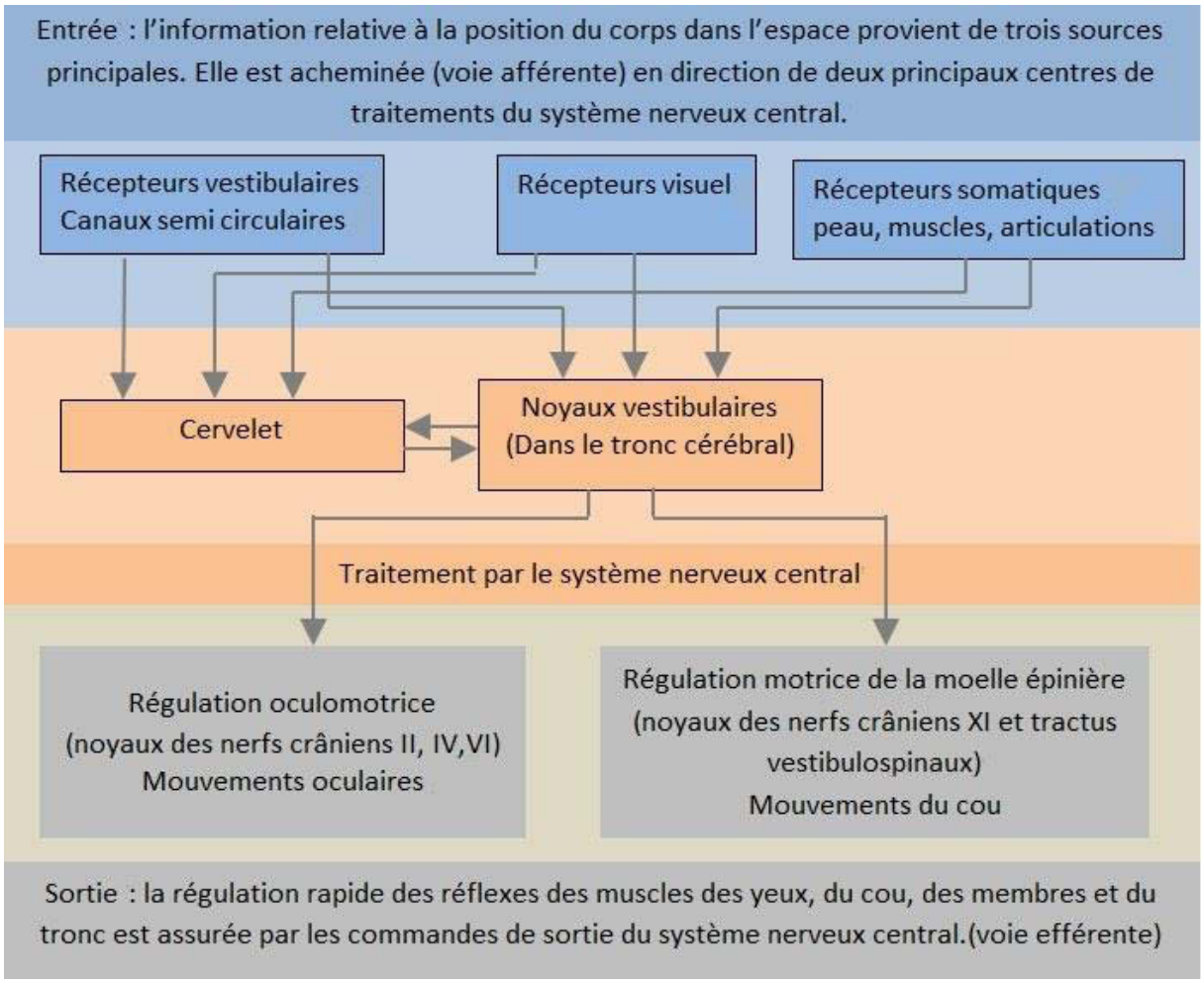


Figure A.2 Schéma explicatif sur le fonctionnement du système proprioceptif, inspiré d'un tableau du livre de Marieb et Hoehn (2010).

Porter son attention sur un objet, comme un mot, nécessite la mise en action d' « un processus multi-sensoriel complexe qui suppose la cohérence de la proprioception de l'ensemble du corps » (Roll, Kavounoudias, Escoffier & Roll, 2003, p. 62)

Annexe B. Expérimentation

B - 1. Questions

Voici les questions écartées du questionnaire pour le texte 1 de l'expérimentation (elles ont été renumérotées pour rester en cohérence avec celles du document principal) :

(7) La vie du paysan au Xème et XIème siècle :

A C'est une vie de soldat dans l'armée du seigneur.

B C'est une vie facile, le paysan mange à sa faim et dispose de temps pour ses propres affaires.

C Le paysan est soumis aux prélèvements sur les récoltes et aux corvées.

D Le paysan se déplace de seigneurie en seigneurie.

(8) Qu'est-ce qu'une seigneurie ?

A Le territoire sur lequel s'exerce le pouvoir du seigneur.

B Un ensemble de terres, ainsi que des droits et des redevances qui leurs sont attachés.

C Le château fort d'un seigneur.

D Une bataille menée par un seigneur.

(9) Les paysans ont amélioré les pratiques agricoles après que certaines communautés paysannes se sont soulevées.

A Vrai.

B Faux.

C Je ne peux pas répondre.

De la même manière, voici les questions écartées du questionnaire pour le texte 2 de l'expérimentation :

(7) Qu'est ce qui explique l'augmentation de la population mondiale au XX siècle ?

A Les progrès sanitaires.

B Le développement de l'agriculture.

C L'exode rural important.

D Les besoins grandissants.

(8) Les progrès sanitaires provoquent l'exode rural.

A Vrai.

B Faux.

C Je ne peux pas répondre.

(9) La croissance de la population mondiale continue mais l'indice de fécondité diminue.

A Vrai.

B Faux.

C Je ne peux pas répondre.

(10) Dans les pays développés la population est vieillissante et urbaine.

A Vrai.

B Faux.

C Je ne peux pas répondre.

B - 2. Appel à participation à l'expérimentation

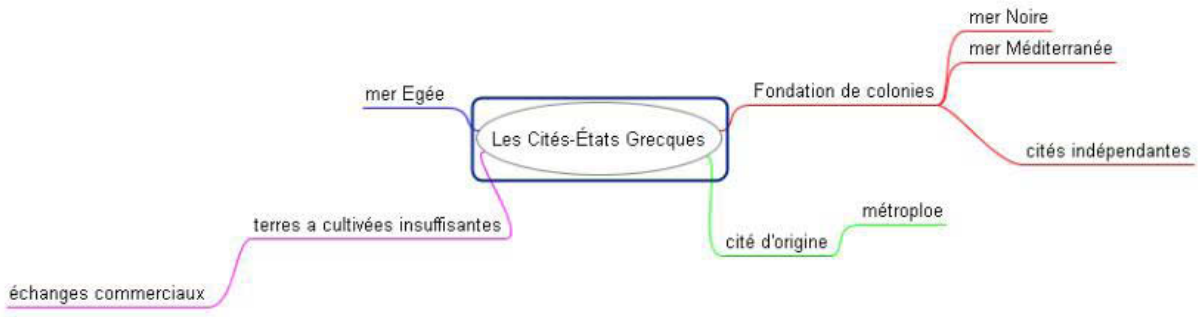
Voici le document rédigé pour faire la publicité de l'expérimentation LICI auprès des parents d'enfants dys, des orthophonistes et des autres personnes susceptibles d'être intéressées par la mise en place de celle-ci :

Appel à participation au projet LICI

1) Le Projet :

Le projet LICI est mené par Mario Laurent, dans le cadre de sa thèse en Sciences du Langage. Mario Laurent est doctorant au Laboratoire de Recherche sur le Langage (LRL) à Clermont-Ferrand, sous la direction de Thierry Chanier, professeur et directeur du LRL. Ce projet est réalisé au sein de Médialexie, une entreprise innovante spécialisée dans l'aide aux personnes atteintes de troubles de la communication. Médialexie développe des logiciels d'aide dont l'objectif est de s'ajouter (et non de remplacer) aux soutiens déjà existants pour les enfants dyslexiques : les aménagements scolaires et l'aide des professionnels de santé, comme les orthophonistes. Ces logiciels vont permettre de compenser leurs difficultés de lecture et d'écriture et ainsi favoriser leur réussite scolaire.

Les recherches de Mario Laurent visent à mieux expliquer les difficultés des enfants dyslexiques et à proposer de nouvelles solutions pour les aider. Au centre de ce travail se trouve la réalisation du LICI (Logiciel Intelligent de Cartographie Inversée). Le LICI a pour but d'aider les collégiens sujets à des troubles du langage dans leurs tâches scolaires. Pour cela, il va permettre de générer automatiquement une représentation visuelle d'un texte sous forme schématique. Cette représentation visuelle contient les éléments d'information principaux du texte et les met en valeur. Elle va donc venir en complément du texte pour aider à sa compréhension.



Exemple de carte heuristique simple réalisée à partir d'une leçon d'Histoire de collège

2) L'expérimentation :

Une expérimentation est menée dans le cadre du projet. Elle se présente sous la forme d'un questionnaire en ligne, à l'intention de collégiens. Il s'agit de lire deux textes courts, de manuels scolaires, et de répondre pour chacun à un QCM de six questions ; le tout dure environ 30 minutes. L'un des deux textes sera accompagné d'une carte, comme celle présentée à la page précédente. Nous émettons l'hypothèse que cette carte va aider les élèves à retrouver plus vite les réponses aux questions. Nous avons besoin de réponses de

collégiens dyslexiques mais également de normo-lecteurs, qui feront partie du groupe contrôle.

Le questionnaire est accessible directement via un lien internet. La participation est entièrement anonyme, il est seulement demandé un prénom et des informations génériques comme l'âge, utiles pour l'analyse des résultats.

L'objectif est de mesurer l'impact de la visualisation de la carte sur les réponses des élèves. Le groupe de normo-lecteurs est également important car il permettra de vérifier si les dyslexiques ont effectivement davantage de difficultés de compréhension mais surtout de voir si l'apport de la carte est significativement différent entre les deux groupes.

Le contact pour participer et recevoir le lien du questionnaire : lici.recherche@gmail.com

Merci de votre participation !

Mario LAURENT

Université Blaise Pascal - LRL
Médialexie
6 rue Nicolas Joseph Cugnot
63100 Clermont-Ferrand
lici.recherche@gmail.com

Thierry CHANIER

Université Blaise Pascal
Maison des Sciences de l'Homme – LRL
4, rue Ledru
63057 Clermont-Ferrand cedex 01
Tél : +33 3 4 73 34 68 39
thierry.chanier@univ-bpclermont.fr

Le projet LICI bénéficie du soutien de l'ANRT via une bourse CIFRE.

B - 3. L'environnement de l'expérimentation

Avant de lancer l'expérimentation, j'ai implémenté celle-ci dans deux environnements différents : sur la plateforme d'apprentissage Moodle et sur l'environnement de création de

questionnaires LimeSurvey. L'implémentation sur Moodle a été plus difficile et m'a obligé à utiliser un format intermédiaire, le format SCORM. Je vais revenir ici sur l'origine et les caractéristiques de ce format.

B - 3.1. SCORM

Les spécifications de SCORM sont établies par l'ADL (Advanced Distributed Learning) qui a été créée en 1997 par le Département de la Défense américain (Bruet, 2013). SCORM signifie Sharable Content Object Reference Model. SCORM est considéré comme une norme, mais il s'agit en réalité d'un modèle utile pour créer des contenus pédagogiques pour la formation à distance par internet. Il existe d'autres standards du même type, comme IMS ou AICC.

