Analyse syntaxique automatique d'un corpus du français médiéval

Jingyu Liu

Analyse syntaxique automatique d'un corpus du français médiéval

Jingyu
LIU

Sous la direction de Olivier Kraif

Stage co-encadré par Maximin Coavoux, Corinne Denoyelle et Julie Sorba

Laboratoire : LIG

UFR LLASIC
Département Sciences du Langage

Mémoire de master 2 mention Sciences du Langage - 20 crédits

Parcours : Industrie de la Langue

Année universitaire 2020-2021

Analyse syntaxique automatique d'un corpus du français médiéval

Jingyu
LIU

Sous la direction de Olivier Kraif

Stage co-encadré par Maximin Coavoux, Corinne Denoyelle et Julie Sorba

Laboratoire : LIG

UFR LLASIC
Département Sciences du Langage

Mémoire de master 2 mention Sciences du Langage - 20 crédits

Parcours : Industrie de la Langue

Année universitaire 2020-2021

Remerciements

Ce stage a été financé grâce à la chaire MIAI « Artificial Intelligence and Language »[1]. Un grand merci à Laurent Besacier et François Portet pour leur soutien dans ce projet.

Je tiens à remercier ma direction du stage, Olivier Kraif, pour sa patience et ses conseils durant le stage, et pour son amabilité, ses remarques et ses critiques lors la réalisation de ce mémoire.

Je remercie fortement Maximin Coavoux, Corinne Denoyelle et Julie Sorba, mes trois co-encadrants du projet, pour avoir partagé leurs connaissances et leurs expériences, et pour m'avoir permis de participer à un tel projet.

Je tiens également à remercier Claude Ponton pour avoir pris le temps de me suivre pendant ce stage, pour ses conseils et pour l'aide qu'il m'a apportée.

Enfin, j'aimerais également remercier Aurélie pour la relecture et la correction, ainsi que tous mes amis et ma famille pour leurs encouragements.

[1] https://miai.univ-grenoble-alpes.fr/research/chairs/perception-interaction/artificial-intelligence-language-850480.kjsp?RH=6499587813011763

DÉCLARATION ANTI-PLAGIAT

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

PRENOM : ……

NOM : ……

DATE : ……

Sommaire

Introduction

Langue de culture et de littérature, le français médiéval est bien conservé. De ce fait, on peut analyser son caractère et sa structure avec une grande précision. De nombreuses études sur la langue française ne peuvent se passer de la connaissance du français médiéval. La structure du texte en français médiéval étant principalement composée d'expressions stéréotypées récurrentes, pour l'étude de la phraséologie étendue dans la structuration des genres textuels, les unités lexicales dans les textes sont, de manière appropriée, l'objet de cette recherche.

Ce projet propose de repérer et d'étudier les unités phraséologiques structurant précisément le genre romanesque médiéval. Le corpus principal du projet de recherche est le genre littéraire du XIIIe siècle qui contient des romans en prose et en vers, ainsi que des chroniques. Grâce au développement des outils modernes de traitement automatique des textes, une étude plus systématique et numérique de la langue médiévale dans le cadre littéraire est possible. Lorsque la recherche empirique constate que des formules différentes sont utilisées selon les genres littéraires, la linguistique de corpus outillée permet de reconnaître d'autres phénomènes phraséologiques qui ne se limitent plus aux syntagmes et aux phrases[2].

Mon travail de stage consiste à fusionner l'annotation réalisée par l'analyseur syntaxique Hops (Grobol & Crabbé, 2021) avec celle faite par la plateforme LGeRM (Bazin-Tacchella & Souvay, 2020)[3]. Les deux annotations sont obtenues à partir d'œuvres littéraires médiévales. Une fois fusionnées, elles sont intégrées dans notre outil de fouille textuelle, le Lexicoscope, qui est une plate-forme développée à l'Université Grenoble-Alpes (Kraif, Diwersy, 2012 ; Kraif, 2019). Dans le même temps, il est nécessaire d'effectuer une analyse syntaxique sur les résultats d'annotation. L'analyse se fait sur deux niveaux : 1) au niveau de la relation entre les tokens et 2) au niveau de l'étiquette des tokens. Cette analyse est également pertinente pour le lemme du token. Les résultats de cette analyse pourront servir de base à l'amélioration des systèmes d'analyse futurs. Ainsi, les résultats de l'analyse automatique seront plus précis, et on pourra mieux étudier et repérer avec finesse les unités phraséologiques du genre romanesque médiéval en prose.

[2] https://lidilem.univ-grenoble-alpes.fr/node/16/axes-recherche/phraseologie-et-genres-textuels-cas-roman-medieval

[3] Lemmes Graphies et Règles Morphologiques (LGeRM)

Ce mémoire est composé de trois parties. La première partie met en perspective le contexte du projet comprenant l'introduction du domaine, la problématique et la présentation de l'état de l’art. Ensuite, la deuxième partie se concentre sur les différentes étapes et le contenu de mon travail. Enfin, la troisième partie présente les résultats obtenus et les perspectives d'amélioration.

Partie 1

-

Contexte du projet

Chapitre 1. Contexte et problématique

Le premier chapitre présente les membres/groupes du projet ainsi que le corpus et les outils d'analyse utilisés dans la recherche. La problématique et l'objectif de ce stage sont également soulignés dans ce chapitre.

1. Contexte et ressources

1.1. Contexte

Ce travail de stage a été effectué dans le cadre du projet de « Phraséologie et genres textuels : le cas du roman médiéval » qui explore les liens entre phraséologie et genres textuels en ancien français, mené par les coordinatrices, Julie Sorba et Corinne Denoyelle, et réalisé dans le Laboratoire de Linguistique et Didactique des Langues Étrangères et Maternelles (LIDILEM). Il s'appuie également sur une solide collaboration avec plusieurs chercheuses du Laboratoire Litt&Arts UMR 5316[4] et du Laboratoire d'Informatique de Grenoble (LIG). En outre, des chercheurs d'autres organisations sont également impliquées dans ce projet en tant que membres extérieurs.

1.1.1. Le laboratoire de linguistique et didactique des langues étrangères et maternelles (LIDILEM)

Le Lidilem est un laboratoire de recherche pluridisciplinaire né en 1987 de la fédération de cinq centres de recherche. Il regroupe actuellement plus de 60 membres permanents et environ 70 doctorants.

Le contenu de la recherche du laboratoire comprend principalement les quatre axes suivants : la description et la modélisation linguistiques, le Traitement Automatique de la Langue (TAL), la constitution et l'exploitation de corpus ; la sociolinguistique qui intègre les thématiques des identités, cultures, interactions et usages ; l'acquisition du langage (multimodalité, variabilité et contexte) ; et la didactique des langues, s'intéressant notamment à l'analyse et à l'évaluation des processus d'enseignement/apprentissage[5]. Les responsables du premier axe sont également les principaux responsables de ce projet d'analyse concernant la phraséologie et les genres textuels en ancien français. Ils sont

[4] Arts et pratiques du texte, de l'image, de l'écran et de la scène
[5] Source : https://lidilem.univ-grenoble-alpes.fr/node/16

responsables des travaux liés au traitement automatique et à la constitution du corpus du projet. En outre, en tant que responsables du projet, ils doivent coordonner également diverses tâches et prennent l'initiative de communiquer avec les collègues d'autres laboratoires.

1.1.2. Le laboratoire Litt&Arts UMR 5316 et le laboratoire d'informatique de grenoble (LIG)

L'unité de recherche Litt&Arts UMR 5316 a été créée en 2016 et compte environ 70 enseignants-chercheurs en Lettres et Arts du spectacle, une chercheuse du CNRS[6], trois ingénieurs en humanités numériques (ELAN[7]) et quatre personnels administratifs. Quatre grands axes constituent les activités de recherche de Litt&Arts UMR 5316, à savoir : les nouvelles philologies ; la traduction, transmission, réception des textes littéraires ; les expériences de la création ; la transversalité des humanités numériques. Ce laboratoire est également engagé dans la recherche sur la « convergence numérique » qui permet de traiter les textes ainsi que les images, les voix, les gestes d'une manière innovante. Dans ce projet, le personnel du laboratoire Litt&Arts UMR 5316 a joué un rôle important dans la constitution d'un corpus de textes originaux de romans du XIIIe siècle en français médiéval.

