MÉMOIRE DE MASTER

Présenté par
Marceau Hernandez et Didier Kouame

Master 1
« Langue et informatique »

Dirigé par :
Gaël Lejeune

Détection de Sauts stylistiques et de sauts qualitatifs dans les corpus - Quels rapprochements entre les deux tâches ?

Présenté et soutenu le 27 Juin 2023

Sorbonne Université
Faculté des Lettres

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

D'un format à un autre : Impact sur le contenu

L'essor du numérique permet de nos jours d'accéder à une infinité de contenus en ligne. Certains contenus sont même pensés avant tout pour être accédés et consultés depuis un terminal électronique, pour être partagés sur internet, ou pour être liés à d'autres œuvres numériques. D'autres, quant à eux, sont rendus numériques, c'est-à-dire consultables sur un terminal électronique. Cependant, ce processus peut être fait avec plus ou moins de soin, plus ou moins de détail, et les informations contenues dans le document d'origine peuvent, après numérisation, être difficiles d'accès ou complètement disparaître de la version numérique.

Documents textuels - Frontière entre le nativement numérique et le reste

Nous catégoriserons alors les documents textuels par deux classes distinctes, les documents nativement numériques et les documents numérisés.

Documents nativement numériques

Nous définissons comme nativement numériques les documents textuels numériques, c'est-à-dire consultables sur des terminaux électroniques, dans lesquels, le texte (en tant que chaîne de caractères) et ses propriétés, telles que la structure (titres de partie, paragraphes, etc.), mais aussi la police, la couleur, etc., sont entièrement représentées de façon textuelle. C'est par exemple le cas des fichiers aux formats *plain text*, JSON ou encore XML.

Dans ce type de documents, toute l'information relative au texte est donc encodée de manière textuelle.

Notons que dans cette définition, nous n'excluons pas la possibilité pour un document d'avoir du contenu en plus du texte, des images, de l'audio et d'autres éléments.

Nous considérerons que tant que la partie textuelle est nativement numérique, le document est un document textuel nativement numérique.

Avec cette définition, nous classons également les retranscriptions de textes non nativement numériques et qui répondent à ces critères : si un document textuel (ou sa partie textuelle) est entièrement encodé de manière textuelle, ce dernier est un document nativement numérique, qu'il soit issu ou non d'un document non nativement numérique. Évidemment, la retranscription d'un document se fait au travers de choix, c'est donc un autre document, différent, qui en résulte.

À noter que, dans cette définition, nous faisons volontairement abstraction de la manière d'encoder le texte.

Documents rendus numériques ou documents numérisés

Cette catégorie désigne tout simplement les documents textuels consultables sur support électronique dont au moins une partie des données textuelles ne répond pas aux critères d'un document nativement numérique. C'est par exemple le cas d'un document PDF dans lequel le texte n'est pas directement sélectionnable ou encore d'une image prise d'une page de texte. Le document numérisé se différencie ainsi du document nativement numérique dans lequel toutes les informations liées au texte figurent (qu'elles aient été réinterprétées depuis un document source ou non).

Ce qu'induit le support

Ainsi, il est important de noter que, selon notre définition, ces documents, peu importe s'ils ont pour origine un autre document, sont des documents à part entière. Cette distinction est faite car dans le cas où un document source existe, les informations contenues diffèrent forcément : on peut perdre des données (une image d'un texte), on peut en créer (ajout de méta-données), ou on peut modifier le contenu (on passe d'un glyphe à un caractère, dans une certaine police, ...). Ainsi, le support, qu'il soit numérique ou physique, influe le document, car il impose des contraintes et offre des possibilités qui diffère des autres supports.

Nous étudierons alors la question de l'accessibilité à l'information conte-

nue dans des documents non nativement numériques en considérant deux exemples : deux corpus de natures très différentes qui soulèvent cependant des problèmes similaires, l'information contenue dans le support original, non nativement numérique, s'altérant lors de la numérisation. Ainsi, tout au fil de ce mémoire, nous oscillerons entre ces deux corpus, afin d'aborder cette question sous deux angles :

Altération des données : Sauts qualitatifs

Afin de couvrir l'angle des sauts qualitatifs, nous utiliserons le corpus Antonomaz, collection de Mazarinades. Les Mazarinades, de courts textes imprimés du milieu 17^e siècle, sont alors rendus disponibles par numérisation, puis retranscrits à l'aide d'un outil d'OCR.

Cependant, ce processus de retranscription est imparfait, d'une part, car les outils d'OCR eux-mêmes sont imparfaits et d'autre part, car les documents sont très loin d'être parfaits (usure, pages fines, tâches, ...). Ainsi les textes, nativement numériques, obtenus en sortie comportent de nombreuses erreurs, réparties au sein du texte. Notre tâche sera alors de comprendre comment déterminer si un texte est bruité (contient des erreurs) et à quel point il l'est, voire de pouvoir localiser plus précisément les passages bruités (à l'échelle de la page par exemple).

Pour répondre à la question de l'accessibilité à la ressource au travers du prisme des sauts qualitatifs, nous nous poserons la question de l'impact du bruit, notamment sur les traitements automatiques, mais également sur la lecture de tels documents. Nous nous intéresserons également au coût de la correction manuelle de telles erreurs et à l'intérêt que présenterait une détection précise des erreurs.

L'objectif serait alors de pouvoir détecter les sauts stylistiques et de déterminer leur impact sur les usages des documents, que ce soit dans le cadre de l'utilisation d'outils de TAL (statistiques, détection de langue, date, ...) ou que ce soit pour une consultation des documents.

Des données textuelles difficiles d'accès : Sauts stylistiques

L'objectif de ce mémoire pour les sauts stylistiques est d'arriver à détecter les variations d'écriture ou d'expressions entre des structures textuelles, plus précisément repérer des changements parfois brusques qui peuvent se produire entre les sections d'articles scientifiques du corpus taln-archives, corpus qui regroupe des articles scientifiques en TAL rédigés de 1997 à 2022

des conférences TALN [1] et RECITAL [2].

Les sauts stylistiques pour ce travail couvrent deux aspects que nous avons tenté d'explorer : les changements ou variations stylistiques entre les sections d'un même article et les marques stylistiques permettant d'attribuer la rédaction d'une section à un auteur ou plusieurs s'il y a plusieurs auteurs qui ont rédigé l'article.

La détection de sauts stylistiques entre sections soulèvent des difficultés déjà connues des experts du TAL, Giguet et al. [2020] parlent des difficultés d'extraction de structure dans leurs travaux sur la *ToC* [3] extraction des documents anglais et français. Les documents en format *plain text* ou PDF ont des structures textuelles qui sont en apparence facilement accessibles lorsqu'on les observe visuellement mais difficilement accessibles lorsqu'on applique certaines tâches de TAL, typiquement celles explorées dans nos travaux.

Nous allons donc essayer de rapprocher dans notre travail les tâches afférentes à la détection de ces sauts pour lesquelles nos approches peuvent légèrement différer.

Conclusion

Ainsi, après avoir présenté nos corpus et les différentes méthodes explorées pour les étudier sous le prisme de la question de l'accessibilité à la ressource, nous présenterons les résultats succincts ces études. Par le suite, nous apporterons quelques éléments de conclusion, en essayant de généraliser les résultats de nos études à la question posée précédemment. Pour finir, nous présenterons les pistes d'amélioration et d'approfondissement qui nous semblent être les plus envisageables.

1. Traitement Automatique du Langage Naturel
2. Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues
3. Table of Content

Chapitre 2

Comment détecter les variations au sein d'un texte ou d'une collection de textes

2.1 Détection de variations qualitatives - Quand le texte se dégrade

L'objectif de cette partie est de montrer comment nous pouvons mesurer la qualité de la retranscription, créée automatiquement, d'un texte en employant différentes métriques. Il faut alors considérer deux grands types de mesures d'évaluation :

2.1.1 Comment mesurer le bruit - Évaluation supervisée

Les évaluations supervisées se basent sur une version correctement retranscrite du texte, qui fait office de référence. Pour la retranscription de textes, deux méthodes principales existent consistant à mesurer les erreurs en fonction de chaque mot (un mot est faux si un caractère ou plus ne correspond pas) ou alors en fonction de chaque caractère (mesure la distance entre le mot de la correction et celui de la retranscription). Ces deux mesures sont appelées *WER* (taux d'erreurs de mots) et *CER* (taux d'erreurs de caractères) leurs calculs, inspirés de la distance de Levenstein, sont alors très similaires [Kettunen and Koistinen 2019].

$$WER = \frac{S_m + D_m + I_m}{N_m}$$

$$CER = \frac{S_c + D_c + I_c}{N_c}$$

Où S représente le nombre de substitutions, D le nombre de délétions, I le nombre d'insertions et N le nombre total d'éléments dans la version de référence. $_m$ et $_c$ représentant l'unité mesurée (mot ou caractère). Par exemple D_m représente le nombre de mots retirés.

Ces méthodes offrent, *a priori* une grande fiabilité des résultats. Cependant, elles nécessitent cette fameuse version de référence (ou "correction"), c'est-à-dire une version manuellement retranscrite. Ce qui rend ces méthodes particulièrement coûteuses et quasiment inutilisables sur un grand corpus, c'est qu'il faut tout retranscrire à la main, ou bien corriger des retranscriptions automatiques. On se contente donc souvent de retranscrire des échantillons qui feront office de "sondage" de la qualité de la retranscription à l'échelle du corpus. Cependant, même pour un échantillon, une retranscription manuelle bien faite s'avère très coûteuse, souvent réalisée en double, par deux personnes distinctes afin de s'assurer de l'exactitude de cette dernière.

2.1.2 Comment mesurer le bruit - Évaluation non-supervisée

Les évaluations non-supervisées ne se basent pas sur une référence. À la place, elles se servent de méthodes détournées pour évaluer le texte. L'idée étant que d'autres caractéristiques, que l'on peut mesurer, permettent de déterminer si le texte est bruité ou non. Par exemple, si l'on sait que, généralement, nos mots ont une longueur comprise entre 3 et 8 caractères, un texte avec beaucoup de mots de longueur 15 est surprenant, on peut s'attendre alors a ce qu'il manque des espaces dans ce texte. Une évaluation de ce type se révèle alors bien moins coûteuse, elle ne nécessite plus la réalisation d'une version de référence. On peut alors aisément appliquer la méthode à l'ensemble du corpus et donc de ne plus se baser sur les résultats d'un échantillon. Cependant, il peut s'avérer compliqué de trouver les bonnes caractéristiques à mesurer, car elles nécessitent de pouvoir juger de la qualité d'un résultat sans connaître la valeur attendue. Dans notre cas, cela revient à savoir à quel point un texte est bien retranscrit sans savoir quels mots sont attendus, ni dans quel ordre ils le sont.

Avoir une idée précise de la qualité d'un texte nous offrirait plusieurs avantages :

— Pour un corpus que nous voudrions rendre disponible à la lecture, nous pourrions alors trier les documents par niveau de bruit, un document peu bruité se retrouvant dans les premiers proposés à l'utilisateur.

— Dans un contexte où les erreurs au sein de notre corpus seraient corrigées manuellement, on pourrait proposer les textes moyennement

bruités à la correction, les textes qui demanderaient peu de retour au texte, au lieu de faire corriger les textes très bruités qui demanderaient une post-édition manuelle complète. De plus, si l'on pouvait localiser précisément les passages à corriger, cela réduirait de nouveau la charge sur l'annotateur. Dans le cadre d'un projet collaboratif comme celui de l'ANR MATOS [1], pouvoir proposer des passages précis semble essentiel pour maximiser l'attention de l'annotateur. On peut également prendre l'exemple du mode de fonctionnement de *Rigor Mortis* [Fort et al. 2020] [2] où le processus d'annotation est transformé en jeu. Dans le cadre d'un jeu, pouvoir localiser plus précisément l'erreur ainsi qu'en évaluer le niveau de bruit pourraient grandement améliorer l'expérience de l'annotateur, avec, par exemple, des niveaux de difficulté.

Dans le cadre d'une correction humaine, il pourrait alors être intéressant d'être capables d'identifier la cause du bruit. Par exemple, si l'on a plusieurs mots faux d'une seule lettre, un humain pourrait facilement les corriger, sans retourner au texte, de même s'il manque une espace entre deux mots, ou s'il y a des caractères en trop au début ou à la fin de la page. De plus, selon la méthode, nous pourrions même déterminer quelle(s) partie(s) du texte est (sont) responsable(s) de sa mauvaise qualité globale.

— Dans le cas d'une utilisation plus automatique, connaître le niveau de bruit d'un texte ou d'une section permettrait d'adapter les analyses en conséquence. Selon le niveau de bruit, on pourrait alors exclure certaines analyses et favoriser celles qui, en dépit du bruit, donnent tout de même de bons résultats. Par exemple, nous pourrions déterminer la langue du texte, le dater [Abiven and Lejeune 2019], y trouver les entités nommées, découper le texte en fonction des thématiques abordées (*text-tiling* [Hearst 1997]). Tout en gardant en tête que les textes sont bruités, et que cela peut altérer nos résultats, certaines tâches seraient alors tout de même réalisables.

2.1.3 Bruit intermittent - Comment détecter des phénomènes locaux

Dans la plupart des cas, on peut remarquer que le bruit, lorsqu'il est très prononcé, n'est pas présent tout au long du document, mais plutôt localisé, il est intermittent. Il peut être présent uniquement sur certaines pages ou sur

1. Projet ANR MATOS : `https://postedition.anr-matos.fr`
2. Projet Rigor Mortis : `http://rigor-mortis.org`

certaines sections de pages. Ainsi, n'avoir qu'une métrique globale pourrait cacher certains cas d'erreurs localisées et placer dans la même catégorie les textes légèrement bruités tout du long et ceux auxquels il manquerait toute une page.

Il vient alors se poser la question d'une analyse plus fine du bruit, non plus à l'échelle du document, où la moyenne peut masquer bien des variations, mais plutôt à plus petite échelle, comme celle de la page, du paragraphe ou encore de la ligne. On pourrait alors imaginer pouvoir qualifier les pages bruitées, mais aussi analyser les sauts qualitatifs entre les pages ou sections. Cela permettrait alors de représenter au mieux ces évènements locaux, qui semblent représenter une grande proportion du bruit au sein de notre corpus. Dans *Daniel@FinTOC'2 Shared Task : Title Detection and Structure Extraction* [Giguet et al. 2020], les chercheurs explorent les questions de création et de détection de tables des matières. A cet égard, ils exploitent, d'une part, des indices extra-langagiers, tels que la mise en page, la police du texte, les passages soulignés, en gras, et cætera en récupérant ce qui est le plus saillant, ce qui se distingue du reste. D'autre part, ils exploitent également une approche textuelle, se servant alors de la longueur des passages du texte ou encore de la fréquence des n-grammes qui la compose (notamment pour la détection de tableaux).

On pourrait ainsi concevoir, sur les textes peu bruités, de pouvoir faciliter la navigation de l'utilisateur au sein de divers ouvrages. Ainsi, une approche permettant de catégoriser les sections d'un texte selon le niveau de bruit pourrait permettre de fournir des données supplémentaires aux outils de création ou de récupération de tables des matières dans l'optique d'améliorer ces traitements. Cette granularité des mesures semble alors être encore plus adaptée pour la détection de structures plus locales. On pourrait alors ne rechercher des listes (ordonnées ou non) [Giguet and Lejeune 2021] seulement si les pages ou sections sont jugées de qualité suffisante (par exemple à l'aide d'un seuil sur la mesure utilisée).

De même, dans l'article *Spatial Named Entity Recognition in Literary Texts : What is the Influence of OCR Noise ?* [Koudoro-Parfait et al. 2021], les auteurs se rendent compte de l'impact du bruit dans la transcription dur la détection d'entités nommées. Ils notent alors que certains types d'erreurs, notamment la disparition d'espaces, sont particulièrement problématiques. Ainsi, la problématique de pouvoir détecter plus précisément le type d'erreurs rencontrées se pose une nouvelle fois. Par exemple, si un document semble avoir beaucoup de fois des erreurs sur les espaces, le seuil de confiance pour relever une entité nommée dans ce texte pourrait être plus élevé.

2.1.4 Avantages d'une évaluation non-supervisée à l'échelle du corpus

Nous pouvons ainsi nous demander dans quels cas l'utilisation d'une ressource bruitée influe sur les résultats, et avec quelle intensité. Nous pouvons faire le parallèle avec la thèse de Jean-Baptiste Tanguy [Tanguy 2022] au sujet de la motivation de l'OCR, mais également de son cadre d'utilisation. L'auteur explore par ailleurs l'impact de l'analyse textuelle à partir de corpus bruités au travers de deux méthodes, décrivant ainsi l'inégalité des méthodes face au bruitage des données d'entrée.

Nous pouvons aussi concevoir une approche se servant de plusieurs outils d'OCR qui les évalueraient avec les mêmes métriques et pourraient ainsi déterminer, à l'échelle de la page, quelle retranscription produite est la plus "fidèle" au texte d'origine. Ainsi, on pourrait palier les manquements de différents OCR (pages de garde, pages sombres, pages très marquées par l'usure, ...) en n'utilisant que les parties de la retranscription dans lesquelles chaque outil est le meilleur.

2.1.4.1 Applications indirectes d'une évaluation par page

Au-delà de la détection d'erreurs au sein d'un corpus produit par OCR, on peut également se demander ce qui produit ces erreurs. Détecter les causes du bruit pourrait participer à la capacité d'estimer quand un OCR produira un mauvais résultat. Cela rejoint le point précédent, en effet, il peut être très coûteux d'utiliser simultanément plusieurs OCR par page. Le meilleur outil d'OCR en moyenne globale n'est en effet pas forcément le meilleur sur chacune des pages ou sur chacun des documents. Par exemple, un phénomène rare, comme une page de garde, pourrait être mieux retranscrit par un outil d'OCR particulièrement bien entraîné sur ce cas de figure, mais qui serait moins bon sur le cas plus classique d'une page comportant du texte parfaitement aligné. Le second cas étant plus fréquent, cet outil afficherait une qualité moyenne inférieure à un outil très mauvais sur les pages de garde, mais très bon sur les pages de texte.

Nous pourrions alors décider en amont du meilleur logiciel à utiliser selon les caractéristiques de notre page et ainsi diminuer fortement les ressources en temps et en énergie requises par l'approche précédente.

2.1.4.2 Reconnaissance de schémas d'erreurs

De plus, si nous sommes en mesure de déterminer plus précisément les schémas habituels qui entraînent une mauvaise reconnaissance ou des pro-

duits habituels d'une mauvaise reconnaissance : Par exemple, les caractères insérés en début ou fin de page ne seraient-ils pas séparables du reste de la page ? Si oui, pourrions-nous déterminer quand les retirer sans altérer le véritable contenu de la page ?

Par exemple, dans la capture d'écran en figure 2.1, page 9 de l'un des textes de notre corpus [3] , nous pouvons voir plusieurs erreurs fréquentes :

— Une à plusieurs séries de caractères ne correspondant à aucun mot de la page. Souvent présents aux extrémités des pages, comme c'est le cas du `2` sur cette page. Ces erreurs sont ainsi dues à une mauvaise reconnaissance des zones de texte sur la page, l'OCR se retrouve alors à chercher des caractères là où il n'y en a pas. Comme il fait ce qu'on lui demande, il en trouve souvent, nos textes n'étant pas particulièrement bien conservés, il y a souvent des tâches, des marques d'usures sur les pages, et d'autres artefacts qui ressemblent parfois, même pour un humain, à des lettres. Ces symboles vides de sens peuvent alors sembler faciles à détecter en fin de chaîne. Mais, en réalité, ils peuvent facilement porter à confusion : par exemple, le `9` sur la page ne correspond pas à ce problème, mais est simplement le numéro de la page ;
— Les codes d'échappement : Reliquats du format XML, ils sont une manière de représenter certains symboles dans ce langage ;
— Les problèmes d'espacement, certains espaces sont ajoutés et d'autres retirés. Dans notre capture d'écran, c'est par exemple le cas de `cemal`, une erreur de retranscription de `ce mal` ou `Ma zarin`, erreur de retranscription de `Mazarin`. Dans cet exemple précis, ces erreurs semblent être causées par la mauvaise qualité de la page, laissant apercevoir le texte de l'autre côté de la page, en filigrane.

3. *Avis sincère du mareschal de l'Hospital donné à sa Maiesté dans Sainct Denys avec les raisons pour lesquelles on l'a fait arrester en Cour*, exemple de texte pris du corpus des Mazarinades : `https://books.google.fr/books?id=KVCXy9OUe8UC`

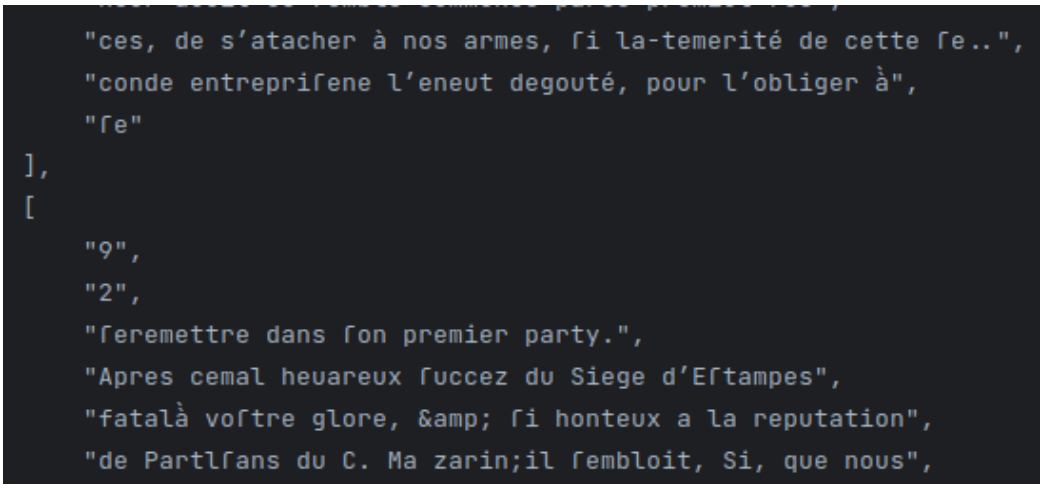


FIGURE 2.1 – Exemple d'erreurs fréquentes au sein d'un texte du corpus des Mazarinades.

2.2 Détection de variations stylistiques entre sections d'un document

Les tâches relatives à la détection de variations qualitatives pour le corpus Antonomaz nécessitent donc des métriques telles que *Word Error Rate (WER)* et le *Character Error Rate(CER)* pour déterminer les textes bruités et localiser de manière précise les passages bruités. Dans notre tâche de détection de changements stylistiques, nous utiliserons des mesures de similarité comme la distance cosinus pour identifier des variations de style éventuellement brusques des sections. Dans le cas de la problématique de mesure du bruit d'OCR, une page ne contient pas toujours un texte complètement linéaire. Il peut être nécessaire de détecter des notes de bas de pages ou qu'un bloc de texte est une citation. On pourra peut-être comparer les scores de similarité si on enlève les marques de ponctuation, par exemple pour déterminer l'intérêt de ce traitement.

La tâche de détection des sauts stylistiques est donc consacrée à l'analyse des changements ou variations de style dans les différentes sections d'un article scientifique. Un article scientifique peut, en effet, être rédigé par un ou plusieurs auteurs. L'idée de ce travail est donc de trouver ce qui relève du style d'écriture d'une personne. On appelle cela **auctorialité**. Pour cela, on peut déjà évoquer quelques traitements qui permettront de faire ressortir un style d'écriture particulier :

— La syntaxe : l'analyse par séquence des éléments de syntaxe [Baledent

and Lejeune 2019] dans une phrase. Les étiquettes morphosyntaxiques concourent à repérer les écarts linguistiques dans un texte.

— Des métriques d'évaluation qui permettent d'évaluer la distance entre les sections, c'est-à-dire à quel point deux sections sont similaires ou non.

— L'étude du taux de lexicalité, [Tanguy 2022] utilise la mesure du taux de lexicalité pour évaluer le ratio de mots correctement identifiés par un logiciel d'OCR, en les comparant à un lexique et [Abiven et al. 2021] qui l'utilisent également pour calculer, pour une page, la proportion de mots présents dans la sortie d'OCR appartenant au lexique LGeRMATILF [2017] (Souvay et Pierrel 2009). Le taux de lexicalité permettra de mesurer la richesse lexicale de chaque section ; et donc permettra de savoir si on a des variations en mesurant l'évolution des changements. On utilisera le GLÀFF [Sajous et al. 2013] [4] et le corpus de l'Est Républicain [5] [ATILF and CLLE 2020].

— La stylométrie, consistant à faire ressortir des traits particuliers d'un auteur avec des statistiques sur la ponctuation, la longueur des mots ou des phrases.

Le nombre d'auteurs joue également dans la détection de la variation du style. Plus on aura d'auteurs, plus on aura des styles différents. Un autre aspect pour la détection de style est lié au fait qu'un document peut être une agrégation de différents genres. Par exemple les articles de presse récupérés par Web Scraping peuvent contenir non seulement les écrits d'un auteur mais aussi d'autres éléments qui ne sont pas écrits par lui qui peuvent être des commentaires des internautes. Également pour nos deux corpus contenant des articles scientifiques de la conférence TALN qui inclus ceux de RECITAL, il peut aussi utile de détecter le style de rédaction des auteurs étant donné que nous sommes dans le domaine du TAL. Il y a peut-être une manière de s'exprimer ou de rédiger les articles qui est spécifique soit au domaine, soit aux écrits de TALN qui regroupe les articles des experts du TAL et de RECITAL qui regroupe le plus souvent les articles des jeunes étudiants. Cette détection de style peut se faire par années pour analyser la variation de style non plus entre sections mais entre documents.

2.2.1 Problèmes liés à l'extraction textuelle

Le corpus sur lequel est menée l'analyse des sauts stylistiques est l'archive TALN qui contient des articles scientifiques des conférences TALN et RECI-

4. http://redac.univ-tlse2.fr/lexiques/glaff.html
5. http://redac.univ-tlse2.fr/corpus/estRepublicain.html

TAL, tenues de 1997 à 2022 [6] est constitué d'articles en format `PDF`. C'est un langage de description de pages qui est uniquement utilisé pour afficher et visualiser des documents numériques. Bien qu'il soit possible de travailler avec ce format de document, il n'est pas adapté pour faire des analyses sur corpus, car la plupart des outils sont conçus pour traiter des chaînes de caractères. De plus, traiter des `PDF` est potentiellement gourmand en ressources, notamment parce que ce format conserve tel quel les modifications de polices (gras, italique...) faites par l'auteur du document. À l'opposé, un document en `TXT`, parfois nommé document en texte brut, demande moins de ressources et est plus facile à traiter. Pour exploiter notre corpus, on extrait donc les informations à l'aide de cet outil *pdftotext*, GROUIN [2014] utilise cet outil pour l'extraction de textes sur des articles également.

Dans notre cas, les articles scientifiques comportent des tableaux, des figures, des formules mathématiques et bien d'autres. L'outil d'extraction mentionné n'est pas sans faille sur ce type de données. Certains documents du corpus ont une structure riche, voire complexe, et c'est à ce niveau que pdftotext montre ses limites. Nous rencontrons les mêmes problèmes d'extraction avec cet outil que GROUIN [2014]. Lors de la conversion, la structure des documents se retrouve modifiée, il y a des problèmes de codage et des phrases qui se retrouvent sur plusieurs lignes. Mais l'idée ici sera d'utiliser un outil approprié qui permettra d'extraire de manière structurées les informations, plus précisément de découper en sections.