Les objectifs de SCORM sont, d'une part de répertorier et classer les objets d'apprentissage et d'autre part de rendre ces objets utilisables et modifiables de façon générique (Bruet, 2013). La version actuelle est la version SCORM 2004, datant de 2006. Le modèle SCORM emploie le langage XML et définit des unités d'apprentissages appelées SCO. Les SCO sont des éléments autonomes, comme par exemple des textes, des vidéos, des sons, des pages HTML ou encore des QCM. Des métadonnées leurs sont associées et elles contiennent des informations générales (quoi ?), des informations techniques (comment ?) et des informations pédagogiques (pourquoi ?). Le modèle SCORM permet de constituer des cours interactifs complets, ils sont structurés selon une méthode bien définie. Le fichier principal est un fichier compressé, qui contient un fichier "imsmanifest.xml" contenant toutes les métadonnées générales, accompagné de plusieurs modules, contenant chacun des SCO (Claroline, 2008). Chaque SCO est compilé et accompagné d'un fichier XML appelé "manifest". Ce fichier contient des métadonnées (titre, description du cours, ...), la séquence des SCO qui constitue le cours, les noms de fichier qui permettent de lancer les SCO en question et enfin, la liste de tous les fichiers ressources du cours (Claroline, 2008). Il existe également une version SCORM Lite, qui n'impose pas la création d'un fichier "manifest" et dont le fichier principal contient toutes les ressources nécessaires au cours, sans appel à des modules extérieurs.

L'intérêt du modèle SCORM est de pouvoir créer des contenus génériques et réutilisables. Ces contenus peuvent être intégrés dans la pluparts des LMS (Learning Management System). Ainsi, le recours à ce format assure la pérennité des modules développés, contrairement à ceux qui seraient développés sur un LMS particulier. Le modèle assure aussi l'interopérabilité, c'est à dire la communication entre contenus et plateforme (Deruy & Ferrandino, 2011).

L'usage du modèle SCORM a aussi plusieurs inconvénients. Il existe différentes méthodes pour générer du contenu au format SCORM, certaines sont efficaces mais payantes et dans tous les cas beaucoup d'étapes sont nécessaires entre l'idée originale et le rendu final sur le LMS. Le logiciel Dreamweaver permettait de générer des cours au format SCORM, mais cette

fonctionnalité n'est plus mise à jour. Il est toujours possible de créer des activités au bon format avec HotPotatoes, qui permet de visualiser assez directement ce que l'on fait et qui inclut une option d'export automatique.

Enfin, lorsque le module SCORM est prêt, il y a deux possibilités pour l'intégrer dans *Moodle* (Deruy & Ferrandino, 2011) :

- l'ajouter en tant que ressource, lorsque l'on veut seulement utiliser des SCO au sein de l'environnement *Moodle*.
- l'ajouter en tant qu'activité (voir Figure B.1), lorsque l'on souhaite importer l'ensemble du module SCORM. C'est cette méthode qui permet de récupérer les scores à un test, le temps passé sur les activités et autres traces de navigation.

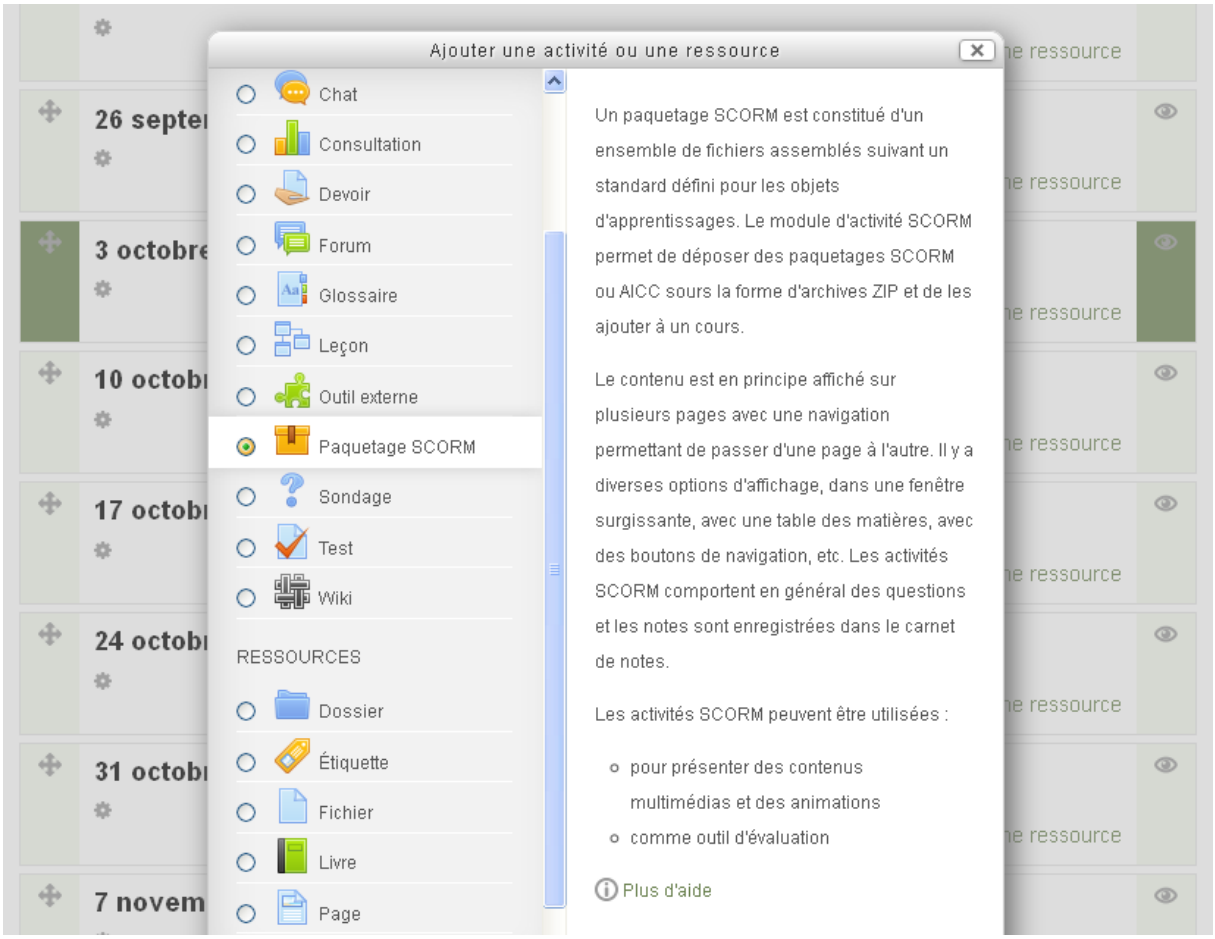


Figure B.1 Intégration d'un module SCORM en tant qu'activité sur la plateforme Moodle.

Finalement, pour tester l'implémentation de mon expérimentation sur *Moodle*, j'ai donc utilisé l'outil *JQuiz*, inclut dans le logiciel *Hot Potatoes* (Half-Baked Software Inc., 2013), afin de créer un questionnaire. J'ai ainsi pu exporter ce questionnaire au format SCORM pour pouvoir ensuite l'importer en tant qu'activité dans ma plate-forme *Moodle*.

B - 4. Résultats

B - 4.1. Tableau de données brutes

L'annexe la plus importante concernant l'expérimentation est le tableau réunissant l'ensemble des données brutes. C'est à partir de celles-ci que j'ai effectué toutes mes analyses et elles doivent être mises à disposition afin de permettre la vérification de mes calculs, ou encore la possibilité pour d'autres chercheurs d'effectuer d'autres analyses.

Le tableau des données de l'expérimentation ne sera cependant disponible qu'en version électronique, sous la forme d'un fichier *excel*.

B - 4.2. Histogrammes

Je propose dans cette section quelques histogrammes complémentaires, réalisés lors de l'analyse des résultats, mais qui n'ont pas été utilisés dans le déroulement de la discussion de mon document principal de thèse.

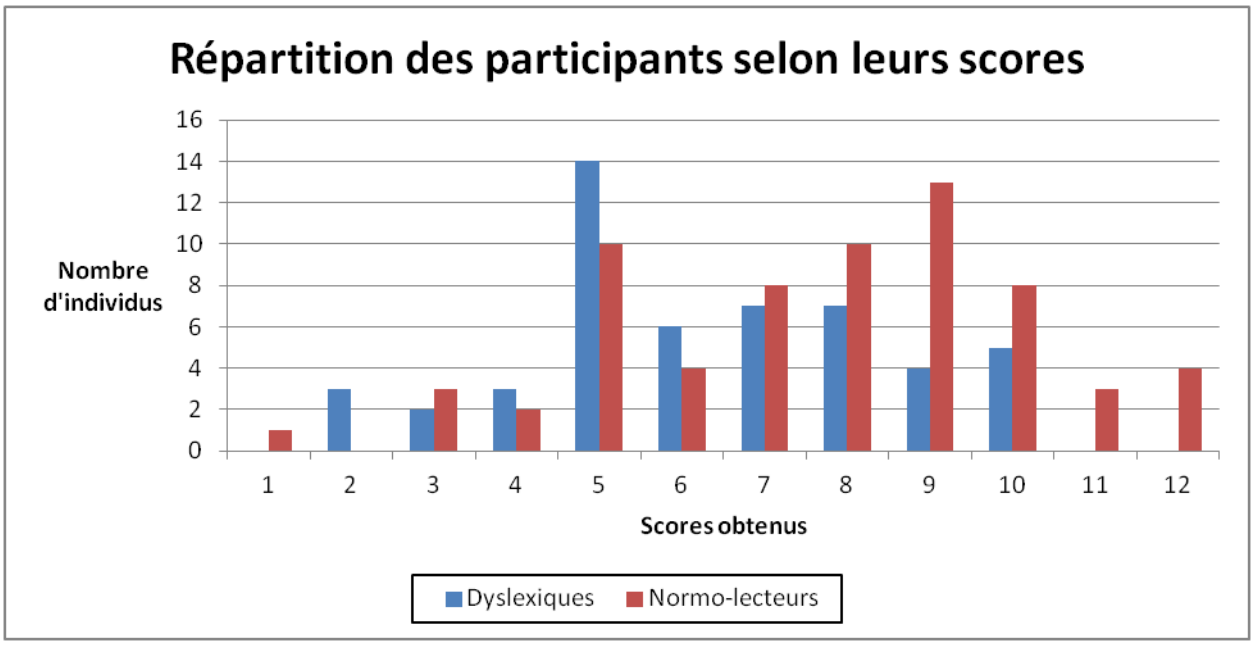


Figure B.2 Histogramme présentant le nombre de participants par classe de score obtenu.

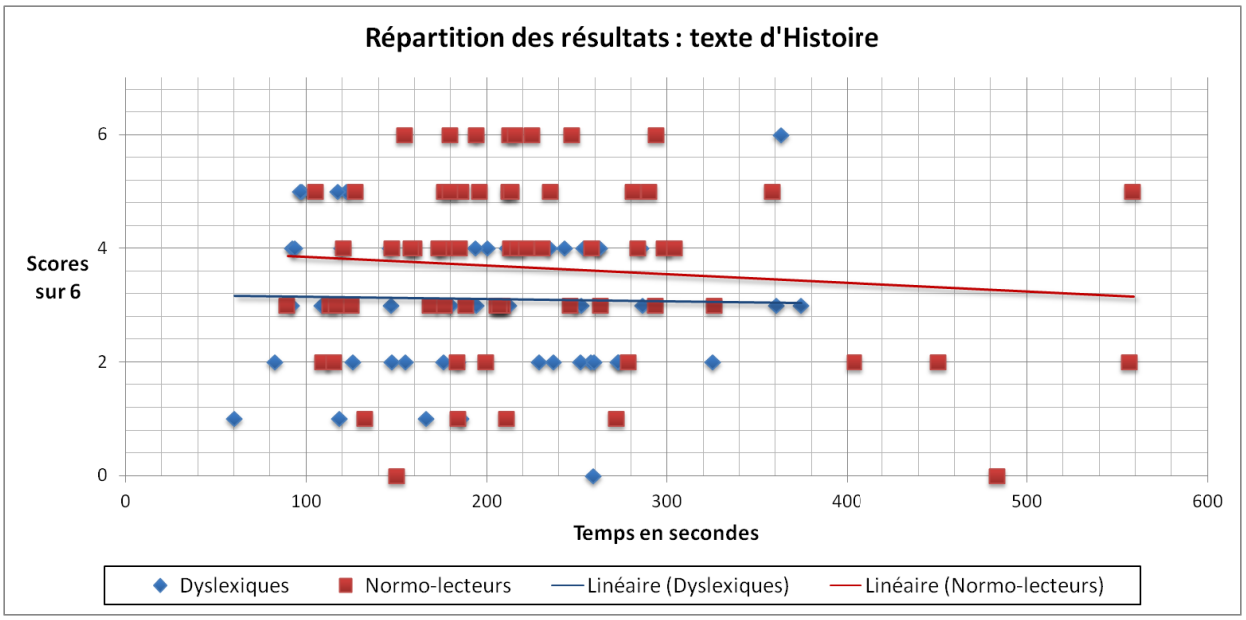


Figure B.3 Répartition des participants en fonction de leur score et de leurs temps pour la lecture et la réponse aux questions du texte d'Histoire.