Créé en 2007, le LIG est un laboratoire spécialisé dans le développement informatique. Ce laboratoire rassemble près de 500 personnes et 24 équipes de recherche afin de contribuer au développement des aspects fondamentaux de l'informatique (modèles, langages, méthodes, algorithmes) et de développer une synergie entre les défis conceptuels, technologiques et sociétaux associés à cette discipline[8]. Grâce à ces larges champs de recherche et la qualité des études menées, le laboratoire jouit d'une reconnaissance internationale dans ces domaines. Dans ce projet, Maximin Coavoux, chercheur au LIG, a co-encadré le stage sur divers aspects, et notamment sur la mise en oeuvre de l'analyse syntaxique.

1.2. Ressources

Les ressources utilisées dans ce projet proviennent principalement de romans écrits en ancien français au XIIIe siècle, notamment des romans en prose et en vers, et des chroniques (cf. Annexe 1). Ils ont été analysés et numérisés par des chercheurs de

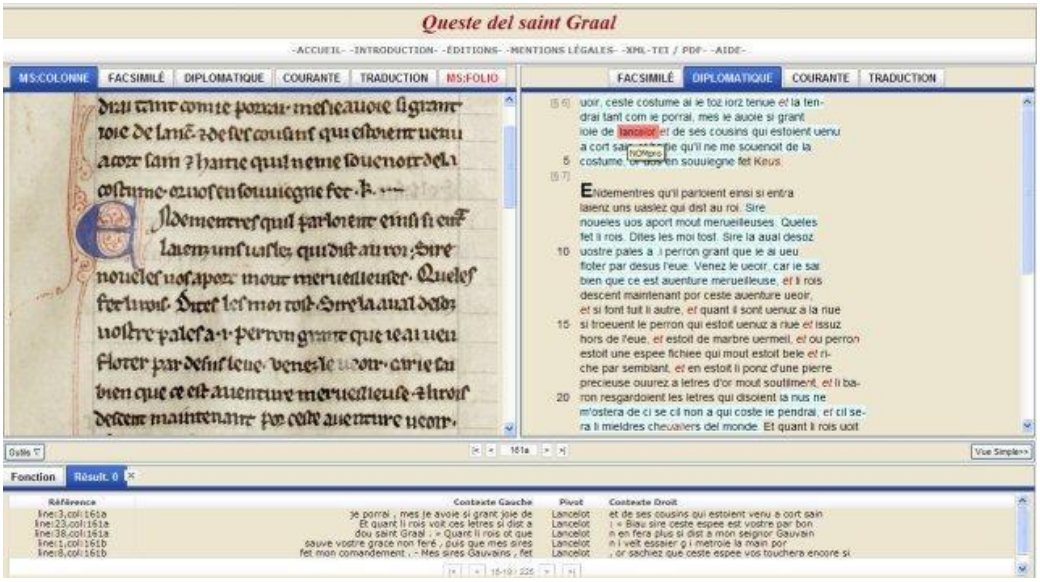
[6] Centre National de la Recherche Scientifique
[7] Équipe Littératures et Arts Numériques
[8] Source : https://www.liglab.fr/fr/presentation/presentation-du-laboratoire

différents laboratoires et traités sous différents formats. Mon travail consiste à étudier un corpus constitué d'une partie des romans ciblés par le projet PhraséoMédiéval, et ce corpus sera utilisé pour mettre en place la chaîne de traitements utile à l'analyse. Je ne présenterai que les parties pertinentes des documents encodés aux formats XML et CSV, car je n'ai pas eu accès aux autres parties dans le cadre de mon travail.

1.2.1. Le corpus

Le corpus de ce projet se compose principalement des textes de romans du Moyen Âge, concernant des histoires et des légendes de l'époque. Avec l'apport de la technologie numérique, nous pouvons analyser le contenu du corpus à l'aide de différents outils d'analyse linguistique automatique.

Pour le manuscrit original, le texte doit d'abord être transcrit. L'affichage peut se faire en une ou deux colonnes, ce qui permet d'avoir, par exemple, la photographie numérisée du texte manuscrit et sa transcription diplomatique, ou l'édition courante et sa traduction, etc. (cf. Figure 1). Ces fonctionnalités sont possibles grâce à la plateforme TXM (Heiden, S. 2010)[9], qui est un environnement d'analyse de texte/corpus basé sur Unicode et XML[10], qui peut traiter les balises XML de type TEI[11]. Il propose également un moteur de recherche qui permet de trouver facilement des mots, des constructions et des catégories grammaticales dans le texte et de les trier et les combiner.



[9] https://sourceforge.net/projects/txm/
[10] Extensible Markup Language
[11] Text Encoding Initiative

Figure 1. Texte original et texte de la transcription en utilisant l'outil de TXM pour le roman de « Queste del saint Graal »[12]

Pour le texte de la transcription, il est possible d'ajouter des balises et d'effectuer l'annotation de la structure syntaxique et lexicale de chaque phrase en utilisant l'outil d'exportation XML sous TXM et en s'appuyant sur les paramètres du corpus SRCMF (Stein & Prévost, 2013)[13]. Ce traitement de textes a été réalisé selon les mêmes guides de transcription et de normalisation.

De même, afin d'obtenir une lemmatisation et un étiquetage performant des parties du discours, on peut utiliser d'utiliser LGeRM, une plate-forme de lemmatisation du français médiéval qui s'appuie sur un lexique et la prédiction des étiquettes pour désambiguïser et pour faire le traitement automatique des fichiers texte ou en XML (Souvay, 2020). Après avoir été déposé, le fichier est lemmatisé avec LGeRM et désambiguïsé à l'aide de TreeTagger et de procédures internes d'élimination et de correction. L'utilisateur peut ensuite visualiser les résultats de la lemmatisation et exporter un fichier au format CSV contenant les tokens, les étiquettes morpho-syntaxiques et les lemmes (cf. Figure 2).

```
Or  ADVgen  OR2
s'  PROper  SE1
en  PROadv  EN2
vet VERcjg  ALLER
Galehout    NOMpro  GALEHAUT
entre   PRE ENTRE
lui PROper  LUI
et  CONcoo  ET
son DETpos  SON4
compaignon  NOMcom  COMPAGNON
,   PONfbl  ,
liés    ADJqua  LIÉ
et  CONcoo  ET
dolens  ADJqua  DOLENT
:   PONfbl  :
```

Figure 2. La sortie de LGeRM pour un texte court

La quantité de données utilisée actuellement par ce programme est relativement conséquente. Toutefois, ce qui est réellement nécessaire pour mon travail ne concerne qu'une fraction de la totalité des données. Les résultats des données actuelles sont principalement générés sur la base de LGeRM, et j'ai donc besoin de fusionner les résultats

[12]Source : https://a.fsdn.com/con/app/proj/txm/screenshots/282255.jpg/max/max/1
[13] Syntactic Reference Corpus of Medieval French (SRCMF)

d'un autre analyseur syntaxique à des fins de comparaison et de complétion des relations de dépendance.

1.2.2. Le Lexicoscope

Développé en 2011, le Lexicoscope est un outil d'exploration de corpus spécifiquement conçu pour explorer des profils combinatoires lexico-syntaxiques de mots ou d'expressions basés sur des dépendances syntaxiques telles que les relations sujet-verbe ou verbe-objet[14]. Il permet d'extraire des *words sketches* (Kilgariff *et al.,* 2004) indiquant, pour un pivot donné, son plus fort cooccurrent pertinent en fonction des relations syntaxiques impliquées, par exemple : sujet, objet direct, modificateur, etc. Il permet également d'étudier la combinatoire non seulement des mots et d'expressions polylexicales, mais aussi des structures complexes définies par des configurations de traits multidimensionnels telles que des traits syntaxiques, morphosyntaxiques, lexicaux, sémantiques, etc. Cela en fait un outil adapté à l'étude des motifs phraséologiques (Legallois *et al*., 2016).

En tant qu'architecture spécifiquement conçue pour l'exploration de corpus arborés, « *le Lexicoscope permet un accès rapide aux cooccurrents syntaxiques des unités étudiées dans de grands corpus multilingues, mais sans avoir besoin de maîtriser un formalisme complexe pour formuler des requêtes* » (Kraif, 2016). La richesse des outils sous-jacents reste accessible aux utilisateurs non experts grâce à l'utilisation de principes ergonomiques originaux, tels que la reformulation de requêtes basée sur l'exemple.