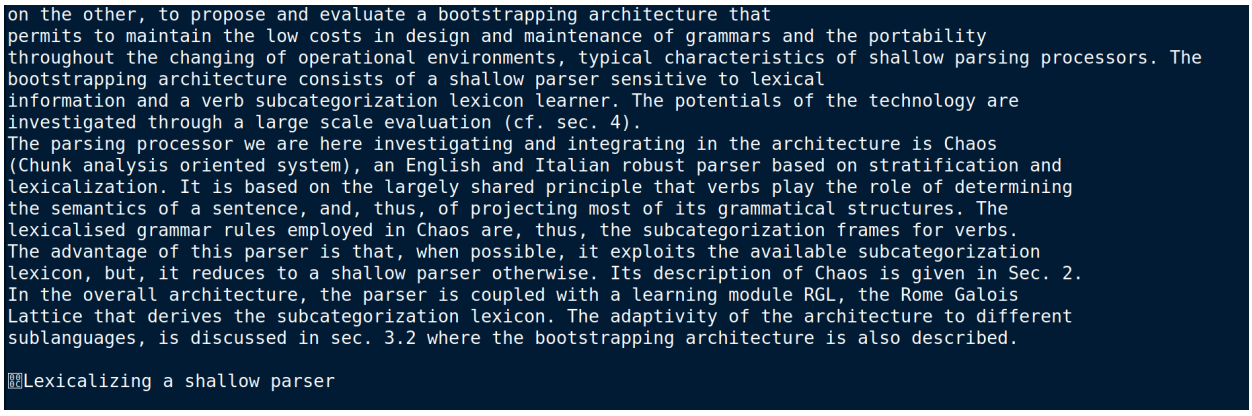


FIGURE 2.2 – Exemples d'erreurs de conversion de l'outil pdftotext

6. Dépôt Gitlab : `https://gitlab.com/parmenti/taln-archives/`

2.2.2 Les difficultés liées à la séparation en paragraphes

L'une des premières difficultés relative à l'étude du style entre les différentes sections d'un corpus est celle de l'exploitation du corpus. Pouvoir séparer de manière précise les différentes sections d'un article scientifique et ensuite comparer leur similarité ou dissimilarité est une tâche difficile. Il existe à notre connaissance peu de travaux en informatique sur les sauts stylistiques en ce qui concerne la détection de changement de style entre sections ; toutefois des travaux du même genre ont été faits sur des écrits d'écrivains.

2.2.3 Le bruit dans l'extraction de contenu non structuré

À partir de la récurrence de ces motifs, on peut être en mesure de détecter un changement de style.

Étudier les particularités d'écrits de différents auteurs est abordable comme tâche. Mais, le faire entre différentes sections d'un article scientifique requiert d'abord de segmenter l'article en sections.

Avant d'en arriver à une telle qualité de séparation textuelle, on commence par séparer l'article selon le double saut de ligne `\n\n` puis selon un seuil de caractères, par exemple 2000. Ce nombre est donc le nombre de caractères de chaque paragraphe à comparer. Le problème avec cette méthode est qu'elle est imprécise parce qu'elle ne découpe pas selon les sections, mais selon un flot de caractère spécifié dans chaque paragraphe.

Chapitre 3

État de l'art : Comment mesurer le bruit et la similarité et exploiter la ressource en conséquence

3.1 Évaluation de l'OCR

3.1.1 Métriques d'évaluation supervisée

Dans l'article *Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER)* [Leung 2021], l'auteur nous présente les deux méthodes les plus traditionnelles d'évaluation des sorties d'OCR : le taux d'erreur au mot (WER) et le taux d'erreur au caractère (CER). On comprend alors les différences entre les deux mesures, aussi bien en termes de granularité que de coût de calcul. Cependant, ces mesures sont supervisées, c'est-à-dire qu'elles nécessitent une vérité de terrain (*ground-truth*), une version retranscrite manuellement, afin de permettre l'évaluation de l'OCR. Ces approches sont donc très intéressantes, à condition de disposer des ressources nécessaires en annotateurs, en temps, mais aussi d'avoir accès au matériau de base (sans ce dernier, il peut être compliqué de réaliser une retranscription).

Un bon exemple de l'utilisation de ces métriques est l'article *OCR performance prediction using cross-OCR alignment* [Ben Salah et al. 2015], dans cet article, les auteurs, en exploitant des mesures supervisées, présentent l'intérêt de réaliser une analyse de la qualité des sorties d'OCR. En effet, les taux de confiance fournis par les OCR ont tendance à sur-estimer la qualité de la retranscription. Et, cela semble devenir de plus en plus prononcé à mesure

que la qualité de la retranscription ne se dégrade. Par la suite, les auteurs emploient des méthodes supervisées d'évaluation de sorties d'OCR pour entraîner un modèle d'évaluation. Développer un tel modèle permettrait alors de ne plus avoir à recourir à une vérité de terrain, et ainsi s'affranchir de la nécessité d'annotation manuelle.

3.1.2 Métriques d'évaluation non-supervisée

Dans sa thèse, Tanguy [Tanguy 2020] cherche alors une méthode pour comparer les sorties d'OCR, sans avoir à recourir à une vérité de terrain, qui est le plus souvent absente. Il approfondit alors la question de l'évaluation non-supervisée abordée par Ben Salah et al. 2015 Sur le corpus des Mazarinades[1], corpus de textes produits en plein essor de la presse en France, que nous vous présenterons au sein du prochain chapitre, il compare alors les résultats de deux outils de reconnaissance optique de caractères : *Kraken*[2] et *Tesseract*[3] [Smith 2011].

Il décide alors de s'attaquer au problème en utilisant des modèles de langue. Pour ce faire, il établit une vérité de terrain sur de brefs passages afin de comparer les résultats du modèle créé avec des méthodes supervisées (CER en l'occurrence). Afin d'estimer la qualité de la sortie d'un OCR à partir du modèle de langue, il va ainsi se fier à la probabilité estimée de la production des séquences de caractères de la retranscription.

Cependant, en vue de la taille relativement limitée du corpus exploité, l'emploi d'un modèle de langue pour qualifier la qualité de la retranscription semble alors infructueux : aucun modèle de qualité ne semble pré-exister sur l'état de langue étudié, et malheureusement, trop peu de données sont disponibles pour obtenir un modèle de la fiabilité requise.

Vers une utilisation de ressources bruitées pour l'évaluation

Cependant, la réalisation d'un modèle de langue n'est pas la seule piste pour l'élaboration d'un outil d'évaluation non-supervisée. Une autre piste, requérant moins de données en entrée est explorée par Abiven et al. [2021]. Ces derniers approfondissent, dans un premier temps, l'utilisation de méthodes non-supervisées pour l'évaluation de l'OCR. Ils se tournent alors vers l'utilisation du taux de lexicalité avec une ressource lexicale que nous présenterons plus tard, le *LGeRM*. Ceci en combinaison avec les taux de confiance

1. Lien du compte GitHub Antonomaz : `https://github.com/Antonomaz`
2. Lien du dépôt GitHub Kraken : `https://github.com/mittagessen/kraken`
3. Lien du dépôt GitHub Tesseract `https://github.com/tesseract-ocr/tesseract`

fournis par le logiciel d'OCR, qui, comme nous l'avons vu, peuvent se révéler biaisés.

Puis, dans un second temps, les auteurs analysent les conséquences du bruit au sein d'un corpus sur diverses tâches de textométrie. Les auteurs abordent également l'intérêt de l'OCRisation. De tels ouvrages, alors que les mazarinades sont un sujet assez connu des spécialistes, leur faible accessibilité les rend assez difficilement exploitables, et donc peu exploités. C'est ainsi que la retranscription de ces textes, même imparfaite, permet dans certains cas une analyse jusqu'ici impossible à réaliser. Ils dénotent également, à l'aide de quelques retours au matériau d'origine, les obstacles que présente ce corpus. En évoquant notamment les obstacles dus à une mauvaise qualité du papier sur lequel est imprimé le document océrisé, provoquant par exemple un effet de transparence, nous laissant ainsi apercevoir, en filigrane, le texte du verso de la page.

En résumé, cet article présente une approche complète pour l'étude des mazarinades burlesques de la Fronde, en mettant l'accent sur la délimitation du corpus, l'acquisition des données, leur analyse statistique et algorithmique, ainsi que l'interprétation des résultats. Nous essaierons alors d'avoir une approche plus centrée sur la mesure de la qualité en essayant de compléter la mesure par taux de lexicalité, tout en introduisant de nouvelles mesures.

3.1.3 Utilisation de sorties bruitées

K. Abiven et G. Lejeune [2019] s'intéressent particulièrement à la définition de "corpus" des mazarinades. Ils mettent ainsi l'accent sur le caractère très varié des textes qui le composent. Puis, sur la sous-partie de ce "corpus" retranscrite par OCR, ils essaient de l'exploiter en l'état, c'est-à-dire avec les erreurs de retranscriptions. Ainsi, ils développent une méthode de datation automatique des textes à l'aide d'une approche basée sur les caractères. En utilisant des paramètres de support minimal et maximal, c'est-à-dire le nombre minimal et maximal de documents comportant la suite de caractères, ce qui leur permet de négliger l'impact du bruit (les formes bruités ont peu de chance de se retrouver dans un grand nombre de textes), ils arrivent à obtenir des résultats intéressants : En fonction du sous-corpus, ils arrivent à obtenir, dans la plupart des cas, de meilleurs résultats que les approches basées sur une approche en mots.

À l'opposé, [Chiron et al. 2017] soulèvent l'influence des erreurs de retranscription pour la recherche plein texte, en insistant sur l'impact exacerbé de ces erreurs sur les mots les moins fréquents, ces derniers étant généralement les mots à plus forte valeur sémantique [Spärck Jones 1972]. Une sortie bruitée, même sur quelques pages, pourrait alors profondément changer l'in-

dexation du document. Par exemple, si une partie est mal retranscrite, le champ lexical d'un thème entier pourrait alors être manquant à l'index du document.

Synthèse

Ainsi, l'approche de cette partie sera de s'interroger sur l'utilisation des métriques d'évaluation non-supervisée : Approfondir certaines approches précédemment utilisées, comme le taux de lexicalité et essayer de les lier à de nouvelles mesures dans l'espoir d'être capables de localiser plus finement les problèmes de transcription. Nous essaierons également d'explorer la cause de ces erreurs, serait-il possible de prédire certaines erreurs ? Et, dans le cas où elles correspondent à un schéma précis, pourrions-nous les corriger ?

3.2 Calcul de la similarité et analyse stylistique textuelle

Le calcul de la similarité textuelle consiste à comparer des textes ou des documents dans le but de voir s'il y a des rapprochements possibles ou pas entre eux, c'est-à-dire que l'on procède par comparaison de mots, de phrases ou de caractères entre deux textes, par exemple pour repérer des particularités partagées. Dans *Comparaison de textes : quelques approches...* [Negre 2013], Negre pose le contexte de l'analyse de la similarité entre textes. Nous avons par exemple un seul texte dont on souhaite comparer le contenu avec d'autres textes. Lorsqu'on compare ce texte avec d'autres textes, on attend en sortie d'avoir les textes qui ont une similarité proche de 1 avec le texte de référence. Aussi, la similarité textuelle peut être généralement calculée lexicalement ou sémantiquement [Gomaa et al. 2013] ; lexicalement correspond à comparer les chaînes de caractères tandis que sémantiquement correspond à la comparaison de la similarité des unités selon leur sens ou selon leur contexte d'apparition.

3.2.1 Méthode non supervisée de calcul de la similarité textuelle

Dans la thèse de Jean-Baptiste Tanguy [Tanguy 2022], on utilise une méthode non-supervisée pour le calcul de similarité entre des parties de pages dans le corpus des mazarinades. Il utilise l'alignement textuel ; aligner des textes, si l'on part du principe que "S1 et S2 sont des ensembles de phrases",

c'est ainsi *trouver un sous ensemble du produit cartésien S1xS2*, [Kraif 1999]. Il s'agit donc du repérage dans deux ensembles différents des éléments qui sont repris dans lesdits ensembles.

Pour rapprocher donc des pages de même type et identifier des passages avec des reprises, Tanguy procède au calcul de la similarité entre textes : méthode qu'on utilisera également pour établir la similarité entre les sections d'un article scientifique. Il commence d'abord par vectoriser le contenu des pages puis calcule la similarité avec des distances. L'utilisation de ces deux méthodes va lui permettre de trouver des pages contenant des reprises ou pas. En ce qui concerne la vectorisation, il propose de vectoriser de manière "creuse" [4] ou "dense" [5].

3.2.2 L'approche sémantique dans le calcul de la similarité textuelle

La prise en considération de la sémantique dans le calcul de la similarité entre des textes est une deuxième technique au calcul de similarité classique, c'est-à-dire à la comparaison entre caractères ou mots vectorisés dans les textes. La manière la plus populaire de calculer ces similarités aujourd'hui prend son origine dans les travaux de chercheurs de Google [Mikolov et al. 2013] qui sont à l'origine de cette technique qu'on appelle aussi plongements lexicaux. Elle passe par l'apprentissage *d'embeddings* (plongements) qui sont de vecteurs denses. Les *embeddings* étant des vecteurs, on peut désormais rapprocher ces vecteurs par des calculs que de simplement calculer la distance entre deux vecteurs avec une distance cosinus ou une autre. Un usage de cette technique se retrouve dans l'outil *Google translate*. L'outil utilise en effet un plongement multilingue dans lequel on retrouve un **espace sémantique** pour les mots dans les deux langues. L'idée est en effet de capturer les relations sémantiques entre les vecteurs et de les rapprocher en conséquence.

Les vecteurs proches ou similaires vont apparaître très souvent dans les mêmes contextes d'énonciation. Ce qui va permettre de les mettre sur le même plan au niveau du sens.

Toujours dans son travail, Tanguy [Tanguy 2022] utilise cette méthode pour chercher les vecteurs les plus proches avec la distance cosinus appliquée à des mots. La particularité cette fois-ci est qu'en plus d'analyser les vecteurs en fonction de leur signification, les mots à analyser sont basés sur le corpus étudié avec une approche dite *corpus based*. L'application de la technique de la représentation sémantique sur ce corpus est une tâche intéressante.

4. Matrice qui contient beaucoup de zéros, des éléments nuls
5. Matrice où l'on regroupe les colonnes pour limiter le nombre de zéros

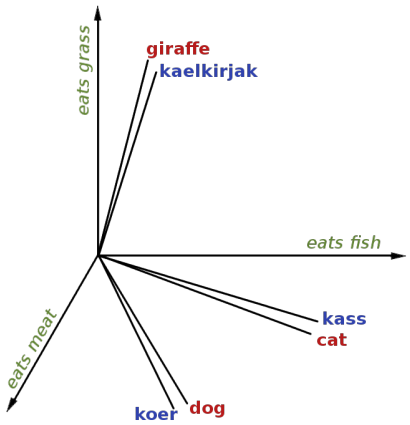


FIGURE 3.1 – Exemple de vectorisation sémantique

Source : `https://www.marekrei.com/blog/multilingual-semantic-models`

en effet, on a l'habitude d'utiliser ces plongements lexicaux sur des corpus contemporains, alors que dans le cadre de sa thèse, on est en face d'un corpus ancien du 17e siècle, en réalité des imprimés de 5 000 entrées, liste établie notamment par le travail de Célestin Moreau [1851].

3.2.3 Analyse comparative des schémas syntaxiques récurrents pour l'analyse stylistique

Les schémas syntaxiques récurrents que l'on appelle aussi *motifs syntaxiques* sont des séquences d'éléments de langue qui permettent de trouver des structures linguistiques du type étiquettes morphosyntaxiques. Dan Kraif and Tutin [2017] les auteurs citent André Salem [1987] qui utilise une méthode dite de segments répétés mais aussi [Granger et al. 2008] dans [Disentangling the phraseological web] qui appellent ces segments répétés des *n-grams* ou *clusters*. Pour notre travail de détection de motifs syntaxiques, il s'agit de n-grammes de longueur 2, 3 voire 5. L'idée avec cette tâche est de trouver donc des séquences d'éléments d'une section qui sont reprises dans la section suivante pour tous les articles du corpus. Cette tâche offre une perspective de poursuite intéressante ; si l'on trouve des intersections de n-grammes entre sections pour un article en particulier, on peut aller chercher ce qui se *cache* derrière ces étiquettes et/ou aller chercher ces étiquettes dans les autres articles du corpus pour éventuellement repérer les particu-

larités d'expression d'un auteur ou des particularités d'expression générale, mais qui sont récurrentes dans certains articles. Cela peut nécessiter de faire en amont une tâche de détection d'entités nommées pour trouver le ou les auteurs des motifs trouvés. Aussi, les travaux de Anaëlle Baledent et Gaël Lejeune dans Baledent and Lejeune [2019] se sont révélées inspirantes pour *l'analyse stylistique* en ce qui concerne les sections d'articles scientifiques. Les auteurs travaillent en effet sur des corpus de Dumas qui contiennent 9 œuvres pour 1 267 651 tokens avec un contre-corpus de Féval de 1 236 781 tokens. Ils utilisent une méthode de *clustering*, donc une tâche non supervisée pour regrouper les données pour calculer les motifs syntaxiques avec contraintes sur la longueur : 1 <= minimum(len(motif)) <= 5 et 1 <= maximum(len(motif)) <= 5. On ne peut pas en effet travailler sur des longueurs plus petites parce qu'on n'obtiendrait pas de bons résultats ou simplement, on aurait beaucoup de bruits étant donné que les bigrammes et trigrammes sont les éléments les plus fréquents dans la langue.

Ils obtiennent donc des résultats pertinents du point de vue des particularités stylistiques des deux auteurs. Dans les figures 3.2 et 3.3 ci-dessous, on peut voir que certains motifs qui ont donc une longueur définie sont propres à chaque auteur, on peut le voir en regardant le taux de croissance ou *Growth Rate* de chaque motif. On peut voir également que le *clustering* hiérarchique selon le lexique se révèle efficace pour rapprocher les œuvres de chaque auteur.

Motif	Taux Croissance	#Dumas	#Feval
NC_CL_CL_V_DET	0.3013	47	156
PUNC_V_NPP_PROREL	**0.3313**	55	166
CL_CL_V_DET	0.3951	162	410
DET_NC_CL_CL_V	0.4012	67	167
NC_CL_CL_V	0.4036	111	275
...			
P_NC_P_NC	0.9822	2542	2588
NC_P_DET_NC	0.9849	10543	10705
P_NC_P_DET_NC	0.9849	1825	1853
V_P_DET_NC_P	0.9874	1566	1586
DET_NC_CC_DET_NC	0.9918	1455	1467
DET_NC_PROREL_V	1.0025	2375	2369
DET_NC_ADJ_ADJ	1.003	1007	1004
NC_P_DET_NC_P	1.0062	2287	2273
V_P_DET_NC	**1.013**	6071	5993
...			
PUNC_DET_NPP_PUNC	3.6667	187	51
NC_P_PROREL_CLS	3.9324	291	74
NC_P_PROREL_CLS_V	**3.9355**	244	62
DET_NC_P_PROREL_CLS	4.641	181	39
DET_I_PUNC_DET	5.9833	359	60

FIGURE 3.2 – Motifs classés par taux de croissance - Anaëlle Baledent et Gaël Lejeune

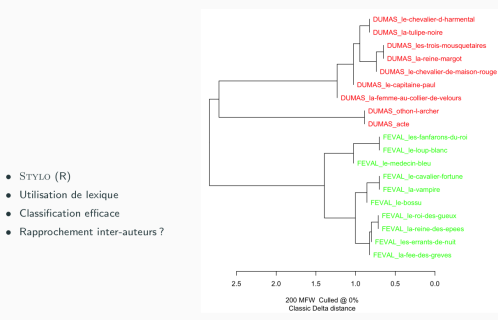


FIGURE 3.3 – *Clustering* hiérarchique (lexique) - Anaëlle Baledent et Gaël Lejeune

Deux autres travaux incontournables en ce qui concerne les motifs syntaxiques ou *syntactic patterns* sont ceux de Jean-Gabriel Ganascia dans Ga-

nascia [2001] et Audras and Ganascia [2005]. Il a, en effet, créé un outil **d'extraction de motifs syntaxiques** qui s'appelle le **Littératron** qu'il a réalisé au LIP 6. L'intérêt de cet outil est triple :

- Dans le domaine de la didactique des langues : supprimer les barrières linguistiques grâce à l'informatique
- Dans le domaine de la linguistique computationnelle : utile pour les applications en informatique appliquée à la linguistique
- L'enseignement assisté par ordinateur

Un bref aperçu de la technique d'extraction de motifs syntaxiques de Ganascia

Il utilise deux outils, la première est un "un analyseur [6] morphosyntaxique du français [Vergne and Giguet 1998] qui construit un arbre syntaxique à partir de productions écrites" est un algorithme de calcul de dépendances . L'analyseur prend un texte en entrée et le divise en groupes de mots non récursifs, c'est-à-dire que les groupes de mots sont combinés de manière linéaire et donc ne contiennent pas des sous-groupes de mots qui se répètent. Les sorties de cet analyseur sont transformées en ASO (Arbres Stratifiés Ordonnés) ou SOT (Stratified Ordred Trees). L'ASO est un arbre dans lequel l'ordre de gauche à droite entre les frères et sœurs est significatif. Toutes les données séquentielles peuvent évidemment être représentées avec un arbre ordonné de profondeur 1 [Ganascia 2001]. L'intérêt de l'ordonnancement et de la stratification de ces arbres est que les arbres ordonnés augmentent "la puissance de la représentation" dit-il, laquelle représentation permet de détecter des sous-arbres similaires et donc de détecter des motifs généraux qui ont plusieurs occurrences approximatives : *"Ordered trees increase representation power and it is possible to detect similar sub- trees, with respect to this data organization. It is also possible to extract general patterns that have multiple approximate occurrences. For instance, any specific recurrent sequence of syntactical groups extracted from a parsing tree may be detected without considering the corresponding words or their categories."*

6. Analyseur au départ créé par Jacques Vergne puis modifié par le Groupe Syntaxe [7] du GREYC [8])

Pour résumer, nous avons dessiné (voir figure 1) l'ensemble de la chaîne de traitement qui prend, en entrée, un texte en langage naturel et qui engendre, en sortie, un ensemble de motifs récurrents.

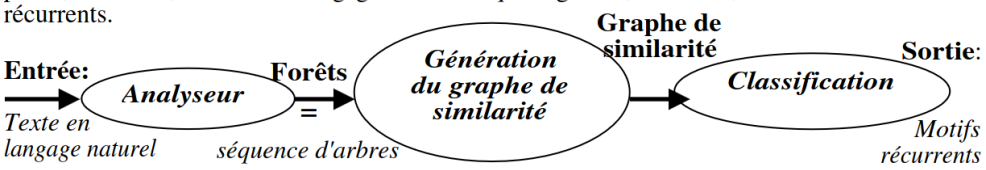


Figure 1 : chaîne de traitement

Dans la première sous-partie, les entrées du processus ainsi que l'étape d'analyse sont présentées, tandis que, dans la seconde, ce sont les ASO (Arbres Structurés Ordonnés) qui sont détaillés, car l'algorithme d'extraction repose en grande partie sur l'utilisation de ces structures de données.

FIGURE 3.4 – Chaine de traitement de l'analyseur linéaire Vergne et dégagement des motifs récurrents par le Littératron - Article TALN 2001, Tours, 2-5 juillet 2001 : Analyses comparatives de motifs syntaxiques de francophones et d'apprenants du français arabophones, à l'aide d'outils d'extraction automatique du langage - Isabelle Audras et Jean-Gabriel Ganascia

3.2.4 Des méthodes avancées de la stylométrie : le problème d'Attribution d'Auteur ou *Writeprints*

Selon le TLFi [ATILF 2002] la stylométrie est *la science qui utilise les statistiques pour l'étude du style*. Par statistiques, on entend l'analyse quantitative des données linguistiques. Par exemple, pour appliquer des techniques de stylométrie à une section de document, on peut suivre ces étapes :

- Découper le texte en mots
- Compter le nombre de phrases
- Compter le nombre de mots
- Compter le nombre de caractères
- Calculer la moyenne des mots
- Calculer la moyenne des phrases

Une fois que l'on a ces caractéristiques, on pourra être en mesure dans un premier temps de faire ressortir les caractéristiques d'une section ou d'un texte. Chaque section ayant donc ses caractéristiques, on peut alors, dans un second temps, mener une étude comparative sur cette base.

Ce type d'analyse existe depuis la fin du XIX^{e} siècle, on considère souvent qu'elle a été créée par Wincenty Lutosławski (1863–1954) [1898].

La langue peut être définie comme la somme des moyens d'expressions dont nous disposons pour mettre en forme l'énoncé, tandis que le style est l'aspect et la qualité qui résultent du choix entre ces moyens d'expression. La somme des moyens d'expressions dont nous disposons contient les éléments d'énonciations tel que "je" qui est une expression apparemment immédiate

de point de vue (PDV) et auquel à chaque changement de point de vue correspond un nouvel **énonciateur**, ou encore la syntaxe des phrases et aussi les motifs syntaxiques. Ces moyens d'expressions forment donc une donnée individuelle qui peut être trouvée par des méthodes statistiques.

La tâche d'Attribution d'Auteur est une sous-tâche de la stylométrie ; *"L'hypothèse sous-jacente est qu'un auteur laisse involontairement dans son message textuel des indices qui peuvent mener à son identification"* [Brixtel et al. 2015] dans *Attribution d'Auteur : approche multilingue fondée sur les répétitions maximales*. Ces travaux apportent un plus à l'attribution d'auteur en contexte multilingue avec une alternative aux n-grammes de caractères par des *répétitions maximales*. Par *répétitions maximales*, on entend des motifs qui apparaissent plusieurs fois dans un texte et qui se répètent plusieurs fois de manière non consécutive. Les n-grammes étant des séquences consécutives de caractères, les auteurs tentent donc cette approche des *répétitions maximales*. Cela permet de réduire les données sur lesquelles on travaille, on a donc un gain de coût et de temps en termes de performance et l'on gagne en précision d'Attribution d'Auteur. Comme détaillé plus haut, les traits stylistiques personnels [Koppel et al. 2009] sont extraits grâce à des mesures simples telles que la longueur des mots, des suites de mots et bien d'autres. Dans une liste de traits d'Attribution d'Auteur définies par Abbasi and Chen [2008], l'approche par n-grammes de caractères semble être la plus adaptée à toutes les langues, mais tout en contrôlant la valeur des n du fait des particularités morphologiques différentes des langues.

Dans notre travail de détection de particularités entre sections, nous avons travaillé sur n-grammes de mots et d'étiquettes morphosyntaxiques, des tâches auxquelles les auteurs proposent donc leur approche des n-grammes de caractères : *"Les expérimentations menées dans cet article soulignent les caractéristiques principales des méthodes d'AA fondées sur les chaînes de caractères (en opposition aux mots ainsi qu'à l'exploitation de leurs étiquettes morphosyntaxiques)."*

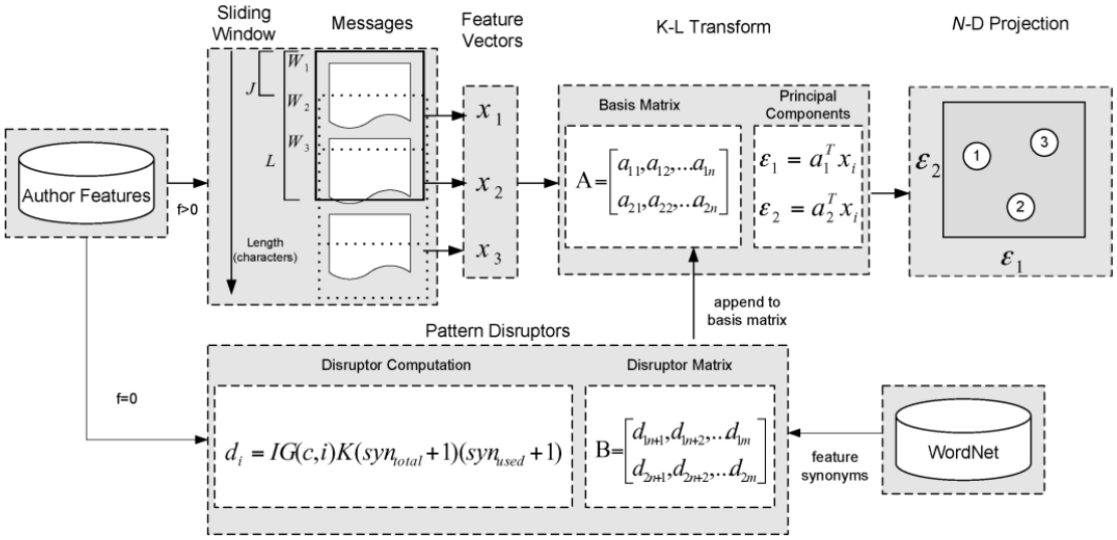


Fig. 3. Writeprints creation illustration.