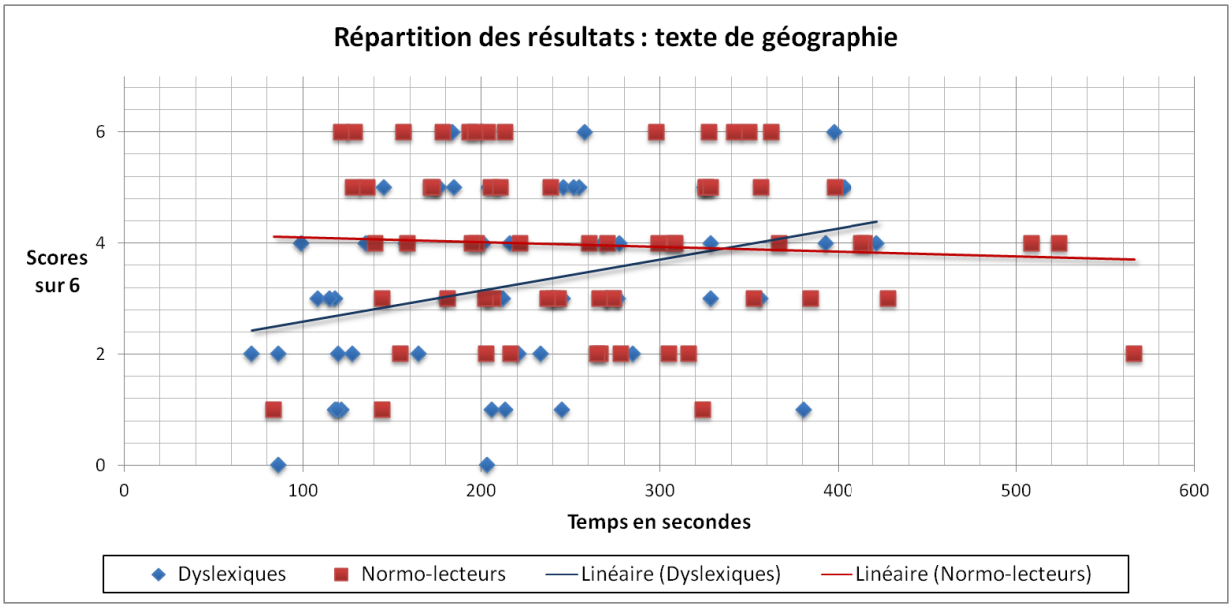


Figure B.4 Répartition des participants en fonction de leur score et de leurs temps pour la lecture et la réponse aux questions du texte de géographie.

Annexe C. Le LICI

C - 1. Exemple d'utilisation du LICI

Le texte sur la prise de la Bastille, extrait de Wikipédia et présenté dans le chapitre 10 du document principal est le suivant :

"La prise de la Bastille, survenue le mardi 14 juillet 1789 à Paris, est l'un des événements inauguraux et emblématiques de la Révolution française.

Cette journée, durant laquelle la Bastille est prise d'assaut par des émeutiers, est, dans la tradition historiographique, considérée comme la première intervention d'ampleur du peuple parisien dans le cours de la Révolution et dans la vie politique française.

Le siège et la reddition de la forteresse royale s'inscrivent dans une période de vide gouvernemental, de crise économique et de tensions politiques à la faveur de la réunion des États Généraux et de leur proclamation par le Tiers en Assemblée constituante. L'agitation du peuple parisien est à son comble suite au renvoi de Necker (11 juillet) et du fait de la présence de troupes mercenaires aux abords de la ville.

Si son importance est relative sur le plan militaire, l'événement est sans précédent par ses répercussions, par ses implications politiques et son retentissement symbolique.

La forteresse était défendue par une centaine d'hommes qui firent près de cent morts parmi les assiégeants. Il y en eut six parmi les assiégés, dont le gouverneur de Launay. Cependant, la reddition de la Bastille fit l'effet d'un séisme, en France comme en Europe, jusqu'en Russie impériale.

D'emblée, l'événement est considéré comme un tournant radical dans le cours des événements par les Parisiens et le pouvoir royal. Il marque l'effondrement de l'administration royale et provoque une révolution municipale. La capitale puis le pays se mobilise derrière les Constituants. De plus, il est immédiatement mis en scène et célébré par ses partisans. Il revêt par la suite une charge symbolique extrêmement forte dans la culture politique républicaine.

La Fête de la Fédération fut organisée à la même date l'année suivante, pour coïncider avec le premier anniversaire de l'évènement. La date du 14 juillet fut choisie en 1880 pour célébrer la fête nationale française en souvenir de cette double commémoration.

Un faisceau de causes

« C'était, dit un observateur, c'était un temps orageux, lourd, sombre, comme un songe agité et pénible, plein d'illusions, de trouble. Fausses alarmes, fausses nouvelles ; fables, inventions de toutes sortes. »

— Jules Michelet, Histoire de la Révolution française

La Forteresse de la Bastille.

Le 14 juillet 1789, la forteresse ne comptait que sept prisonniers : quatre faussaires, dont le procès était en cours d'instruction ; deux fous, Auguste Tavernier et de Francis Xavier Whyte dit chevalier de Whyte de Malleville ; un noble, criminel, enfermé à la demande de sa famille, le comte de Solages. Les autres prisonniers, comme le marquis de Sade ou Anne-Gédéon Lafitte de Pelleport avaient été transférés ou libérés peu avant. « Quasi vide sans doute, mais surchargée : surchargée de la longue histoire entretenue entre la monarchie et sa justice ». Ainsi, une légende noire enveloppait la forteresse et en faisait le symbole du despotisme ministériel ou de l'arbitraire royal. Cette image, dont témoigne les Cahiers parisiens explique pour une part l'« émotion populaire » de cette journée du 14 juillet.

La légende noire d'une prison d'État

Frontispice des Mémoires sur la Bastille de Linguet, 1783.

Le premier témoignage écrit sur la prison est le livre des pseudo-mémoires d'un calviniste français, Constantin de Renneville, qui livre une histoire fantasmée de la Bastille. Il en fait un bastion terrifiant du pouvoir arbitraire et inquisitorial et l'oppose à la Tour de Londres. Les récits « antibastillonnaires » se multiplient dans les années 1770 - 1780. Deux ouvrages publiés à l'étranger participent grandement à associer la Bastille et l'arbitraire : Mirabeau avec Des lettres de cachet et des prisons d'État (Hambourg, 1782) et Simon-Nicolas-Henri Linguet, Mémoires sur la Bastille (Londres, 1783). Ce dernier ouvrage se fait fort de distinguer le « bon roi » justicier de ses mauvais et tyranniques ministres. Louis-Sébastien Mercier qui en donne une peinture effroyable dans son Tableau de Paris, prophétise sa chute dans l'An 2440, rêve s'il en fut jamais.

L'imagerie pré et post-révolutionnaire, notamment par les gravures, a largement contribué à entretenir le mythe d'une Bastille abritant des cachots où pourrissaient les victimes de la monarchie. Cependant, la Bastille avait perdu pour partie sa fonction de prison d'État. La plupart de ses détenus n'étaient pas des victimes de l'« arbitraire » royal, les lettres de cachet ne concernant des affaires d'État que pour une très faible minorité du corpus, de l'ordre de 4 ou 5 %. La majorité de ses résidents était constituée de grands personnages et des jeunes gens incarcérés pour dissipation ou libertinage, le plus souvent à la demande de leur propre famille ou de leur entourage. Elle était enfin la forteresse qui rappelait dans le paysage parisien, l'usage que pouvait en faire le pouvoir en période de troubles, en particulier au cœur du populaire faubourg Saint-Antoine.

L'attitude de l'administration royale et du gouvernement

La Bastille où le baron de Besenval avait fait entreposer la poudre de l'arsenal, était connue pour sa faiblesse stratégique. Son gouverneur était désavoué par ses supérieurs et Besenval lui-même affirme avoir cherché à lui donner un remplaçant au début du mois de juillet. La situation exige des moyens humains et militaires supplémentaire. Broglie demande à son lieutenant de renforcer les effectifs de trente gardes suisses et envoie des canonniers pour examiner « si les pièces sont en état, et les servir, si cela devenait nécessaire, ce qui serait bien malheureux, mais est heureusement dénué de toute vraisemblance. »

La mobilisation populaire

Le peuple de Paris était inquiet depuis plusieurs jours, craignant que les troupes étrangères massées autour de la capitale depuis juin ne finissent par être utilisées contre les États-généraux ou pour servir un hypothétique massacre de la population des « patriotes ». Les échos et la publicité des débats de l'Assemblée ont autant compté dans la mobilisation populaire que « la colère et des peurs cumulés dans les différentes strates de la population parisienne », peur d'un « complot aristocratique », peur de la disette alimentée par les fantasmes d'un « pacte de famine ». Au 14 juillet, le prix du pain atteint son maximum depuis le règne de Louis XIV. La question frumentaire est au cœur de l'insurrection. Le portrait des émeutiers confirme ces préoccupations de subsistance. « Gens de métiers », artisans, commis de boutiques, les cortèges sont composés de salariés des faubourgs, père de famille, pour les deux tiers alphabétisés.

Les prémices du siège de la Bastille

Article détaillé : Émeutes des 12 et 13 juillet 1789.

Dès le 12 juillet, une milice bourgeoise de 48 000 hommes sans armes avait été constituée par l'Assemblée des électeurs de Paris, souhaitant éviter et maîtriser les débordements populaires et soutenir et défendre l'action de l'Assemblée nationale.

Le matin du dimanche 12 juillet 1789, les Parisiens sont informés du renvoi de Necker, la nouvelle se répand dans Paris. À midi, au Palais-Royal, un avocat et journaliste alors peu connu, Camille Desmoulins, monte sur une table du café de Foy et harangue la foule des promeneurs et l'appelle à prendre les armes contre le gouvernement du roi». Dans les rues de Paris et dans le jardin du Palais-Royal de nombreuses manifestations ont lieu, les bustes de Jacques Necker et de Philippe d'Orléans sont portés en tête des cortèges. Le régiment de cavalerie, le Royal-allemand charge la foule amassée aux Tuileries. On compte plusieurs blessés dont des femmes et enfants, et trois tués parmi les émeutiers. En début de soirée, Pierre-Victor de Besenval à la tête des troupes installées à Paris, donne l'ordre aux régiments suisses cantonnés au Champ-de-Mars d'intervenir.

À une heure du matin, quarante des cinquante barrières (postes d'octroi) qui permettent l'entrée dans Paris sont incendiées. La foule des émeutiers exige la baisse du prix des grains et du pain. Une rumeur circule dans Paris : au couvent Saint-Lazare et à la

Bastille seraient entreposés les grains ; le couvent est pillé à six heures. Deux heures plus tard, une réunion des « électeurs » de la capitale se tient à l'Hôtel de ville (ceux qui, au deuxième degré, ont élu les députés des États généraux). À leur tête se trouve le prévôt des marchands de Paris, Jacques de Flesselles. Au milieu d'une foule déchaînée, ils décident de former un « comité permanent » (appelé « municipalité insurrectionnelle », elle se substitue à la vieille municipalité royale) et prennent la décision de créer une « milice bourgeoise » de 48 000 hommes, afin de limiter les désordres. Chaque homme portera comme marque distinctive une cocarde aux couleurs de Paris, rouge et bleu. Pour armer cette milice, les émeutiers mettent à sac le Garde-Meuble où sont entreposées des armes de collections anciennes. Sur ordre de Jacques de Flesselles, on lance la fabrication de 50 000 piques. La foule obéissant aux ordres qui semblaient provenir du Palais-Royal[Informations douteuses], parlait de prendre la Bastille. À 17 heures, une délégation des électeurs parisiens se rend aux Invalides pour réclamer les armes de guerre qui y sont entreposées. Le gouverneur refuse. La Cour ne réagit pas. Les électeurs n'obtiennent pas les armes.