Dans mon travail, après avoir lemmatisé, étiqueté et analysé en dépendances syntaxiques le corpus original, il était nécessaire de démontrer dans une étude pilote un exemple des fonctionnalités du Lexicoscope appliquées au corpus. Cet outil d'exploration a pu identifier avec précision la fréquence d'occurrences des unités phraséologiques dans le texte et montrer à l'utilisateur l'ensemble des possibilités dans différents contextes du texte en fonction des conditions limitées. L'objectif est d'étudier plus intuitivement l'interrelation entre les types d'unités phraséologiques et la construction d'une séquence textuelle particulière.

[14] Source : http://phraseotext.univ-grenoble-alpes.fr/lexicoscope_2.0/

2. Problématique et objectif du stage

2.1. Problématique et méthodologie

Ce mémoire s'inscrit dans le cadre d'un projet visant à étudier le rôle de la phraséologie étendue dans la structuration des genres textuels en ancien français, et consistait à effectuer le traitement et l'analyse syntaxique automatique des données du corpus existant. Afin d'intégrer le corpus d'étude dans l'outil de fouille textuelle - Lexicoscope, nous avons besoin d'avoir un corpus avec les lemmes et les dépendances. Deux outils différents sont à notre disposition : LGeRM et Hops. Cependant, LGeRM permet de lemmatiser mais n'extrait pas de dépendances ; Hops extrait les dépendances mais ne lemmatise pas. Par conséquent, afin d'obtenir une analyse en dépendance et une lemmatisation pour le corpus d'étude, le problème à résoudre est de savoir comment fusionner ces deux sorties provenant d'outils différents, en tenant compte des différences éventuelles de segmentations en phrase, de tokenisation et de ponctuation.

Afin d'atteindre cet objectif, nous proposons de procéder en deux étapes d'un point de vue méthodologique. D'une part, nous effectuons le travail de traitement du corpus, tel que la lemmatisation et l'étiquetage avec LGerM et l'analyse avec Hops. Nous comparons et observons les résultats en sortie de différents outils de traitement, après les avoir analysés et évalués autant que possible afin de les fusionner de la façon la plus avantageuse. D'autre part, nous les intégrons dans notre outil de fouille de textes, le Lexicoscope, afin de permettre aux linguistes d'analyser les unités préconstruites en s'appuyant sur la cooccurrence de ces unités lorsqu'elle est statistiquement significative. La cooccurrence étant en effet un phénomène qui joue un rôle central dans la construction de la textualité (Viprey, 2006).

2.2. Missions et objectifs du stage

Ce travail a été réalisé entre le 1er mars 2021 et le 31 août 2021. Comme indiqué précédemment, afin de pouvoir faire l'analyse syntaxique automatique du corpus de manière plus précise, deux outils différents sont utilisés pour effectuer l'analyse de corpus et intégrer les résultats dans le Lexicoscope. Pour ce faire, nous partageons le travail en trois parties. Premièrement, le corpus doit être retraité et converti au format CoNLL-U

(Nivre *et al.*, 2016)[15] pour passer à l'étape suivante, en tenant compte des différences dans l'analyse du corpus par les différents outils. Par exemple, il est difficile d'analyser la ponctuation à l'aide de l'analyseur syntaxique Hops, qui s'appuie sur le modèle de langue pré-entraîné FlauBERT (Le *et al.*, 2019)[16], mais il est possible de le faire sur la plate-forme de LGeRM. Deuxièmement, les nouvelles données sont analysées sous deux niveaux différents : 1) au niveau de la relation entre les tokens et 2) au niveau de l'étiquette des tokens. C'est pour cela que, les données sont traitées de manière différente. Pour terminer, les données finales sont étiquetées et importées dans le Lexicoscope. Des exemples d'applications qui accèdent à des données pertinentes de ce corpus sur la plate-forme sont également présentés.

[15] Les annotations sont encodées dans des fichiers de texte brut avec trois types de lignes :
1. des lignes de mots contenant l'annotation d'un mot/token dans 10 champs séparés par des caractères de tabulation simples.
2. les lignes vides marquant les limites de la phrase.
3. les lignes de commentaires commençant par un dièse (#).

[16] French Language Understanding via Bidirectional Encoder Representations from Transformers

Chapitre 2. État de l'art

Comme mentionné ci-dessus, l'objectif de ce travail est d'analyser syntaxiquement des corpus de romans du français médiéval en utilisant deux outils différents et d'intégrer les résultats des données à la plateforme du Lexicoscope. Elle permet un meilleur accès et une meilleure analyse des résultats de données. Avant de détailler chaque étape de mon travail et les résultats, je vais d'abord présenter le domaine de la phraséologie d'un point de vue linguistique et d'un point de vue du TAL et décrire brièvement les outils qui sont utilisés.

1. Étude sur l'approche phraséologique du roman en français médiéval au niveau linguistique

1.1. Phraséologie et genres textuels

La phraséologie, comme une discipline relativement jeune, est une branche de la linguistique qui étudie les combinaisons de mots partiellement ou totalement figées, appelées unités phraséologiques[17]. Dans le même temps, elle est aussi « *la congruence à la fois syntaxique et sémantique qui lie les unités lexicales entre elles pour donner lieu à des unités polylexicales qui se distinguent par une fixité d'emploi conditionnant leur fonctionnement interne et leur combinatoire externe* » (Grossmann, Mejri & Sfar, 2017 :7).

Il existe une tendance à étendre le domaine de la phraséologie, initialement limité aux syntagmes et aux phrases, à la linguistique du discours et aux études textuelles. Cette conception étendue de la phraséologie ne traite plus son objet comme un phénomène marginal, mais comme le noyau des modèles linguistiques qui postulent un principe phraséologique de la langue (Denoyelle, Sorba, 2020). Cela démontre que le phénomène de la phraséologie est lié à l'organisation du discours. En effet, l'appartenance d'un texte à un genre conditionne les variations lexicales, morphosyntaxiques et discursives que l'on y trouve par rapport aux autres genres.

En ce qui concerne la littérature médiévale, la question du genre est complexe (Denoyelle, Sorba, 2020). Les auteurs médiévaux auraient varié considérablement le genre dans leurs manuscrits, en fonction du sujet, du contenu, etc. Ainsi, il est difficile pour les chercheurs contemporains de distinguer clairement les genres de chaque œuvre. Les

[17] Source : https://fr.wikipedia.org/wiki/Phras%C3%A9ologie_(linguistique)

experts ont émis une nouvelle conjecture à cet égard : l'approche stylistique et contrastive, dans laquelle on situe les études phraséologiques, permet peut-être d'aborder le problème sous un autre angle, en se concentrant sur la question de savoir si l'étude des entrées phraséologiques peut contribuer efficacement à la caractérisation des genres médiévaux. L'objectif méthodologique est de renforcer l'approche contrastive, en incluant dans le corpus des textes d'autres genres médiévaux (chroniques, chansons de geste, vies de saints, etc.).

1.2. Historique et étude existante

Depuis les années 1970, la phraséologie est apparue comme une discipline affirmée. Cependant dans le cas du français, Michel Bréal (1897) mentionne certains groupes de mots qu'il appelle formules, locutions ou groupes articulés. Ils se caractérisent par leur fixité et leur opacité sémantique (Rey, 2000). Il s'agit en quelque sorte d'un précurseur du concept de phraséologie. Ce n'est qu'après plusieurs années de recherche, sur la base d'une théorie formulée par des linguistes tels que Charles Bally (1909), que la phraséologie s'est réellement constituée en discipline indépendante[18]. En tant que discipline en plein essor, les chercheurs qui travaillent sur la phraséologie sont plus nombreux dans les années 2010. La phraséologie a un large caractère interdisciplinaire, car les unités phraséologiques sont étudiées sous plusieurs angles : lexical, syntaxique, stylistique, sémantique, et aussi avec des liens avec l'étymologie. Son champ d'étude est donc très large avec la syntaxe, la pragmatique, la sociolinguistique, la psycholinguistique et même le domaine de la culture en général (Piirainen *et al.,* 2008). Les chercheurs peuvent mener des études pertinentes à partir de diverses perspectives linguistiques ou d'une perspective bilingue ou multilingue.

L'exploration des liens entre la phraséologie et les genres textuels est également l'une des nombreuses études connexes. Il s'agit de montrer comment les unités phraséologiques peuvent caractériser les genres textuels et permettre de les distinguer les uns des autres.