Figure: Writeprints system design overview

FIGURE 3.5 – Un aperçu d'une chaîne de traitement de l'Attribution d'Auteurs - [Abbasi and Chen 2008]

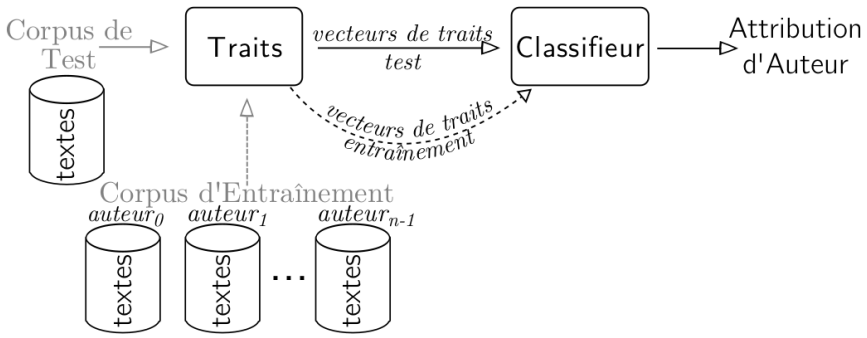


FIGURE 1 – Chaîne de traitement utilisée pour l'Attribution d'Auteur.

FIGURE 3.6 – Chaîne de traitement - TALN2015a - R. Brixtel, C. Lecluze, G. Lejeune

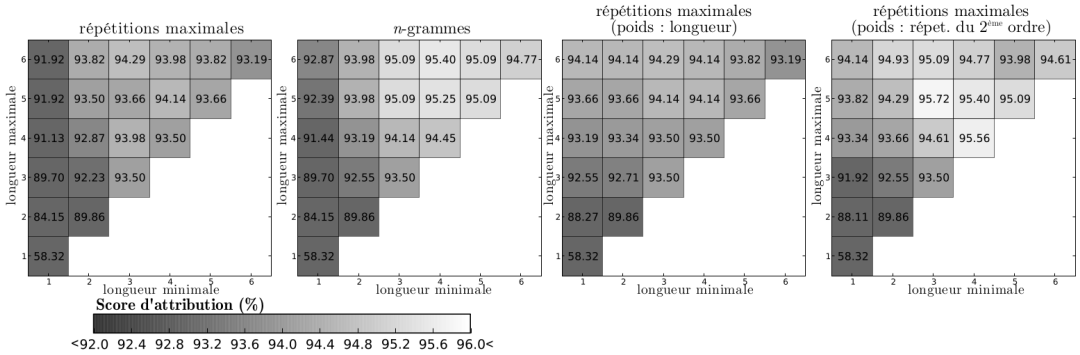


FIGURE 5 – Score d'attribution sur le corpus EBG-40.

FIGURE 3.7 – Score d'attribution d'auteur pour le corpus anglais EBG - TALN2015a - R. Brixtel, C. Lecluze, G. Lejeune

Résultats de leur approche [Brixtel et al. 2015]

Ils travaillent donc sur deux corpus : un corpus de textes en anglais EBG [9] [Brennan et al. 2012] et un corpus en français LIB [10] Ils ont considéré les répétitions maximales de deux façons :

— Pondérés par leur longueur (w_{len})

— Par les répétitions maximales du 2ème ordre (w_{2nd}) sans pondération

Les scores d'attributions pour le corpus EBG sont consultables dans la figure 3.7 :

Pour obtenir les meilleurs paramètres de longueur des motifs, ils calculent la moyenne des scores d'attribution de chaque matrice pour chaque intervalle de longueurs [*min, max*]. Ils obtiennent donc de meilleurs paramètres avec les **répétitions maximales du 2ème ordre (w_{2nd})**.

La table 3.8 montre les scores moyens selon chaque trait, le meilleur paramètre qui sort du lot est le motif du second ordre ***motifs*$_{2nd}$**.

9. EXTENDED BRENNAN GREENSTADT adversarial corpus (Brennan et al., 2012)
10. Corpus d'articles de presse.

	meilleurs paramètres de longueur $[min, max]$	score moyen
n-grammes	$[4, 6]$	84,61%
$motifs$	$[4, 6]$	83,69%
$motifs_{len}$	$[4, 6]$	83,88%
$motifs_{2nd}$	$[4, 5]$	**85,39%**

TABLE 4 – Meilleurs paramètres en fonction du score moyen sur les corpus LIB-40, EBG-40 et MIXT-80.

FIGURE 3.8 – Meilleurs paramètres en fonction du score moyen - TALN2015a - R. Brixtel, C. Lecluze, G. Lejeune

Enfin, ils travaillent aussi sur l'évolution de la qualité en fonction du nombre de traits et d'auteurs. la tendance générale observée est que le score d'attribution décroît quand le nombre d'auteurs augmente, mais les répétitions maximales du 2$^{\text{ème}}$ ordre (w_{2nd}) donnent de meilleurs résultats que les répétitions maximales et les n-grammes.

Chapitre 4

Corpus d'études : deux matériaux de nature différente

4.1 Explosion de la presse en France au 17^{e} siècle, les Mazarinades

4.1.1 Un corpus qui n'en est pas un ? Documents variés au premier essor de la presse en France

Le corpus utilisé dans cette partie est celui des Mazarinades, réalisé dans le cadre du projet Antonomaz [1].

L'appellation "Mazarinades" caractérise de courts documents publiés principalement au milieu du 17^{e} siècle, lors de la Fronde, une grande période de révolte en France : En pleine guerre contre l'Espagne, la France est alors régie par Anne d'Autriche, Louis XIV, l'héritier légitime, étant alors mineur. L'imprimerie était alors en plein essor, les techniques de presse étant devenues de plus en plus répandues en un peu moins de vingt ans, ce qui a entraîné une grande diminution des coûts de production. Ce faible coût permit alors de rendre plus accessible cesdits écrits. Les différents auteurs de l'époque ont alors pu s'exprimer massivement, le support privilégié était alors assez semblable à un journal, était constitué de quelques pages qui présentaient une qualité variable (papier parfois extrêmement fin). D'une importante viralité, ces écrits avaient ainsi la capacité de grandement influencer la réputation des personnes citées, que ce soit pour en faire le blâme ou l'apologie

La cible la plus connue de ces attaques fût alors le cardinal Mazarin, l'un des ministres de cette époque et la raison de l'antonomase "Mazarinade".

1. Dépôt GitHub du corpus Antonomaz : `https://github.com/Antonomaz`

Les Mazarinades ont un format assez libre, elles peuvent alors être de divers genres littéraires : pamphlet, discours retranscrits, poésie, et cætera. Ainsi, comme nous l'avons vu lors de la sous-section 3.1.3, les auteurs de ce corpus remettent eux-mêmes en question cette dénomination de "corpus" [Abiven and Lejeune 2019]. Cependant, en reprenant la définition de Sinclair [1996] vue dans le cours *Corpus, ressources et linguistique outillée* [Fort 2023]

`"A corpus is a collection of pieces of language that are selected and ordered according to explicit linguistic [and/or extra-linguistic] criteria in order to be used as a sample of the language"`

nous pouvons déterminer que les critères d'"éléments du langage" et "sélectionnés en accord à un critère extra-linguistique explicite" collent suffisamment à cette collection de textes pour que, au moins dans le cadre de ce mémoire, mais aussi dans le but d'éviter d'essayer de répondre à une question qui pourrait constituer une étude en elle-même, nous qualifierons les Mazarinades de corpus.

Ces écrits se sont alors parfaitement inscrits dans l'actualité de l'époque, que ce soit pour leur valeur officielle, parodique ou bien satirique. Ils pouvaient également se révéler particulièrement cinglants, comme par exemple ce texte insultant sans trop de détour le Cardinal [2]. Le corpus Antonomaz est alors constitué, à l'heure actuelle, de 3065 textes issus de ces Mazarinades. Il ont alors été OCRisés, convertis d'un format "papier" à un format nativement numérique, à partir de 3 bibliothèques numériques principales :

— Bibliothèque mazarine [3] ;
— Gallica, bibliothèque numérique de la BNF [4] ;
— Google Livres [5].

Ils sont alors disponibles au format *XML-TEI* au sein du dépôt maintenu par l'équipe Antonomaz [6].

Par ailleurs, ces bibliothèques sont loin d'appliquer le même standard de qualité, cela fait écho à la question de l'accessibilité de la ressource évoquée précédemment, la ressource est "présente" mais bien des informations sont altérées. Nous aurons l'occasion de présenter des exemples par la suite mais pour résumer brièvement, certaines numérisations particulièrement erronées, bruitées, entraînant alors un bruit dans la retranscription par OCR : les numérisations fautives étant principalement issues de Google Livres.

2. *Les proprietez diaboliques*, comparant le cardinal Mazarin au diable `https://short.marceau-h.fr/prop-diabo`
3. Bibliothèque mazarine : `https://www.bibliotheque-mazarine.fr/fr`
4. Gallica : `https://gallica.bnf.fr/accueil/fr/content/accueil-fr`
5. Google Livres : `https://books.google.fr`
6. Dépôt GitHub du corpus Antonomaz : `https://github.com/Antonomaz/Corpus`

Les textes retranscrits, présents sur le dépôt du projet, ont une longueur moyenne de 13.9 pages, pour 3292,5 mots et 19385,2 caractères.

La première difficulté inhérente à ce corpus sera donc le *parsing* de chaque fichier XML, c'est-à-dire qu'il nous faudra parcourir ces fichiers afin d'en extraire le texte, correctement formaté, mais aussi les diverses méta-données qui seront clef pour essayer de modéliser la cause d'une mauvaise retranscription.

4.1.2 Chaîne de traitements

Notre premier objectif sera alors de pouvoir, à partir du fichier XML, être capables d'isoler le texte, puis de le séparer en fonction des lignes, mais aussi des pages. Cependant, en fonction des fichiers, nous avons plusieurs manières de séparer ces éléments : Certaines lignes sont séparées par une balise `<lb\>` signifiant "barrière de ligne", d'autres sont contenues dans un couple de balises `<l> </l>` signifiant "ligne", d'autres encore sont doublement signalées, avec donc ces deux systèmes de balisage.

Dans la figure 4.1, nous présentons ainsi deux utilisations différentes du système de balisage :

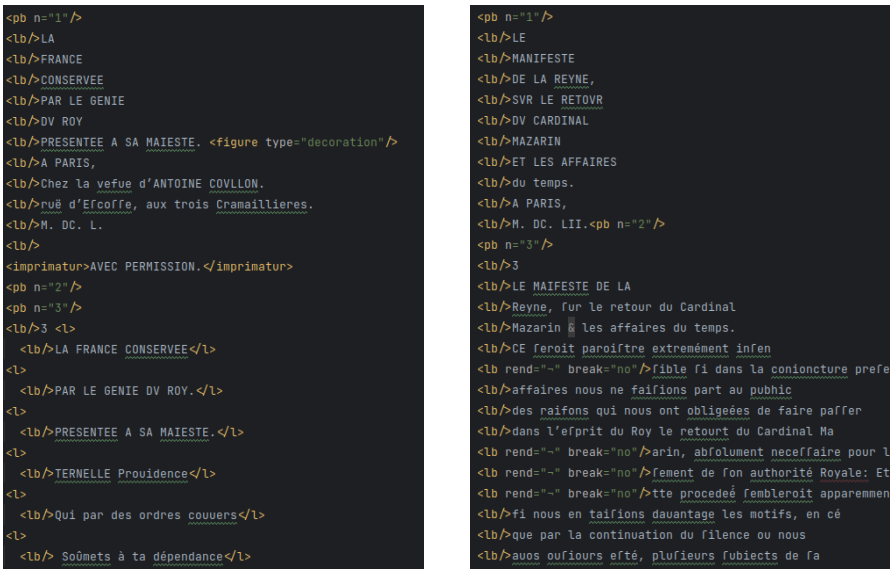
— À droite, nous avons un début utilisant les barrières de lignes. Puis, à partir de la page 3 (représentée par la balise `<pb n="3"/>`), nous avons l'utilisation combinée du couple de balises `<l> </l>` avec les barrières de lignes (`<lb\>`) ;

— À gauche, nous avons un cas d'utilisation plus classique où seules les barrières de lignes sont utilisées.

De plus, il nous faut aussi reconnaître les caractères échappés dans les textes. Du fait que le langage XML utilise des caractères pour représenter la structure de ses fichiers, l'utilisation de ces mêmes caractères au sein du texte rendrait la structure invalide. Pour pallier ce problème, les caractères utilisés par le langage de balisage sont alors échappés. L'échappement consiste à ne plus écrire les caractères tels quels dans le texte, leur présence alors est signifiée par une suite de caractères qui permet de lever l'ambiguïté qu'ils généraient.

Dans le cas du langage XML, les caractères échappés commencent par l'esperluette (`&`) et finissent par un point virgule (`;`). Par exemple, le chevron ouvrant, servant à encoder le début d'une balise en XML, se retrouve représenté par `&lt;`, *lt* signifiant "inférieur à" (lower than). Vous pouvez retrouver la liste des codes d'échappement en XML sur le site de Microsoft [7].

7. Liste des codes d'échappement en XML et XAML par fournis Miscrosoft : `https://short.marceau-h.fr/xml-esc`



(a) Texte avec retours à la ligne doublement balisés

(b) Texte avec retours à la ligne simplement balisés

FIGURE 4.1 – Exemples de balisage des textes

Nous avons donc pu extraire le texte en utilisant deux méthodes : l'utilisation du *parser lxml* à l'aide du module *BeautifulSoup4*, pour délimiter le texte. Puis une fois le segment du XML extrait, quelques expressions régulières ont alors servi à séparer le texte en pages et en lignes tout en prenant en compte les divers cas et en éliminant des sauts de lignes multiples.

Par la suite, nous utilisons la librairie *xmltodict* afin de parser les informations contenues dans l'entête du fichier. Nous récupérons quelques informations clés, les mots clés, la date de création, de modification et la langue. À noter que ces informations ne sont pas toujours toutes renseignées.

Pour finir, nous complétons les méta-données d'origine avec quelques métriques nécessaires à la suite de notre analyse. Grâce aux pré-traitements effectués, nous pouvons compter le nombre de pages, de lignes, de mots et de caractères de chaque texte.

4.2 L'archive TALN : un corpus nativement numérique du 20e siècle et 21e siècle

4.2.1 Description de l'archive TALN

À la différence du corpus des **Mazarinades** qui est un corpus OCRisé puis "convertit à un format nativement numérique", le corpus dédié à l'étude des sauts stylistiques est celui de **taln-archives**. C'est une archive numérique francophone qui contient des articles des conférences TALN [Boudin 2013] (Traitement Automatique des Langues Naturelles) et RECITAL (Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues) en TAL de 1997 à 2022 dont l'ensemble du travail est consultable sur Github [8].

Dans l'archive on retrouve un fichier contenant les métadonnées de la conférence et de chaque article, métadonnées qui ont été utiles pour l'extraction de section de chaque article. Les deux conférences ont une taille de 855,4 Mo dont 1545 fichiers pour TALN et 318 fichiers pour RECITAL en format .pdf.

4.2.2 Comment explorer le corpus ?

Pour exploiter donc ces deux corpus, on peut commencer par extraire des segments déterminés par le ou les auteurs qui sont marqués par exemple par des alinéas, comme les paragraphes [Hearst 1997] ou également par des segments motivés par la structure de sous-thèmes (*TextTiles)* [Hearst and Plaunt 1993]. La chaîne de traitement pour extraire la structure textuelle sera donc présentée dans le paragraphe suivant.

Extraction de structure textuelle : la détection de paragraphes Pour commencer, nous essayons de trouver les paragraphes avec un double saut de ligne `\n\n` . Étant donné que le format de document ne permet pas de traiter facilement la structure textuelle avec le double saut de ligne, les PDF du corpus sont tous convertis en *TXT* avec pdftotext bien que cet outil produit des erreurs au niveau du découpage en paragraphes ; mais ces erreurs de conversion peuvent être résolues avec des méthodes de Cabrera-Diego et al. [2013] d'après GROUIN [2014]. À partir de ce format, nous avons un format de données plus facilement exploitable. Les paragraphes peuvent être facilement extraits, mais alors quelle est l'utilité des paragraphes ? Le dictionnaire CNRTL définit le paragraphe comme suit : *Section d'un texte en*

8. `https://github.com/didierkouame/Sauts-Stylistiques/tree/main`

prose, manuscrit ou imprimé, développant un point bien délimité de l'exposé en cours, pouvant comporter plusieurs alinéas et constituant elle-même une subdivision d'un ensemble plus important (généralement un chapitre). Au vu d'une telle définition, on peut dire qu'un paragraphe est un segment motivé [Hearst 1997], ça signifie qu'à chaque transition de paragraphe dans un texte, il y a une idée qui y est développée. Cependant, ce n'est pas toujours le cas, le paragraphe peut être utilisé pour améliorer l'apparence visuelle du texte ou tout simplement pour rendre la lecture plus confortable [Stark 1988]. L'objectif en essayant de détecter les paragraphes n'est pas de les diviser selon des segments qui développent des idées à chaque fois qu'il y a une transition, mais d'extraire la structure des articles en paragraphes de manière approximative, c'est-à-dire que c'est un découpage qui ne sera pas parfait. La technique qui permet de découper les textes en paragraphes ou en unités cohérentes avec une approche basée sur la sémantique est le *TextTiling* qui utilise *TF-IDF* [Hearst 1993]. Par exemple, dans la figure 4.2, on peut voir ce que donne un découpage en paragraphe avec `\n\n` avec une taille minimale 500 pour chaque paragraphe avec une condition pour les lignes qui sont courtes, `len(paragraphe) < 10`. Une variable tampon joue le rôle de construction des paragraphes en accumulant temporairement les lignes de textes. On sépare donc les lignes du texte en fonction de leur taille. Le découpage n'est pas parfait, mais nous avons à peu près la même taille de caractères pour chaque paragraphe, ce qui est suffisant pour comparer des variations de style entre eux.

Un outil plus précis d'extraction de structure textuelle : le découpage en section avec *GROBID*

Pour améliorer le découpage et obtenir une structure textuelle un peu plus précise, on utilise *GROBID* (GeneRation Of BIbliographic Data) [GRO 2023] qui est un outil pour extraire et parser des documents, dans notre cas en qui sont en format PDF. Il dispose de plusieurs fonctionnalités d'extraction comme l'extraction de références et d'en-têtes de documents. La fonctionnalité qui nous intéresse est celle de l'extraction plein texte. Avec cette fonctionnalité, on va pouvoir extraire la structure d'un document en le segmentant par paragraphes, titres de sections, références, pied de pages et bien d'autres. Il s'utilise en ligne de commande dans un terminal ou en API WEB.

```
Articulation du domaine spatio-temporel
Articulation du domaine spatio-temporel
Yann Mathet
GREYC, Université de Caen, Campus II
14032 Caen Cedex
mathet@info.unicaen.fr
http://www.info.unicaen.fr/~mathet
L'expression de la spatio-temporalité est traditionnellement scindée en deux paradigmes, la
localisation et le déplacement. La localisation exprime alors un certain nombre de relations
entre une entité à localiser et des sites, tandis que le déplacement exprime un changement de
ces relations dans le temps. Pourtant, c'est omettre l'autonomie et la richesse du déplacement

--------------------
que de l'exprimer par rapport à la localisation, et c'est aussi opposer deux paradigmes qui
partagent un certain nombre de types de contraintes (topologie, distance, etc.). Nous
proposons donc d'observer le domaine spatio-temporel et ses articulations d'une façon qui ne
les oppose pas mais qui montre au contraire ce qu'ils partagent. Ce travail de formalisation et
d'analyse est destiné à l'élaboration de mécanismes de compréhension automatique.
Introduction
Nous nous plaçons dans une tâche de compréhension automatique de textes français,

--------------------
autour du projet TACIT, au laboratoire GREYC de Caen et en association avec l'ELSAP.
Notre travail portant sur la compréhension spatiale, il nous est nécessaire de formaliser le
domaine, ou plus exactement d'en proposer une certaine représentation formelle, amenant
notamment à la description des entités manipulées lors de l'expression du spatial. Si cette
question semble amener à un large consensus dans le domaine du temps, les entités
concernées étant généralement des instants ou des intervalles temporels, la question reste très

--------------------
ouverte dans notre domaine.
L'objet de cet exposé n'est pas de discuter ce modèle [Mathet 99], mais les contraintes et
surtout les relations sémantiques portant sur les éléments de ce modèle. Mentionnons
seulement que celui-ci est composé de trois niveaux : l'espace support, tout d'abord, est un
espace spatio-temporel produit d'un espace euclidien E et d'un axe temporel T. A partir de ce
support ExT, on crée un espace cognitif consistant en un ensemble d'objets fonctions de T
dans E, proposant une représentation formelle de « lieux » et « entités du monde » ; parmi ces

--------------------
objets du modèle, une classe est particulièrement importante puisqu'elle est le support de
```

FIGURE 4.2 – Découpage en paragraphes des articles

GROBID génère des documents en format XML-TEI, format de données qui décrit du contenu semi-structuré. Avec ce format, les documents sont mieux structurés et on a une séparation stricte entre contenu et mise en forme.

On a donc une structure plus précise désormais que l'on peut encore plus facilement exploiter. On peut désormais extraire toutes les sections des articles avec leur titre qui sont entre les balises ouvrantes et fermantes `<div> </div>`. Ce qui va permettre de mener plus aisément les analyses entre sections (voir découpages des sections dans la figure A.10. Cependant, *GROBID* n'est pas sans faille car pour des raisons non connues, l'outil n'a pas traité tous les 318 PDF de RECITAL et tous les 1545 PDF de TALN. On se retrouve avec 289 XML pour RECITAL et 1355 XML pour TALN. Il n'arrive pas également à parser certains éléments de références pour certains documents (voir FIGURE 4.5). Concernant le temps d'exécution, *GROBID* prend 9 minutes pour générer les 289 fichiers RECITAL et 36 minutes pour

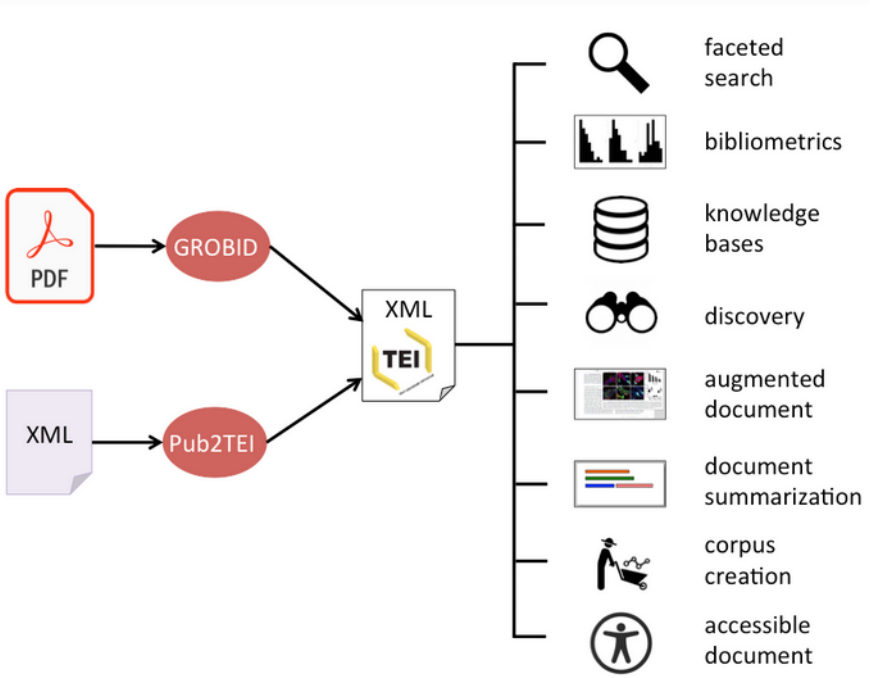


FIGURE 4.3 – Traitement de documents scientifiques par GROBID - Source : Grobid Documentation

les 1355 fichiers TALN en XML. Pour extraire le contenu des balises `<div>` en Python, on utilise la bibliothèque *BeautifulSoup* avec un parser HTML en ignorant le contenu des blocs `<div>` qui sont inférieurs à 100 car on a des blocs courts de contenu.

Conférence	Caractères	Phrases	Mots
RECITAL	10 047 222	60 740	1 526 182
TALN	42 605 195	262 920	6 591 748

TABLE 4.1 – Statistiques des articles pour les deux conférences TALN et RECITAL rédigés de 1997 à 2022 avec le format TXT

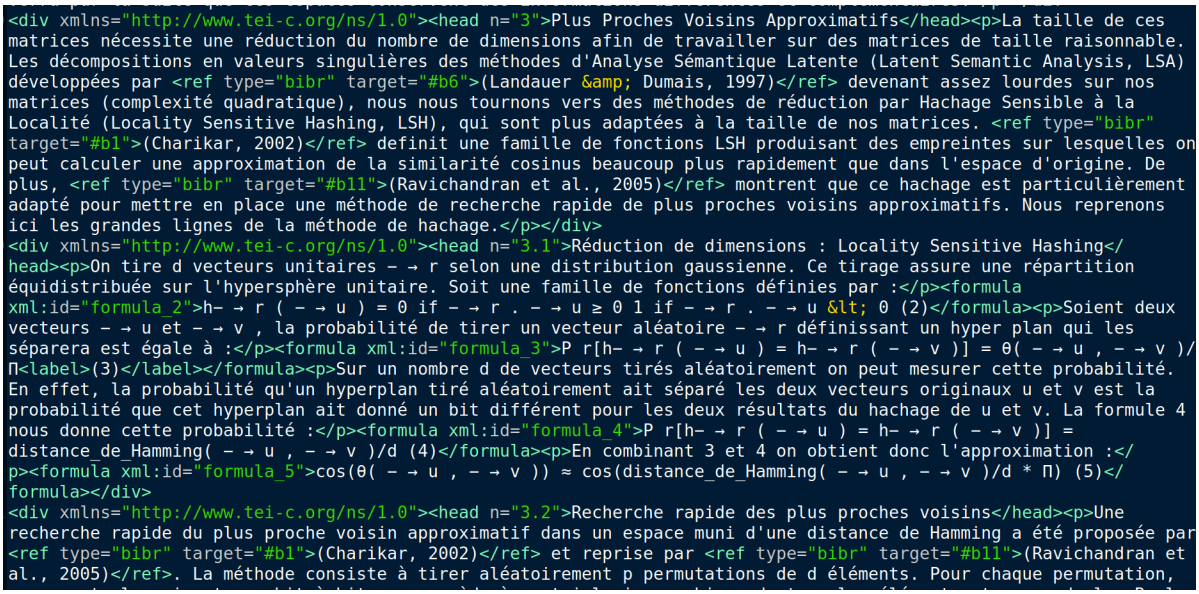


FIGURE 4.4 – Résultats de l'extraction de GROBID

Chapitre 5

Méthodes d'analyse de la variation

5.1 Reconnaissance optique de caractères

La Reconnaissace Optique de Caractères, abrégée en anglais par le sigle OCR, désigne les approches permettant de transformer un document physique, ou un document non nativement numérique de type image, en un document numérique. Pour ce faire, ces derniers essaient de localiser des zones de texte, puis des glyphes, pour finir, ils essaient d'associer chaque glyphe ainsi détecté à un caractère précis. Ainsi les techniques d'OCR permettent d'obtenir, à partir d'une image (formats traditionnels d'image comme jpg mais aussi depuis un pdf), un fichier texte (certaines sorties intermédiaires existent et référencent la position précise de chaque caractère, de chaque ligne ou encore de chaque zone de texte).

L'objectif de ce traitement est donc de rendre le document exploitable de la même manière que le serait un document nativement numérique, on peut, bien sûr, penser à une utilisation par un humain directement (soft wrap, citation, retour au texte, simple lecture, ...). Mais dans notre cas, obtenir un texte réellement numérique nous permet de faire de nombreux traitements : permettre une recherche plein texte, obtenir des statistiques sur nos corpus, construire un lexique, réaliser des plongements sémantiques, retrouver la structure du document, et cætera.

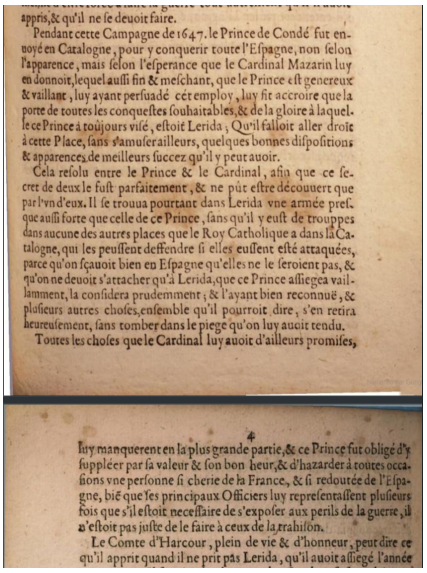
Cependant, transformer des documents non nativement numériques ne se produit pas sans erreurs. En effet, les techniques d'OCR comportent plusieurs étapes qui peuvent altérer le résultat de la transcription. Par exemple, avant de pouvoir décider quel caractère correspond à ce que l'on peut voir sur une partie de l'image, il faut d'abord être capable de déterminer quelle partie de

l'image représente des caractères, mais aussi quelle serait la séparation entre chacun d'entre eux.