Le 14 juillet, 10 h, les émeutiers s'emparent des fusils entreposés aux Invalides. Devant le refus de son gouverneur, une foule composite — près de 80 000 personnes dont un millier de combattants — se présente pour s'en emparer de force. Pour défendre l'Hôtel des Invalides il existe des canons servis par des invalides mais ceux-ci ne paraissent pas disposés à ouvrir le feu sur les Parisiens. À quelques centaines de mètres de là, plusieurs régiments de cavalerie d'infanterie et d'artillerie campent sur l'esplanade du Champ-de-Mars, sous le commandement de Pierre-Victor de Besenval. Celui-ci réunit les chefs des corps pour savoir si leurs soldats marcheraient sur les émeutiers. Informé de leur refus, Besenval décide d'abandonner sa position et de mettre ses troupes en route vers Saint-Cloud et Sèvres. La foule escalade les fossés, défonce les grilles, descend dans les caves et s'empare des 30 000 à 40 000 fusils à poudre noire qui y sont stockés ainsi que vingt pièces de bouches à feu et d'un mortier. Les Parisiens sont désormais armés. Il ne leur manque que de la poudre à canon et des balles. Le bruit court qu'il y en a au château de la Bastille. Besenval avait en effet donné l'ordre de stocker la poudre dans la forteresse.

Le siège de la Bastille

La Prise de la Bastille.

À 10 h 30, une délégation de l'Assemblée des électeurs de Paris se rend à la Bastille. Les membres du Comité permanent n'envisagent pas de prendre le bâtiment par la force mais souhaitent ouvrir la voie des négociations. Pressés par la foule des émeutiers, notamment ceux du faubourg populaire de Saint-Antoine où l'affaire Réveillon a été un épisode marquant de la pré-révolution, les électeurs envoient une délégation au gouverneur de la Bastille, Bernard-René Jordan de Launay. Ce dernier a pris soin de la mettre en défense en calfeutrant des fenêtres, surélevant des murs d'enceinte et en plaçant des canons sur les tours et derrière le pont-levis. La délégation a pour mission de demander le retrait des canons et la distribution de la poudre et des balles aux Parisiens qui forment la « milice bourgeoise ». En effet, au-dessus du portail

monumental de la Bastille construit en 1643, se trouve un arsenal, magasin d'armes et de poudre. Par mesure de sécurité, de Launay les fait déplacer la nuit précédente vers une cour intérieure. Cette délégation est reçue avec amabilité, elle est même invitée à déjeuner, mais repart sans avoir eu gain de cause. Cependant, la foule s'impatiente et certains imaginent que la délégation est retenue prisonnière. Ce quiproquo aggrave les tensions.

À 11 h 30, une deuxième délégation mené à l'initiative de Jacques Alexis Thuriot, accompagné de Louis Ethis de Corny, procureur de la ville, se rend au fort de la Bastille. Thuriot qui souhaite éviter un affrontement, presse les Invalides pour passer la seconde enceinte, inspecte les lieux et demande des garanties. Le gouverneur s'engage à ne pas prendre l'initiative des tirs. La foule des émeutiers armée des fusils pris aux Invalides se rassemble devant la Bastille. Elle amène avec elle cinq des canons pris la veille aux Invalides et au Garde-Meubles, dont deux canons damasquinés d'argent offerts un siècle auparavant par le roi de Siam à Louis XIV. Ils sont servis par des militaires ralliés à la foule et tirent sur les portes de la forteresse.

Une explosion, prise à tort par les émeutiers comme une canonnade ordonnée par le gouverneur, déclenche les premiers assauts. Des émeutiers pénètrent dans l'enceinte par le toit du corps de garde et attaquent à coups de hache les chaînes du pont-levis19.

À 13 h 30, les quatre-vingt-deux invalides défenseurs de la Bastille et trente-deux gardes suisses détachés du régiment de Salis-Samade ouvrent le feu sur les émeutiers qui continuent leurs assauts sur la forteresse, faisant une centaine de tués21. Durant trois heures et demie, la Bastille est alors soumise à un siège régulier.

Une troisième délégation se rend à la Bastille à 14 h, dans laquelle se trouve l'abbé Claude Fauchet, suivi à 15 h d'une quatrième avec de nouveau Louis Ethis de Corny, accompagné de Louis-Lézin de Milly, son secrétaire, du comte Piquod de Sainte-Honorine, de Poupart de Beaubourg, Boucheron, Fleurie, Jouannon et Six. Cette dernière délégation, voulue dans les formes par le comité permanent de l'Hôtel de Ville, affublée d'un tambour et d'un drapeau pour afficher son caractère officiel, se présente devant le marquis de Launay mais n'obtient toujours rien. Pire, les parlementaires reçoivent une décharge de mousqueterie qui toucha la foule. Les soldats de la garnison de la Bastille et les assiégeants échangent des tirs. Dans la confusion, même cette dernière délégation fut prise à partie par la foule des assiégeants. Les négociations sont dès lors closes, et c'est par la force que l'on compte prendre la forteresse.

La Prise de la Bastille, Charles Thévenin, 1793, Musée Carnavalet.

À 15 h 30, un détachement de soixante-et-un garde-françaises sous le commandement de Pierre-Augustin Hulin et Jacob Job Elie, anciens sergents aux Gardes-Suisses se présentent devant la Bastille avec cinq canons. Ces canons sont mis en batterie contre les portes et le pont-levis du château. L'attaque proprement dite de la forteresse débute.

La reddition de la Bastille

À 17 h, la garnison de la Bastille rend les armes, sur promesse des assiégeants qu'aucune exécution n'aura lieu s'il y a reddition. De Launay n'a que peu de confiance sur les forces de ses garnisons et trop peu de vivres pour tenir un siège. Les émeutiers, parmi lesquels on dénombre une centaine de tués et soixante-treize blessés envahissent la forteresse, s'emparent de la poudre et des balles, puis libèrent les sept captifs qui y étaient emprisonnés. La garnison de la Bastille, prisonnière, est conduite à l'Hôtel de Ville pour être jugée. Sur le chemin, de Launay est massacré. Sa tête est découpée au canif par un garçon cuisinier nommé Desnot, avant d'être promenée au bout d'une pique dans les rues de la capitale. Les têtes de de Launay et de Jacques de Flesselles, prévôt des marchands de Paris sont promenées au bout d'une pique dans les rues de la capitale jusqu'au Palais-Royal. Plusieurs des invalides trouvent aussi la mort pendant le trajet. De Flesselles est assassiné sur l'accusation de traîtrise.

Outre les prisonniers, la forteresse héberge les archives du lieutenant de police de Paris qui sont soumises à un pillage systématique. Ce n'est qu'au bout de deux jours que les mesures sont prises par les autorités afin de conserver ces traces de l'histoire. Même Beaumarchais, dont la maison est située juste en face, n'avait pas hésité à puiser dans les papiers. Dénoncé, il doit d'ailleurs les restituer.

À 18 h, ignorant la chute de la Bastille, Louis XVI ordonne aux troupes d'évacuer Paris. Cet ordre est apporté à l'Hôtel de Ville à deux heures du matin.

Réactions des contemporains

De la Bastille à la Cour

La Bastille dans les premiers jours de sa démolition, Hubert Robert, 1789, Musée Carnavalet

Le 14 juillet 1789, en rédigeant son journal intime, le Roi qui revenait d'une partie de chasse, écrira pour cette même date : « Rien » car il était revenu bredouille de la chasse.

Attention à ne pas s'y méprendre, puisque la matière ordinaire de son journal est composée de chasses, réceptions, cérémonies civiles ou religieuses, voyages, etc…

De plus ce carnet a aussi servi le 17 juillet pour indiquer que le roi s'était rendu à l'Hôtel de Ville de Paris.

La légende rapporte que le Roi ne put être tenu informé des événements parisiens le jour même, qu'au lendemain, le 15 juillet, à Versailles, à 8 heures, au moment de son réveil, que le duc de La Rochefoucauld-Liancourt annonça à Louis XVI la prise de la Bastille.

Le dialogue suivant a souvent été cité par les historiens du XIX :

— « C'est une révolte ? » demanda Louis XVI.

— « Non sire, ce n'est pas une révolte, c'est une révolution. » répondit le duc de La Rochefoucauld.

Néanmoins, il est avéré que dès le jour même Versailles et le roi étaient au courant de la prise de la Bastille. Le récit est de la marquise de La Rochejaquelein : « Le 13 juillet 1789, les régiments de Bouillon et de Nassau arrivèrent à Versailles. On les logea dans l'Orangerie ; nous fumes les voir. Le lendemain, 14 juillet, une foule brillante et nombreuse se promenait dans le parterre du midi, au-dessus de l'Orangerie. Les officiers avaient rassemblé la musique, qui jouait des airs charmants ; la joie brillait sur tous les visages : c'était un tableau ravissant ; mais jamais je n'oublierai le changement subit qui s'opéra. Nous entendîmes d'abord des chuchotements. M. de Bonsol, officiers des gardes du corps, vint à nous, et dit tout bas : Rentrez, rentrez, le peuple de Paris est soulevé ; il a pris la Bastille ; on dit qu'il marche sur Versailles. Nous nous dirigeâmes aussitôt vers notre appartement. Partout la crainte succédait à la gaieté, et en un instant les terrasses furent désertes ». On note par ailleurs une première vague d'émigration massive aux lendemains de la prise de la forteresse. Le 16|juillet, le comte d'Artois et le prince de Condé, colonel général de l'infanterie, avait déjà gagné les frontières du royaume. S'ensuivent les départs des principaux Secrétaires d'États, de Villedeuil, de Broglie et de La Vauguyon.

L'Armée royale

Un fossé déjà ancien s'est creusé davantage au sein des armées royales après les événements parisiens. Les officiers n'ont plus confiance en leurs hommes. Le 14 juillet cinq de six bataillons des Gardes-Françaises s'étaient mutinés et certains avaient rejoint les émeutiers. La semaine précédant les événements, on dénombrait déjà soixante-neuf désertions dans le Régiment de Provence et vingt-neuf dans celui de Vintimille. Le régiment allemand Royal-Hesse-Darmstadt, alors cantonné à Strasbourg apprit la prise de la Bastille le 23 juillet 1789. Il accueillit la nouvelle avec force joie et tapages, ce qui lui valut d'être envoyé en garnison à Neufbrisach. Néanmoins, son ardeur patriote (il fut le premier à adopter la cocarde tricolore) lui valut un retour triomphal à Strasbourg, où il fut acclamé par les bourgeois de la ville.

"Adieu Bastille !" (1789), gravure anonyme représentant le Tiers-État en liesse et le déclin des privilégiés.

Les Constituants

Cette section est vide, insuffisamment détaillée ou incomplète. Votre aide est la bienvenue !