[18] Source : http://www.phraseonet.com/fr/la-phraseologie

2. Étude sur l'approche phraséologique du roman en français médiéval au niveau du TAL

2.1. Lien entre deux domaines

Le TAL et la linguistique de corpus jouent un rôle décisif dans le développement de la phraséologie. En tant que domaine situé entre la lexicologie et la syntaxe, la phraséologie est consacrée à l'étude des combinaisons de mots récurrentes et arbitraires. On les appelle souvent « collocations » ou « unités phraséologiques » (Benson, M., 1989). Bien qu'elles soient des éléments constants du discours, les combinaisons de mots lexicalisées restent un élément subliminal du langage, refusant d'être facilement identifiées. « Souvent absentes des dictionnaires traditionnels, elles parsèment le discours d'effets de "déjà entendu" et ne se présentent pas à l'esprit distinctement comme des unités minimales de la langue. » (Pecman, 2005) L'apport des corpus textuels informatisés et des programmes qui les exploitent a permis aux chercheurs en linguistique d'établir des schémas lexico-grammaticaux dans la langue qui étaient auparavant inaccessibles aux observateurs (Gledhill, Ch. J., 2000), c'est-à-dire qu'il a permis de reconnaître des séquences linguistiques récurrentes.

2.2. Avantages et limites

Le TAL et les corpus présentent un avantage important et direct dans la recherche sur la phraséologie. Dans de nombreux cas, par exemple dans d'autres disciplines de la linguistique, la contribution des corpus et du TAL consiste principalement à garantir l'objectivité, mais en phraséologie les corpus peuvent être considérés comme les véritables piliers de la recherche. Un exemple intuitif est donné par M. Pecman (2005). Elle explique qu'il n'est pas facile pour un chercheur de déterminer si une séquence aussi usuelle que *to start a car* est une combinaison libre d'éléments lexicaux ou une combinaison restrictive. Elle ressemble beaucoup plus à une structure libre. Pour ses deux éléments constitutifs, le verbe *start* et le nom *car*, tous deux ont une très grande possibilité de combinaisons acceptables (*to start [a lesson/ a play/ a book/ studying/ cooking/ singing/...], to [buy/ drive/ wash/ lend/ break/ look at/...] a car*). Cependant, il est possible de connaître la probabilité exacte des occurrences de leur apparition mutuelle grâce à l'utilisation de techniques liées au TAL et de méthodes de la statistique lexicale. Le chercheur peut en conclure et affirmer leur statut phraséologique (Smadja, F., 1993).

Dans le même temps, l'utilisation du TAL et des technologies liées aux corpus présente certaines limites qu'il est actuellement difficile d'éviter. Lorsqu'il s'agit du traitement automatique de corpus pour l'identification d'unités phraséologiques, les résultats obtenus par ces techniques rencontrent d'importantes limitations, à savoir : l'exclusion des unités phraséologiques à fréquence faible (Curado, F. A. 2001), et l'incertitude de la précision de l'identification des unités phraséologiques. Cela s'explique par le fait que le statut des unités phraséologiques est flou : les notions de figements, d'opacité, de fréquence constituent des continuums, et il est toujours arbitraire de tracer des frontières dans un continuum. Les unités phraséologiques sont des objets complexes qui ne peuvent pas être caractérisées d'une seule manière. Cela pourrait devenir un problème potentiel pour les études pertinentes. Par conséquent, le traitement automatique des phrases nécessite parfois une relecture manuelle des résultats obtenus à la fin de l'exploitation du corpus.

2.3. Outils utilisés

Dans le domaine du TAL, la lemmatisation automatique est l'objet de nombreuses recherches. Par conséquent, de nombreux outils ont été développés. Ils ont leurs propres caractéristiques en raison de leurs différents besoins en matière de traitement des données. Il existe de nombreuses plate-formes et outils, tels que : TreeTagger (Schmid, 1994), LGeRM, Pie (Manjavacas et al., 2019), UDPipe (Straka & Straková, 2017), etc. Il existe également de plus en plus d'analyseurs avec différents modèles. Dans mon travail, j'utilise à la fois la plateforme LGeRM et l'analyseur syntaxique Hops.

2.3.1. LGeRM

LGeRM est un outil développé depuis 2004 par G. Souvay et *al.* pour aider à la lemmatisation du moyen français. Avec l'augmentation des fonctionnalités, LGeRM peut traiter non seulement le français médiéval (1330-1500), mais aussi s'adapter au français du XVIe et XVIIe siècle, ainsi qu'au français moderne. Il contient deux lexiques morphologiques : l'un adapté au français médiéval et l'autre adaptéau français du XVIe et du XVIIe siécle. LGeRM est une plate-forme de lemmatisation qui permet aux utilisateurs d'effectuer une analyse des textes encodés selon la norme XML TEI et d'étudier leur vocabulaire[19].

[19] Source : http://stella.atilf.fr/LGeRM/

L'un des points forts de la plate-forme est sa bonne performance en matière de désambiguïsation. En général, le lemmatiseur fournit toutes les analyses possibles pour une forme donnée, ce qui génère du bruit lors de l'analyse lexicale du texte. En revanche, il utilise l'étiquette définie par TreeTagger pour réduire le taux d'ambiguïté et donner un résultat plus précis.

L'évaluation de LGeRM par G. Holgado et *al.* (2021) donne une bonne indication de sa performance. L'expérience est basée sur l'évaluation de plusieurs outils différents pour la lemmatisation automatique du français médiéval, en utilisant un même ensemble de textes qui couvre la période du IXe jusqu'au XVe siècle. Il est divisé en 10 parties d'évaluation, comprenant des corpus de textes de caractéristiques linguistiques et de tailles différentes. Le tableau 1 montre les résultats en termes de précision. Comme on peut le constater, grâce à un lexique très riche du français médiéval, LGeRM a obtenu le meilleur résultat de tous les outils, avec une précision moyenne de 83%.

T	**CA**		**CC**			**TreeTagger**		**LGeRM**		**UDPipe**		**Pie**	
	tokens	%	tokens	m.inc.	%	tout	inc.	tout	inc.	tout	inc.	tout	inc.
1	177 050	41,1	254 094	28 976	11,4	0,75	0,07	0,83	0,82	0,67	0,12	0,74	**0,36**
2	383 164	88,9	47 965	1275	2,6	**0,86**	0,09	0,83	**0,84**	**0,76**	0,19	0,71	0,22
3	425614	98,7	5530	770	13,9	0,73	0,07	0,78	0,56	0,65	0,09	0,60	0,21
4	395 832	91,8	35 312	5399	15,3	0,72	0,07	0,85	0,75	0,70	0,13	0,61	0,20
5	413 123	95,8	18 021	2313	18,8	0,77	0,10	0,86	0,63	0,69	0,12	0,69	0,26
6	420 109	97,4	11 035	1405	12,7	0,81	0,20	**0,90**	0,71	0,71	0,20	0,69	0,23
7	408 375	94,7	22 769	2008	8,81	0,79	0,12	0,89	0,77	0,73	0,20	**0,75**	0,31
8	426 052	98,8	5092	1622	31,8	0,43	0,02	0,61	0,39	0,42	0,06	0,45	0,16
9	420 652	97,6	10 492	1711	16,3	0,74	0,13	0,82	0,54	0,68	0,09	0,67	0,16
10	419 163	97,2	11 981	2380	19,9	0,76	**0,29**	**0,90**	0,77	0,64	**0,21**	0,66	0,17
	Moyenne					0,74	0,12	0,83	0,68	0,66	0,14	0,66	0,23

Tableau 1. La précision des outils différents (CA corpus d'apprentissage ; CC corpus de contrôle)

LGeRM est également agréable à utiliser. Pour l'utilisateur moyen, l'outil est disponible dans une version simple en ligne. L'utilisateur dépose simplement le fichier sur la plate-forme dans le bon format et le résultat de la lemmatisation s'affiche immédiatement à l'écran. Dans mon travail, comme il fait partie d'un projet collaboratif, il permet de modifier le corpus concerné et de télécharger directement les résultats des données en CSV.

2.3.3. Hops

Hops est un analyseur syntaxique en dépendances pour le français, qui est relativement au niveau de l'état de l'art sur la plupart des corpus de référence, car

l'analyseur syntaxique est basé sur une représentation lexicale très riche, issue de FlauBERT (Grobol *et al.*, 2021).

FlauBERT est l'un des modèles de langue pré-entraînés les plus performants à l'heure actuelle, et permet d'obtenir une amélioration significative des résultats dans de nombreuses tâches TALN[20] (Le *et al.*, 2020). Ces modèles peuvent être entrainés d'une manière non supervisée sur une quantité colossale de textes bruts disponibles, permettant ainsi l'extraction de représentations continues de mots et l'analyse contextuelle au niveau de la phrase (Le *et al.*, 2020). Tout comme les modèles analogues centrés sur l'anglais dans le sillage de BERT (REF), FlauBERT constitue un ensemble de modèles entrainés sur de grands corpus de français hétérogène. Il est ainsi adaptable à de nombreuses tâches liées à l'analyse du français.