Ainsi, ne serait-ce qu'être capable de savoir où se trouvent les caractères ne se résume pas à trouver les parties de la page où il y aurait un contraste important : Certains documents peuvent être très vieux, avec, par exemple, des taches de café, et des pages burinées, marquées par l'usure. Ainsi, plus l'information de la page est bruitée, plus il peut y avoir des erreurs de reconnaissance. De plus, cette mauvaise reconnaissance des zones de caractère peut être exacerbée sur des pages sans textes, qui laissent parfois entrevoir, en filigrane, le texte du verso, et qui sont parfois couvertes de taches et autres marques d'usures. Dans les images ci-dessous, en figure 5.1, nous pouvons retrouver un exemple de mauvaise détection de parties avec caractères sur des pages pourtant relativement bien conservées.

À titre de comparaison, c'est un peu comme si vous essayiez de trouver des formes dans les nuages, ils auront beau n'être que des amas difformes, si vous essayez d'y trouver des objets, vous en trouverez. De plus, il faut savoir sur l'OCR travaille sur une version binarisé de l'image. En effet, avant de donner la page à l'OCR, on retire toutes les nuances de cette dernière, ne laissant alors que des pixels tout blancs et tout noirs, dans l'objectif de simplifier le traitement mais également de retirer une partie du bruit. Cependant, ce traitement manichéen rend les taches, qu'il n'aura pas fait disparaître, encore plus visibles, facilitant alors la méprise de ces dernières pour des caractères.

(a) Exemple d'erreurs d'OCR


appris,& qu'il ne ſe deuoit faire.

Pendant cette Campagne de 1647. le Prince de Condé fut envoyé en Catalogne, pour y conquerir toute l'Eſpagne, non ſelon l'apparence, mais ſelon l'eſperance que le Cardinal Mazarin luy en donnoit, lequel auſſi fin & meſchant, que le Prince eſt genereux & vaillant, luy ayant perſuadé cét employ, luy fit accroire que la porte de toutes les conqueſtes ſouhaitables, & de la gloire à laquelle ce Prince à toûjours viſé, eſtoit Lerida; Qu'il falloit aller droit à cette Place, ſans s'amuſer ailleurs, quelques bonnes diſpoſitions & apparences de meilleurs ſuccez qu'il y peut auoir.

Cela reſolu entre le Prince & le Cardinal, afin que ce ſecret de deux le fuſt parfaitement, & ne pût eſtre découuert que par l'vn d'eux. Il ſe trouua pourtant dans Lerida vne armée preſque auſſi forte que celle de ce Prince, ſans qu'il y euſt de trouppes dans aucune des autres places que le Roy Catholique a dans la Catalogne, qui les peuſſent deffendre ſi elles euſſent eſté attaquées, parce qu'on ſçauoit bien en Eſpagne qu'elles ne le ſeroient pas, & qu'on ne deuoit s'attacher qu'à Lerida, que ce Prince aſſiegea vaillamment, la conſidera prudemment; & l'ayant bien reconnuë, & pluſieurs autres choſes, enſemble qu'il pourroit dire, s'en retira heureuſement, ſans tomber dans le piege qu'on luy auoit tendu.

Toutes les choſes que le Cardinal luy auoit d'ailleurs promiſes,

luy manquerent en la plus grande partie, & ce Prince fut obligé d'y ſuppléer par ſa valeur & ſon bon heur, & d'hazarder à toutes occaſions vne perſonne ſi cherie de la France, & ſi redoutée de l'Eſpagne, biẽ que ſes principaux Officiers luy repreſentaſſent pluſieurs fois que s'il eſtoit neceſſaire de s'expoſer aux perils de la guerre, il n'eſtoit pas juſte de le faire à ceux de la trahiſon.

Le Comte d'Harcour, plein de vie & d'honneur, peut dire ce qu'il apprit quand il ne prit pas Lerida, qu'il auoit aſſiegé l'année

(b) Page source

FIGURE 5.1 – Erreurs d'OCR, résultat et source

Le texte pris en exemple est disponible à partir de ce lien-ci. [1]

De plus, une fois que le caractère à retranscrire a été localisé sur la page, il faut encore le transcrire proprement, c'est par exemple le problème dans l'exemple suivant :

`"Toutes les choſes que le Cardnal luy auoit d'ailleurs promiſes,"`

Qui est retranscrite avec deux erreurs, le mot ***T****outes* devenant ***F****outes* et le mot *cho****f****es* devenant *cho****l****es*. Ces erreurs peuvent être dues à une mauvaise reconnaissance au niveau du caractère (un T peut ressembler à un F). Mais, cela pourrait aussi être dû à une mauvaise reconnaissance au niveau des suites de lettres. Par exemple, si l'outil voit souvent la suite de lettres `les`, dans le doute, il privilégiera la suite la plus probable, en l'occurrence `les`. On peut également penser à l'échelle du mot. Et, bien sûr, le résultat proposé est le produit de la somme de ces traitements. Pour reprendre notre exemple, si l'outil voit souvent la suite de lettres `les`, dans le doute entre f et `l`, il privilégiera la lettre formant la suite la plus probable, en l'occurrence `les`

Ainsi, ces mauvaises retranscriptions nous posent deux problèmes :

— Le texte ne pourrait ne plus être lisible par un humain sans faire un retour à l'original, c'est-à-dire sans aller consulter la version non nativement numérique du document, sous réserve que cette dernière soit accessible. Et dans une moindre mesure, un texte plus légèrement bruité pourrait demander un effort important pour le déchiffrer, surtout quand il s'agit de plusieurs pages, rebutant ainsi plus d'un lecteur.

— Les traitements automatiques que l'on appliquerait aux textes pourraient en être faussés. Que ce soit pour pouvoir reconnaître la classe grammaticale ou le lemme d'une forme incorrectement retranscrite que pour pouvoir faire des statistiques à partir d'un texte très bruité.

Il nous faudrait alors pouvoir mesurer à quel point un texte est bruité, cependant comme observé au sein de l'article *OCR performance prediction using cross-OCR alignment* [Ben Salah et al. 2015] dans leur figure 1, reprise en figure 5.2, les estimations de performance implémentées dans les outils conventionnels d'OCR ne représentent que peu la réalité.

Nous avons donc pensé à plusieurs mesures qui pourraient nous permettre de mesurer plus fidèlement, sur un large corpus de textes numérisés, la qualité de la transcription, du document "papier" au document nativement numérique.

1. Exemple d'erreur de retranscriptiton : *Factum, servant au procez criminel fait au cardinal Mazarin, touchant ses intelligences avec les estrangers ennemis de l'Estat. Premiere partie* `http://bit.ly/OCR-slong`

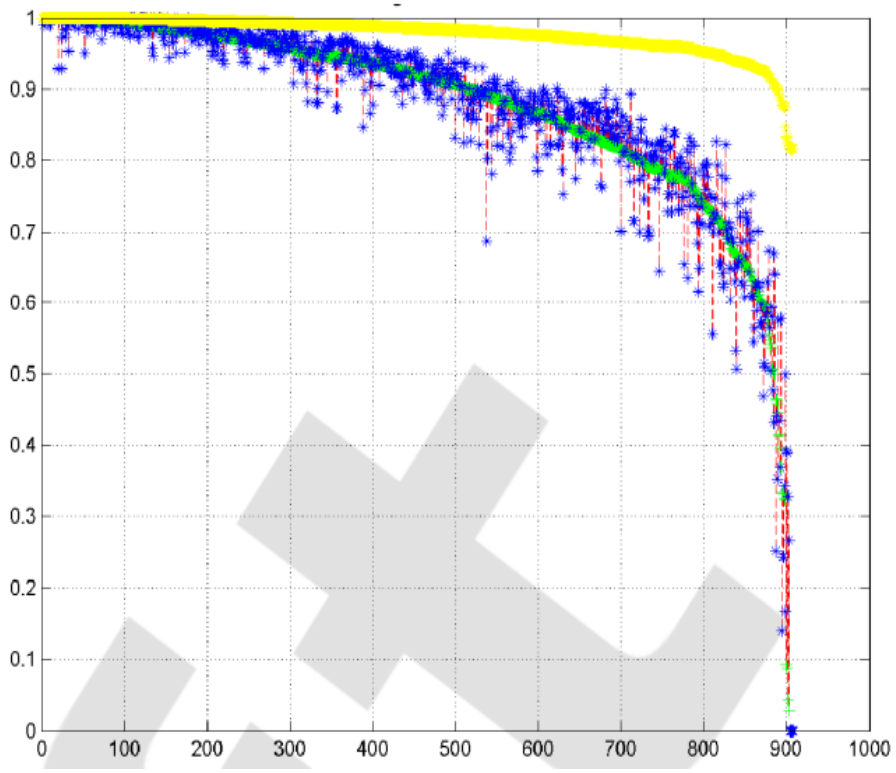


FIGURE 5.2 – Estimation du taux de reconnaissance d'OCR. Les points jaunes indiquent le taux calculé à partir des données fournies par le logiciel de reconnaissance. Les points verts décrivent le véritable taux de reconnaissance calculé dans l'article OCR performance prediction using cross-OCR alignment [Ben Salah et al. 2015] duquel ce schéma est issu.

5.2 Mesures

L'objectif est de pouvoir discriminer les fichiers OCRisés, rendus numériques, en fonction de la qualité de la retranscription obtenue. Cependant, du fait de la grande quantité de données à gérer, 3065 textes de plusieurs pages dans le cas de notre corpus, une approche manuelle n'est donc pas envisageable. Il nous faudrait alors une ou plusieurs mesures simples et capables de rendre compte de la qualité de n'importe quel document.

Dans le cadre de ce mémoire, nous nous sommes donc intéressés à quelques mesures que nous détaillerons à la suite de cette partie introductive :

— Le Type-Token Ratio (TTR), qui consiste à mesurer le ratio entre la taille du vocabulaire et la taille du texte au total (le nombre de formes).

L'idée derrière cette mesure étant qu'un texte bruité aura plus de formes uniques proportionnellement aux formes totales en comparaison à un texte non bruité. Cependant, il reste encore la question du style de l'auteur, qui pourrait faire varier cette mesure.

Ainsi, un texte de 10 000 formes avec un vocabulaire de 8.000 formes aura donc un TTR de 0.8.

— La proportion d'hapaxes, où l'on mesurera simplement le nombre d'hapaxes du texte selon la longueur de ce dernier.

Cette mesure pourrait nous permettre d'obtenir des informations supplémentaires sur la qualité d'un texte. En effet, on peut s'attendre à ce que, au sein d'un texte, les erreurs de transcription qui conduisent à une erreur au sein d'un mot, par opposition à celles qui trouvent des lettres là où nous n'en avons pas, provoquent des hapaxes en majorité.

Ainsi, un texte de 10 000 formes avec un total de 1.400 hapaxes aura donc un TTR de 0.14.

— Le taux de lexicalité, cette dernière mesure consiste à déterminer le pourcentage de formes que l'on connaît, grâce à un lexique.

Cette donnée supplémentaire nous offrirait de nouvelles informations sur le niveau de bruit d'un texte, notamment pour les plus courts. Cependant, cette dernière nécessite l'utilisation d'un lexique, qui doit être assez fidèle à la langue employée, mais également dans notre cas à l'état de la langue employée.

Ainsi, un texte de 10 000 formes avec 2.000 de ces formes dans le lexique aura donc un TTR de 0.2.

Le premier obstacle que nous rencontrons, avant même de réaliser ces mesures, est donc de savoir comment différencier l'impact du style de l'auteur sur les mesures que nous emploierons.

5.3 Taux de lexicalité

Ainsi, l'une des premières mesures explorées est celle du taux de lexicalité : On calcule le taux de lexicalité en déterminant le rapport entre le nombre de tokens présents dans un lexique de référence et le nombre de tokens complets. Dans notre cas, nous entendons par lexique l'ensemble des formes lexicales (conjugaison, pluriel et cætera) donné d'un état de langue. Bien sûr, à la manière d'un dictionnaire, aucun lexique ne couvre 100% d'un

état de langue et aura toujours des termes qui sont absents. À noter que les chiffres (arabes et romains) seront, dans notre cas, toujours considérés comme des mots lexicalisés.

Il nous faut alors extraire le texte de chaque XML. Ainsi, après avoir résolu la manière d'extraire le texte convenablement des fichiers XMLs, puis la manière de tokeniser, il nous faut encore un lexique, correspondant à la langue étudiée.

Le premier enjeu ici est que nous n'étudions pas le français contemporain, mais celui du 17ème et début 18ème siècle.

5.3.1 LGeRM

Afin de remplir cette fonction, le premier lexique utilisé fut celui du LGeRM [ATILF 2017], facilement accessible en ligne [2] et avec une version 16ème et 17ème siècles, donc très proche de ce dont nous avons besoin, la majorité de nos textes datant du 17ème siècle.

De plus, le lexique comporte un grand nombre de formes, presque 3 millions, ce qui normalement nous permettra d'avoir peu de silence (d'éviter de ne pas reconnaître des mots de la langue uniquement par leur absence du lexique utilisé). Il restera évidemment certains mots qui ne seront pas détectés, en guise d'évaluation de ce lexique, l'ATILF atteste qu'il recouvre 83.4% des formes présentes dans Frantext [ATILF 2023b], ce qui représente 99.3% des occurrences.

Cependant, lors de l'utilisation de ce lexique, nous avons fait face à deux problèmes, l'encodage du lexique et la mauvaise représentation de notre corpus.

L'encodage

Au téléchargement du fichier, certains symboles semblent mal encodés : Lorsque l'on se rend sur le site, nous sommes seulement informés qu'il a été encodé en "iso", sans détails supplémentaires. Cependant, lors du téléchargement, le fichier se retrouve en UTF-8 et aucune solution de récupération automatique n'a été trouvée. Vous retrouverez en figure 5.3 un extrait de la version corrompue du lexique, le fichier est également disponible [3]. On aurait pu, par exemple, envisager une récupération avec une distance minimale face à un lexique de référence, le problème étant que nous n'avons à ce mo-

2. Lien vers l'archive du lexique LGEeRM : `http://bit.ly/443tfXa`

3. Version corrompue du LGeRM, obtenue depuis le site de l'ATILF : `http://bit.ly/42Qg1vT`

ment aucun lexique de référence. Nous avons donc opté pour une solution semi-automatique :

— On crée une liste de caractères acceptables dans le français moderne et moyen,
— On regarde quels mots ont des caractères absents de cette liste,
— On essaie de trouver une correspondance entre ces caractères et un caractère de la langue manuellement, en se basant sur des recherches internet.

Le lexique en ressort grandement nettoyé, cependant certaines formes qui sont malheureusement restées trop bruitées pour pouvoir déterminer leur version d'origine.

La mauvaise représentation

Au final, ce lexique ne couvre pas l'ensemble de notre corpus. En effet, ce dernier comporte certains parlers régionaux et des textes en latin. Nos taux de lexicalité en demeurent alors grandement affaiblis, beaucoup de textes se retrouvent avec un taux de lexicalité inférieur à 0.2 alors que ces derniers semblent peu bruités.

Nous avons donc cherché d'autres lexiques complémentaires, plusieurs ont alors été envisagés :

— Morphalou
— DMF
— TLFi
— Ducange
— GLÀFF.

5.3.2 Morphalou

Morphalou [ATILF 2023a] est, dans sa version 3 qui est la plus récente à l'heure de ce mémoire, un lexique du français contemporain. Il fut constitué à partir de plusieurs lexiques, dont certains produits par l'ATILF, en voici la liste détaillée présente sur leur page [4] :

— Morphalou 2 (version de décembre 2013),
— DELA (version de décembre 2011),
— Dicollecte (version 4.3),
— LGLex et LGLexLefff (version 3.4),
— Lefff (version 2.1 avril 2006).

4. Lien vers le lexique Morphalou : `https://www.ortolang.fr/market/lexicons/morphalou`

```
"Ã¦gilopiné",        "gangréneuses",
"Ã¦gilopine",        "gangréneux",
"Ã¦gilopinées",      "gangrénevs",
"Ã¦gilopinees",      "gangrénevse",
"Ã¦olis",            "gangrénevses",
"Ã¦oliz",            "gangrénevx",
"Ã¦pyornis",         "gangreÌneux",
"Ã¦pyorniz",         "gangreÌneus",
"Ã¦sthésiomètre",    "gangreÌneux",
"Ã¦sthesiometre",    "gangreÌnevs",
"Ã¦sthesiomètre",    "gangreÌnevx",
"Ã¦sthesiométre",    "gangster",
"Ã¦sthésiometre",    "gangster",
"Ã¦sthésiométre",    "gangsters",
"Ã¦théogame",        "gangstérisme",
"Ã¦theogame",        "gangsterisme",
"Ã¦théogamie",       "gangué",
"Ã¦theogamie",       "gangue",
"Ã¦theogamye",       "gangué",
"Ã¦théogamye",       "gangve",
"Ã¦tite",            "gangvé",
"Ã¦titte",           "gangue",
```

FIGURE 5.3 – Problèmes d'encodage dans le lexique LGeRM

Il est composé de 522.662 formes, ce qui permet une couverture adéquate du français contemporain.

Après l'avoir téléchargé, dans sa forme qui nous semblait la plus propice (en CSV), nous l'avons élagué afin de n'en tirer que les formes. Pour y parvenir, nous en avons soustrait les premières lignes, puis utilisé la librairie pandas afin de pouvoir parcourir efficacement le CSV et soutirer les lemmes de chaque ligne.

On peut alors se servir de ce dernier pour compléter nos taux de lexicalité.

Cependant, comme ce dernier représente un état de langue grandement postérieur à notre corpus ; Il n'améliore qu'au final la lexicalité de très peu de textes, à peine un peu plus d'un pourcent du corpus.

5.3.3 Ducange

Afin de compenser le manque de nos deux premiers lexiques, nous avons alors envisagé l'utilisation d'un troisième corpus, couvrant pour sa part le Latin. C'est ainsi que nous sommes tombés sur une énième ressource lorraine, Ducange [Du Cange et al. 1887]. Nommé après l'auteur éponyme, ce glossaire fut créé pour représenter l'état de langue du latin des 17-19 èmes siècles.

Ducange est, dans sa version actuelle, un lexique du latin composé de 103 083 formes.

Il n'a pas de version directement accessible en ligne, mais il est possible de l'interroger en ligne. Ainsi, à l'aide du module *requests* et de l'exploration du trafic *XHR* du navigateur, il est assez aisé de récolter les XMLs qui permettent l'affichage de la réponse à nos requêtes.

Il en résulte alors un XML par initiale que nous parserons, puis concaténerons, afin d'obtenir l'ensemble des formes décrites par ce lexique.

Nous obtenons ainsi un lexique, formaté en JSON, d'un total de 103 083 formes avec lequel nous sommes en mesure de compléter notre corpus.

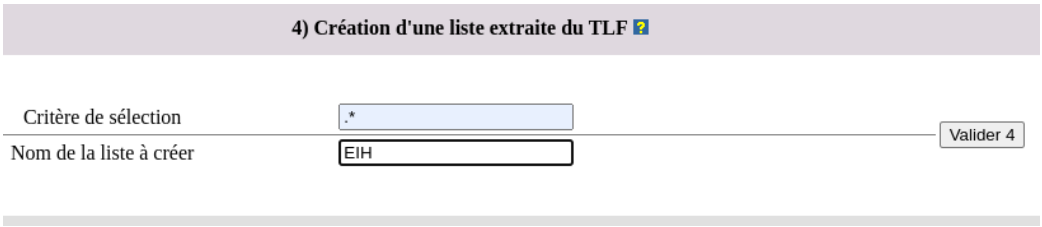
Malheureusement, au final, ce dernier n'améliore aucun des taux de lexicalité de nos textes. En l'état, il nous est donc d'aucune utilité. Par ailleurs, sa taille modeste, relativement aux autres lexiques, pourraient l'expliquer. Cependant, cela nous amène à nous interroger de nouveau sur la nécessité de traitements supplémentaires : Faudrait-il concaténer les lexiques ? Avons-nous besoin d'écarter les éléments qui ne feront jamais partie de nos lexiques ? On pense particulièrement aux noms propres et aux chiffres.

5.3.4 TLFi

Le Trésor de la Langue Française informatisé [ATILF 2002], abrégé en TLFi, est une ressource linguistique développée par l'ATILF qui a majoritairement pour vocation à être utilisée par des linguistes afin d'accéder à un grand nombre d'informations en ligne : C'est en quelque sorte un dictionnaire retranscrivant différents états de la langue française auquel on aurait ajouté l'étymologie ainsi que les premières attestations connues d'usage de l'acception.

En explorant le site de l'ATILF, nous nous sommes rendu compte que l'on pouvait récupérer des listes de formes à partir d'une requête. De plus, nous pouvons intégrer des expressions régulières à ces dernières.

Il nous suffit alors de requêter une liste de formes qui répondent à la requête `.*`, comme en figure 5.4a, qui se traduirait par "n'importe quel caractère, n'importe quel nombre de fois", pour obtenir une liste incluant toutes les formes présentes. Une fois cette liste générée, il suffit de l'extraire de la réponse HTML, puis de la copier dans un fichier texte. On en profite au passage pour retirer les caractères non alphabétiques uniques qui avaient été retournés par la précédente requête. Vous pouvez trouver un échantillon de cette liste en figure 5.4b.



(a) Requêtage de la liste de formes

(b) Résultat fourni par le site du TLFi

FIGURE 5.4 – Collecte des formes du TLFi

L'avantage du TLFi est sa large couverture de la langue au fil du temps.

Ce lexique reste cependant relativement petit, notre extraction est composée de 365 737 formes.

Ainsi, nous pouvons désormais nous servir de ce lexique pour compléter notre arsenal : ce faisant, nous obtenons un meilleur taux de lexicalité pour un peu plus de 30% de notre corpus. Cette amélioration est loin d'être négligeable, et provoque également une nette augmentation de notre taux de lexicalité moyen. Ce lexique nous permet donc de pallier le manque de couverture de notre premier lexique, le LGeRM.

```
ratio_lexiques = {k: (v/len(files))*100 for k, v in
  counter_lexiques.items()}
ratio_lexiques
Executed in 6ms, 10 Apr at 18:10:41

 {'ducange': 0.0,
  'LGERM': 67.66721044045677,
  'tlfi': 30.701468189233278,
  'morphalou': 1.598694942903752}
```

FIGURE 5.5 – Pourcentage de textes pour lesquels chaque lexique donné est le plus représentatif. C'est-à-dire, celui qui donne le meilleur taux de lexicalité.

5.3.5 GLÀFF

Le Gros Lexique À Tout Faire du Français [Sajous et al. 2013] (ou GLÀFF), est un lexique particulièrement conséquent. En effet, il est composé de 1 406 857 formes dans sa version actuelle, ce qui en fait le seul de cette liste à partager l'ordre de grandeur du LGeRM. Constitué à partir du Wikitionnaire, sa première version a été publiée en 2013. Quant à sa version actuelle, elle date de 2017. Ainsi, malgré son grand nombre de formes, il représente un état de langue assez éloigné de notre corpus.

Ainsi, avec une moyenne du taux de lexicalité assez faible, aucun texte ne se retrouve mieux représenté par ce lexique.

5.3.6 Un lexique pour les rassembler tous ? Synthèse sur l'utilisation d'une ressource externe

Caractéristiques générales des lexiques présentés

Ainsi, à l'aide de l'utilisation de ces différents lexiques, nous avons pu constater l'importance de faire le bon choix afin d'avoir la mesure la plus proche de la réalité : Un texte avec un faible taux de lexicalité par rapport à un lexique de référence ne veut pas forcément dire que ce dernier est bruité, le lexique pourrait tout aussi bien ne pas correspondre à l'état de langue du texte. Il demeure également la question des emprunts et du langage argotique. En effet, du fait de la forme très libre de ces textes, certains auteurs pourraient prendre considérablement plus de libertés que d'autres, ainsi, du fait de cette seule mesure, leurs textes pourraient nous apparaître bruités. Nous revenons donc à la question de l'impact du style de l'auteur.

Vous trouverez ci-dessous un tableau résumant les différentes propriétés des lexiques que nous avons utilisés.

Lexique	Nombre de formes	Langue	Etat de langue	Meilleur %	Moyenne lexicalité
LGeRM	3 049 125	Français	16 et 17 è siècles	67,5%	44,4%
Morphalou	740 830	Français	Contemporain	1,6%	39,5%
Ducange	103 083	Latin	Moyen-âge	0,0%	10,7%
TlFi	365 737	Français	Contemporain et antérieur	30,8%	44,2%
GLAFF	1 406 857	Français	Contemporain	0,0%	37,3%

TABLE 5.1 – Caractéristiques des lexiques utilisés en vue de la mesure du taux de lexicalité.

Comment comparer les vocabulaires de ces lexiques ? UpSet plot

Au final, nos lexiques ne sont utilisés qu'en tant que simples ensembles de formes lexicales. Nous avons alors pensé à analyser ces derniers comme tel. Nous nous servirons alors de l'UpSet plot pour représenter le rapport entre ces derniers. L'analyse des ensembles de données est une tâche courante dans de nombreux domaines, de la biologie à l'analyse des données commerciales. Cependant, lorsque les ensembles de données deviennent complexes, il devient de plus en plus difficile de comprendre les relations entre les différents sous-ensembles. C'est là que l'UpSet Plot (ou diagramme UpSet) entre en jeu [Lex et al. 2014].

Pour lire efficacement un UpSet Plot, il est important de comprendre les éléments clés de la visualisation. Voici les principaux éléments à prendre en compte :

Axes horizontaux : Chaque axe horizontal représente un ensemble. La longueur de l'axe indique le nombre d'éléments dans cet ensemble.

Barres verticales : Les barres verticales connectent les axes horizontaux. Chaque barre représente une intersection entre les ensembles correspondants.

Intersection size : La taille de chaque barre verticale indique le nombre d'éléments communs aux ensembles correspondants. Plus la barre est longue, plus l'intersection est grande.

Encodage de la cardinalité : La partie supérieure de chaque axe horizontal affiche une ligne de nombres. Ces nombres représentent la cardinalité de l'ensemble, c'est-à-dire le nombre total d'éléments qu'il contient. Ici, nous avons décidé de l'exprimer en pourcentage.

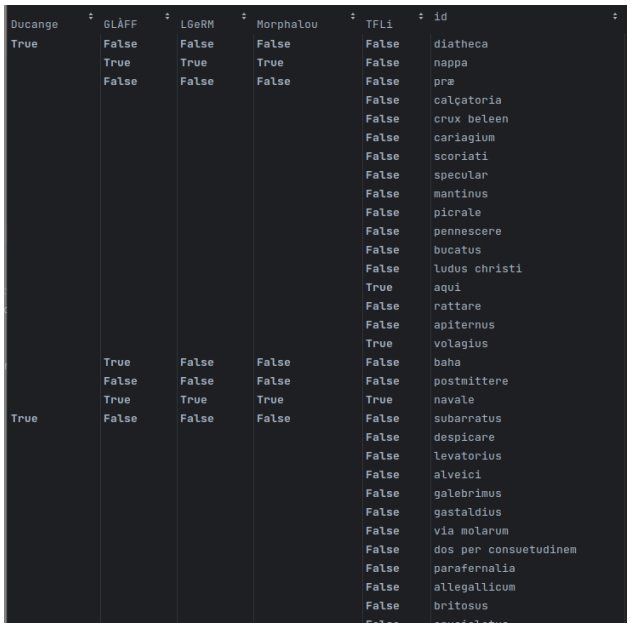
Ordre des colonnes : Les colonnes d'intersection sont triées par la valeur absolue du degré de déviation. Nous pouvons ainsi noter ce qui sort le plus de l'habituel.

En analysant ces éléments, nous pouvons extraire des informations précieuses à partir d'un UpSet Plot. Par exemple, nous pouvons identifier rapidement les ensembles qui ont une forte intersection ou ceux qui sont totalement distincts.

Avant de créer un UpSet Plot, nous devons préparer nos données. Pour chaque lexique que nous souhaitons comparer, nous devons représenter les mots sous forme d'un ensemble. Cela signifie que chaque mot unique dans le lexique constitue un élément de l'ensemble. Une fois que nous avons préparé les ensembles pour chaque lexique, nous les combinons en un Dataframe. Dans ce dernier, chaque ligne représente une forme, et chaque colonne, à l'exception de celle pour la graphie, représente son appartenance à nos ensembles. Vous en trouverez un extrait dans la figure 5.6.