Les observateurs étrangers

Dès le 16 juillet, le duc de Dorset, ambassadeur d'Angleterre et familier du comte d'Artois, écrivait au Foreign Office : « Ainsi, mylord, s'est accomplie la plus grande révolution dont l'Histoire ait conservé le souvenir, et, relativement parlant, si l'on

considère l'importance des résultats, elle n'a coûté que bien peu de sang. De ce moment, nous pouvons considérer la France comme un pays libre. »

Pour Charles James Fox, c'est « le plus grand événement qui soit jamais arrivé au monde »

Ce spectacle inouï provoque chez Edmund Burke un tel étonnement qu'il ne sait s'il doit y souscrire ou le condamner

L'édifice après l'insurrection

Outre les prisonniers, la forteresse héberge les archives du lieutenant de police de Paris qui sont soumises à un pillage systématique. Les Gardes-françaises les dispersent en partie dans les fossés de la forteresse. Dès le 15 juillet, les autorités municipales tentent de les récupérer. Beaumarchais, dont la maison est située juste en face, n'hésite pas à puiser dans les papiers. Dénoncé, il doit d'ailleurs les restituer. En 1798, elles sont conservées à la Bibliothèque de l'Arsenal - dont le directeur est alors Hubert-Pascal Ameilhon - et cataloguées depuis le XIXe siècle (60 000 dossiers comprenant 600 000 feuillets, essentiellement des lettres de cachet, interrogatoires, suppliques au roi, correspondances de l'embastillé).

La Bastille fut ensuite démolie à partir du 15 juillet sous la direction de l'entrepreneur privé Pierre-François Palloy. Son chantier fait l'objet de nombreuses visites, Beaumarchais, Mirabeau, attirés par la poétique des ruines ou voulant participer à cet événement. Celui-ci monta un commerce annexe en transformant les chaînes de la Bastille en médailles patriotiques et en vendant des bagues serties d'une pierre de l'ancienne forteresse.

Pierre de la Bastille (conservée à la mairie de Pontoise)

Miniature de la Bastille sculptée dans une pierre de la Bastille (musée Carnavalet).

Palloy fit faire également des maquettes de l'édifice qui furent envoyées dans tous les chefs-lieux des départements français). On peut y ajouter la transformation en objets de piété et de culte, de tout ce qu'il put récupérer sur les boiseries et les ferronneries de la vieille forteresse. La plus grande part a servi à construire le pont de la Concorde. Le marquis de La Fayette envoya une des clés de la Bastille à George Washington, l'une des grandes figures de la Révolution américaine et premier président des États-Unis. Elle est aujourd'hui exposée à la résidence de Mount Vernon, transformée en musée. Une autre des clés fut envoyée à Gournay-en-Bray, lieu de naissance, du premier révolutionnaire à être entré dans la Bastille, Maillart. Cette dernière clé a depuis disparu. C'est à la fonderie de Romilly, dans l'Eure, qu'ont été conservées jusqu'à sa fermeture l'horloge et les cloches de la forteresse. Le carillon quant à lui se trouve actuellement au Musée européen d'art campanaire, à L'Isle-Jourdain (Gers) La disparition de la Bastille n'empêche pas son mythe de renaître dès la Révolution sous la forme d'une mode « à la Bastille » (bonnet, souliers, éventails)."

C - 2. Code du LICI

Cette partie de l'annexe est destiné à fournir le code en Python du programme LICI aux personnes de la communauté de recherche en TAL qui pourraient être intéressés par celui-ci. J'ai retiré dans ce code les éléments propres à l'intégration du programme dans la barre d'outils de Médialexie, notamment tout ce qui concerne la structure xml spécifique aux fichiers ctm. Cela correspond à l'intégralité de la classe "Graphique".

C - 2.1. Programme principal

Je présente ici le code du programme LICI tel qu'il était dans sa dernière version béta. Il s'agit de la v16.5, le code a ensuite été encore modifié, optimisé et corrigé avant la sortie de la version commerciale du logiciel. Le code est commenté, c'est à dire qu'il est accompagné d'explication sur le rôle des fonctions appelées et séparé en partie délimitées par des titres. Toutes les lignes commençant par le caractère # sont des lignes de commentaires, toutes les autres lignes sont des lignes qui sont effectivement prises en comptes par l'interpréteur Python et ont un rôle à jouer dans l'exécution du programme. Il est possible que les commentaires contiennent de nombreuses fautes d'orthographe puisqu'ils ont été écrit lors du développement et n'ont pas été relu en dehors de leur aspect technique.

Voici d'abord le code du programme principal :

```
# -*- coding: utf-8 -*-
import codecs
import re
from nltk.probability import FreqDist
from nltk.collocations import *
from nltk.tokenize import RegexpTokenizer
import nltk.data
import os.path
import glob
import operator

##////////////////////////////##
##    Programme principal    ##
##////////////////////////////##
```

Annexe C

```
#On importe le module
import Lici_Classes

#On importe les classes de fonction du module
resume=Lici_Classes.Resumeur()
graphiq=Lici_Classes.Graphique()

##//////////////////##
## Accès au Corpus ##
##//////////////////##

from nltk.corpus import PlaintextCorpusReader
#Importation du module qui permet de lire des fichiers textes

dossier_corpus='textes'
#ici, entre guillemets on met le chemin d'accès au dossier qui contient ton texte

textes_corpus=PlaintextCorpusReader(dossier_corpus,'.*')
#cette fontion stocke les textes de ce dossier dans une variable

##//////////////////////////////////////##
## Traitement de chaque texte du corpus ##
##//////////////////////////////////////##

for fi in textes_corpus.fileids():
        #nom du fichier traité
        nomFichier=fi.replace(".txt","")

        #création du fichier texte de sortie :
        ResumerSortie="ResumeDe" +fi #On crée le nouveau fichier

        #création de la carte de sortie :
```

```
        CarteSortie="carte%s.ctm" %nomFichier #On crée le nouveau fichier


        #Récupération des données du fichier texte dans une chaine de
caractère
        texte=textes_corpus.raw(fi)




        ##////////////////////////////##
        ## Création du fichier ctm     ##
        ##////////////////////////////##


        # ici se trouvait dans le programme la structure xml propre au
format ctm de Médialexie, que j'ai donc retiré dans les annexes de la thèse


        #on récupère le texte sous forme de chaine de caractères
        texte2=texte.decode('latin-1')


        #On récupère l'éventuel titre !
        titreTexte=resume.titres(texte2)


        #On coupe le titre du texte
        texteCoupe=texte2.replace(titreTexte,"",1)


        #On segmente le texte en graphie
        texte3 = resume.segmentation(texte2)


        #On retire les majuscules inutiles pour le traitement
        texte3maj = resume.maj(texte3)


        #on définit :
        #la limite sert à déterminer les paliers pour changer la couleur
des liens (limite)
        #le nombre minimum de relations pour tracer des liens entre les
éléments (épaisseurMini)
```

```
        #le nombre de mots-clés maximum, qui concerne les mots simples,
avant calcul des collocations (nbrMotcles)

        #le nombre maximum de termes-clés (nbrTermesclesMax)

        #le nombre maximum de liens dans la carte (nbrliensMax)

        #le nombre maximum de liens par nœuds (nbrliensMaxNoeud)

        #la limite sert aussi à calculer le nombre d'occurrences minimal
pour qu'une collocation soit dans la liste des collocations


        if os.path.isfile("Parametres.txt"):

                param=resume.fichierReglage()

                if len(texte3)<1000:

                        limite = len(texte3)//300

                        epaisseurMini=param[0]

                        nbrMotcles=param[1]

                        nbrTermesclesMax=param[2]

                        nbrliensMax=param[3]

                        nbrliensMaxNoeud=param[4]

                else:

                        if len(texte3)<3000:

                                limite = len(texte3)//350

                                epaisseurMini=param[5]

                                nbrMotcles=param[6]

                                nbrTermesclesMax=param[7]

                                nbrliensMax=param[8]

                                nbrliensMaxNoeud=param[9]

                        else:

                                limite = len(texte3)//400

                                epaisseurMini=param[10]

                                nbrMotcles=param[11]

                                nbrTermesclesMax=param[12]

                                nbrliensMax=param[13]

                                nbrliensMaxNoeud=param[14]

        else:

                if len(texte3)<1000:
```

```
                        limite = len(texte3)//300
                        epaisseurMini=1
                        nbrMotcles=7
                        nbrTermesclesMax=9
                        nbrliensMax=9
                        nbrliensMaxNoeud=2
                else:
                        if len(texte3)<3000:
                                limite = len(texte3)//350
                                epaisseurMini=2
                                nbrMotcles=8
                                nbrTermesclesMax=9
                                nbrliensMax=14
                                nbrliensMaxNoeud=3
                        else:
                                limite = len(texte3)//400
                                epaisseurMini=3
                                nbrMotcles=9
                                nbrTermesclesMax=9
                                nbrliensMax=20
                                nbrliensMaxNoeud=4


        #On récupère également la localisation des fichiers d'exceptions
dans les paramètres !!
        localisation="defaut"
        if os.path.isfile("Parametres.txt"):
                localisation=param[16]


        #Enfin, on récupère dans les paramètres le nombre de phrases-clés
souhaitées par l'utilisateur
        phC=param[15]


        #On récupère les phrases-clés
        resultat = resume.resumer(texteCoupe,texte3,phC,localisation)
```

```
        texte4 = "Voici les "+str(phC)+" phrases cl\xe9s du texte : \n \n"
+ resultat.encode('latin-1')


        #On récupère les mots-clés

        mots = resume.mots_cles(texte3maj,nbrMotcles,localisation)


        #On prépare l'écriture des mots-clés dans un fichier texte

        motscles =
resume.scores_mots_cles(texte3maj,nbrMotcles,localisation)

        motscles2=""

        for mc in motscles:

                motscles2 = motscles2 + u"""\n %s  %i""" %(mc[0],mc[1])
+u"""\n"""

        motscles3 = "\n \nVoici les mots-cl\xe9s du texte : \n \n" +
motscles2.encode('latin-1')


        #On supprime les doublons singulier/pluriel de la liste de mots-
clés

        mcPluriels = resume.pluriels(motscles)


        for plur in mcPluriels:

                mots.remove(plur)


        #On récupère les meilleures collocations

        motscol = resume.collocations(texte3maj,limite,localisation)


        #On supprime les doublons singulier/pluriel de la liste de
collocations

        collocPluriels = resume.pluriels2(motscol)


        for plur2 in collocPluriels:

                motscol.remove(plur2)


        #crétion de la liste des triples

        bigrams=resume.collocations2(texte3maj,limite)
```

```
        #On génère tous les bigrams du texte et on les fusionne pour
trouver les trigrams

        listeTrigram=resume.triples(bigrams)


        #On prépare l'écriture des collocations dans un fichier texte

        motscol2=""

        listeTriplet=""

        i=0

        for paires in motscol:

                motscol2=motscol2+"\n"+motscol[i][0]+" "+motscol[i][1]

                i+=1

        for tripl in listeTrigram:

                listeTriplet=listeTriplet+"\n"+tripl[0]+" "+tripl[1]+"
"+tripl[2]

        motscol3 = "\n \nVoici les paires de mots en collocation
fr\xe9quente dans le texte : \n" + motscol2.encode('latin-1') + """

\nVoici les triplets de mots en collocation fr\xe9quente dans le texte :
\n""" + listeTriplet.encode('latin-1')


        #création de la liste des termes-clés

        termesCles2=resume.concat(mots,motscol)


termesCles3=resume.concatTriples(termesCles2,listeTrigram,localisation)

        termesCles4=resume.quadruples(termesCles3)

        termesCles=resume.quadruples2(texte2,termesCles4)


        #on transforme les listes de collocation en chaine de caractères

        Ltermes=[]

        apostrophes=["l'","L'","d'","D'"]

        for t in termesCles:

                if isinstance(t,tuple) or isinstance(t,list):

                        chaine=t[0]

                        for j in range (len(t)-1):

                                if t[j] in apostrophes:

                                        chaine=chaine+t[j+1]
```

```
                                else:
                                        chaine=chaine+" "+t[j+1]
                        else :
                                chaine=t
                        Ltermes.append(chaine)


        # On supprime de la liste les collocations calculées qui
n'apparaissent pas dans le texte
        # par exemple deux mots en collocations fréquentes mais qui ne se
suivent jamais tels quels dans le texte
        nouvelleListe=[]
        mauvaiseListe=[]
        indexListe=[]
        ind=0
        for element in Ltermes:
                apparitions=resume.occurrences(texte2,element)
                if apparitions>0:
                        nouvelleListe.append(element)
                else:
                        mauvaiseListe.append(element)
                        indexListe.append(ind)
                ind+=1
        Ltermes=nouvelleListe


        #On veut récupérer les mots clés exclus à cause d'une collocation
fréquente mais n'apparaissant pas dans le texte tel quel
        listeRecup=resume.restauration(Ltermes,
mauvaiseListe,mots,indexListe)
        for couple in listeRecup:
                Ltermes.insert(couple[0],couple[1])


        #On vérifie qu'il ne reste pas plus d'éléments que souhaité dans la
liste
        #les termes sont encore classés dans l'ordre de leur fréquence,
donc on enlève les derniers éléments en premier, avec "pop"
```

```
        #on va garder ses éléments pour la liste des autres termes-clés !

        AutresTermes=[]

        while len(Ltermes)>nbrTermesclesMax:

                autre=Ltermes.pop()

                AutresTermes.append(autre)


        #On tri la liste des termes-clés en fonction de leur ordre
d'apparition dans le texte !

        termes=resume.triTermes(texte2,Ltermes)


        #On récupère la liste des termes clés

        listeTermes="\n\n".join(termes)

        termes2 = "\n \nVoici la liste des termes cl\xe9s apparaissants
dans la carte : \n" + listeTermes.encode('latin-1')


        #On créé une concordance pour chaque terme clé

        listeTotaleConcor=[]

        for term in termes:

                concord=resume.concordances(texte2,term)

                listeConcor="""\nConcordance dans le texte pour le terme
'%s' :\n""" %term +"%s" %concord

                listeTotaleConcor.append(listeConcor)

        Concordances="\n".join(listeTotaleConcor)

        Concordances2="\n \n \nConcordanceur des termes cl\xe9s du texte :
\n" + Concordances.encode('latin-1')


        ##////////////////////////////////////////////////##

        ##   Ecriture du fichier de sortie au format txt    ##

        ##////////////////////////////////////////////////##


        #on écrit la liste de phrases-clés et la liste de mots-clés, dans
le fichier de sortie au format texte

        fichier1 = open(ResumerSortie, "w") # On ouvre le fichier en
écriture.

        fichier1.write(texte4)
```

```
        fichier1.write(motscles3)
        fichier1.write(motscol3)
        fichier1.write(termes2)
        fichier1.write(Concordances2)
        fichier1.close()


        ##////////////////////////////////////////////////##
        ##   Ecriture du fichier de sortie au format ctm    ##
        ##////////////////////////////////////////////////##

        #On écrit le titre dans la carte
        titreCarte=graphiq.titre(titreTexte)

        #On créé la liste des termes-clés pour le fichier ctm
        ListeNoeuds=[]
        identifiantMot=0
        couleurMot=0
        placement=0
        premierTerme=mots[0]
        for m in termes:
                chaine="""(^|\s)%s(\s|$)"""%premierTerme
                if re.search(chaine,m):
                        couleurMot=255
                occ=resume.occurrences(texte2,m)
                if occ==0:
                        continue
                CONC=resume.concordances(texte2,m)
                bigrams2=resume.collocations3(texte3maj,limite)
                nouveauTerme=resume.determinants(m,bigrams2)
                if nouveauTerme == " ":
                        n=[m,identifiantMot,couleurMot,occ,CONC,m]
                        ListeNoeuds.append(n)
```

```
                else:
n=[nouveauTerme,identifiantMot,couleurMot,occ,CONC,m]
                        ListeNoeuds.append(n)
                identifiantMot += 1
                couleurMot=0

        #On créé la liste des paragraphes du texte
        paragra = resume.paragraphes(texte2)

        #On calcul les liens entre les termes clés
        ListeLiens=[]
        idMot1=0
        liens = u""
        for m in termes:
                idMot2=0
                for n in termes:
                        if m!= n:
                                if (idMot1<idMot2):
                                        epaisseur =
resume.MemePara(paragra,m,n)
                                        if epaisseur>=epaisseurMini:
                                                l=[idMot1,idMot2,epaisseur]
                                                ListeLiens.append(l)
                        idMot2 +=1
                idMot1 +=1

        #On tri la liste de liens en fonction de leurs épaisseurs
        ListeLiens.sort(key=operator.itemgetter(2),reverse=True)


        ListeLiens2=[]
        for i in range(len(termes)):
                y=0
```

```
                for l in ListeLiens:
                        if l[0]==i:
                                if y<nbrliensMaxNoeud:
                                        ListeLiens2.append(l)
                                        y += 1


        #On tri la nouvelle liste de liens en fonction de leurs épaisseurs générales puisqu'ils se retrouvent dans l'ordre de l'épaisseur des liens par mot
        ListeLiens2.sort(key=operator.itemgetter(2),reverse=True)


        #On supprime des liens de la liste, jusqu'à ce qu'on ne dépasse plus le nombre de liens maximum
        while len(ListeLiens2)>nbrliensMax:
                ListeLiens2.pop()


        #On écrit les liens calculés au format xml pour les intégrer au ficher de sortie ctm
        #On calcul en même temps la liste des termes clés qui sont connectés à d'autres par des nœuds
        #Les termes non-connectés seront dans la boite de vocabulaire "autres termes-clés"
        ListeDesTermesConnectes=[]
        for li in ListeLiens2:
                lienxml=graphiq.lien(li[0],li[1],li[2],limite)
                if li[0] not in ListeDesTermesConnectes:
                        ListeDesTermesConnectes.append(li[0])
                if li[1] not in ListeDesTermesConnectes:
                        ListeDesTermesConnectes.append(li[1])
                liens +=lienxml


        #On va connecter par des liens faibles, les noeuds non connectés


        #On sélectionne les termes-clés à afficher en nœud de la carte et ceux à ajouter dans la boite de vocabulaire
        balise = u""
```

```
        n=0
        idlienFaible=600
        for tc in ListeNoeuds:
                if tc[1] in ListeDesTermesConnectes:

noeudxml=graphiq.noeud(tc[0],tc[1],tc[2],tc[3],tc[4],n,tc[5])
                        balise +=noeudxml
                        n +=1
                else:

lienFaiblexml=graphiq.lienFaible(tc[1],idlienFaible)
                        idlienFaible+=1
                        liens += lienFaiblexml

noeudxml=graphiq.noeud(tc[0],tc[1],tc[2],tc[3],tc[4],n,tc[5])
                        balise +=noeudxml
                        n +=1
                        AutresTermes.insert(0,tc[0])


        ChaineAutresTermes=""
        for autr in AutresTermes:
                ChaineAutresTermes += "- %s\n"%autr
        #cadreVocab=graphiq.autres(ChaineAutresTermes)


        #On crée les cadres de termes-clés et de phrases-clés
        cadreTC=graphiq.cadreTermes(listeTermes)
        cadrePC=graphiq.cadrePhrases(resultat)


        #On concatène l'ensemble du fichier xml
        final=intro+titreCarte+cadreTC+cadrePC+balise+liens+fin


        #on écrit l'arbre xml, contenant les mots-clés dans le fichier de
sortie de la carte au format ctm
        fichier2 = codecs.open(CarteSortie, 'w', encoding='utf-16-le')
```

```
        fichier2.write(final)
        #On ferme le fichier
        fichier2.close()
```