Cependant, il existe également d'autres modèles pré-entraînés pour la langue française, tels que CamemBERT (Martin *et al.*, 2019), et des variantes de BERT (Devlin *et al.*, 2019). L'une des raisons pour lesquelles les concepteurs de Hops ont choisi FlauBERT s'explique par le fait qu'il s'agit d'un outil avec d'excellentes performances attestées sur différents corpus de test. En 2021, les développeurs de Hops, Grobol & Crabbé, ont effectué une série de tests sur différents jeux de données afin de vérifier les performances de leur système avec FlauBERT. Les expériences de développement sont réalisées sur le sous-corpus de développement du corpus UD_FRENCH-GSD (Guillaume *et al.*, 2019), qui est le deuxième plus grand corpus en français disponible en Universal Dependencies (Zeman *et al.*, 2020). Les résultats observés montrent que les résultats avec FlauBERT sont meilleurs lorsqu'on compare les performances du modèle avec mBERT ou sans représentation contextuelle (no BERT) (cf. Tableau 2). De plus, sa vitesse d'exécution, notamment sur GPU, s'est aussi améliorée (cf. Tableau 3).

Modèle	UPOS	UAS	LAS	CLAS
mBERT	*98,13*	93,95	92,08	88,00
FlauBERT	**98,56**	95,67	**94,19**	**91,16**
no BERT	97,24	91,74	89,04	84,02
Martin *et al.*	*98,18*	—	92,57	—
Stanza	97,30	91,38	89,05	84,38
UD-Pipe 2	97,98	92,55	90,31	—

Tableau 2. Le résultat des performances de Hops et d'autres analyseurs pour UD_FRENCH-GSD

[20] Traitement automatique du langage naturel

Modèle	Vitesse CPU (phrases/s)	Vitesse GPU (phrases/s)	Mémoire vive (GiB)
FLAUBERT	4,09±0,06	20,32±0,08	5,5
no BERT	22,96±0,12	33,03±0,79	3,5

Tableau 3. Les vitesses de CPU/GPU pour différents modèles

Un modèle de l'analyseur Hops est utilisé dans mon travail, et ses paramètres sont affinés sur le corpus SRCMF dans sa version Universal Dependencies (UD). Cet outil s'est révélé très utile pour le travail d'étiquetage et d'analyse du corpus en dépendances syntaxiques.

Partie 2

-

Analyse syntaxique automatique d'un corpus du français médiéval

Chapitre 3. Analyse syntaxique au niveau de la relation entre les tokens

Comme nous l'avons déjà expliqué dans la première partie, ce travail a utilisé deux outils, LGeRM et Hops, pour aider à l'analyse syntaxique du français médiéval. Dans ce chapitre, je vais décrire comment l'analyse syntaxique des relations entre les tokens peut être complétée avec ces outils.

Pour ce faire, trois étapes sont nécessaires : 1) la normalisation des données au format CoNLL pour le traitement avec le parseur[21] Hops ; 2) l'identification des phrases pour lesquelles les résultats de segmentations des tokens des deux outils diffèrent ; 3) et la conversion des résultats en graphiques pour la visualisation.

1. Normalisation des données

Afin de répondre aux exigences de format de l'analyseur Hops, le format des données doit d'abord être normalisé au format CoNLL-U avant de pouvoir être analysé.

CoNLL-U est un format qui offre un large éventail d'applications. Il est généralement utilisé pour les informations sur l'annotation d'un mot/token en dix champs, séparés par des caractères de tabulation simples (cf. Annexe 2). Chaque champ a sa propre composition spécifique. Dans cette phase de l'étude, nous nous concentrons sur les premier, deuxième, quatrième, septième et huitième champs. Comme le montre la figure 3, le premier champ est ID représentant l'indice du mot. Chaque nouvelle phrase est comptée à partir de 1. Le deuxième champ est FORM, qui indique la forme du mot ou le symbole de ponctuation. Le quatrième est UPOS, c'est-à-dire Universal part-of-speech (POS). Les septième et huitième champs sont HEAD et DEPREL : le premier indiquant la tête du mot courant, une valeur ID ou zéro (0), et le second indiquant « Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one »[22]. Le contenu représenté par ces champs nous permet de représenter le résultat d'une analyse syntaxique des phrases. Ce sont également les données importantes dont dépendent les travaux de conversion des données en représentation graphique sous forme d'arbre.

[21] Analyseur syntaxique/interpréteur

[22] Source : https://universaldependencies.org/format.html#conll-u-format
En français : la relation de dépendance universelle avec le HEAD (racine si HEAD = 0) ou un subtype spécifique à un langage défini de l'un d'eux.

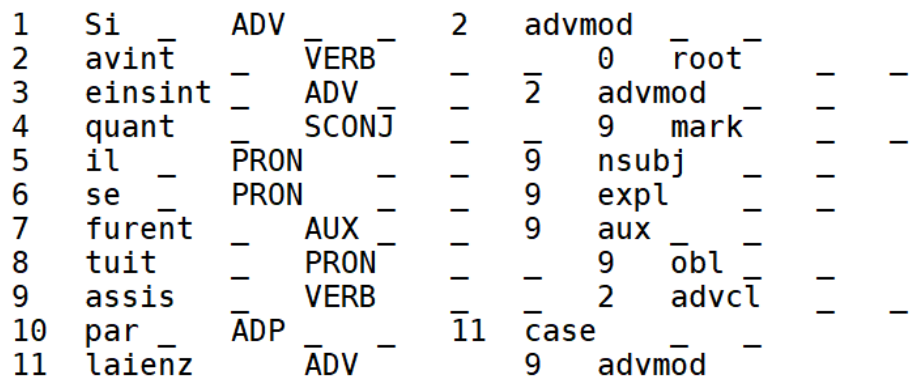


Figure 3. La sortie avec les champs complets

En tant que fichier d'entrée pour l'analyseur syntaxique, le format attendu consiste à remplir uniquement les premier et deuxième champs au format CoNLL, c'est-à-dire ID et FORM (cf. Figure 4). Cependant, les données originales sont les fichiers CSV des résultats de la lemmatisation fournis par LGeRM, et les fichiers XML traités avec le support SRCMF. Les fichiers LGeRM n'ont que trois champs pour chaque token : forme, étiquette et lemme (cf. Figure 5).

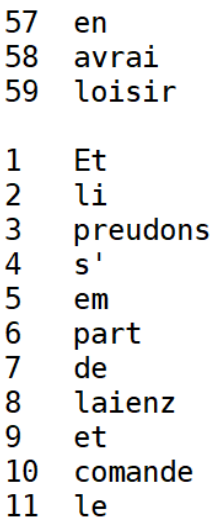


Figure 4. Le format attendu pour le fichier Input

	A	B	C	D
1	et	CONcoo	ET	
2	mes	DETpos	MON1	
3	sires	NOMcom	SIRE	
4	Gauvains	NOMpro	GAUVAIN	
5	lor	PROper	LEUR1	
6	demande	VERcjg	DEMANDER	
7	,	PONfbl	,	
8	coment	ADVsub	COMMENT2	
9	il	PROper	IL	
10	l'	PROper	LE	
11	ont	VERcjg	AVOIR1	

Figure 5. La sortie avec trois champs pour chaque token

Compte tenu de la nécessité d'extraire les champs requis sans affecter le contenu des données d'origine, il est important de réfléchir au prétraitement des données. En même temps, en raison de la nature de l'analyseur syntaxique, nous devons également supprimer toute la ponctuation des données existantes, et ajouter une ligne vide entre les phrases (l'analyseur n'ayant pas été entrainé pour analyser les ponctuations).

De plus, comme les données originales ne sont pas accompagnées de ID, il est nécessaire de renuméroter chaque token qui commence par 1, en utilisant la fin de la phrase comme marqueur.