À l'aide de la bibliothèque Python "UpSetPlot", nous pouvons créer facilement un UpSet Plot à partir de notre Dataframe. En utilisant la classe "UpSet" de cette bibliothèque, nous pouvons styliser les ensembles de mots que nous souhaitons comparer et générer le diagramme UpSet correspondant. L'utilisation de l'UpSet Plot en tant qu'outil visuel pour comparer les lexiques offre une perspective puissante sur les relations entre les mots. En analysant les intersections, les cardinalités et les ensembles uniques, nous pouvons identifier rapidement les similarités et les différences significatives entre les lexiques. La combinaison de Python et de la bibliothèque UpSetPlot facilite l'utilisation de cette technique dans l'analyse des données textuelles et permet une interprétation visuelle claire et concise.

On peut ainsi visualiser les liens entre les différents ensembles de formes lexicales issues de nos lexiques. Cependant, au vu du nombre d'intersections



Ducange	GLÀFF	LGeRM	Morphalou	TFLi	id
True	False	False	False	False	diatheca
	True	True	True	False	nappa
	False	False	False	False	præ
				False	calçatoria
				False	crux beleen
				False	cariagium
				False	scoriati
				False	specular
				False	mantinus
				False	picrale
				False	pennescere
				False	bucatus
				False	ludus christi
				True	aqui
				False	rattare
				False	apiternus
				True	volagius
	True	False	False	False	baha
	False	False	False	False	postmittere
	True	True	True	True	navale
True	False	False	False	False	subarratus
				False	despicare
				False	levatorius
				False	alveici
				False	galebrimus
				False	gastaldius
				False	via molarum
				False	dos per consuetudinem
				False	parafernalia
				False	allegallicum
				False	britosus

FIGURE 5.6 – Représentation sous forme de Dataframe des ensemble de formes lexicales issues des différents lexiques. Les colonnes contenant "True" ou "False" décrivent l'appartenance à l'ensemble donnant son nom à la colonne. La dernière colonne donne la graphie. (Les cases vides signifient "pareil qu'au-dessus")

possibles, la figure est assez conséquente, nous avons donc décidé de la placer en annexe (A.1). Par la suite, nous réaliserons un second UpSetPlot, en se passant cette fois-ci du lexique Ducange pour la comparaison, étant donné qu'il fournit assez peu de valeur ajoutée.

On peut ainsi constater que, proportionnellement à la taille de ces ensembles, les intersections entre le LGeRM et, respectivement, le GLÀFF et Morphalou, sont plus faibles que statistiquement attendues (intersections entourées en vert et en jaune, respectivement). Cela s'explique assez facilement par le fait que le LGeRM est, de loin, notre plus gros lexique, et que les deux autres soient les deux plus grands après lui, tout en couvrant un état de langue assez éloigné de ces deux lexiques. Ainsi, cela nous confirme une différence significative entre le LGeRM, lexique du moyen français, et nos lexiques les plus récents. Enfin, cela explique pourquoi, lorsque le LGeRM nous fournit un bon taux de lexicalité, Morphalou et le GLÀFF nous fournissent des taux de lexicalité relativement faibles. Cela nous permet aussi de s'assurer que les deux lexiques les plus récents ont une importante similarité grâce au grand degré de déviation de leur intersection (entouré en orange).

Mais, un constat plus intéressant est celui de la similarité importante entre le LGeRM et le TLFi (entouré en rouge). Ils semblent ainsi représenter un état de langue similaire, nous pouvons alors nous demander si leur association pourrait nous permettre d'obtenir de meilleurs taux de lexicalité, sans pour autant générer trop de bruit.

Ainsi, à partir des résultats obtenus, nous décidons de réaliser un second UpSetPlot, en excluant cette fois-ci le lexique *Ducange*, nous obtenons alors la figure suivante (5.7). On obtient alors une figure beaucoup plus petite, plus condensée, sur laquelle nous pouvons retrouver nos observations précédentes : Le LGeRM a peu de points communs avec le GLÀFF et Morphalou, relativement à leurs tailles respectives. Et toujours relativement aux tailles des sets, à peine plus de points communs avec le TLFi ; Le Morphalou et le GLÀFF ont beaucoup de points communs relativement à leur taille.

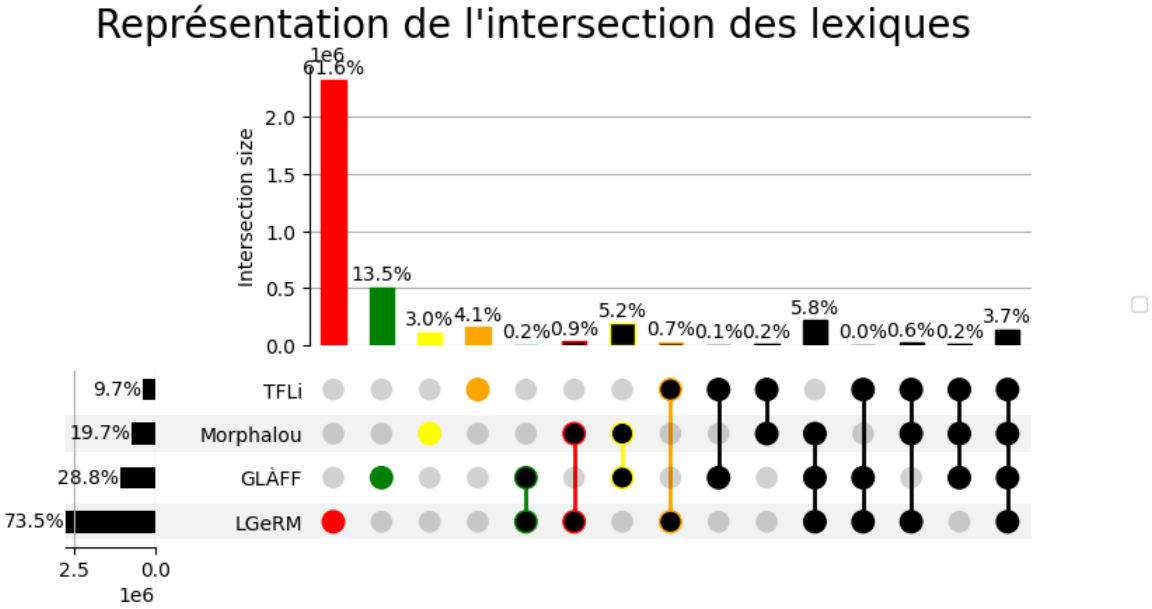


FIGURE 5.7 – UpSet plot des ensembles de mots issus des lexiques. Chaque couleur pleine représente la partie unique d'un lexique. Les couleurs de contour représentent les principales intersections en fonction du degré de déviation.

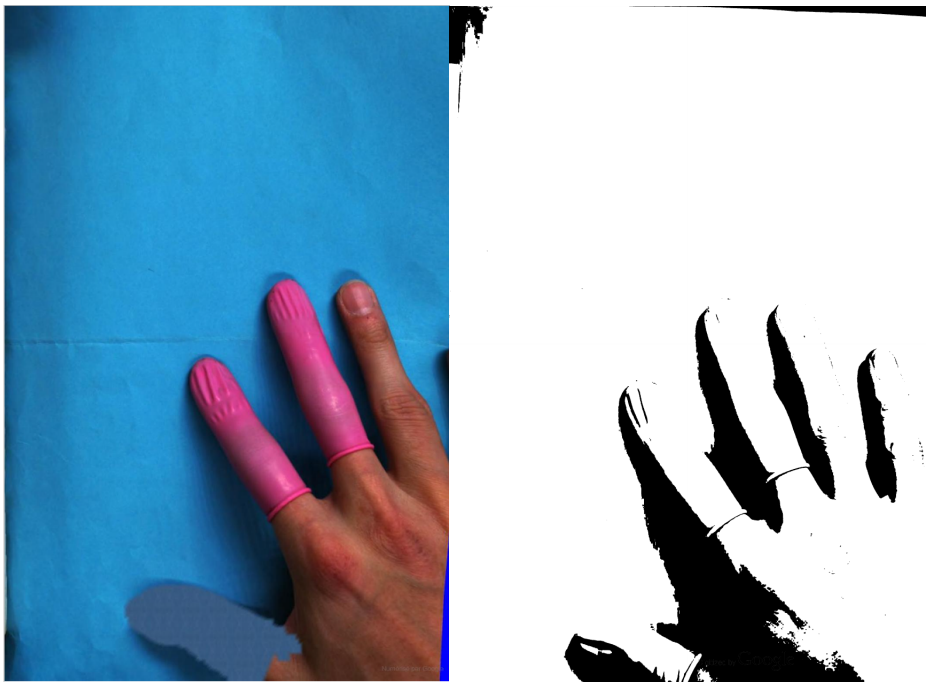
5.4 Variation de la lexicalité

Par la suite, nous avons pu tester les différentes mesures présentées sur notre corpus. La méthode qui semblait la plus intéressante, et celle que nous avons le plus développée, est celle du taux de lexicalité. Cependant, l'hétérogénéité de notre corpus, notamment en nombre de pages par document et en nombre de mots, constitue un frein à notre analyse, qui ne peut être compensé par un simple ratio du nombre de mots.

Nous avons alors décidé de nous essayer à une nouvelle approche et de regarder la variation du taux de lexicalité au sein d'un même document, nous nous intéresserons ainsi à la variation du taux de lexicalité selon la page. Plus précisément, nous nous demandons si une variation entre deux pages précises pourrait être significative, par exemple, entre la première page et la deuxième. On pourrait alors peut-être détecter la présence de pages de gardes, de pages vierges, de pages mal retranscrites, et cætera. De plus, nous pouvons également observer d'autres mesures issues de cette variation, comme la moyenne ou l'écart-type de la variation. Plus précisément, nous regarderons la moyenne et l'écart-type des valeurs absolues des différences de lexicalité entre les pages, ainsi que le signe moyen des différences (nous informant si en moyenne, on diminue ou augmente en taux de lexicalité). Cette mesure pourrait également nous permettre de négliger les erreurs dues à nos pré-traitements. En effet, en s'intéressant aux différences entre les pages, s'il nous manque certains mots à nos corpus cela devrait avoir peu d'influence sur la différence relativement à l'impact sur la mesure du taux de lexicalité telle quelle.

Au travers de cette méthode, nous cherchons également à s'affranchir de la mesure globale, à l'échelle du texte. En effet, cette dernière nous cache une grande partie des problèmes de transcriptions : elle est très utile pour trouver les textes mal retranscrits du début à la fin, mais beaucoup moins pour trouver les phénomènes locaux. Par exemple dans la thèse de J.-B. Tanguy [Tanguy 2022], nous pouvons retrouver l'image en figure 5.8. où une personne a tout simplement numérisé sa main. L'image de droite (5.8b) est issue de la version binarisé de la numérisation, c'est-à-dire, une version dans laquelle nous n'avons plus que deux types de pixels, les pixels blancs (0) et les pixels noirs (1), d'où cette appellation. C'est le genre de problèmes que la numérisation / mise à disposition du document non nativement numérique peut engendrer. Par ailleurs, on peut constater sur le document source, à gauche de la figure (5.8a), que sa main ne semble même pas reposer sur l'une des pages, de plus, l'erreur revient plusieurs fois au sein du même document.

L'ouvrage est accessible au lien suivant sur Google Books [5].



(a) Version brute (b) Version binarisé

FIGURE 5.8 – Numérisation de la 12ème page de La Nappe renversée, chez Renard, en vers burlesques, consultée sur Google Livres [Tanguy 2022]. Cette page de bruit aurait dû être retirée au moment de la mise en ligne. Mais, manifestement, les numérisations sur Google Livres ne subissent pas un contrôle-qualité des plus rigoureux. L'outil d'OCR essayant de trouver du texte sur la page, du bruit en résulte.

Ainsi, cette méthode nous a permis de découvrir une nouvelle balise qui caractérisait l'omission d'une partie du document de la transcription finale, la balise `<gap\>`. Dans le schéma XML du projet [6], nous pouvons trouver la phrase suivante : *"indicates a point where material has been omitted in a transcription"*, qui se traduirait par : "indique un endroit où le matériel a été omis de la transcription". Vous trouverez en annexe (A.2) la partie du schéma évoquant ladite balise. Cette balise apparaît alors quelques fois au sein de

5. *La nappe renversee chez Renard* : `https://bit.ly/3JnaIgq`
6. ODD du projet Antonomaz `https://bit.ly/46c37Lp`

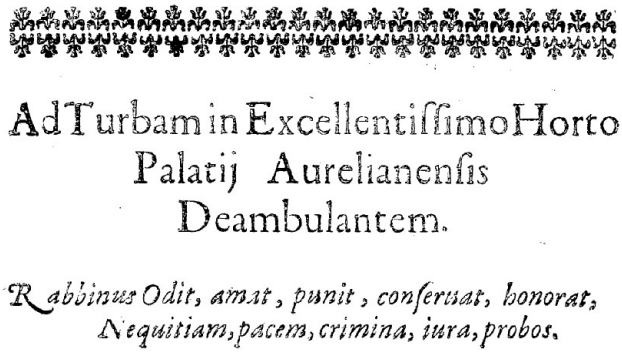
notre corpus pour signaler la non-transcripion de pages entières, comme nous pouvons le voir dans la figure, également en annexe (A.3), reprenant un exemple au sein de notre corpus [7].

En effet, au sein de nos listes d'écarts, l'omission de la transcription d'une page entière se caractérise par la présence d'une page avec un taux de lexicalité nul, ce qui ne se produit que très rarement pour une page avec un contenu. D'autre part, de très faibles taux de lexicalité nous permet le plus souvent de trouver d'autres phénomènes. Par exemple, dans la figure 5.9, nous pouvons retrouver une page entièrement en latin lorsque le reste du document est en français. Ainsi, malgré la sélection du lexique le plus adapté, cette page n'obtient qu'un taux de lexicalité de 0,2. À l'aide de nos écarts, nous pouvons alors voir que la moyenne des écarts est assez faible alors que le saut entre deux des pages est très conséquent. Cela nous permet alors d'observer rapidement que, malgré un taux de lexicalité plutôt correct, dans ce cas 0,57, une page a été particulièrement mal retranscrite. Cet exemple est tiré du document numérisé suivant [8].

Disparité dans la mise à disposition

Un second exemple est celui d'un document [9] dans lequel, après la dernière page, nous retrouvons un total de 20 pages qui sont une alternance entre une répétition de la dernière page et une version zoomée de cette même page. Ainsi, on retrouve, de nouveau, une erreur qui survient avant le processus de transcription et que l'on pourrait plutôt attribuer à la numérisation / la mise en disposition du document en version "image". Par ailleurs, on remarque une tendance dans la qualité des numérisations / mises à dispositions. La plupart des erreurs que nous avons pu trouver à cette étape-ci proviennent de Google Books. Comme l'évoquent les auteurs du Corpus étudié [Abiven et al. 2022], la qualité de ce qui est mis à disposition est très influencé par la plateforme. De manière générale, les textes sur Gallica semblent les mieux numérisés (pas de texte apparent en filigrane, pages droites et entières) et les documents de la bibliothèque Mazrine sont les plus complets (numérisation des 4 couvertures, de la tranche, mire pour se rendre compte des différences de couleur). Le même soin ne semble pas avoir été apporté aux ouvrages disponibles sur la plateforme de Google. Ainsi, les transcriptions issues de ces numérisations, qui sont en quelque sorte déjà bruitées, ne peuvent être de bonne qualité.

7. `https://bit.ly/43E22ul`
8. *Rymaille sur les plus celebres bibliotieres de Paris* : `https://bit.ly/43KN1Xv`
9. *Les profanations mazariniques* : `https://bit.ly/46dOCGY`


Ad Turbam in Excellentissimo Horto
Palatij Aurelianensis
Deambulantem.

Rabbinus Odit, amat, punit, conservat, honorat,
Nequitiam, pacem, crimina, iura, probos.

FIGURE 5.9 – Exemple d'une page en latin au sein d'un document en français.

5.5 Vecteurs liés à la lexicalité

Afin de pouvoir analyser plus globalement les expériences évoquées dans la sous-section précédente, nous avons décidé d'exploiter les vecteurs précédemment obtenus. Pour rappel, nous avions ainsi des vecteurs décrivant le taux lexicalité par page et d'autres décrivant les différences de lexicalité entre les pages pour chaque document retranscrit de notre corpus. L'idée étant qu'en séparant nos textes sur ces métriques, nous espérons obtenir des *clusters* plus bruités que d'autres. Ainsi, après une rapide analyse manuelle, pouvoir être capable de qualifier un texte comme bruité en ne regardant que le *cluster* auquel il appartient.

5.5.1 Dimensions variables

Cependant, en l'état, nous ne pouvons obtenir une matrice décrivant les différents textes de notre corpus. De fait, la longueur de nos vecteurs étant directement liée au nombre de pages du document qu'ils décrivent et notre corpus étant très hétérogène sur ce point, entre une page pour les plus courts et de plus de 600 pages pour les plus longs, nos vecteurs sont de longueur très variée. Nous avons alors envisagé deux méthodes :

Sous-corpus en fonction du nombre de pages

La première consisterait à séparer les vecteurs en groupe selon leur taille. Ainsi, le corpus serait partitionné par longueur de document, puis un *clustering* serait réalisé sur chaque groupe. Cependant, cela provoquerait deux soucis majeurs, dans l'analyse de ce corpus :

— Cela nous rajouterait une division, alors que l'idée de base était de séparer les textes uniquement par le clustering, cette approche créerait

une seconde séparation. La lecture en serait alors complexifiée, tout comme la "qualification" des groupes, c'est-à-dire déterminer s'ils sont bruités ou non.
— Certains documents (surtout dans les grands nombres de pages) sont les seuls à avoir ce nombre de pages, ils ont alors un vecteur d'une taille différente de tous les autres.

Réduction de dimmensionnalité

La seconde méthode consisterait quand à elle à réduire le nombre de dimensions des vecteurs, on prend alors tous les vecteurs de dimension n auxquels on applique un algorithme de réduction de dimensionnalité, retirant ainsi une dimension, on obtient alors des vecteurs de dimension n-1, que l'on réduit, en même temps que les vecteurs n-1 préexistants.

Cette opération est répétée jusqu'à arriver à une dimension de 2. Nous avons choisi de réduire jusqu'à 2 le nombre de dimensions, d'une part, pour des questions de visualisation. En effet, nous allons en grande partie analyser les données à la main pour essayer de paramétrer au mieux le nombre de *clusters*, mais également pour trouver lesquels correspondraient à une sortie bruitée.

D'autre part, cette réduction à 2 se base sur les travaux autour de la question du fléau de la dimensionnalité. En effet, dans l'article *Classification supervisée et non supervisée des données de grande dimension* [Bouveyron and Girard 2009], les auteurs explorent la question de la réduction de dimension dans le cadre de l'évaluation non supervisée, en utilisant notamment le *scree-test* [Cattell 1966] (test d'accumulation de variance, souvent utilisé pour produire la "méthode du coude"). Ils trouvent ainsi que pour leur corpus, une dimension faible, aux alentours de 4 correspond au mieux.

De plus, réduire jusqu'à une dimensionnalité de 2 permet d'intégrer à notre *cluster* tous les documents avec un vecteur d'origine d'au moins deux dimensions, c'est-à-dire au moins deux pages dans le cas des vecteurs sur la lexicalité et au moins trois pour les vecteurs sur les différences.

5.5.2 Concaténation de vecteurs

Additionnellement, nous avons essayé de construire des vecteurs en concaténant ceux précédemment vus, cela nous donne un vecteur qui encode à la fois le taux de lexicalité et la différence avec la page précédente, permettant alors de donner du "poids" aux deux métriques simultanément. Nous voulions rajouter ces vecteurs à nos expériences car nous pensions qu'il pourraient apporter une vraie valeur. Les vecteurs de lexicalité encodent peu les différences

entre les pages lorsque les vecteurs de différences n'encodent pas du tout la lexicalité. Ainsi, cette concaténation nous semblait intéressante à étudier.

5.6 Discrimination des textes - Clustering

À l'aide des vecteurs précédemment obtenus, nous pouvons désormais essayer des méthodes de *clustering*. Le *clustering*, que l'on pourrait traduire par regroupement en français, consiste à séparer les vecteurs en fonction de leurs coordonnées, formant alors des *clusters* (ou groupes) regroupant les points en fonction de leur adjacence. Plus deux points sont proches, plus leurs coordonnées sont similaires. Ainsi, l'adjacence de deux points permet aussi de caractériser la similarité entre leurs vecteurs correspondants car les coordonnées des points sont identiques à celles de leur vecteur.

5.6.1 Algorithme de clustering

Après avoir essayé plusieurs algorithmes, nous avons décidé d'employer l'algorithme *KMeans* [Sinaga and Yang 2020]. Ce dernier est assez adapté pour des vecteurs/points à deux dimensions (géométrie plate). De plus, il s'exécute bien plus rapidement que les autres algorithmes testés (DBSCAN, AgglomerativeClustering, AffinityPropagation, Spectral*Clustering* et OPTICS) [Pedregosa et al. 2011], ce qui nous permet de tester divers ajustements dans le temps imparti. Il ne faut cependant pas négliger les défauts de l'algorithme : Sa mauvaise gestion des groupes à taille hétérogène et d'un trop grand nombre de groupes [Buitinck et al. 2013] [10]. De plus, contrairement à certaines autres méthodes (notamment DBSCAN et OPTICS) , le k-means nécessite que l'on trouve le nombre adéquat de *clusters*.

5.6.2 Déterminer le nombre optimal de *clusters*

Lorsque l'on ne connaît pas le nombre de classes (groupes) souhaités, plusieurs méthodes sont envisageables. Dans le cadre de cette étude, nous utiliserons deux mesures distinctes :

Méthode du coude - WCSS

Le *Within-Cluster-Sum-of-Squares (WCSS)* qui calcule la somme de la distance entre chaque point et le centroïde (le point moyen) de son cluster.

10. Documentation ScikitLearn pour le *clustering* : `https://scikit-learn.org/stable/modules/clustering.html`

Cette mesure permet de réaliser la "méthode du coude" consistant à créer une courbe représentant le WCSS en fonction du nombre de *clusters*. Cette mesure décrivant la distance entre chaque point et son centroïde, au fur et à mesure que le nombre de *clusters* augmente, en employant l'algorithme k-means, cette distance ne peut que diminuer ou rester identique. Et ce, jusqu'à ce que chaque point ne soit son propre *cluster* (WCSS = 0). L'idée de représenter la fonction avec une courbe est alors de regarder à quel moment la fonction diminue plus lentement (tangente plus proche de 0) en trouvant le "coude" de la fonction, c'est-à-dire l'endroit auquel la courbe tourne abruptement. Cela permet de savoir à quel moment augmenter le nombre de *clusters* ne crée plus autant des groupes spécifiques qu'auparavant. [Mahendru 2019] [Tomar 2022]

Silhouette score

La seconde mesure que nous utiliserons est la *Silhouette score*. Pour résumer brièvement, contrairement au WCSS qui ne mesure seulement la distance entre un point et son centroïde, cette mesure permet de mesurer a quel point chaque point est similaire au reste de son cluster, en comparaison au reste des points. Pour ce faire, on calcule la différence entre la distance moyenne du point avec ceux du *cluster* le plus proche (a l'exception du sien) et la distance moyenne du point avec ceux de son cluster, le tout divisé par le maximum des deux moyennes. [KumarI 2023]

$$a(i) = i \notin C, \min(\frac{\sum_{j \in C} d(i,j)}{len(C)})$$

$a(i)$ représente ainsi la plus petite distance moyenne entre les points d'un autre *cluster* (C) et i.

$$b(i) = i \in C, \frac{\sum_{j \in C, i \neq j} d(i,j)}{len(C) - 1}$$

$b(i)$ représente ainsi la distance moyenne entre i et les points de son *cluster* (C).

$$silhouette(i) = \frac{a(i) - b(i)}{\max(a(i), b(i))}$$

Nous pouvons alors nous servir de cette mesure de deux façons :

— La première, assez semblable à la méthode du coude, consiste à tracer la courbe du score de silhouette moyen (moyenne de tous les points) pour un nombre donné de *clusters*. Vous trouverez un exemple en annexe A.5.

Cette fois-ci, contrairement au WCSS, on recherche le nombre de *clusters* permettant d'avoir le meilleur silhouette score. Ainsi, contrairement à la méthode du coude, où, pour un vrai jeu de données, le "coude" peut être difficile à trouver, la courbe de la silhouette est généralement plus facile à lire. On pourrait également demander à combien de *clusters* le silhouette score atteint son maximum. Cependant, cette mesure est encore très globale et peut masquer d'importants écarts entre les *clusters* [Mahendru 2019].

— La seconde méthode consiste à afficher, en séparant par cluster, le silhouette score de chaque point sous forme d'histogramme croissant. [Buitinck et al. 2023] Cela permet de représenter visuellement la distribution complète, et ainsi de se rendre compte si certains points sont mal représentés par leur cluster, de voir si la taille des *clusters* est homogène ou non, de voir si un *cluster* en particulier présente un mauvais silhouette score et cætera. Ainsi, on cherchera a optimiser le silhouette score moyen, mais également l'homogénéité en taille des *clusters* et la proportion de points bien représentés dans chaque cluster. [Tomar 2022] Empiriquement, on constate que ce dernier paramètre, à quel point chaque point est bien représenté par son cluster, semble être la métrique la plus importante. Car dans notre cas, le corpus étant assez hétérogène, de très petits *clusters* peuvent se former. Vous trouverez un exemple de cette représentation, tiré de nos analyses au sein de la figure 5.10.

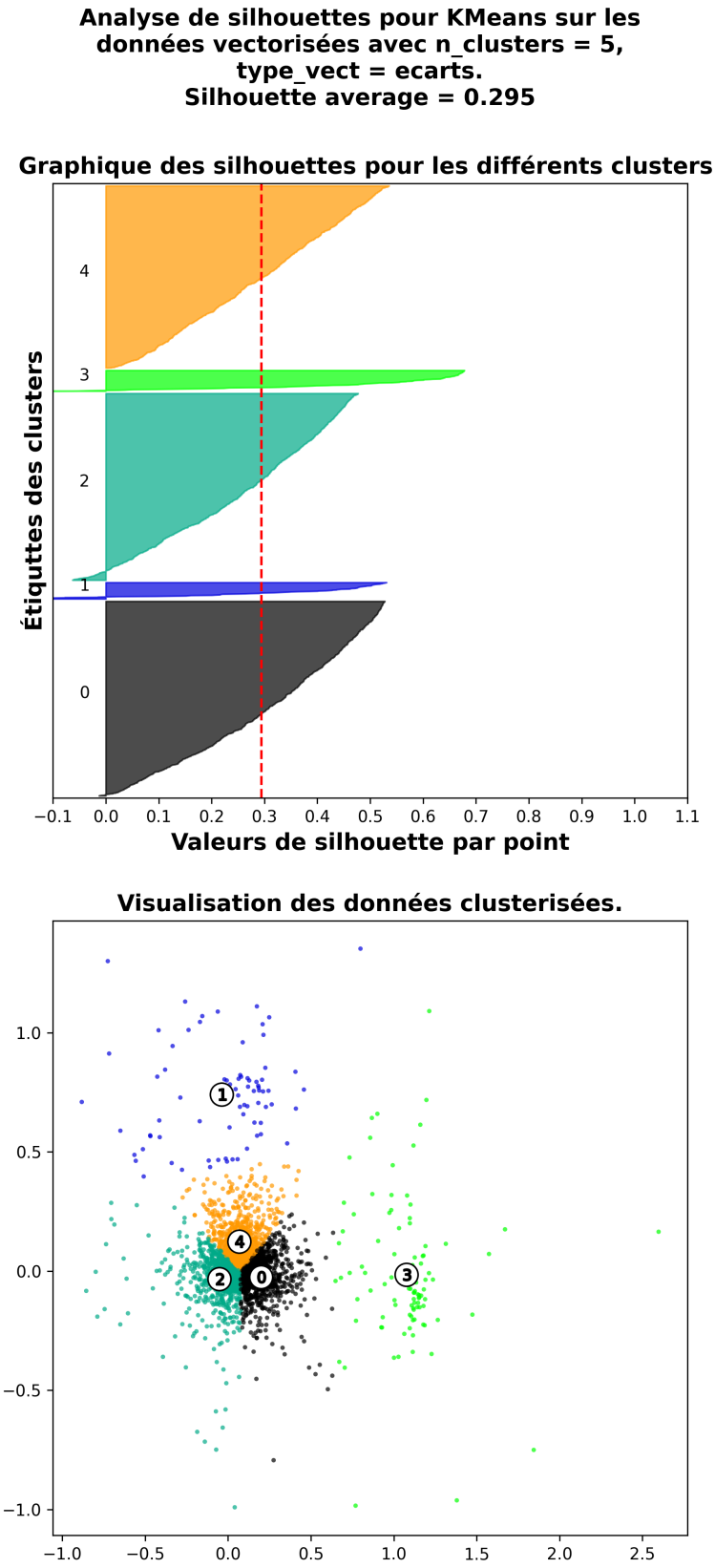


FIGURE 5.10 – Exemple d'analyse des silhouette-scores. En haut, on retrouve l'histogramme de la valeur des silhouettes par point, séparés par *clusters*. En dessous, on retrouve la représentation des points selon leur coordonnées. Le même *cluster* a la même couleur dans ces deux graphes.