Voici ensuite le code de la classe "Resumeur", comprise dans un autre fichier, qui contient aussi la classe "Graphique", que je ne présente pas ici :

```
import codecs

##///////////////##
##     LICI     ##
##///////////////##

import re
from nltk.probability import FreqDist
from nltk.collocations import *
from nltk.tokenize import RegexpTokenizer
import nltk.data
import os.path
import glob
import operator

##//////////////////////////////##
##     Classes de fonctions     ##
##//////////////////////////////##
class Resumeur:

        def segmentation(self,texte):
            #fonction de segmentation d'un texte en graphies. Renvoie la
liste des graphies
                segmentation_graphies = r'''(?x)
                  aujourd'hui    # exception 1
                | prud'hom\w+ # exception 2
                | \d+(,\d+)?\s*[%€$] # les valeurs
```

```
                | \d+ (\d+)*            # les hautes valeurs regroupées par
paquets
                | \d+                  # les nombres
                | \w'                  # les contractions d', l', j', t', s'
                | \w+(-\w+)+          # les mots composés
                | (\d|\w)+            # les combinaisons alphanumériques
                | \w+                 # les mots simples
                '''
                segmenter = RegexpTokenizer(segmentation_graphies,
flags=re.UNICODE|re.IGNORECASE)
                texteSeg = [graphie for graphie in
segmenter.tokenize(texte)]
                return texteSeg


        def paragraphes (self, texte):
            #fonction de segmentation d'un texte en paragraphes. Renvoie la
liste des paragraphes
                segmentation_para = r'''(?x)
                  .*\r
                | .*\n
                '''
                segmenter = RegexpTokenizer(segmentation_para,
flags=re.UNICODE|re.IGNORECASE)
                texteSeg = [chaine for chaine in segmenter.tokenize(texte)]
                return texteSeg


        def titres (self, texte):
                #segmentation des titres !
                segmentation_titre = r'''(?x)
                .+ \n
                '''
                titre = re.match(segmentation_titre,texte,flags=re.UNICODE)
                titreCarte = "Carte sans titre"
                if titre!= None:
                        if len(titre.group(0)) < 60:
```

```
                                titreCarte = titre.group(0)

                return titreCarte


        def MemePara (self, paras, mot1, mot2):

                #on prend en entrée la liste des paragraphes du texte et
deux termes clés

                #on renvoie le nombre de paragraphes où les mots sont
communs

                lien = 0

                info = []

                for para in paras:

                        para2=para.lower()

                        if para2.find(mot1.lower())!= -1:

                                if para.find(mot2.lower()) != -1:

                                        lien+=1

                return lien


        def collocations(self,texteSeg,palier,loc):

                #fonction de détection des collocations, renvoie la liste
des meilleurs collocations du texte

                filtre = palier/3+2

                #prend en entrée un texte segmenté !

                #recherche des bigrames

                bigram = nltk.collocations.BigramAssocMeasures()

                colloc =
BigramCollocationFinder.from_words(texteSeg,window_size = 3)

                #on définit le texte dans lequel on recherche et la taille
de la fenêtre de recherche

                colloc.apply_freq_filter(filtre)

                #filtre : au moins n apparitions de la collocation

                mg=self.fichierExceptionCollocations(loc)

                if not mg:

                        mg =
set(["etc",u'dirig\xe9',"est","eut","www","toutes","faut","même",u'm\xeame'
,"fait","toujours","dire","question","réponse",u'r\xe9ponse',"oui","non","t
emps","entre","chose","y","avait","faire","va","aller","bien","dit","est-
ce","dont","autres","pendant","durant","pouquoi","cependant","déjà","lui","
```

```
non","ont","ainsi","cet","s","l","d","c","c'","j","j'","aux","jusqu","fut",
"n'","4","5","alors","au","aucuns","aussi","autre","avant","avec","avoir","
bon","car","ce","cela","ces","cette","ceux","chaque","ci","comme","comment"
,"dans","de","des","du","dedans","dehors","depuis","deux","devrait","doit",
"donc","début",u'd\xe9but',"elle","elles","en","encore","essai","est","et",
"eu","fait","faites","fois","font","haut","hors","ici","il","ils","je","jus
te","la","le","les","leur","leurs","lors","là",u'l\xe0',"l'","d'","un","une
","ma","maintenant","mais","mes","moins","mon","mot","même",u'm\xeame',"ni"
,"nommés",u'nomm\xe9s',"notre","nous","nouveaux","on","ou","où","par","parc
e","parole","pas","personnes","peut","peu","plupart","pour","pourquoi","qua
nd","que","quel","quelle","quelles","quels","qui","sa","sans","ses","seulem
ent","si","sien","son","sont","sous","soyez","sur","ta","tandis","tellement
","tels","tes","ton","tous","tout","trop","très",u'tr\xe8s',"tu","valeur","
voie","voient","vont","votre","vous","vu","ça",u'\xe7a',"étaient",u'\xe9tai
ent',"était",u'\xe9tait',"étions",u'\xe9tions',"été",u'\xe9t\xe9',"être",u'
\xeatre',"1","2","3","a","b","se","ne","qu","qu'","plus","après",u'apr\xe8s
',"à",u'\xe0',"s'"])

                #création d'une liste de mots grammaticaux

                colloc.apply_word_filter(lambda m: len(m) <2 or m.lower()
in mg)

                #filtre : on garde les mots non inclus dans la liste des
mots grammaticaux

                res = colloc.nbest(bigram.likelihood_ratio, 15)

                #on récupère les 15 meilleurs résultats

                return res


        def collocations2(self,texteSeg,palier):

                filtre = palier/3+2

                #prend en entrée un texte segmenté !

                #recherche des bigrames

                bigram = nltk.collocations.BigramAssocMeasures()

                #on définit le texte dans lequel on recherche et la taille
de la fenêtre de recherche

                colloc =
BigramCollocationFinder.from_words(texteSeg,window_size = 3)

                #filtre : au moins n apparitions de la collocation

                colloc.apply_freq_filter(filtre)

                #on récupère les 50 meilleurs résultats

                res = colloc.nbest(bigram.likelihood_ratio, 50)

                return res


        def collocations3(self,texteSeg,palier):
```

```
                filtre = palier/3+2
                #prend en entrée un texte segmenté !
                #recherche des bigrames
                bigram = nltk.collocations.BigramAssocMeasures()
                #on définit le texte dans lequel on recherche et la taille
de la fenêtre de recherche
                colloc =
BigramCollocationFinder.from_words(texteSeg,window_size = 2)
                #filtre : au moins n apparitions de la collocation
                colloc.apply_freq_filter(filtre)
                #on récupère les 50 meilleurs résultats
                res = colloc.nbest(bigram.likelihood_ratio, 50)
                return res


        def determinants(self,terme,listeCollocations):
                nouveauTerme=" "
                det=["le","la","les","un","une","des",u'l\x27']
                if not (isinstance(terme,tuple) or isinstance(terme,list)):
                        for col in listeCollocations:
                                if col[1]==terme and col[0].lower() in det:
                                        if col[0].lower() == u'l\x27':
                                                nouveauTerme=col[0]+col[1]
                                        else:
                                                nouveauTerme=col[0]+"
"+col[1]
                                        break
                return nouveauTerme



        def triples(self,listeCollocations):
                trigram=[]
                i=0
                for col in listeCollocations:
                        for col2 in listeCollocations[i+1:]:
```

```
                                if col[1] == col2[0]:
                                        complem=(col[0],col2[1])
                                        if complem in listeCollocations:
triple=[col[0],col[1],col2[1]]
                                                if triple not in trigram:
trigram.append(triple)
                                if col2[1] == col[0]:
                                        complem=(col2[0],col[1])
                                        if complem in listeCollocations:
triple=[col2[0],col2[1],col[1]]
                                                if triple not in trigram:
trigram.append(triple)
                        i+=1
                return trigram

        def quadruples(self, termesCles):
                listeSortie=[]
                for tc in termesCles:
                        ajout=0
                        if isinstance(tc,list):
                                for tc2 in termesCles:
                                        if isinstance(tc2,list):
                                                if tc2[0]==tc[1]
and tc2[1]==tc[2]:
nouveauTerme=[tc[0],tc2[0],tc2[1],tc2[2]]
                                                        ajout=1
                                                        if
nouveauTerme not in listeSortie:
listeSortie.append(nouveauTerme)
                                                if tc[0]==tc2[1]
and tc[1]==tc2[2]:
```

```
nouveauTerme=[tc2[0],tc[0],tc[1],tc[2]]
                                                                ajout=1
                                                                if
nouveauTerme not in listeSortie:

listeSortie.append(nouveauTerme)
                        if ajout==0:
                                listeSortie.append(tc)
                return listeSortie


        def quadruples2(self, texte, termesCles):
                listeSortie=[]
                for tc in termesCles:
                        ajout=0
                        if isinstance(tc,list):
                                for tc2 in termesCles:
                                        if isinstance(tc2,tuple):
                                                if tc2[0]==tc[2]:

nouveauTerme=[tc[0],tc[1],tc[2],tc2[1]]
                                                        ajout=1
                                                        if nouveauTerme not
in listeSortie:

chaine=tc[0]+" "+tc[1]+" "+tc[2]+" "+tc2[1]
                                                                if
self.occurrences(texte,chaine) > 2:

listeSortie.append(nouveauTerme)
                                                                else:

ajout=0
                                                if tc2[1]== tc[0]:

nouveauTerme=[tc2[0],tc2[1],tc[1],tc[2]]
                                                        ajout=1
```

```
                                                    if nouveauTerme not
in listeSortie:

chaine=tc2[0]+" "+tc2[1]+" "+tc[1]+" "+tc[2]

                                                        if
self.occurrences(texte,chaine) > 2:

listeSortie.append(nouveauTerme)

                                                        else:

ajout=0

                        if isinstance(tc,tuple):

                                for tc2 in termesCles:

                                        if isinstance(tc2,list):

                                                if tc[0]==tc2[2]:

nouveauTerme=[tc2[0],tc2[1],tc2[2],tc[1]]

                                                    ajout=1

                                                    if nouveauTerme not
in listeSortie:

chaine=tc2[0]+" "+tc2[1]+" "+tc2[2]+" "+tc[1]

                                                        if
self.occurrences(texte,chaine) > 2:

listeSortie.append(nouveauTerme)

                                                        else:

ajout=0

                                            if tc[1]== tc2[0]:

nouveauTerme=[tc[0],tc[1],tc2[1],tc2[2]]

                                                    ajout=1

                                                    if nouveauTerme not
in listeSortie:

chaine=tc[0]+" "+tc[1]+" "+tc2[1]+" "+tc2[2]

                                                        if
self.occurrences(texte,chaine) > 2:
```

```
listeSortie.append(nouveauTerme)
                                                                    else:
ajout=0
                        if ajout==0:
                                listeSortie.append(tc)
                return listeSortie


        def mots_freq (self, texteSeg, nb_mots ):
                # calculer la fréquence de chaque graphie
                graphies = [graphie for graphie in texteSeg]
                frequence_graphies = FreqDist(graphies)
                # créer la liste des graphies les plus fréquentes
                plus_frequentes_graphies = [pair[0] for pair in
frequence_graphies.items()[:nb_mots]]
                #renvoyer la listes des mots clés
                return plus_frequentes_graphies


        def scores_mots_freq (self, texteSeg, nb_mots ):
                # calculer la fréquence de chaque graphie
                graphies = [graphie for graphie in texteSeg]
                frequence_graphies = FreqDist(graphies)
                # créer la liste des graphies les plus fréquentes
                plus_frequentes_graphies = [pair for pair in
frequence_graphies.items()[:nb_mots]]
                #renvoyer la listes des mots clés
                return plus_frequentes_graphies


        def scores_mots_cles (self, texteSeg, nb_mots, loc):
                # calculer la fréquence de chaque graphie
                stops=self.fichierExceptionTermes(loc)
                if not stops:
stops=set(["objet","bonne","ensuite","surtout","fortement","prendre","autou
```

```
r","notamment","choses","chose","etc","est","dirigé",u'dirig\xe9',"avez","v
oir","eut",u'o\xf9',"alla","où","fort","n","i","e","p","www","toutes","faut
","même",u'm\xeame',"fait","toujours","dire","question","réponse",u'r\xe9po
nse',"oui","non","temps","entre","chose","y","avait","faire","va","aller","
bien","dit","est-
ce","dont","autres","pendant","durant","pouquoi","cependant","déjà","lui","
non","ont","ainsi","cet","s","l","d","c","c'","j","j'","aux","jusqu","fut",
"n'","4","5","alors","au","aucuns","aussi","autre","avant","avec","avoir","
bon","car","ce","cela","ces","cette","ceux","chaque","ci","comme","comment"
,"dans","de","des","du","dedans","dehors","depuis","deux","devrait","doit",
"donc","début",u'd\xe9but',"elle","elles","en","encore","essai","est","et",
"eu","fait","faites","fois","font","haut","hors","ici","il","ils","je","jus
te","la","le","les","leur","leurs","lors","là",u'l\xe0',"l'","d'","un","une
","ma","maintenant","mais","mes","moins","mon","mot","même",u'm\xeame',"ni"
,"nommés",u'nomm\xe9s',"notre","nous","nouveaux","on","ou","où","par","parc
e","parole","pas","personnes","peut","peu","plupart","pour","pourquoi","qua
nd","que","quel","quelle","quelles","quels","qui","sa","sans","ses","seulem
ent","si","sien","son","sont","sous","soyez","sur","ta","tandis","tellement
","tels","tes","ton","tous","tout","trop","très",u'tr\xe8s',"tu","valeur","