Afin de répondre efficacement à toutes les exigences, j'ai écrit un script en langage python qui peut être exécuté simplement en suivant les commentaires pour obtenir les données et le format dont nous avons besoin. C'est-à-dire qu'il faut d'abord enlever toute la ponctuation, sauf le point. Il identifie ensuite les coupures de phrases et ajoute des lignes vides en fonction de la position du point. Enfin, chaque ligne de données est précédée du numéro correspondant (ID). L'idée principale est d'utiliser les étiquettes des données sources comme règle pour déterminer si le programme est exécuté ou non. La partie principale du script est comme suit :

```
for line in data.values:
    #supprimer les ponctuations faibles
    if line[1]!='PONfbl':
        #supprimer les ponctuations fortes et mettre les segmentations du phrases
        if line[1]!='PONfrt':
            f.write((str(i+1)+'\t'+str(line[0])+'\n'))
            i=i+1
        else:
            f.write(('\n'))
            i=0
```

Figure 6. Une partie principale de « traitement_fichier_1.py »

Après avoir testé des données de corpus de trois œuvres (« *Aucassin et Nicolete* », « *Queste del saint Graal* » et « *Le Roman de la Rose* »), il apparait qu'il est possible d'utiliser cette méthode pour transformer des données originales au format CoNLL comme fichier d'Input pour l'analyseur syntaxique.

2. Extrait et analyse des phrases différentes

En raison de la différence de segmentation des phrases et des tokens, les données de LGeRM et du SRCMF auront des valeurs différentes dans le champ de la forme après avoir été normalisées au format CoNLL-U. Or, comme Hops a été entrainé sur le SRCMF, il est important d'évaluer si ces différences de tokenisation et de segmentation ont un effet sur la qualité des sorties. Dans une première phase d'observation, nous avons donc comparé les données du fichier Input sur un même texte qui est tokenisé de deux manières différentes - selon le SRCMF et selon LGeRM.

Par conséquent, afin d'analyser plus efficacement les différentes relations au sein des phrases qu'ils produisent en raison des différentes tokenisations, il est nécessaire d'identifier toutes les phrases qui ont des tokenisations différentes les unes des autres, et les extraire dans deux nouveaux fichiers CoNLL. Pour ce faire, l'idée de ce script est de créer quatre listes. Deux fichiers d'Input sont écrits ligne par ligne dans chacune des deux listes (*list_LGerme* et *list_SRCMF*). Lorsqu'une ligne vide est rencontrée, c'est-à-dire à la fin d'une phrase, tout le contenu de cette liste est écrit dans l'autre liste (*list_L* et *list_S*) comme un élément. La liste précédente est vidée en même temps, et le résultat de l'analyseur est écrit à nouveau, ligne par ligne, jusqu'à ce que la prochaine ligne vide soit rencontrée. Cette opération est répétée jusqu'à ce que toutes les données aient été parcourues. Après cela, les deux listes (*list_L* et *list_S*) sont comparées pour voir si elles sont identiques, c'est-à-dire que s'il y a des éléments dans une liste qui sont identiques à ceux de l'autre, alors les deux éléments sont supprimés et ce qui reste est la phrase qui n'est pas identique. Après avoir parcouru tous les éléments, il ne restera que des phrases différentes. La partie principale du script est comme suit :

```
#mettre toutes les phrases du corpus de LGeRM dans une liste, qui sont séparés par les lignes vides
with infopen_LGerme as files:
    for i in files:
        list_LGerme.append(i)
        if i == "\n":
            list_L.append(list_LGerme)
            list_LGerme = []
#mettre toutes les phrases du corpus de SRCMF dans une liste, qui sont séparés par les lignes vides
with infopen_SRCMF as filee:
    for i in filee:
        list_SRCMF.append(i)
        if i == "\n":
            list_S.append(list_SRCMF)
            list_SRCMF = []
#Supprimer les phrases qui sont exactement correspondent dans les deux listes
for i in range(len(list_L)):
    for j in list_L:
        if j in list_S:
            list_L.remove(j)
            list_S.remove(j)
#écrire les deux fichiers qui contiennent des phrases de tokenisation différentes dans deux nouveaux fichiers.
list_L= eval('[%s]'%repr(list_L).replace('[', '').replace(']', ''))#Supprimer les listes redondantes
with outfopen_LGerme as outfilee:
    for i in list_L:
        outfilee.write(str(i))
list_S= eval('[%s]'%repr(list_S).replace('[', '').replace(']', ''))
with outfopen_SRCMF as outfile:
    for i in list_S:
        outfile.write(str(i))

#close
```

Figure 7. Une partie principale de « comparer_les_phrases_1.py »

Une fois que les deux fichiers, qui contiennent les phrases dont la tokenisation diffère, ont été extraits avec succès, ils sont placés dans l'analyseur syntaxique comme fichiers d'input pour pouvoir analyser les différences de résultats. Le résultat représente deux fichiers contenant les dix champs complets. Afin de voir les résultats plus directement, je les ai fusionnés en un seul tableau. À l'aide de la commande *diff*, j'ai analysé les deux résultats séparément pour les différences de token et de relation. Avec ce fichier, il est possible de voir clairement les différences qui ont été marquées (cf. Tableau 4).

									Roman_Rose											
			(LGeRM à gauche, SRCMF à droite)														Différence de rattachement		Différence de token	
1	1	Qui	_	PRON	_	_	3	nsubj	_	_	1	Qui	_	PRON	_		3	nsubj		_
	2	me	_	PRON	_	_	3	iobj	_	_	2	me	_	PRON	_		3	iobj		_
	3	donroit	_	VERB	_	_	0	root	_	_	3	donroit	_	VERB	_	DIFF	18	csubj	DIFF	_
											4	.iiii.	_	NUM	_	DIFF	5	nummod	DIFF	_
	1	iiii	_	PRON	_	_	0	obl	_	_									DIFF	
	1	besanz	_	ADV	_	_	2	obl	_	_	5	besanz	_	NOUN	_	DIFF	3	obj	DIFF	_
	2	conbien	_	ADV	_	_	14	advcl	_	_	6	conbien	_	ADV	_	DIFF	3	advmod	DIFF	_
	3	que	_	SCONJ	_	_	5	mark	_	_	7	que	_	SCONJ	_	DIFF	9	mark		_
	4	deboner	_	ADJ	_	_	5	amod	_	_	8	deboner	_	ADJ	_	DIFF	9	amod		_
	5	saie	_	VERB	_	_	2	advcl	_	_	9	saie	_	NOUN	_	DIFF	3	obl	DIFF	_
	6	se	_	SCONJ	_	_	11	mark	_	_	10	se	_	SCONJ	_	DIFF	15	mark		_

Tableau 4. Le résultat de la tokenization différente des phrases

Les résultats des données montrent que les documents traités par LGeRM diffèrent de ceux traités par SRCMF dans le processus de tokenisation. Les taux des phrases différentes sont indiqués dans le tableau 5 :

Titre de l'œuvre	« *Aucassin et Nicolete* »	*« Queste del saint Graal »*	« *Le Roman de la Rose* »
Nombre total de phrases	460	1315	281
Nombre de phrases différentes	0	278	68
Taux de phrases différentes **(NB différentes/NB total*100)**	**0**	**21.14%**	**24.19%**

Tableau 5. Les taux des phrases différentes

Le taux des phrases différentes est actuellement d'environ vingt pour cent, ce qui est un taux relativement élevé. Après avoir soigneusement analysé les raisons qui ont conduit à cette différence dans la segmentation des phrases, j'ai trouvé deux possibilités principales. L'une est la différence entre « l'en » (qu'un seul token) et « l' »+ « en » (deux tokens), et l'autre est la différence qui peut résulter de l'utilisation de chiffres romains[23], par exemple: « .i. », « .ii. », etc. La première possibilité est une différence dans la tokenisation qui, à mon avis, doit être conservée afin d'identifier les raisons de son apparition. La seconde possibilité, en revanche, s'apparente davantage à une simple erreur de traitement. Après en avoir discuté avec des experts de ce projet, nous pensons que cette différence dans la tokenisation différente des phrases peut être évitée en écrivant un script permettant de normaliser ces différences.

Après avoir écrit un programme pour fusionner les deux points qui avaient été séparés et les lettres intermédiaires (c'est-à-dire de « . » « i » « . » à « .i. »), nous pouvons alors normaliser le fichier de sortie LGeRM au début. Le fichier traité est ensuite utilisé pour la conversion du format CoNLL et pour l'extraction des phrases qui ne sont pas identiques. Les nouvelles données finales sont ensuite traitées par l'analyseur syntaxique et un résultat plus précis a été obtenu (cf. Tableau 6).

[23] Les chiffres dans le système romain sont représentés par les lettres latines I, V, X, L, C, D et M, correspondant respectivement aux nombres 1, 5, 10, 50, 100, 500 et 1000. Apparaît dans les données comme étant un point plus un chiffre plus un point (par exemple « .i. »).