5.7 Métriques permettant le calcul de similarités entre sections

L'évaluation de la similarité textuelle consiste à comparer des textes pour savoir à quel point des textes sont éloignés ou non par un score de similarité appelé *métrique*. Les métriques utilisées sont des mesures qui varient entre 0 et 1 (inclus). 0 signifie que la section précédente et la section suivante ne sont pas similaires et 1 signifie que les sections sont similaires. Pour ces métriques qui varient dans cet intervalle nous avons :

La distance cosinus : cette métrique s'effectue sur des valeurs réelles. C'est la métrique la plus couramment utilisée [Baeza-Yates et al. 1999]. Elle mesure la ressemblance entre sections précédente et section suivante *section 0* et *section 1* et ainsi de suite en calculant le cosinus de l'angle entre les représentations vectorielles des sections.

$$\cos_{\text{sim}}(\vec{section}1, \vec{section}2) = \frac{\vec{section}1 \cdot \vec{section}2}{\|\vec{section}1\|\|\vec{section}2\|}$$

$\vec{section}1$ et $\vec{section}2$ sont des vecteurs.

Avant donc d'appliquer cette métrique tout comme les autres également, il faut procéder par une vectorisation en matrice dense ou creuse. Nous avons le choix entre ces deux vectorisations (voir 5.2) pour les tokens des sections, la matrice creuse qui contient des éléments non nuls, c'est-à-dire que chaque section vectorisée contient des vecteurs non nuls, les zéros n'apparaissent jamais en mémoire ou la vectorisation dense dans laquelle on a tous les vecteurs. On applique les deux vectorisations avec `.toarray()` et `.tocsr()` sur les mots des sections pour un document ou sur les n-grammes d'étiquettes morphosyntaxiques. Dans le tableau 5.2, on ne constate pas de différence entre les deux vectorisations :

Section	Similarité cosinus (toarray)	Similarité cosinus (tocsr)
0	0.97829	0.97829
1	0.58771	0.58771
2	0.58957	0.58957
3	0.44080	0.44080
4	0.50028	0.50028
5	0.63289	0.63289
6	0.70259	0.70259
7	0.55180	0.55180
8	0.47239	0.47239
9	0.45834	0.45834
10	0.79093	0.79093

TABLE 5.2 – Comparaison des similarités cosinus pour les représentations `toarray()` et `tocsr()` - valeurs arrondies à 5 chiffres

Distance de Jaccard : L'indice de Jaccard [Jaccard, 1901] est le rapport entre la taille de l'intersection des éléments considérés et la taille de l'union des documents considérés. Les sections ne sont pas représentés en vecteurs, mais comme des ensembles de termes et la similarité est située entre [0, 1].

$$J(section1, section2) = \frac{section1 \cap section2}{section1 \cup section2}$$

Distance de Canberra : cette distance[11] a été développée par G.N Lance puis modifiée par W.T. Williams en 1967 d'après Faisal et al. [2020]. Cette distance est utilisée quand on a besoin de trouver la distance entre des paires de points ou les données sont autour de l'origine dans l'espace vectoriel, selon Nagar and Khunteta [2016] et Moh'd B and Roberts [1996]. Le score de similarité n'est plus entre [0, 1]. C'est en réalité une mesure de dissimilarité, elle donne par exemple cette valeur `1016.7005677072997`. Plus cette valeur est élevée, plus les sections ne vont pas se rapprocher.

$$d_{section1,section2} = \sum_{k=1}^{n} \frac{|section_i - section_i|}{|section_i| + |section_i|}$$

d est la distance entre section1 et section2. La métrique suivante la distance de Levenshtein ne se situe pas également entre 0 et 1. Cela est intéressant puisqu'on pourra comparer les deux métriques.

11. `https://en.wikipedia.org/wiki/Canberra_distance`

Distance de Levenshtein : La distance de Levenshtein [Levenshtein et al. 1966] est une mesure de dissimilarité (comme la métrique précédente) entre deux séquences par exemple les mots, pour calculer le nombre d'opérations nécessaires pour transformer une chaîne A en une chaîne B et donne le nombre de caractères à supprimer, insérer ou remplacer pour passer de A à B. Cette distance est utilisée également pour calculer le taux d'erreur d'une transcription de mots ou WER comme mentionné dans l'introduction pour les variations qualitatives. La distance de Levenshtein en ce qui concerne la détection de variations stylistiques "calcule la similarité entre les représentations sous forme de chaînes de caractères des documents d_1 et d_2" [Negre 2013], dans notre cas *section 1* et *section 2*. Les opérations suivantes se déroulent pendant le calcul de distance entre section précédente et section suivante :

- substitution d'un caractère de *section 1* en un caractère de *section 2*,
- ajout dans *section 1* d'un caractère de *section 2*,
- suppression d'un caractère de *section 1*.

La distance de Levenshtein obtenue sera le score de similarité ou dissimilarité entre la section précédente (section 1) et la section suivante (section 2).

Dans les résultats de comparaison des similarités, nous aurons d'autres similarités comme la distance de *Kulsinki*, *Bray - Curtis* etc. pour observer les variations de similarité avec plus de métriques. Il faut noter également que ces similarités sont comparées avec des vectorisations de type sac de mots et avec TF-IDF.

5.7.1 Vectorisation en sac de mots (ou *bag of words)*

La vectorisation en sac de mots est une technique très souvent utilisée en TAL et Recherche d'Information (RI) : *"Elle permet d'associer à un texte un descripteur unique basé sur l'ensemble des mots-formes qu'il contient."* [Claveau 2014].Dans notre cas quand on utilise le *BOW* pour représenter numériquement notre texte ou notre section, on attribue à chaque mot uniquement un vecteur numérique. Cette technique n'est pas forcément adaptée pour certaines tâches de TAL car on ne prend pas en compte la structure interne du texte. Un corpus n'est donc pas un sac de mots ; dans Rastier [2005], l'auteur affirme que : *(...) l'objet empirique de la linguistique est fait de textes oraux ou écrits, non de mots ou de phrases - qui ne s'observent pas à l'état isolé (...)*.. Dans notre tâche, nous le faisons en Python `CountVectorizer`

de Scikit-Learn. La chaîne de traitement est la suivante :

- Tokenisation : on découpe chaque section en mots.
- Élaboration du vocabulaire : le vocabulaire est élaboré à partir des mots trouvés.
- Calcul des occurrences de chaque mot sans prendre en compte le poids des mots (les mots outils sont pris en compte)
- Représentation vectorielle : chaque section est vectorisé et elle contient des vecteurs numériques.

Il faut donc pouvoir prendre en compte le poids des mots en calculant le taux d'apparition des mots dans chaque section. C'est ce que va permettre TF-IDF dans la prochaine sous-section.

5.7.2 Vectorisation avec prise en compte de l'importance du poids des mots : TF-IDF

Avec TF-IDF, on traite plus les sections comme des sacs de mots. TF-IDF ou $TF_{(m,section)}$ = *Term-Frequency*est le taux d'apparition d'un mot m dans une section d et $IDF_{(m)}$ = *Inverse Document Frequency* est une mesure globale pour un mot m. Dans la mesure TF si le mot est fréquent il est important dans la section 1 par rapport aux autres mots de la section 2. La mesure se trouve dans l'intervalle $[0, \infty]$. Cette mesure quantifie l'importance d'un mot pour la caractérisation d'une section. Si un mot se trouve à la fois dans la section 1 et la section 2, il ne contribue pas de manière significative et son score IDF sera faible.

$$\boxed{\text{TF-IDF}_{(m,\text{section})} = \text{TF}_{(m,\text{section})} \times \text{IDF}_{(m)}} \tag{5.1}$$

5.8 Mesurer la richesse lexicale de chaque section avec le taux de lexicalité

Le taux de lexicalité permet de déterminer la proportion du nombre de mots présents dans un dictionnaire de référence par rapport au nombre de mots dans chaque section. Ce taux de lexicalité va permettre de mesurer la variabilité des sections de chaque article et pouvoir observer par la suite à quels endroits nous détectons des changements ou à quel point une section est différente d'une autre. Le lexique de référence ne couvre certainement pas tous les mots de la langue comme dit dans la section 5.3 il faut un lexique assez fournit pour pouvoir comparer.

$$\boxed{\text{Taux de lexicalité} = \frac{\text{Nombre de mots communs}}{\text{Nombre de la fréquence de chaque mot}}} \quad (5.2)$$

5.8.1 GLÀFF

Nous avons donc utilisé le GLÀFF (Gros Lexique À tout Faire du Français [Sajous et al. 2013] pour évaluer la variabilité qualitatif du corpus OCRisé du 17ème et 18ème siècle et la variabilité stylistique du corpus numérique datant du 20ème et 21ème siècle. C'est un lexique assez fourni, il contient **1 406 857** formes, sachant que le dictionnaire de l'Académie française contient au moins environs 60 000 mots et le TLFi environ 100 000 mots. [12]

5.8.2 Corpus de l'Est Républicain

Nous utilisons également le corpus de l'Est Républicain [Franck Sajous] qui est un corpus qui contient des articles de journal écrit entre 1999 et 2003. [13]. Nous utilisons donc un seul corpus concatené, des articles parus en 2002 et 2003(le corpus de 1999 contient des tokens avec des formes étiquetées). On obtient donc un corpus composé d'un nombre total de **242 731** mots. Le nombre total de mots a été obtenus avec une expression régulière `[a-zA-ZÀ-ÿ][a-zA-ZÀ-ÿ-]*`.

5.9 *Clustering* des sections en fonctinon des mots

Le *clustering* est tâche d'apprentissage non-supervisée qui consiste à regrouper les éléments d'un jeu de données en groupes similaires appelés *clusters* en calculant les distances entre k *clusters*, k étant le nombre de *clusters*. L'intérêt de la non supervision est de regrouper les données sans chercher une étiquette pré-définie, de mettre en évidence des propriétés attendues et d'en découvrir de nouvelles [Baledent and Lejeune 2019]. Dans notre cas, nous essayons de regrouper les paires de sections qui sont plus ou moins similaires.

12. `https://www.dictionnaire-academie.fr/article/QDL056`
13. `http://redac.univ-tlse2.fr/corpus/estRepublicain.html`

5.9.1 *Clustering* plat des sections sur les mots

Nous utilisons, dans un prermier temps un algorithme de *clustering* appelé *KMeans* Sinaga and Yang [2020] ou *K-moyens* en français. La principale caractéristique des *k-means* est de regarder le nombre de k et d'identifier le nombre de k centroïdes et ensuite affilier chaque point au *cluster* le plus proche. Le nombre de centroïdes est défini par l'utilisateur. Typiquement, ici, on ne connaît pas le nombre optimal de *clusters* raison pour laquelle la détection des variations qualitatives utilise la méthode du coude *WCSS* et le *Silouhette score*. En ce qui concerne le *clustering* des sections avec *k-means*, on détermine le nombre de *clusters* à la volée par exemple `n_\textit{clusters}=5`. Mais cela peut générer des erreurs lorsque le nombre de *clusters* spécifié est inférieur au nombre d'échantillons (nombre de sections). Cela peut être en effet résolu avec la méthode du coude ou bien spécifier un nombre maximum de *clusters* pour prendre en compte les échantillons qui sont inférieurs à ce nombre maximum ou réduire le nombre d'échantillons, en l'occurrence, réduire le nombre de sections que l'on a dans chaque document.

5.9.2 *Clustering* hiérarchique ascendant (CAH) des sections : *BOW* vs *TF-IDF*

Nous avons par la suite tenté d'observer le rapprochement des sections par du *clustering* hiérarchique ; type de *clustering* permettant de construire des *clusters* imbriqués en les fusionnant ou les divisant de manière successive. Les échantillons sont intégrés dans une structure hiérarchique globale. Cette hiérarchie de *clustering* est représentée sous forme d'arbre que l'on appelle *dendogramme*. Le sommet de l'arbre représente un *cluster* unique. On rassemble progressivement les sections qui sont seules dans une classe pour les intégrer dans des classes plus grandes. Nous utilisons donc la méthode de Ward Ward Jr and Hook [1963] qui décrit une procédure pour former de manière hiérarchique des groupes homogènes. L'hypothèse est que les informations les plus précises sont disponibles lorsque chaque individu constitue un groupe. Le critère de Ward est défini par la formule [14] :

Inertie totale du nuage :

$$I_t = \frac{1}{n}\sum_{i=1}^{n} d(e_i, g)^2$$

14. Source : `https://fr.wikipedia.org/wiki/MÃľthode_de_Ward`

Inertie interclasse :

$$I_e = \frac{1}{n} \sum_{i=1}^{k} n_i \times d(g_i, g)^2$$

Inertie intraclasse :

$$I_a = \frac{1}{n} \sum_{i=1}^{k} \sum_{e \in G_i} d(e, g_i)^2$$

Nous observerons donc les résultats des dendogrammes générés avec la méthode *Ward* et la méthode *Complete* dans la section résultats.

5.10 Intersection des n-grammes d'étiquettes morphosyntaxiques entre sections

Nous avons également pour finir, tenté de retrouver les motifs ou n-grammes d'étiquettes morphosyntaxiques de type ('adp', 'det', 'noun', 'det', 'noun'), ('noun', 'adp', 'det', 'noun', 'adp') avec contrainte sur la longueur dans chaque section. Comme mentionné pour les travaux de Baledent and Lejeune [2019] concernant la tâche de recherche de motifs, nous spécifions une longueur maximum de 5 pour la recherche de motif pour plus de spécificité et moins de bruits de motifs fréquents. Nous avons utilisé Spacy Honnibal and Montani [2015] pour le *clustering* des sections selon les unigrammes d'étiquettes morphosyntaxiques. Mais pour cette tâche de recherche d'intersection des étiquettes, on utilise le modèle de Stanford d'étiquettage pré-entrainé en Java, Standford POS Tagger Toutanova et al. [2003] en version 4.2.0 de 2020 disponible en plusieurs langues dont le français dont la principale technique est l'utilisation de dépendances syntaxiques [15]. On utilise donc le modèle du français `french-ud tagger` utilisant le modèle U Nivre and de Marneffe [2016] pour le français entraîné sur le jeu de données French GSD (UD v2.2 McDonald and Nivre [2013] selon les étiquettes définies par Petrov et al. [2012], sachant qu'on a également 3 autres jeu de données : Sequoia Candito and Seddah [2012], Spoken [Lacheret et al. 2014] et PartTUT [Sanguinetti and Bosco 2015]. Le *treebank* UD est le plus fourni en termes de tokens.

15. Un cours sur Universal Dependencies de Karën FORT : `https://members.loria.fr/KFort/files/fichiers_cours/UD_GREW.pdf`

Treebank	#Tokens	#Sentences	Genres
GSD	389,363	16,342	Blogs, News Reviews, Wiki
Sequoia	68,615	3,099	Medical, News Non-fiction, Wiki
Spoken	34,972	2,786	Spoken
ParTUT	27,658	1,020	Legal, News, Wikis
FTB	350,930	27,658	News

Table 1: Statistics on the treebanks used in POS tagging, dependency parsing, and NER (FTB).

FIGURE 5.11 – Statistiques pour les jeux de données french UD - Martin et al. [2019]

Les sections étant donc étiquetées, nous calculons les n-grammes les plus fréquents de chaque section avec `nltk.FreqDist` de la bibliothèque NLTK [16]. Avec cela, on est en mesure de ressortir les n-grammes d'étiquettes morpho-syntaxiques de longueur 5 les plus fréquents de chaque section. Par la suite, on a également fait l'intersection des n-grammes de chaque section. La première section de l'article est toujours vide car elle ne fait aucune intersection avec une section précédente. Si la section suivante à une intersection de n-grammes de longueur 5 avec la section précédente, alors on affiche le ou les n-grammes en question. L'affichage de ces n-grammes peut être amélioré avec leurs tokens respectifs, c'est-à-dire que pour un motif 'noun', 'adp', 'det', 'noun', 'adp', on peut l'afficher avec les mots qui correspondent à côté.

16. `https://www.nltk.org/`

Chapitre 6

Résultats et discussion

6.1 Détection de sauts qualitatifs appliqués au corpus des Mazarinades - Pistes explorées

Ainsi, au fil de nos expériences, nous avons pu identifier diverses méthodes permettant de qualifier le bruit au sein de notre corpus.

TTR La première mesure explorée, le *Type-Token Ratio* (TTR) n'a pas été particulièrement concluante. Nous pensions pouvoir détecter un lien entre la proportion entre taille du vocabulaire et taille du texte et le niveau de bruit de ce dernier. Nous pensions alors que plus un texte serait bruité, plus ce dernier aurait un grand vocabulaire, proportionnellement à sa taille globale. Cependant, l'augmentation du TTR s'est plutôt retrouvée corrélée à la taille du document. En effet, plus un document est court, moins les mots qui le composent avaient l'occasion de se répéter. La réutilisation des mots étant rare, la taille du vocabulaire se trouvait alors très proche de celle du texte complet. Dans le corpus des Mazarinades, la taille des documents est très variable, certains faisant une simple page lorsque d'autres dépassent allègrement les 500 pages (et avec des pages de longueur tout autant inégale). Sur un tel corpus, la longueur de chaque texte avait alors un grand impact sur la mesure, la rendant utilisable en l'état. Nous avons alors essayé de pondérer cette mesure par rapport à la taille des textes, mais une nouvelle fois sans succès. À ce moment, nous avons rencontré un nouveau problème avec cette mesure, les textes en latin. Ces derniers, de par les déclinaisons, se retrouvent avec un TTR particulièrement élevé, rajoutant une nouvelle variable faisant varier le

TTR.

Hapax ratio En essayant une mesure différente mais essayant d'exprimer la même idée, le taux d'hapaxes, nous nous sommes retrouvés confrontés au même verrou. L'idée qui nous avait conduit à utiliser cette mesure était que dans la plupart des cas, une erreur de retranscription au sein d'un texte devrait provoquer l'apparition d'une forme unique, un hapax. Dans ce cas, plus un texte aurait eu d'hapaxes, proportionnellement à la taille totale du texte, plus ce dernier aurait eu de chances d'avoir été mal retranscrit. Cependant, cette mesure s'est retrouvée une nouvelle fois corrélée à la taille du texte. De nouveau, plus un texte est long plus les mots qui le composent risquent de se répéter, on a alors moins d'hapaxes et donc un taux d'hapaxes qui diminue [Lardilleux 2010]. Ainsi, pour les mêmes raisons que la première mesure, le taux d'hapaxes semble alors inadapté pour la mesure du bruit au sein de notre corpus.

Taux de lexicalité En nous servant d'une ressource externe, les lexiques, nous avons pu utiliser une autre mesure : le taux de lexicalité. Le taux de lexicalité mesure le taux de mots qui sont lexicalisés, c'est-à-dire qui appartiennent au lexique testé (dans notre cas, nous considérons les chiffres comme lexicalisés). Cette mesure est assez intéressante, car elle permet de s'affranchir en partie de cet obstacle qu'était la longueur des documents. L'objectif étant toujours le même, être capables de localiser lorsqu'une grande quantité d'erreur de transcription surviennent, ainsi, un mot dans lequel un ou plusieurs caractères changent a de grandes chances de donner un mot qui ne fait pas partie du lexique. De ce fait, hormis pour les entités nommées et les quelques mots hors-lexique, le taux de lexicalité nous a permis d'encoder assez efficacement les erreurs au sein de nos documents.

Dans la figure 6.1 vous retrouverez un graphique de la distribution du taux de lexicalité au sein de notre corpus. Cette mesure nous permet alors de récupérer les textes qui sont extrêmement mal OCRisés, c'est par exemple le cas du texte ayant le pire taux de lexicalité (32 %) [1] ou ceux qui sont presque parfaitement OCRisés [2].

Cependant, lorsqu'un texte comporte des erreurs plus localisées, son taux de lexicalité est alors assez proche du reste. Certains textes contiennent également beaucoup d'entités nommées [3], rendant alors leur taux de lexicalité assez faible. Pour finir, cette mesure agrège

1. Exemple de texte dans une langue non couverte : `http://bit.ly/OCR-allemand2`
2. Texte avec le meilleur taux de lexicalité : `http://bit.ly/OCR-best2`
3. *Rymaille sur les plus celebres bibliotieres de Paris* : `https://bit.ly/43KN1Xv`

toutes les informations contenues au sein du texte. nous ne savons alors pas si un taux moyen est dû à un faible bruit constant ou à un fort bruit ponctuel (par exemple une page entière de bruit).

Vecteurs de lexicalité et clustering Afin d'essayer de pallier le problème du bruit localisé, nous avons alors créé des vecteurs à partir des taux de lexicalité par page ainsi que des écarts de lexicalité par page. Une fois ces vecteurs réduits aux mêmes dimensions, nous avons alors utilisé l'algorithme de *clustering* k-means afin de séparer nos vecteurs. À cette étape, nous nous sommes servis du silhouette score combiné au WCSS afin de déterminer le nombre idéal de *clusters* pour chaque type de vecteur. Cependant, dans certains cas, nous n'étions pas sûrs du nombre optimal de *clusters*, nous avons alors décidé de sélectionner deux nombres de *clusters* pour chaque vectorisation. Nous nous retrouvons alors avec un total de 6 *clusters*. Par la suite, nous avons annoté manuellement un texte de chaque cluster, pour chaque cluster. Afin de pouvoir couvrir l'ensemble de ces textes, nous avons alors pris le parti de faire une annotation assez sommaire : après lecture des textes, nous leur attribuons une note sur dix.

Une fois les textes annotés, nous nous en sommes servis pour évaluer les autres textes : chaque évaluation donnait une note au cluster, ainsi pour chaque clustering, les textes se voyaient attribués la note de leur cluster, par la suite une moyenne était calculée afin d'attribuer une note sur dix à chaque texte. Nous avons ainsi une note pour chaque texte, en accord avec leur appartenance à chaque cluster. Cela nous permet alors de trouver assez facilement les textes les plus bruités, il nous suffit de regarder les pires notes pour retrouver des textes avec des erreurs de retranscription. Ainsi, on obtient 257 textes avec une note inférieure ou égale à 4, jusqu'à cette note, les retranscriptions sont particulièrement altérées [4].

4. Exemple de texte ayant reçu la note de 4/10 `https://bit.ly/3XgANDH`

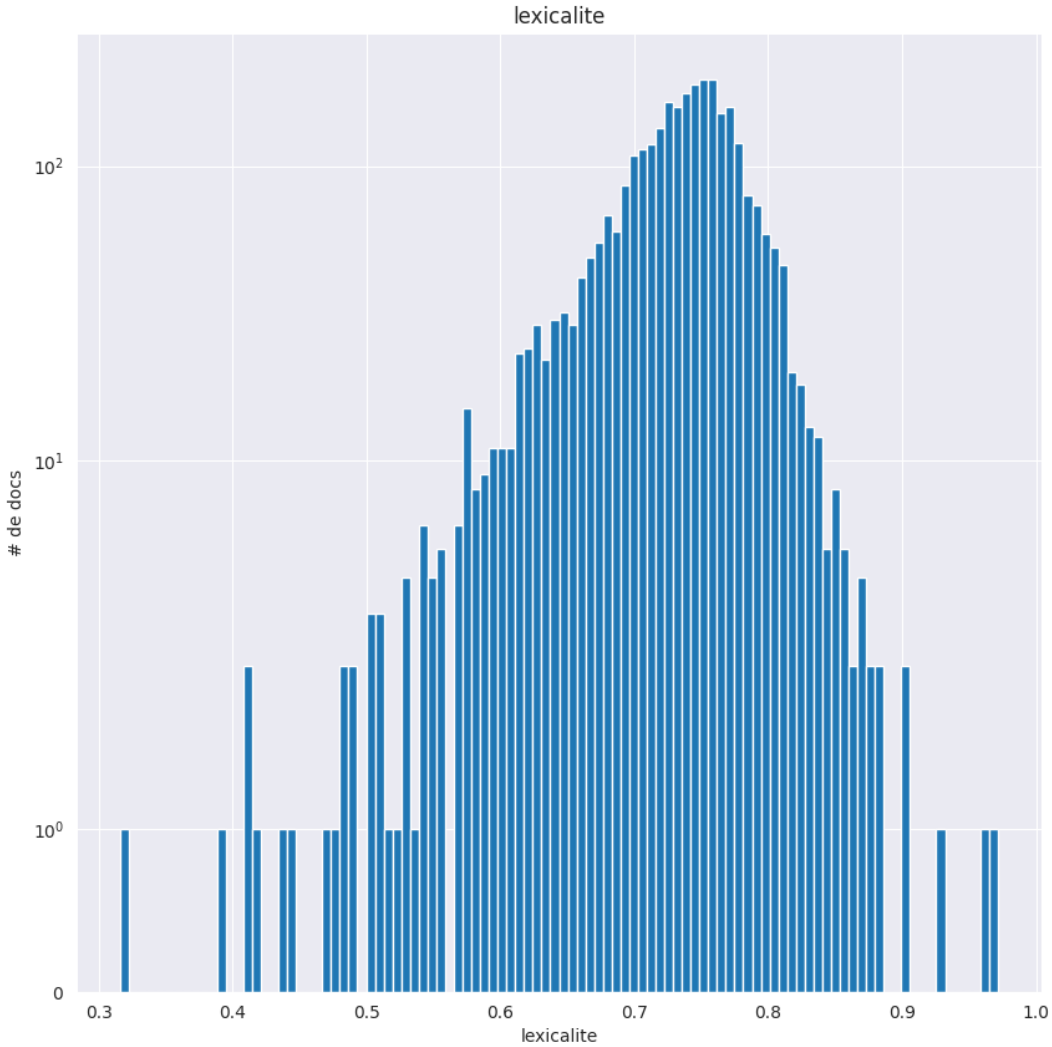


FIGURE 6.1 – Distribution des textes du corpus en fonction du taux de lexicalité. On peut remarquer une distribution gaussienne. Les valeurs les plus basses représentent alors les textes très bruités, cependant on ne peut pas détecter les textes qui n'ont que de courts passages bruités.

Cette mesure nous permet alors de trouver efficacement un grand nombre de textes bruités. Cependant, nous remarquons également certaines limitations :

— Les textes dans des langues étrangères, par exemple en allemand [5] sont assez souvent retrouvés parmi les plus bruités. Ce comportement est sûrement dû aux lexiques employés, il faudrait alors pouvoir énumérer l'ensemble des langues apparaissant dans ce corpus et trouver des lexiques correspondants. On peut aussi imaginer une méthode venant compléter le taux de lexicalité, par exemple avec le TTR.

— Lorsque des textes sont multilingues, les passages dans la langue minoritaire sont considérés comme du bruit. Ceci est encore une fois dû au taux de lexicalité, ou plutôt à la manière dont on le mesure : nous mesurons le taux de lexicalité par lexique de tout le texte, puis ne gardons que le meilleur. Ainsi, un texte ne peut être lexicalisé que par un seul texte à la fois. On pourrait alors résoudre ce problème en mesurant à un grain plus fin (page ou ligne) laissant la possibilité de combiner les lexiques au sien d'un seul texte. On peut également penser à la concaténation de ces lexiques, mais cela risque de masquer certaines erreurs (pus il y a de mots, plus une retranscription incorrecte risque de correspondre à un mot d'une des langues, par hasard).

— Les textes utilisant beaucoup de césures, c'est-à-dire des retours à la ligne pour finir un mot, peuvent être détectés comme bruités. Il faudrait alors détecter lorsqu'un mot en fin de ligne est hors lexique puis s'il l'est, le concaténer avec le premier mot de la ligne suivante.