voie","voient","vont","votre","vous","vu","ça",u'\xe7a',"étaient",u'\xe9tai
ent',"était",u'\xe9tait',"étions",u'\xe9tions',"été",u'\xe9t\xe9',"être",u'
\xeatre',"1","2","3","a","b","se","ne","qu","qu'","plus","après",u'apr\xe8s
',"à",u'\xe0',"s'"])

                graphies = [graphie for graphie in texteSeg if
graphie.lower() not in stops]

                frequence_graphies = FreqDist(graphies)

                # créer la liste des graphies les plus fréquentes

                plus_frequentes_graphies = [pair for pair in
frequence_graphies.items()[:nb_mots]]

                #renvoyer la listes des mots clés

                return plus_frequentes_graphies


        def maj(self, texteSeg):

                nouvtexte=[]

                for graphie in texteSeg:

                        graphieMaj=graphie.lower()

                        if graphieMaj in texteSeg:

                                nouvgraphie=graphieMaj

                        else:

                                nouvgraphie=graphie

                        nouvtexte.append(nouvgraphie)

                return nouvtexte


        def mots_cles (self, texteSeg, nb_mots, loc):
```

```
                # calculer la fréquence de chaque graphie

                stops=self.fichierExceptionTermes(loc)

                if not stops:


stops=set(["objet","bonne","ensuite","surtout","fortement","prendre","autou
r","notamment","choses","chose","etc","est","dirigé",u'dirig\xe9',"avez","v
oir","eut",u'o\xf9',"alla","où","fort","n","i","e","p","www","toutes","faut
","même",u'm\xeame',"fait","toujours","dire","question","réponse",u'r\xe9po
nse',"oui","non","temps","entre","chose","y","avait","faire","va","aller","
bien","dit","est-
ce","dont","autres","pendant","durant","pouquoi","cependant","déjà","lui","
non","ont","ainsi","cet","s","l","d","c","c'","j","j'","aux","jusqu","fut",
"n'","4","5","alors","au","aucuns","aussi","autre","avant","avec","avoir","
bon","car","ce","cela","ces","cette","ceux","chaque","ci","comme","comment"
,"dans","de","des","du","dedans","dehors","depuis","deux","devrait","doit",
"donc","début",u'd\xe9but',"elle","elles","en","encore","essai","est","et",
"eu","fait","faites","fois","font","haut","hors","ici","il","ils","je","jus
te","la","le","les","leur","leurs","lors","là",u'l\xe0',"l'","d'","un","une
","ma","maintenant","mais","mes","moins","mon","mot","même",u'm\xeame',"ni"
,"nommés",u'nomm\xe9s',"notre","nous","nouveaux","on","ou","où","par","parc
e","parole","pas","personnes","peut","peu","plupart","pour","pourquoi","qua
nd","que","quel","quelle","quelles","quels","qui","sa","sans","ses","seulem
ent","si","sien","son","sont","sous","soyez","sur","ta","tandis","tellement
","tels","tes","ton","tous","tout","trop","très",u'tr\xe8s',"tu","valeur","
voie","voient","vont","votre","vous","vu","ça",u'\xe7a',"étaient",u'\xe9tai
ent',"était",u'\xe9tait',"étions",u'\xe9tions',"été",u'\xe9t\xe9',"être",u'
\xeatre',"1","2","3","a","b","se","ne","qu","qu'","plus","après",u'apr\xe8s
',"à",u'\xe0',"s'"])

                graphies = [graphie for graphie in texteSeg if
graphie.lower() not in stops]

                frequence_graphies = FreqDist(graphies)

                # créer la liste des graphies les plus fréquentes

                plus_frequentes_graphies = [pair[0] for pair in
frequence_graphies.items()[:nb_mots]]

                #renvoyer la listes des mots clés

                return plus_frequentes_graphies


        def pluriels (self, motsCles):

                intru=[]

                for i in range(len(motsCles)):

                        for j in range(i+1,len(motsCles)):

                                pluriel = motsCles[i][0]+"s"

                                pluriel2 = motsCles[i][0]+"x"

                                pluriel3 = motsCles[j][0]+"s"

                                pluriel4 = motsCles[j][0]+"x"
```

```
                                if motsCles[j][0] == pluriel2 or
motsCles[j][0] == pluriel:
                                        if motsCles[j][1] > motsCles[i][1]:

intru.append(motsCles[i][0])
                                        else:

intru.append(motsCles[j][0])
                                if motsCles[i][0] == pluriel3 or
motsCles[i][0] == pluriel4:
                                        if motsCles[i][1] > motsCles[j][1]:

intru.append(motsCles[j][0])
                                        else:

intru.append(motsCles[i][0])
                return intru


        def pluriels2 (self, listeCollocs):
                intru=[]
                for i in range(len(listeCollocs)):
                        for j in range(i+1,len(listeCollocs)):
                                pluriel = listeCollocs[i][0]+"s"
                                pluriel2 = listeCollocs[i][0]+"x"
                                pluriel3 = listeCollocs[i][1]+"s"
                                pluriel4 = listeCollocs[i][1]+"x"
                                pluriel5 = listeCollocs[j][0]+"s"
                                pluriel6 = listeCollocs[j][0]+"x"
                                pluriel7 = listeCollocs[j][1]+"s"
                                pluriel8 = listeCollocs[j][1]+"x"
                                if listeCollocs[j][0] == pluriel2 or
listeCollocs[j][0] == pluriel or listeCollocs[j][0] == listeCollocs[i][0]:
                                        if listeCollocs[j][1] == pluriel3
or listeCollocs[j][1] == pluriel4 or listeCollocs[j][1] ==
listeCollocs[i][1]:

intru.append(listeCollocs[j])
```

```
                                if listeCollocs[i][0] == pluriel5 or
listeCollocs[i][0] == pluriel6 or listeCollocs[j][0] == listeCollocs[i][0]:

                                        if listeCollocs[i][1] == pluriel7
or listeCollocs[i][1] == pluriel8 or listeCollocs[j][1] ==
listeCollocs[i][1]:


intru.append(listeCollocs[j])

                return intru


        def concat(self,liste,listeColloc):

                #fonction de concaténation de la liste des mots-clés et de
la liste des meilleures collocations

                listeConc=[]

                for m in liste:

                        trouve=0

                        for n in listeColloc:

                                if n[0]==m or n[1]==m:

                                        if n not in listeConc:

                                                listeConc.append(n)

                                        trouve=1

                        if trouve==0:

                                listeConc.append(m)

                return listeConc


        def concatTriples(self, termesCles, triples, loc):

                listeSortie=[]

                exceptions=["du","de"]

                exceptions2=["les"]

                exceptions3=["du","de","les","le","la","des","un","une"]

                exceptions4=self.fichierExceptionCollocations(loc)

                det=["Les","Le","La","Des","Une","Un",u'L\x27']

                for tc in termesCles:

                        ajout=0

                        if isinstance(tc,tuple):

                                for triple in triples:
```

```
                                        if triple[0] in det:
                                                triple[0]=triple[0].lower()
                                        if (triple[0]==tc[0] or
triple[1]==tc[0]) and triple[2]==tc[1] and triple[0] not in exceptions and
triple[1] not in exceptions2:
                                                ajout=1
                                                if triple not in
listeSortie :

listeSortie.append(triple)
                                        if triple[0]==tc[0] and
triple[1]==tc[1] and triple[2] not in exceptions3 and triple[2] not in
exceptions4:
                                                ajout=1
                                                if triple not in
listeSortie :

listeSortie.append(triple)
                                if ajout==0:
                                        listeSortie.append(tc)
                        else :
                                listeSortie.append(tc)
                return listeSortie


        def
restauration(self,listeTermes,termesExclus,listeMots,listeIndex):
                listeRestauration=[]
                coupleRest=[]
                i=0
                for te in termesExclus:
                        ind=listeIndex[i]
                        i+=1
                        for mot in listeMots:
                                y=0
                                if mot in te:
                                        for tc in listeTermes:
                                                if mot in tc:
```

```
                                                y=1
                                        if y==0:
                                                coupleRest=[ind,mot]
                                                if coupleRest[1] in (x[1]
for x in listeRestauration):
                                                        pass
                                                else:
listeRestauration.append(coupleRest)
                return listeRestauration


        def phrases_ordonees(self, phrases_sortie, texte ):
                phrases_sortie.sort( lambda s1, s2:
                        texte.find(s1) - texte.find(s2) )
                return phrases_sortie


        def resumer(self, texte, texteSeg, nb_phrases,loc):
                # on va pré-découper le texte en paragraphe, puis faire le
découpage en phrases
                para=self.paragraphes(texte)
                phrases_sortie = []
                plus_frequentes_graphies = self.mots_cles(texteSeg,7,loc)


                # segmenter en phrases : on utilise segments, les phrases
sans majuscules,
                # ensuite on revient à phrases, les phrases originales
                marqueur = nltk.data.load('tokenizers/punkt/french.pickle')
                for p in para:
                        phrases = marqueur.tokenize(p.strip())
                        segments = [phrase.lower() for phrase in phrases]
                        # parcourir la liste des graphies les plus
fréquentes et ajouter au résultat
                        # la première phrase du texte contenant la graphie
pour chacune d'elle
                        for graphie in plus_frequentes_graphies:
```

```
                                for i in range(0, len(segments)):
                                        if len(phrases_sortie) >=
nb_phrases: break
                                        if (graphie in segments[i] and
phrases[i] not in phrases_sortie):

phrases_sortie.append(phrases[i])
                                if len(phrases_sortie) >= nb_phrases: break
                # ranger les phrases sélectionnées dans l'ordre initial
                phrases_sortie = self.phrases_ordonees(phrases_sortie,
texte)
                # joindre les phrases de sortie, séparées par deux retour à
la ligne
                sortie = "\n\n".join(phrases_sortie)
                return sortie


        def concordances(self,texte,terme):
                para=self.paragraphes(texte)
                marqueur = nltk.data.load('tokenizers/punkt/french.pickle')
                concordance=""
                for p in para:
                        phrases = marqueur.tokenize(p.strip())
                        for phrase in phrases:

apparait=self.occurrences(phrase.lower(),terme.lower())
                                if apparait>0:
                                        concordance +="%s\n\n" %phrase
                return concordance


        def occurrences(self,texte,terme):
                i=0
                terme2=terme.lower()
                chaine=r"""(^|[ "\n\r\b('.\d])%s[ ,;.?!:/)\n\b\r\d$"']"""
%terme2
                i=sum(1 for oc in re.finditer(chaine,texte.lower(),
flags=re.MULTILINE))
```

```
                return i


        def triTermes(self,texte,ListeTermes):
                def comp(e1,e2):
                        c1=0
                        c2=0
                        chaine1=r"""(^|[ "\n\b\r\d('.])%s[
,;.?!:/)\n\b\d\r$"']""" %e1
                        chaine2=r"""(^|[ "\n\b\r\d('.])%s[
,;.?!:/)\n\b\d\r$"']""" %e2
                        for it in re.finditer(chaine1,texte,
flags=re.MULTILINE):
                                c1=it.start()
                                break
                        for it in re.finditer(chaine2,texte,
flags=re.MULTILINE):
                                c2=it.start()
                                break
                        if c1 < c2:
                                return -1
                        if c1 > c2:
                                return 1
                        else:
                                return 0
                ListeTermes.sort(cmp=comp)
                return ListeTermes


        def fichierExceptionTermes(self,loc):
                listeException = set()
                if loc=="defaut":
                        loc2="TermesInterdits"
                else:
                        loc2=loc+"\TermesInterdits"
                loc3=loc2+"/*.txt"
```

```
                if os.path.isdir(loc2):
                        fichiers=glob.glob(loc3)
                        for element in fichiers:
                                fichier = open(element,'r')
                                for ligne in fichier:
                                        if ligne[0]=="#":
                                                pass
                                        else:
                                                mot=ligne.replace('\n',"")
listeException.add(mot.decode('latin-1'))
                                fichier.close()
                return listeException

        def fichierExceptionCollocations(self,loc):
                listeException = set()
                if loc=="defaut":
                        loc2="CollocationsInterdites"
                else:
                        loc2=loc+"\CollocationsInterdites"
                loc3=loc2+"/*.txt"
                if os.path.isdir(loc2):
                        fichiers=glob.glob(loc3)
                        for element in fichiers:
                                fichier = open(element,'r')
                                for ligne in fichier:
                                        if ligne[0]=="#":
                                                pass
                                        else:
                                                mot=ligne.replace('\n',"")
listeException.add(mot.decode('latin-1'))
                                fichier.close()
                return listeException
```

```
def fichierReglage(self):
        parametres=[]
        fichier = open("Parametres.txt",'r')
        for ligne in fichier:
                if ligne[0]=="#":
                        pass
                else:
                        if ligne[0]=="@":
                                ligne2=ligne.replace("@","")
                                parametres.append(ligne2)
                        else:
                                parametres.append(int(ligne))
        fichier.close()
        return parametres
```