Titre de l'œuvre	« *Aucassin et Nicolete* »	*« Queste del saint Graal »*	« *Le Roman de la Rose* »
Nombre total de phrases	460	1315	281
Nombre de phrases différentes	0	67	2
Taux de phrase différentes **(NB différentes/NB total*100)**	**0**	**5.09%**	**0.71%**

Tableau 6. Les nouveaux taux des phrases différentes

On peut remarquer qu'avec l'élimination du bruit lié aux chiffres romains, les taux de différence sont maintenant plus bas. C'est une nouvelle encourageante pour notre analyse.

3. Générer des graphiques

Pour faciliter la visualisation des résultats d'analyse et les comparer, nous proposons d'examiner les relations entre les tokens dans une phrase à l'aide de graphiques. Afin de convertir les données au format CoNLL en un graphique, deux méthodes différentes ont été essayées.

La première chose à expérimenter est l'un des outils de UD (Universal Dependency) qui s'appelle Arborator. C'est un outil d'annotation manuelle des dépendances qui prend en charge l'édition des étiquettes POS et des relations de dépendance dans une interface facile à utiliser[24]. Il propose des graphiques de couleurs pour différentes étiquettes et peut lire et écrire des dépendances au format CoNLL. Il existe également un « mode rapide » qui permet simplement de copier-coller les données CoNLL.

L'utilisation de cet outil est très facile car il fournit une démonstration du mode rapide qui peut être exécuté directement en ligne, simplement en copiant-collant les données CoNLL ; même les utilisateurs ayant peu de connaissances informatiques peuvent l'utiliser facilement. Il permet également de déposer plusieurs groupes de données en même temps et de produire plusieurs résultats simultanément (cf. Figure 8).

[24]Source : https://universaldependencies.org/tools.html#arborator

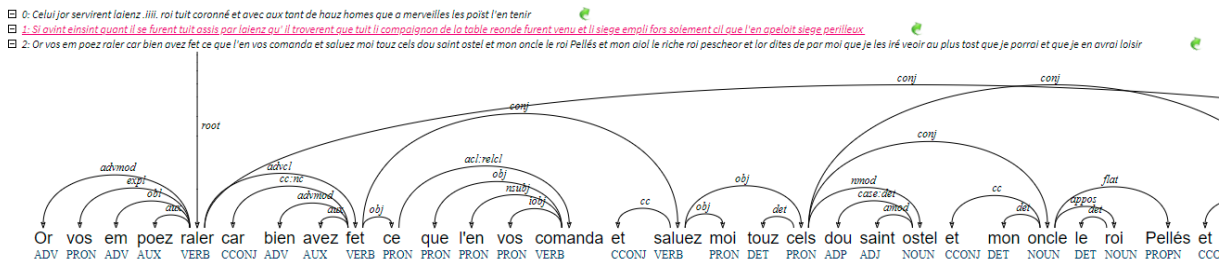


Figure 8. Plusieurs résultats d'analyse sur la page web

Comme le montre la figure ci-dessus, chaque phrase est répertoriée séparément et un graphique correspondant est généré sur la base des données. Chaque groupe de données est converti en un graphique avec un POStag et des relations de dépendance universelles. Ce site permet également de télécharger ces graphiques dans différents formats tels que « png », « jpg », « pdf », etc.

Toutefois, la démonstration de ce site ne permet pas de télécharger plusieurs fichiers en même temps. Cela signifie que chaque graphique généré à partir d'une phrase doit être téléchargé et enregistré manuellement. Par conséquent, si un grand nombre de données doit être analysé, l'outil Arborator, peut ne pas être approprié.

De ce fait, nous nous intéressons à un deuxième outil. Cette fois, nous utilisons un script appelé « conllu_to_viz.py ». Grâce au support de SpaCy, une bibliothèque logicielle Python, ce script peut lire directement les données CoNLL et générer un graphique des relations de dépendance universelles par phrase. Les fichiers générés sont automatiquement numérotés et stockés dans le chemin d'accès correspondant au format *html*, qui peut être consulté directement par l'utilisateur avec un navigateur.

Le script peut être exécuté avec les paramètres correctement configurés. Il s'avère que le programme peut reconnaître nos données et qu'il fonctionne bien. Plusieurs réunions ont été menées afin d'évaluer l'impact de la segmentation sur l'annotation syntaxique, ainsi que pour évaluer les arbres obtenus en sortie. À la fin de ces réunions, nous avons validé l'utilisation de Hops sur notre corpus, car les erreurs sont restées assez rares et n'ont pas affecté l'utilisation.

4. Difficultés rencontrées

Pendant le stage, c'est cette première étape du travail qui a pris le plus de temps. Cela est dû, non seulement, à la nécessité de me familiariser avec le contexte du projet et

de comprendre les informations pertinentes, mais aussi, à la nécessité de m'adapter au contenu du travail et d'apprendre de nouvelles connaissances impliquant ce travail.

Dans la première phase de la tâche, il y avait de grandes différences entre les données originales et les données requises. Les données des deux outils diffèraient également beaucoup après la tokenisation. C'est la raison pour laquelle la réflexion constante sur la manière d'enlever ou de modifier les parties différentes a été l'une des difficultés de cette phase du travail. Je vais donc décrire brièvement l'une des difficultés les plus représentatives lorsqu'on travaille avec des données similaires, et une difficulté plus technique, mais très pertinente pour mon travail et mon apprentissage.

4.1. Chiffres romains

Comme mentionné précédemment, les chiffres romains sous forme de « . » + « lettre » + « . » peuvent interférer avec la fonction de séparation des phrases, par exemple en traitant par erreur les points des chiffres comme des points finaux. Ainsi, ce qui est à l'origine une phrase complète est traité comme trois phrases. C'est le cas dans mon script.

Après réflexion, j'ai réussi à résoudre ce problème quoique cela a été laborieux. L'idée principale était d'ajouter des règles contextuelles pour identifier précisément les chiffres et les combiner en un token pour éviter la possibilité d'erreurs lors de la segmentation des phrases. La partie principale du programme est la suivante :

```
i = 0
for j in range(len(df)):
    # ne pas écrire la ligne de la deuxième ligne des chiffres romains
    if df.iloc[j][4]!='.,ponctuation' and df.iloc[j + 1][0] == '.' and df.iloc[j - 1][0] == '.':
        f.write(('\n'))
    #supprimer les ponctuations et mettre les segmentations du phrases
    elif df.iloc[j][4]!='.,ponctuation':
        f.write((str(i+1)+'\t'+str(df.iloc[j][0])+'\n'))
        i = i + 1
    #regrouper les chiffres romains au seul token
    elif df.iloc[j][0]=='.' and df.iloc[j + 2][0]=='.' and re.match("\w+",df.iloc[j + 1][0]):
        f.write((str(i + 1) + '\t' + str(df.iloc[j][0]) + str(df.iloc[j + 1][0]) + str(df.iloc[j + 2][0]) +'\n'))
        i = i + 1
    #ne pas écrire la ligne de la deuxième point des chiffres romains
    elif df.iloc[j][0]=='.' and df.iloc[j - 2][0]=='.' and re.match("^[a-z].*",df.iloc[j + 1][0]):
        f.write(('\n'))
    #supprimer les autres ponctuations fortes
    else:
        f.write(('\n'))
        i = 0
```

Figure 9. Une partie principale de « traitement_fichier_2.py »

A ce stade, le problème semble être résolu. Néanmoins existe-t-il une meilleure solution ? Après en avoir discuté avec les experts du projet, une nouvelle proposition est faite : certains éditeurs utilisent le pont médian « ·XX· » plutôt que « .XX. ». Comme le point médian n'est pas traité comme un point normal, la possibilité d'erreurs dans le

traitement automatique peut être réduite tout en préservant le texte original. Peut-être que pour les nouveaux textes, remplacer les points encadrants bas par des points médians serait une meilleure solution.

En plus de cela, nous pourrions également envisager la possibilité d'utiliser des expressions régulières pour remplacer ces points directement avant le traitement par LGeRM.

4.2. GPU et accès au serveur à distance

Le traitement de grandes quantités de données à l'aide d'un analyseur syntaxique sur CPU d'un ordinateur personnel prend beaucoup de temps pour accomplir les tâches du travail. En raison du travail à distance, il n'était pas possible de se rendre au laboratoire et d'utiliser l'ordinateur équipé avec GPU.

Pour résoudre cette difficulté, l'encadrant de mon stage m'a patiemment appris à me connecter à distance au serveur dans le laboratoire et m'a expliqué le fonctionnement de nombreuses commandes courantes pour les systèmes Linux. Cela m'a permis non seulement d'obtenir des données plus rapidement sur ce travail, mais aussi d'acquérir de nouvelles connaissances techniques.