— Certaines erreurs localisés nous échappent toujours. Des erreurs très localisées, ou dans des grands textes se faufilent encore parmi nos textes considérés comme bons [6]. Cela est sûrement dû à notre notation. En effet, nous n'avons sélectionné qu'un seul texte par *cluster* pour le représenter, de plus notre évaluation est loin d'être parfaite. Il faudrait alors faire une évaluation basée sur des critères moins arbitraires, comme le CER ou le WER, mais également sur plus de textes. Nous aurions alors des notes qui, je pense, représenteraient bien mieux les *clusters*. Cependant, une telle approche prendrait considérablement plus de temps.

5. Exemple en allemand : `https://bit.ly/ocr-allemand`

6. Exemple de texte avec des erreurs localisées, notamment sur les espaces : `https://bit.ly/46dv0Tk`

6.2 Résultats de l'analyse des variations stylistiques observées entre sections d'articles scientifiques

— Nous avons observé, en calculant le taux de lexicalité de chaque article pour section, une variation qui se produit sur presque tous les documents. Il fallait voir à quel point une section varie d'une autre en comparant avec le GLÀFF et le corpus de l'Est Républicain. Dans les figures A.17 et A.18 pour TALN et RECITAL à gauche et à droite respectivement.
Nous observons une tendance sur ces graphiques A.17 et A.18 pour tous les documents des deux conférences ; la première section à très souvent un taux de lexicalité élevé, qu'on peut nommer T_{lex} comme présenté dans *Exploiter un corpus de données textuelles sans post-traitement : l'écriture burlesque de la fronde.* [Abiven et al. 2021]. On a donc mis un seuil du taux pour détecter plus facilement les sections qui varient beaucoup ou non, si $T_{lex} < 0.5$, le T_{lex} est faible. Les sections qui ont un $T_{lex} > 0.5$ sont la plupart du temps la première section dans laquelle le ou les auteurs résument leur article et presque tous les documents débutent avec une section *"abstract"*. On retrouve donc beaucoup de mots de leur lexique qui sont nos deux corpus de référence. Bien qu'on ait certains articles en anglais, la majeure partie des articles sont en français. C'est la raison pour laquelle la courbe commence avec un fort taux de lexicalité. Le taux de lexicalité de la section *"abstract"* figure A.11 qui contient le résumé de l'article en anglais peut avoir souvent un taux de lexicalité faible du fait de la présence de mots étrangers. Puis on remarque que le taux de lexicalité reste à peu près stable sur les sections qui suivent. On observe à la fin de chaque section un pic de décroissance important. Cela est dû au fait que nous sommes en présence de la section qui contient les références (voir Figure A.13) de l'article avec des noms de personnes ou des lieux par exemple. On se retrouve donc avec un $T_{lex} < 0.5$.
Cette tendance sur les taux de lexicalité pour chaque section dans les différents articles s'observe également avec les scores de similarité avec la distance cosinus. Pour chaque section, on observe donc une variation de la similarité dans les tables A.19 et A.20.
On constate également une certaine ressemblance des courbes des taux de similarité de chaque section avec les courbes du taux de lexicalité même s'il ne s'agit pas des mêmes documents surtout au niveau de la

dernière section dans les figures A.19 et A.20.
En ce qui concerne les vectorisations en sac de mots et *TF-IDF* pour le calcul des distances entre sections les résultats sont présentés dans la figure A.13. On remarque que la vectorisation en *TF-IDF* améliore considérablement le rapprochement entre sections pour la distance cosinus. Pour les autres métriques, on n'observe pas beaucoup de variations. On remarque aussi que la distance de Levenshtein est plus élevée que Canberra lorsque l'on a une pondération *TF-IDF*. Elle est moins élevée en vectorisation *BOW*.

— **Le rapprochement des sections avec le clustering**
L'utilisation du *clustering* avec K-Means a permis d'observer les *clusters* représentés par les sections avec des points dans la figure A.14. On remarque toujours la section dont le contenu inclut des références à l'écart. Cela se remarque également dans le *clustering* de tous les documents qui ont été générés.
Ensuite nous avons observé des résultats intéressants avec le *clustering* hiérarchique en appliquant avec une vectorisation en *TF-IDF* et en *BOW* pour observer le comportement des *clusters*. Le critère de Ward qui minimise la variance entre les *clusters* fonctionne de manière optimale avec la vectorisation TF-IDF A.13. On remarque que les dernières sections sont regroupées dans les mêmes *clusters*.

— **Les intersections entre sections avec des n-grammes d'étiquettes morphosyntaxiques**
La recherche de l'intersection entre des n-grammes d'étiquettes morphosyntaxiques a permis de voir que des motifs peuvent être présents dans les sections selon l'ordre des sections dans l'article. Cette tâche peut être améliorée en recherchant les motifs dans toutes les sections, pas seulement dans des paires de sections qui se suivent. Il y a en effet des motifs de longueur 5 qui sont présents dans deux sections consécutives ; mais il faudra rechercher les tokens qui ont été étiquetés pour connaître les éléments qui sont repris. À défaut de pouvoir faire un UpsetPlot pour l'intersection des n-grammes d'étiquettes entre sections, nous pouvons montrer un exemple d'intersection avec un diagramme de Venn dans la figure A.16.

6.3 Sauts qualitatifs et stylistiques dans les corpus - Synthèse et perspectives

Nous avons pu observer certaines lacunes à nos analyses et avons ainsi déterminé des pistes à explorer :

Taux de lexicalité

Dans notre mesure du taux de lexicalité, plusieurs éléments viennent perturber sa fiabilité.

Premièrement, le fait que, dans le corpus des Mazarinades, de nombreux documents utilisent la césure, le fait de couper un mot dépassant de la ligne pour le continuer sur la suivante, fausse partiellement nos taux de lexicalité : au lieu d'un mot qui existe dans le lexique, nous obtenons deux mots généralement tous deux hors-lexique. Une piste d'amélioration intéressante serait alors de détecter les phénomènes de césure, qu'ils soient ou non signalés par un tiret.

Deuxièmement, la présence de plusieurs langues dans les textes provoque une grande chute du taux de lexicalité. Un texte ne pouvant être mesuré par un seul lexique à la fois, les passages comportant la ou les langues minoritaire.s du texte ont alors un taux de lexicalité très faible. Pour pallier ce problème, nous pourrions alors changer notre mesure afin de sélectionner le meilleur lexique par page, voire par ligne. Cela nous permettrait également d'obtenir des informations sur les langues qui composent nos textes. Par exemple : si un texte est à 30% lexicalisé par le lexique Ducange, nous pourrions alors en déduire que certains passages sont en latin.

Pour finir, certains textes sont écrits avec des langues qui ne sont pas couvertes par notre sélection de lexiques, il pourrait alors s'avérer intéressant de les détecter. Nous pensions alors combiner la librairie `langid` à nos taux de lexicalité actuels. Ainsi, si un texte a un taux de lexicalité faible et qu'il est clairement reconnu par *langid* comme un texte d'une autre langue (avec une valeur seuil sur le taux de confiance donné par *langid*), nous pourrions alors relever les langues non couvertes.

Clustering

Pour la partie du clustering, nous avons utilisé l'implémentation par ScikitLearn de l'algorithme k-means [Pedregosa et al. 2011]. Ce choix était motivé par deux raisons principales : les *clusters* semblaient cohérents et l'algorithme était extrêmement rapide, ce qui nous permettait de faire des tests, de modifier la chaîne de traitement en amont et d'y corriger certaines erreurs.

Cependant, d'autres méthodes de *clustering* semblent intéressantes, notamment OPTICS, un algorithme de *clustering* censé être plus adapté pour des *clusters* de taille et densité hétérogènes ainsi que pour un grand nombre de *clusters*. Il pourrait alors être intéressant d'approfondir les expériences relatives au *clustering* afin d'obtenir *in fine* de meilleures évaluations.

Évaluation

Pour ce qui est de l'évaluation, nous l'avons déjà évoqué quelques paragraphes au-dessus, mais il serait pertinent de créer un critère d'annotation explicite afin d'évaluer une petite partie de ce corpus manuellement. On pourrait alors imaginer faire annoter quelques textes par *clusters* formés par deux personnes pour chacun de ces textes, puis agréger les notes obtenues pour donner une note plus précise à chaque cluster. Cela permettrait sûrement d'améliorer la détection de textes bruités, principalement en diminuant le nombre de faux positifs.

Méthodes non explorées

Comme le présente Claude Martineau dans sa thèse *Compression de textes en langue naturelle* [2001], parmi les nombreuses méthodes qui existent pour compresser les textes, nombre d'entre-elles reposent sur une compression basée sur les mots, suites de mots ou parties de mots. L'auteur prend alors comme exemple des suites de mots comme "Président de la République" ou encore "le prix des agrumes.".

Une méthode complémentaire qui nous semblerait alors pertinente serait d'évaluer le bruit en fonction du taux de compression. Le raisonnement est le suivant : Lorsque l'on compresse un texte, au plus le taux de compression final est important, au plus ce dernier est ordonné, c'est-à-dire que nous avons pu retrouver suffisamment d'éléments d'une longueur significative qui se répétaient suffisamment pour que le fait qu'on les encode permette de diminuer la taille de la donnée finale. On pourrait alors supposer que plus un texte est bruité, plus ce dernier cassera ces structures, créant du désordre dans ce texte et provoquant alors une diminution du taux de compression.

Une étude complémentaire serait d'améliorer la détection de zones de textes, on pourrait alors utiliser un algorithme de reconnaissance de formes que l'on entraînerait une première fois avec un modèle non spécifique à notre tâche, comme celui de Kraken. Puis, à chaque itération, nous pourrions manuellement corriger les zones de texte pour les pages les plus bruitées, ou alors celles dans lesquelles on retrouve les caractères évoqués en 5.1, souvent présents en début et fin de page. On pourrait alors espérer obtenir un modèle

assez efficace tout en économisant du temps d'annotation.

La détection de variations stylistiques en ce qui concerne le rapprochement entre sections peut être améliorée avec le plongement lexical ou *Embeddings*. Ces embeddings vont permettre de mettre en relation les vecteurs en prenant en compte la sémantique avec un algorithme de type *Word2Vec* [Mikolov et al. 2013] qui contient deux modèles *CBOW* et *Skip-gram*.

Nous pouvons également améliorer le *clustering* des sections avec une technique populaire d'analyse de similarité entre textes appelée *LSA (Latent Semantic Analysis)* [Dumais et al. 1996]. Cette technique suppose que les mots qui sont proches au niveau sémantique, apparaissent dans des *pièces de textes similaires* [Gomaa et al. 2013]. Il serait donc intéressant de voir comment les sections seront représentées sur un nuage de points avec *K-Means* et *LSA*. Il est possible de faire ce clustering avec les mots, les n-grammes de mots ou encore les n-grammes d'étiquettes morphosyntyaxiques par années étant donnée qu'on a des articles rédigés de 1997 à 2022 pour observer à quel point il y a de la variation concernant l'usage de ces motifs.

Encore une perspective très intéressante serait, d'observer dans les sections la distribution des segments répétés, comme évoqué dans *Des motifs séquentiels aux motifs hiérar-chiques : l'apport des arbres lexico-syntaxiques récurrents pour le repérage des routines discursives* [Kraif and Tutin 2017] avec des méthodes tirées de la phraséologie et de la stylistique. Les scientifiques ont des particularités d'expressions lorsqu'ils rédigent un article. Il est donc possible d'associer ces segments répétés aux verbes de communication listé dans le lexique scientifique transdisciplinaire du projet [Hatier 2016]. Les mêmes auteurs, Kraif and Tutin [2017], il est possible avec une lemmatisation des segments tels que *ce résultat suggérer que, comme le avoir souligner...* de les repérer dans des écrits. Cela pourrait s'appliquer à l'analyse de variations entre sections de sorte à observer comment se distribuent ces segments. Par exemple on peut émettre l'hypothèse que le segment lemmatisé *nous proposer...*, qui cherchera donc les formes *nous proposons*, apparaissent très souvent en début d'article.

Une perspective de tâche avancée intéressante est celle utilisant un modèle de langue comme BERT [Devlin et al. 2018], qui utilise une technique appelée STS [7]. Cette technique permet de rapprocher des phrases de manière sémantique avec un score de similarité. Mais, appliquer cet algorithme à un coût car il nécessite plus de puissance de calcul, en particulier pour fonctionner sur des textes de grande taille.

7. Semantic Textual Similarity

Annexe A

Figures

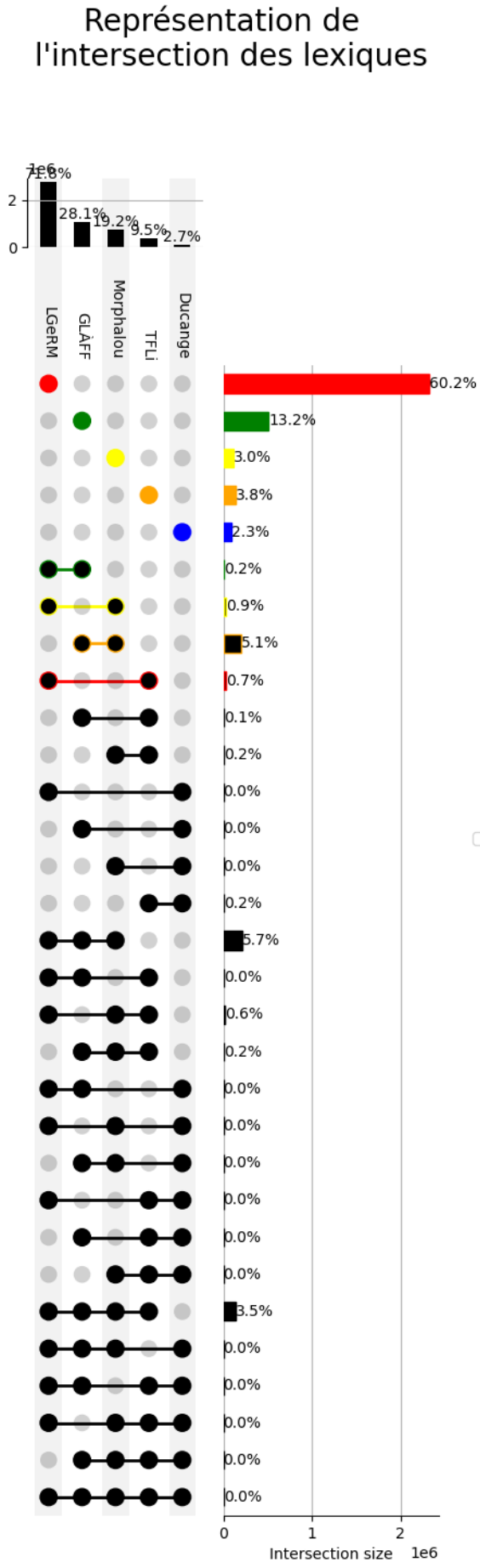


Figure A.1 – UpSet plot des ensembles de mots issus des lexiques. Chaque couleur pleine représente la partie unique d'un lexique. Les couleurs de contour représentent les principales intersections en fonction du degré de déviation.

```
<define name="tei_gap">
  <element name="gap">
     <a:documentation
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">indicates
         a point where material has been omitted in a
         transcription, whether for editorial reasons described in
         the TEI header, as part of sampling practice, or because
         the material is illegible, invisible, or inaudible.
         [3.4.3. Additions, Deletions, and
         Omissions]</a:documentation>
     [...]
     <optional>
        <attribute name="reason">
           <a:documentation
               xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">gives
               the reason for omission
Suggested values include: 1] cancelled; 2] deleted; 3] editorial;
    4] illegible; 5] inaudible; 6] irrelevant; 7]
    sampling</a:documentation>
           <list>
                [...]
           </list>
        </attribute>
     </optional>
     <optional>
        <attribute name="agent">
           <a:documentation
               xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">in
               the case of text omitted because of damage,
               categorizes the cause of the damage, if it can be
               identified.
Sample values include: 1] rubbing; 2] mildew; 3]
    smoke</a:documentation>
           <data type="token">
              <param name="pattern">[^\p{C}\p{Z}]+</param>
           </data>
        </attribute>
     </optional>
     <empty/>
  </element>
</define>
```

FIGURE A.2 – Passage de documentation récupéré depuis l'ODD (le schéma de validation pour des XMLs à la norme TEI) du projet Antomomaz.

```
1  <pb n="9"/>
2  <lb/>ARREST
3  <lb/>DE LA COVR
4  <lb/>DV PARLEMENT
5  <lb/>DE BRETAGNE,
6  <lb/>DONNE LES SEMESTRES ASSEMBLEZ.
7  <lb/>TOVCHANT LA CONVOCATION
8  <lb/> des ſEtats generaux du Royaume, &amp; parti
9  <lb rend="¬" break="no"/>culiers de la Prouince. <figure
       type="decoration"/>
10 <lb/>A PARIS,
11 <lb/>Par les Imprimeurs &amp; Libraires ordinai
12 <lb rend="¬" break="no"/>res du Roy.
13 <lb/>M. DC. XLIX.
14 <lb/>
15 <imprimatur>Auec Priuilege de ſa ſMajete.</imprimatur>
16 <pb n="10"/>
17 <pb n="11"/>
18 <gap/>
19 <pb n="12"/>
20 <lb/>3
21 <lb/>Chapitre &amp; ſCommnnaute dudit ''Rennesdy enuoyer leu
22 <lb/>Deputez, afin de donncr leurs voix àa ceux 'quils ſetimeront
       les
23 <lb/>plus capables pour ſeruir ſa ſéMajet, &amp; la Prouince en cette
24 <lb/>ſoccaion. Et ſur ce édelibere, L a Co vR apres auoir veu les
25 <lb/>Lettres Parentes du Roy éſdonnes àFontainebleau, le quin
```

FIGURE A.3 – Fragment du fichier "Moreau196_MAZ.xml", on y voit l'utilisation de la balise <gap\> afin de signaler l'absence de retranscritpion d'une des pages (à la ligne 18).

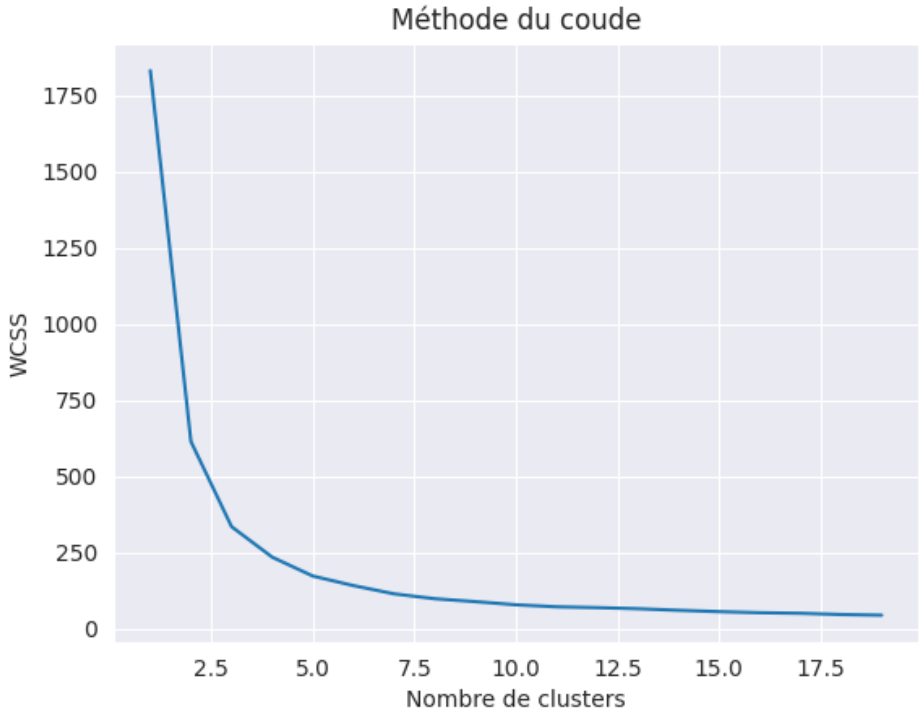


FIGURE A.4 – Méthode du coude. Afin de trouver le nombre optimal de *clusters*, on représente le WCSS (Cf. Partie 5.6.2) en fonction du nombre de *clusters*, puis on relève le moment où la courbe descend moins abruptement. Le nom provient du fait que la partie ou la décroissance ralentit ressemble à un coude.

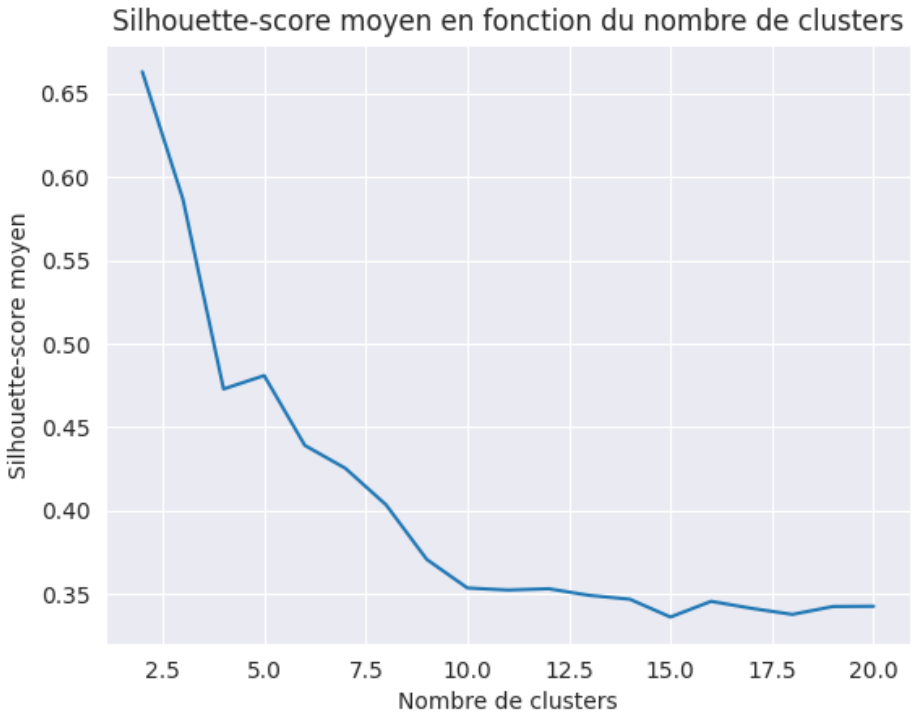


FIGURE A.5 – Représentation du silhouette score (Cf. Partie 5.6.2) en fonction du nombre de *clusters*. Cette représentation sert à choisir le nombre de *clusters* pour lequel le silhouette score est le plus élevé, elle est cependant imparfaite et à compléter avec une analyse du score par point (Cf. Fig. 5.10).

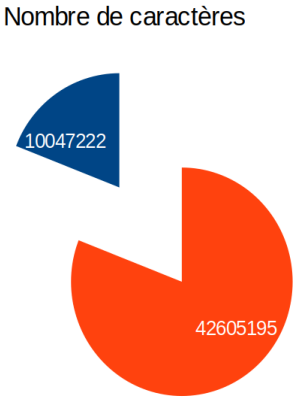


FIGURE A.6 – Nombre de caractères des deux corpus

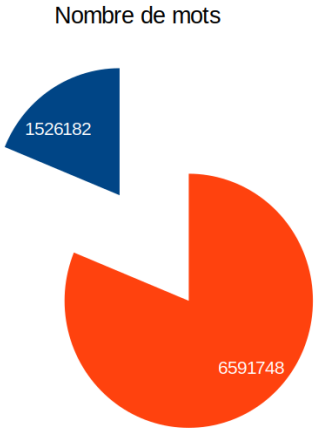


FIGURE A.7 – Nombre de mots des deux corpus

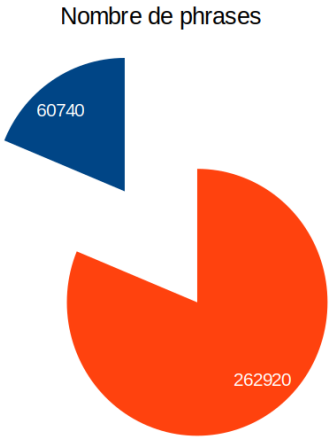


FIGURE A.8 – Nombre de phrases des deux corpus

FIGURE A.9 – Des avertissements d'erreurs de parsing de GROBID

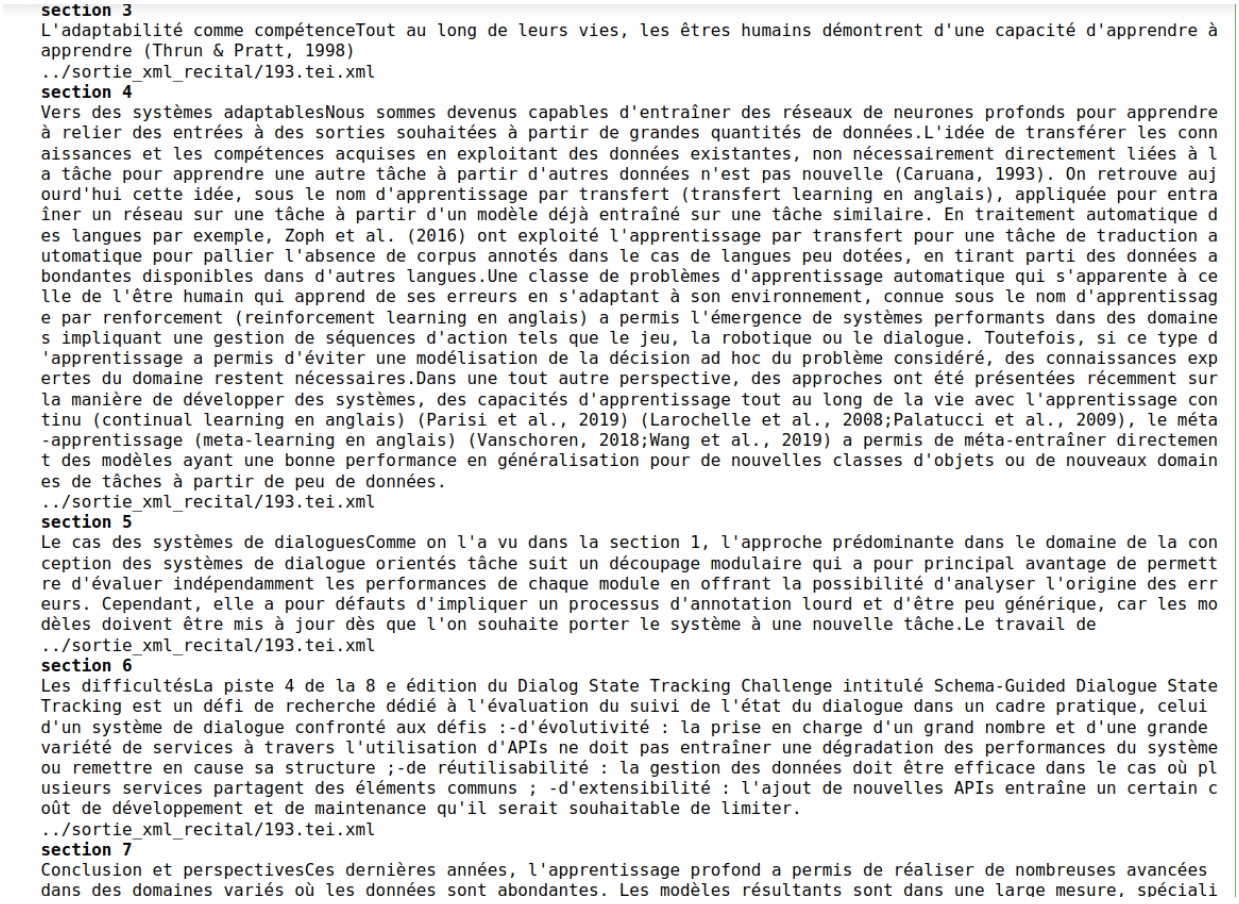


FIGURE A.10 – Découpage en sections dans python après extraction du contenu des balises

```
Article : ../sortie_xml_taln/taln-2006-long-001.tei.xml
Section 1
--------------------------------------------------------------------------------
Nombre de mots dans la section : 286
Présents avec le glaff ---> 39
Absents avec le glaff ---> 122
Présents avec l'est republicain ---> 34
Absents avec l'est republicain ---> 127
Taux de lexicalité avec le glaff : 0.2422360248447205
Taux de lexicalité avec l'est republicain : 0.2111801242236025
Taux de lexicalité avec le glaff inférieur à 0.5
C'est probablement la section 'abstract' ou 'autre chose'
Voici une partie des mots absents avec le glaff : ['sense', 'TALN', 'the', 'language', 'well', 'performed', 'selecte
d', 'overview', 'approach', 'Readable', 'Section', 'translated', 'Machine', 'The', 'from', 'form', 'just', 'will', '
common', 'discriminate', 'Retrieval', 'be', 'terms', 'compared', 'spelling', 'may', 'each', 'methods', 'In', 'Such',
'It', 'non-disambiguated', 'term', 'distinguish', 'MI', 'We', 'readable', 'queries', 'conducted', 'against', 'find',
'process', 'among', 'word', 'work', 'text', 'Two', 'Cross', 'senses', 'query', 'one', 'not', 'then', 'given', 'ranke
d', 'synonyms', 'different', 'polysemy', 'with', 'automatically', 'investigating', 'irrellevant', 'function', 'On',
'results', 'words', 'IntroductionA', 'between', 'disambiguated', 'experiments', 'whereby', 'no', 'Mutual', 'dictiona
ry', 'present', 'several', 'similar', 'were', 'aimed', 'context', 'is', 'was', 'how', 'Language', 'French', 'introdu
```