Références

Assemblée Nationale. LOI n°2005-102 du 11 février 2005 pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées, Pub. L. No. SANX0300217L (2005). France: JOURNAL OFFICIEL DE LA RÉPUBLIQUE FRANÇAISE.

Assemblée Nationale. LOI no 2005-380 du 23 avril 2005 d'orientation et de programme pour l'avenir de l'école (1), Pub. L. No. MENX0400282L (2005). France: JOURNAL OFFICIEL DE LA RÉPUBLIQUE FRANÇAISE.

Berthoz, A. (2013). *Le sens du mouvement*. Odile Jacob.

Bruet, J. (2013). Scorm, qu'est ce que c'est ? [http://www.e-doceo.net/blog/scorm-qu'est-ce-que-c'est/]

Claroline. (2008). Comment créer un contenu SCORM ? [http://doc.claroline.net/fr/index.php/Comment_cr%E9er_un_contenu_SCORM_%3F]

Coulon, E. (2002). *Les acouphènes ou l'impossible silence : étiologie, physiopathologie et tentatives de traitement*. Faculté mixte de médecine et de pharmacie de Rouen.

Deruy, C., & Ferrandino, S. (2011). Des ressources réutilisables et interopérables : SCORM. [http://c.deruy.ouvaton.org/exemples/moodle-scenarisationPedagogique/co/6-scorm.html]

Dubus, S. (2003). *L'oreille*. Roubaix. [http://www.infirmiers.com/pdf/oreille.pdf]

Half-Baked Software Inc. (2013). Hot Potatoes. Victoria, Canada: Half-Baked Software Inc. [https://hotpot.uvic.ca/]

Marieb, E. N., & Hoehn, K. (2010). *Anatomie et physiologie humaines* (8ème). Pearson.

Netter, F. H. (2009). *Atlas d'anatomie humaine* (4ème). Masson.

Roll, R., Kavounoudias, A., Escoffier, E., & Roll, J. P. (2003). Illusions posturales d'origine tactile plantaire chez l'homme. In B. Weber, P. Villeneuve, & P. M. Gagey (Eds.), *Pied, équilibre et traitements posturaux*. Paris: Masson.

Touzalin-Chretien, P. (2009). *Etude des liens entre les systèmes visuel et proprioceptif : approche électrophysiologique et comportementale chez le sujet sain et le patient amputé du membre supérieur*. Université de Strasbourg. [http://scd-theses.u-strasbg.fr/1722/01/TOUZALIN-CHRETIEN_Pascale_2009r_s.pdf]