FIGURE A.11 – Taux de lexicalité d'une section 'abstract'

```
Présents avec l'est republicain ---> 47
Absents avec l'est republicain ---> 241
Taux de lexicalité avec le glaff : 0.1076388888888889
Taux de lexicalité avec l'est republicain : 0.16319444444444445
Taux de lexicalité avec le glaff inférieur à 0.5
C'est probablement la section 'abstract' ou 'autre chose'
Voici une partie des mots absents avec le glaff : ['th', 'NIde', 'SLSmall', 'Corpus', 'sense', 'MCCs-', 'the', 'Und
rstanding', 'based', 'Cross-Language', 'Waterloo', 'Processing', 'DFass', 'Paraphrase', 'Lexical', 'KDahlgren', 'So
e', 'Computational', 'Addis-Abeba', 'Conference', 'Dublin', 'GHirst', 'Mega', 'The', 'TPlate', 'from', 'MA', 'Roche
ter', 'indexing', 'Word-Sense', 'Science', 'Research', 'Academic', 'JAGuthrie', 'Longman', 'Second', 'ambiguity', '
Slator', 'Mateo', 'MJMcgill', 'Journal', 'Analysis', 'artificial', 'GWCottrell', 'W', 'COLING-', 'supervised', 'Tec
nical', 'R', 'PHayes', 'Retrieval', 'Kluwer', 'Development', 'MLesk', 'Simulated', 'LBallesteros', 'analysis', 'Sem
ntic', 'Comprehending', 'Management', 'methods', 'In', 'cone', 'Norwell', 'Advances', 'Usa', 'haystack', 'Lin', 'HS
hütze', 'readable', 'preference', 'WGLehnert', 'Knowledge', 'Guo', 'CRD', 'Using', 'VJ', 'computationally', 'word',
'text', 'IBM', 'Croft', 'Virginia', 'Semantics', 'LMercer', 'San', 'HBian', 'Ph', 'Publishers', 'Resource', 'Lexico
ogy', 'Two', 'neuropsychology', 'MSanderson', 'CIKM-', 'than', 'Computer', 'senses', 'Sense', 'Slator']
Taux de lexicalité avec l'est republicain inférieur à 0.5
C'est probablement la section 'abstract' ou 'autre chose'
Voici une partie des mots absents avec l'est republicain : ['th', 'NIde', 'SLSmall', 'Corpus', 'sense', 'MCCs-', 'U
derstanding', 'based', 'Cross-Language', 'Waterloo', 'Processing', 'DFass', 'Paraphrase', 'Lexical', 'KDahlgren', '
```

FIGURE A.12 – Taux de lexicalité d'une section 'references'

Document	Section	Distance	Similarité (Bag of Words)	Similarité (TF-IDF)
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	hamming	0.2487798075582206	0.25686794031515825
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	canberra	1734.6837801291408	1746.358617341464
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	braycurtis	0.7695006747638327	0.8171279469622378
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	jaccard	0.9685124864277959	1.0
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	matching	0.2487798075582206	0.25686794031515825
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	dice	0.8531187122736419	0.8531187122736419
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	kulsinski	0.9835344535919702	0.9835344535919702
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	russellrao	0.9796402175428811	0.9796402175428811
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	sokalmichener	0.3825420096988835	0.3825420096988835
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	sokalsneath	0.9587337478801583	0.9587337478801583
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	cosine	0.7014347946954146	0.534328195643769
../sortie_xml_recital/recital-2011-long-005.tei.xml	0	levenshtein	1537	1754
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	hamming	0.25686794031515825	0.2706735462278622
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	canberra	1760.5071295905016	1780.7705803890167
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	braycurtis	0.6781841109709962	0.7433483183978945
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	jaccard	0.9489953632148377	1.0
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	matching	0.25686794031515825	0.2706735462278622
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	dice	0.7840073529411765	0.7840073529411765
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	kulsinski	0.9735270924862003	0.9735270924862003
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	russellrao	0.967229117277925	0.967229117277925
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	sokalmichener	0.38436408696631746	0.38436408696631746
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	sokalsneath	0.9355634768302714	0.9355634768302714
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	cosine	0.8813489942526291	0.7120001914581503
../sortie_xml_recital/recital-2011-long-005.tei.xml	1	levenshtein	1559	1845

FIGURE A.13 – Comparaison des scores de similarité en *BOW* et *TF-IDF*

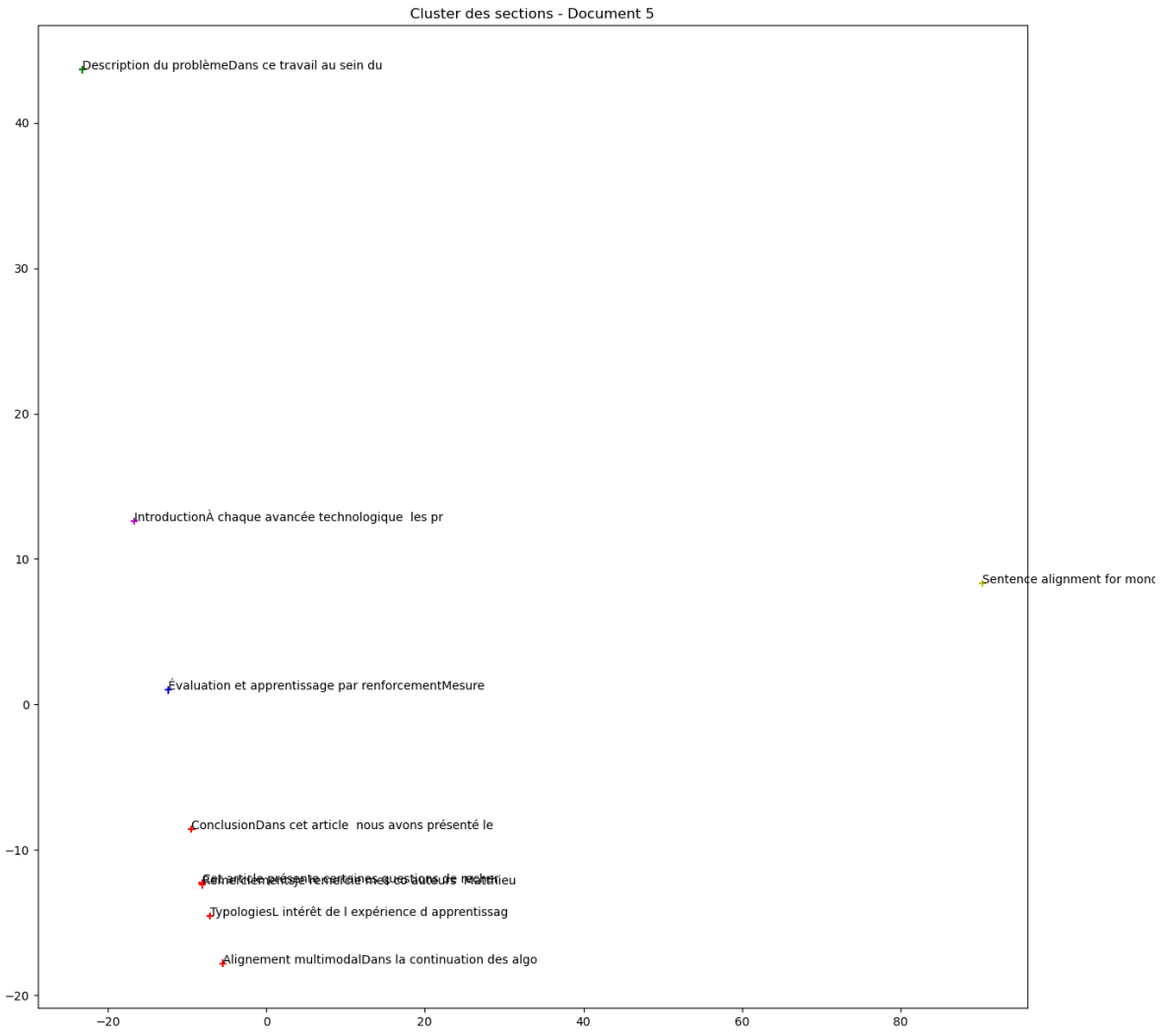


FIGURE A.14 – Exemple de *clustering* pour un document

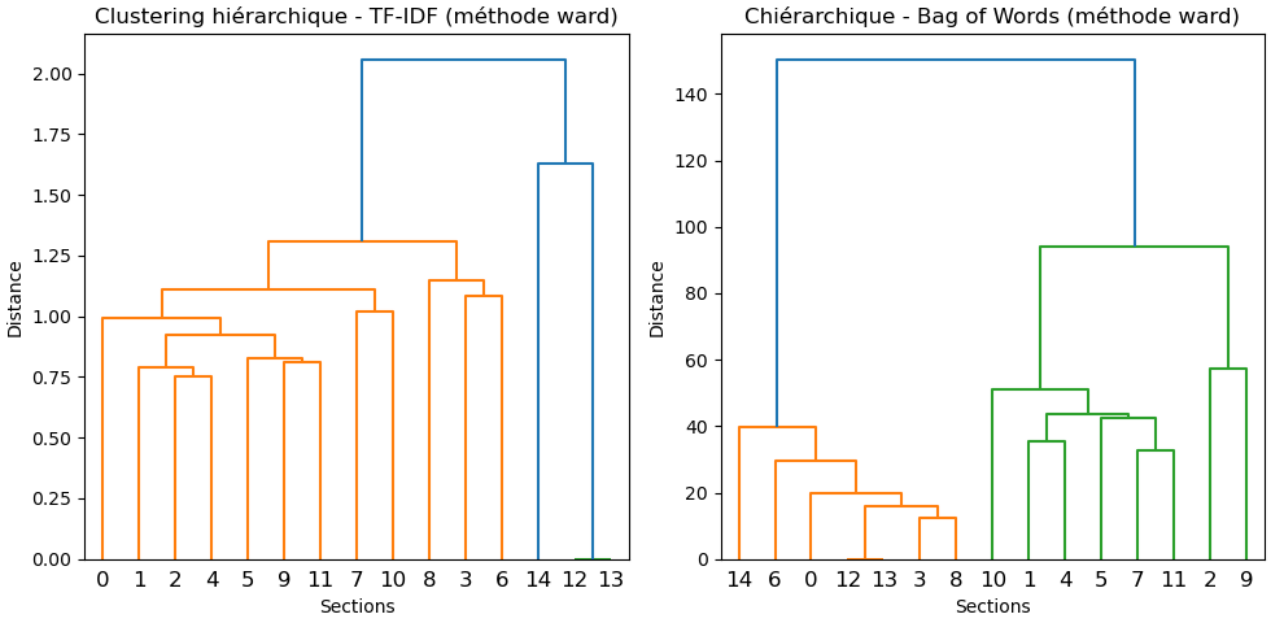


FIGURE A.15 – Exemple de *clustering* hiérarchique pour un document

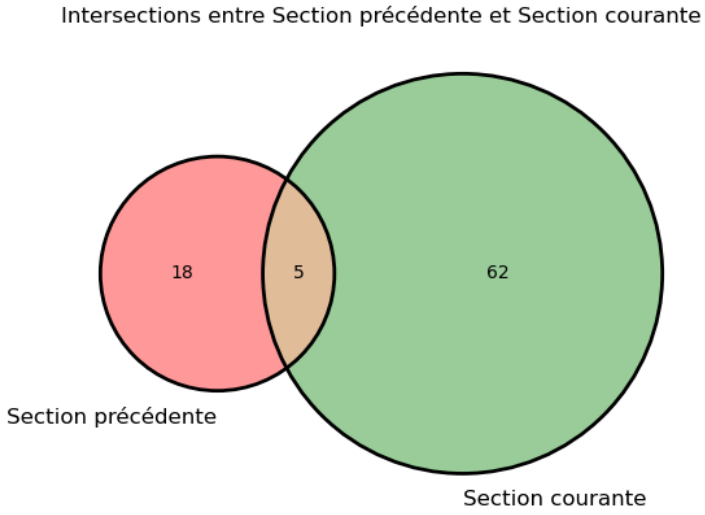


FIGURE A.16 – Intersection des n-grammes d'étiquettes entre sections pour 968.pdf.tei.xml

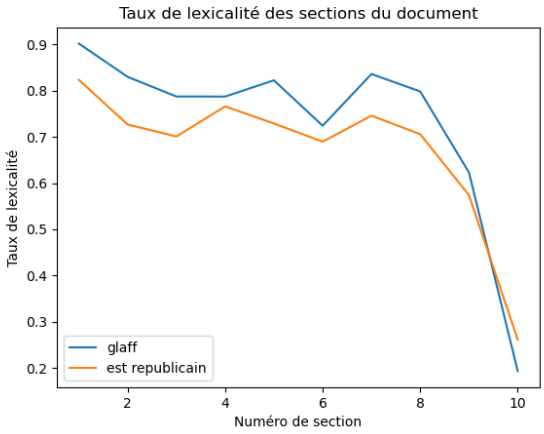


FIGURE A.17 – Variations du taux de lexicalité des sections - article TALN

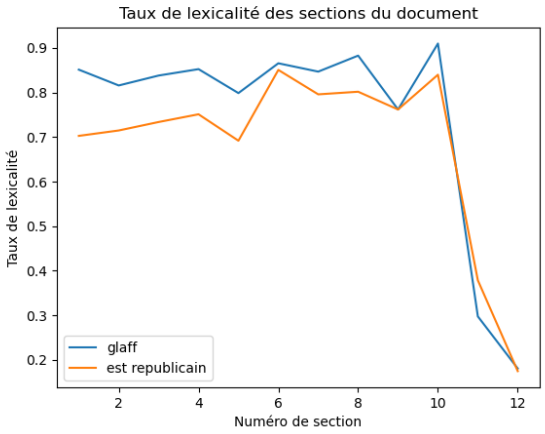


FIGURE A.18 – Variations du taux de lexicalité des sections - article RECITAL

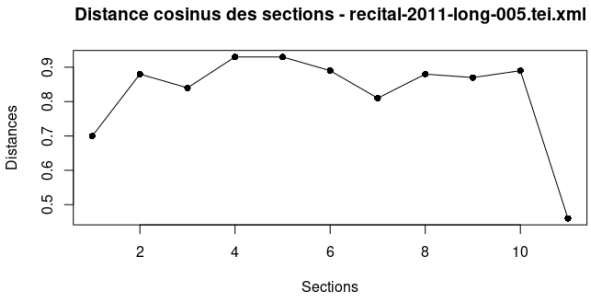


FIGURE A.19 – Variations des distances cosinus des sections - recital-2011-long-005.tei.xml

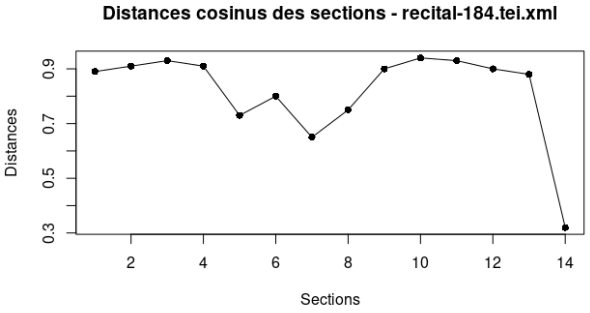


FIGURE A.20 – Variations des distances cosinus des sections-recital-184.tei.xml

Bibliographie

(2008–2023). Grobid. `https://github.com/kermitt2/grobid`.

Abbasi, A. and Chen, H. (2008). Cybergate : a design framework and system for text analysis of computer-mediated communication. *Mis Quarterly*, pages 811–837.

Abiven, K., Bartz, A., Lejeune, G., and Tanguy, J.-B. (2022). Vers une collection numÃ©rique des libelles parus pendant la fronde, ou comment relier des mazarinades. *Le Verger*, (23).

Abiven, K. and Lejeune, G. (2019). Analyse automatique de documents anciens : tirer parti d'un corpus incomplet, hétérogène et bruité. *Recherche dínformation, document et web sémantique*, 2(Numéro 1).

Abiven, K., Tanguy, J.-B., and Lejeune, G. (2021). Exploiter un corpus de données textuelles sans post-traitement : l'écriture burlesque de la fronde. *Humanités numériques*, (4).

ATILF (2002). Trésor de la langue française informatisé (tlfi) [ressource en ligne].

ATILF (2017). Lgerm. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

ATILF (2023a). Morphalou. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

ATILF (2023b). Ressource textuelle frantext (en ligne). *ATILF-CNRS & Université de Lorraine*, pages 1998–2023.

ATILF and CLLE (2020). Corpus journalistique issu de l'est républicain. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

Audras, I. and Ganascia, J.-G. (2005). Analyses comparatives de motifs syntaxiques de francophones et d'apprenants du français arabophones, à l'aide d'outils d'extraction automatique du langage. In *Ingénierie des Langues et Ingénierie de l'Arabe 2005*, pages 59–67. Centre de Recherche scientifique et technique pour le développement de la ….

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Baledent, A. and Lejeune, G. (2019). Analyse stylistique automatique : à la recherche d'indices efficaces et pertinents pour caractériser le style de Dumas. In *Phraséologie et stylistique de la langue littéraire*, Erlangen, Germany.

Ben Salah, A., Moreux, J.-P., Ragot, N., and Paquet, T. (2015). Ocr performance prediction using cross-ocr alignment.

Boudin, F. (2013). Taln archives : une archive numérique francophone des articles de recherche en traitement automatique de la langue. In *Traitement Automatique des Langues Naturelles (TALN)*.

Bouveyron, C. and Girard, S. (2009). Classification supervisée et non supervisée des données de grande dimension. *La revue MODULAD*, 40 :81–102. Accessible en ligne : http ://www.modulad.fr/.

Brennan, M., Afroz, S., and Greenstadt, R. (2012). Adversarial stylometry : Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security (TISSEC)*, 15(3) :1–22.

Brixtel, R., Lecluze, C., and Lejeune, G. (2015). Attribution d'auteur : approche multilingue fondée sur les répétitions maximales. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 208–219.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software : experiences from the scikit-learn project. In *ECML PKDD Workshop : Languages for Data Mining and Machine Learning*, pages 108–122.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J.,

Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2023). `https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html`.

Cabrera-Diego, L.-A., Torres-Moreno, J.-M., and El-Bèze, M. (2013). Segcv : Eficient parsing of résumés with analysis and correction of errors (segcv : traitement efficace de cv avec analyse et correction d'erreurs)[in french]. In *Proceedings of TALN 2013 (Volume 2 : Short Papers)*, pages 707–714.

Candito, M. and Seddah, D. (2012). Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012-19e conférence sur le Traitement Automatique des Langues Naturelles*.

Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2) :245–276. PMID : 26828106.

Chiron, G., Doucet, A., Coustaty, M., Visani, M., and Moreux, J.-P. (2017). Impact of ocr errors on the use of digital libraries : Towards a better access to information. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pages 1–4, Toronto, Canada. IEEE.

Claveau, V. (2014). Agrégation de sac-de-sacs-de-mots pour la recherche d'information par modèles vectoriels. In *14 ème conférence Extraction et Gestion des Connaissances, EGC 2014*, pages 6–p.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

Du Cange, D. et al. (1883-1887). Glossarium mediæ et infimæ latinitatis. *Niort : L. Favre*.

Dumais, S., Furnas, G., Landauer, T., Deerwester, S., and Harshman, R. (1996). Using latent semantic analysis to improve access to textual information. *Proceedings, CHI*, 88.

Faisal, M., Zamzami, E., et al. (2020). Comparative analysis of inter-centroid k-means performance using euclidean distance, canberra distance and manhattan distance. In *Journal of Physics : Conference Series*, volume 1566, page 012112. IOP Publishing.

Fort, K. (2023). `https://members.loria.fr/KFort/files/fichiers_cours/CorpusEtDroits.pdf`.

Fort, K., Guillaume, B., Pilatte, Y.-A., Constant, M., and Lefèbvre, N. (2020). Rigor mortis : Annotating MWEs with a gamified platform. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4395–4401, Marseille, France. European Language Resources Association.

Ganascia, J.-G. (2001). Extraction automatique de motifs syntaxiques. In *Actes de la 8ème conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 192–201.

Giguet, E. and Lejeune, G. (2021). Daniel at the FinSBD-2 task : Extracting Lists and Sentences from PDF Documents : a model-driven end-to-end approach to PDF document analysis. In *Second Workshop on Financial Technology and Natural Language Processing in conjunction with IJCAI-PRICAI 2020*, Proceedings of the Second Workshop on Financial Technology and Natural Language Processing, pages 67–74, Kyoto, Japan.

Giguet, E., Lejeune, G., and Tanguy, J.-B. (2020). Daniel@FinTOC'2 Shared Task : Title Detection and Structure Extraction. In *1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation @COLING'2020*, Proceedings of the 28th International Conference on Computational Linguistics (COLING'2020), Barcelone, Spain.

Gomaa, W. H., Fahmy, A. A., et al. (2013). A survey of text similarity approaches. *international journal of Computer Applications*, 68(13) :13–18.

Granger, S., Paquot, M., et al. (2008). Disentangling the phraseological web. *S. Granger & Meunier (Eds.)*.

GROUIN, C. (2014). Les 10 ans du défi fouille de texte deft.

Hatier, S. (2016). *Identification et analyse linguistique du lexique scientifique transdisciplinaire. Approche outillée sur un corpus d'articles de recherche en SHS*. Theses, Université Grenoble Alpes.

Hearst, M. A. (1993). Texttiling : A quantitative approach to discourse segmentation. Technical report, Citeseer.

Hearst, M. A. (1997). Text tiling : Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1) :33–64.

Hearst, M. A. and Plaunt, C. (1993). Subtopic structuring for full-length document access. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 59–68.

Honnibal, M. and Montani, I. (2015). spacy. `https://spacy.io`. Version 3.5.3.

Kettunen, K. T. and Koistinen, J. M. O. (2019). Open source tesseract in re-ocr of finnish fraktur from 19th and early 20th century newspapers and journals–collected notes on quality improvement. *Digital Humanities in the Nordic Countries*.

Koppel, M., Schler, J., and Argamon, S. (2009). Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1) :9–26.

Koudoro-Parfait, C., Lejeune, G., and Roe, G. (2021). Spatial named entity recognition in literary texts : What is the influence of ocr noise ? In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Geospatial Humanities*, GeoHumanities'21, pages 13–21, New York, NY, USA. Association for Computing Machinery.

Kraif, O. (1999). Identification des cognats et alignement bi-textuel : une étude empirique. In *Actes de la 6ème conférence annuelle sur le Traitement Automatique des Langues Naturelles, TALN*, volume 99, pages 205–214.

Kraif, O. and Tutin, A. (2017). Des motifs séquentiels aux motifs hiérarchiques : l'apport des arbres lexico-syntaxiques récurrents pour le repérage des routines discursives. *Corpus*, (17).

KumarI, A. (2023). Kmeans silhouette score python example. `https://vitalflux.com/kmeans-silhouette-score-explained-with-python-example/`.

Lacheret, A., Kahane, S., Beliao, J., Dister, A., Gerdes, K., Goldman, J.-P., Obin, N., Pietrandrea, P., and Tchobanov, A. (2014). Rhapsodie : un treebank annoté pour l'étude de l'interface syntaxe-prosodie en français parlé. In *SHS Web of Conferences*, volume 8, pages 2675–2689. EDP Sciences.

Lardilleux, A. (2010). Contribution des basses fréquences à l'alignement sous-phrastique multilingue : une approche différentielle.

Leung, K. (2021). Evaluate ocr output quality with character error rate (cer) and word error rate (wer).

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., and Pfister, H. (2014). Upset : Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis)*, 20(12) :1983–1992.

Lutoslawski, W. (1898). Principes de stylométrie appliqués à la chronologie des œuvres de platon. *Revue des Études Grecques*, 11(41) :61–81.

Mahendru, K. (2019). How to determine the optimal k for k-means ? `https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb/`.

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de La Clergerie, É. V., Seddah, D., and Sagot, B. (2019). Camembert : a tasty french language model. *arXiv preprint arXiv :1911.03894*.

Martineau, C. (2001). *Compression de textes en langue naturelle*. Theses, Université de Marne-la-Vallée.

McDonald, R. and Nivre, J. (2013). Yvonne irmbach-brundage, yoav goldberg, dipanjan das, kuzman ganchev, keith hall, slav petrov, hao zhang, oscar täckström, claudia bedini, núria bertomeu castelló, and jungmee lee. universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 92–97.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Moh'd B, A.-D. and Roberts, S. A. (1996). New methods for the initialisation of clusters. *Pattern Recognition Letters*, 17(5) :451–455.

Moreau, C. (1851). *Bibliographie des mazarinades*, volume 3. J. Renouard et cie.

Nagar, S. and Khunteta, A. (2016). A proposed modification over learning vector quantization and k-means algorithms for performance enhancement.

In *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing : ICRCWIP-2014*, pages 671–682. Springer.

Negre, E. (2013). Comparaison de textes : quelques approches... working paper or preprint.

Nivre, J. and de Marneffe, M. (2016). Ginter, f., goldberg, y., hajič, j.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Rastier, F. (2005). Enjeux épistémologiques de la linguistique de corpus. *La linguistique de corpus*, (31-45).

Sajous, F., Hathout, N., and Calderone, B. (2013). GLÁFF, un Gros Lexique Á tout Faire du Français. In *Actes de la 20e conférence sur le Traitement Automatique des Langues Naturelles (TALN'2013)*, pages 285–298, Les Sables d'Olonne, France.

Salem, A. (1987). *Pratique des segments répétés : essai de statistique textuelle*, volume 4. Klincksieck.

Sanguinetti, M. and Bosco, C. (2015). Parttut : The turin university parallel treebank. *Harmonization and development of resources and tools for Italian natural language processing within the PARLI project*, pages 51–69.

Sinaga, K. P. and Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE Access*, 8 :80716–80727.

Sinclair, J. (1996). Preliminary recommendations on corpus typology. Technical report, EAGLES (Expert Advisory Group on Language Engineering Standards).

Smith, R. (2011). Limits on the application of frequency-based language models to ocr. In *ICDAR*, pages 538–542. Won Best Industrial Paper Award.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1) :11–21.

Stark, H. A. (1988). What do paragraph markings do ? *Discourse processes*, 11(3) :275–303.

Tanguy, J.-B. (2020). Exploiter des modèles de langue pour évaluer des sorties de logiciels d'OCR pour des documents français du XVIIe siècle. In Benzitoun, C., Braud, C., Huber, L., Langlois, D., Ouni, S., Pogodalla, S., and Schneider, S., editors, *6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 3 : Rencontre des Étudiants Chercheurs en Informatique pour le TAL*, pages 205–217, Nancy, France. ATALA.

Tanguy, J.-B. (2022). *Océriser pour accéder aux données ? Vers une évaluation non supervisée du bruit dans les données textuelles issues d'OCR de documents du XVIIème siècle*. PhD thesis, Sorbonne Université - Faculté des Lettres. Thèse de doctorat dirigée par Roe, Glenn H. Littérature comparée Sorbonne université 2022.

Tomar, A. (2022). Stop using elbow method in k-means clustering, instead, use this ! `https://towardsdatascience.com/elbow-method-is-not-sufficient-to-find-best-k-in-k-means-clustering-fc820da0631d/`.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Vergne, J. and Giguet, E. (1998). Regards Théoriques sur le "Tagging". In *Fifth annual conference Le Traitement Automatique des Langues Naturelles (TALN 1998)*, Proceedings of the fifth annual conference Le Traitement Automatique des Langues Naturelles (TALN 1998), pages 22–31, Paris, France.

Ward Jr, J. H. and Hook, M. E. (1963). Application of an hierarchical grouping procedure to a problem of grouping profiles. *Educational and Psychological Measurement*, 23(1) :69–81.