Chapitre 4. Analyse syntaxique au niveau de l'étiquette des tokens

Dans la deuxième étape du stage, l'objet de travail est passé de l'étude des relations de dépendance à l'étude des étiquettes des tokens. Pour fournir les données à l'étude, j'utilise les outils cités précédemment, à savoir LGeRM et l'analyseur syntaxique.

En comparant les étiquettes des deux outils, nous cherchons des moyens d'identifier d'éventuelles erreurs afin d'améliorer l'analyse automatique. Dans ce chapitre, je décris comment standardiser les données en fonction d'un jeu d'étiquette commun, sélectionner et fusionner les phrases contenant différentes étiquettes.

1. Standardisation des étiquettes différentes

En linguistique, l'étiquetage morphosyntaxique (également connu sous le nom d'étiquetage grammatical, ou *POS tagging – part-of-speech tagging* – en anglais) est le processus qui consiste à associer aux mots d'un texte les informations grammaticales correspondantes comme la partie du discours, le genre, le nombre, etc. à l'aide d'un outil informatique[25]. Cet outil marque les mots d'un texte en fonction de leur nature (nom, adjectif, adverbe, verbe, etc.). Voici un exemple d'étiquetage morphosyntaxique :

- Texte original : *Nous sommes allées en Bretagne contempler de magnifiques allées couvertes du Néolithique*.
- Texte étiqueté : *Nous/PRO:PER sommes/VER:pres allées/VER:pper en/PRP/en Bretagne/NAM contempler/VER:infi de/PRP magnifiques/ADJ allées/NOM couvertes/VER:pper du/PRP:det Néolithique/NAM ./SENT*

Il existe une variété de jeux d'étiquette disponibles pour le traitement de la langue française, notamment EAGLES (Leech et Wilson, 1996), UD-POS (Petrov et al., 2011) et CATTEX (Prévost et al., 2013). Alors que EAGLES et UD-POS ont été développés comme des standards internationaux, CATTEX, lui, a été conçu spécifiquement pour les textes de la langue française médiévale.

Les deux outils que nous utilisons s'appuient sur UD-POS (pour l'analyseur syntaxique Hops) et CATTEX (pour LGeRM). Selon l'étiquette morpho-syntaxique utilisée, le résultat de l'étiquetage est complètement différent même lorsqu'un fichier similaire est traité. Dans ce cas, il est difficile de faire une comparaison directe de l'étiquette, même si les données du token sont totalement identiques. Ainsi, nous effectuons

[25]Source : https://fr.wikipedia.org/wiki/%C3%89tiquetage_morpho-syntaxique

d'abord un travail de conversion des différentes étiquettes afin de rendre cohérente l'écriture des étiquettes en sortie des deux outils.

Les différences entre les étiquettes CATTEX et UD-POS étant importantes, la méthodologie pour cette tâche consiste donc à convertir toutes les étiquettes d'un côté en étiquettes de l'autre. Après avoir consulté les informations sur Internet, j'ai trouvé un document sur le site de « Le wiki de la liste bfm »[26] (cf. Figure 10 et Annexe 3) sur la correspondance des étiquettes entre un jeu et l'autre.

Cattex2009	upos
ABR	X
ADJcar	ADJ
ADJind	ADJ
ADJord	ADJ
ADJpos	ADJ
ADJqua	ADJ
ADVgen	ADV
ADVgen.PROadv	ADV.ADV
ADVgen.PROper	ADV.PRON
ADVing	ADV
ADVint	ADV
ADVneg	ADV
ADVneg.PROadv	ADV.ADV
ADVneg.PROper	ADV.PRON
ADVsub	ADV
CONcoo	CCONJ

Figure 10. Tableau de conversion pour CATTEX et UD-POS

Nous constatons que le nombre d'étiquettes dans CATTEX est beaucoup plus élevé que le nombre dans UD-POS. Ceci est dû à la division minutieuse de CATTEX concernant la nature des mots dans les phrases (cf. Figure 11 et Annexe 4). Compte tenu du manque d'informations additionnelles, il est difficile de traduire un jeu d'étiquette plus grossier vers un jeu plus fin (par exemple, il est difficile de déterminer automatiquement si un ADV correspond à ADVgen, ADVint ou ADVneg). Je pense donc qu'il sera plus facile de convertir les étiquettes CATTEX en étiquettes UD-POS que le contraire.

[26] Source : https://groupes.renater.fr/wiki/bfm/index

Étiquette	*Catégorie*
ABR	ABRÉVIATION
ADJcar	ADJECTIF cardinal
ADJind	ADJECTIF indéfini
ADJord	ADJECTIF ordinal
ADJpos	ADJECTIF possessif
ADJqua	ADJECTIF qualificatif
ADVgen	ADVERBE (général)
ADVgen.PROper	contraction ADVERBE + PRONOM personnel
ADVint	ADVERBE interrogatif
ADVneg	ADVERBE de négation
ADVneg.PROper	contraction ADVERBE de négation + PRONOM personnel
ADVsub	ADVERBE subordonnant

Figure 11. Les catégories pour les étiquettes CATTEX

Afin de convertir les étiquettes, j'ai procédé au remplacement dans un script. L'idée centrale est de parcourir les données et de tout réécrire dans un nouveau fichier, en remplaçant directement le contenu du nouveau fichier si un caractère correspondant est trouvé, c'est-à-dire en remplaçant directement l'étiquette CATTEX par une étiquette UD-POS. Il s'agit d'un processus complexe mais qui permet de s'assurer que toutes les conditions de conversion sont respectées. Le programme a été testé et fonctionne sans aucun problème. Cependant, l'utilisation d'expressions régulières peut être envisagée pour des travaux futurs afin de simplifier le processus de remplacement. La partie principale du programme ainsi qu'un exemple d'entrée/sortie sont la suivante :

```
with infopen_LGerme as files:
    for i in files:
        i = str(i).replace('ABR', 'X').replace('ADJcar', 'ADJ').replace('ADJind', 'ADJ').replace('ADJord', 'AD.
        list_LGerme.append(i)
```

Figure 12. Une partie principale de « comparer_les_phrases_2.py »

```
1   Apréz   PRE APRÈS
2   la  DETdef  LE
3   mort    NOMcom  MORT1
4   le  DETdef  LE
5   bon ADJqua  BON
6   roy NOMcom  ROI1
7   Artus   NOMpro  ARTHUR
8   qui PROrel  QUI1
9   tant    ADVgen  TANT
10  fu  VERcjg  ÊTRE1
11  nobles  ADJqua  NOBLE1
12  roys    NOMcom  RAI1|ROI1
13  et  CONcoo  ET
14  gentilz ADJqua  GENTIL1
15  et  CONcoo  ET
16  entour  PRE ENTOUR
17  qui PROrel  QUI1
18  fu  VERcjg  ÊTRE1
19  et  CONcoo  ET
20  regna   VERcjg  RÉGNER
```

Figure 13. L'exemple d'entrée de LGeRM de ce script

```
1   Apréz   ADP APRÈS
2   la  DET LE
3   mort    NOUN    MORT1
4   le  DET LE
5   bon ADJ BON
6   roy NOUN    ROI1
7   Artus   PROPN   ARTHUR
8   qui PRON    QUI1
9   tant    ADV TANT
10  fu  VERB    ÊTRE1
11  nobles  ADJ NOBLE1
12  roys    NOUN    RAI1|ROI1
13  et  CCONJ   ET
14  gentilz ADJ GENTIL1
15  et  CCONJ   ET
16  entour  ADP ENTOUR
17  qui PRON    QUI1
18  fu  VERB    ÊTRE1
19  et  CCONJ   ET
20  regna   VERB    RÉGNER
```

Figure 14. L'exemple de sortie de LGeRM de ce script

2. Extrait et fusion des étiquettes

Après avoir converti toutes les étiquettes, nous pouvons commencer le travail d'extraction et de fusion des phrases. Afin d'analyser les causes possibles d'une différence d'étiquetage, il faut trouver les phrases contenant les différences. Pour pouvoir observer les différences directement, les phrases sélectionnées doivent également être fusionnées dans un même fichier.

Cependant, avant de pouvoir le faire, il faut vérifier que les deux fichiers contiennent exactement la même quantité de données, c'est-à-dire que les tokens peuvent correspondre exactement l'un à l'autre. Si ce n'est pas le cas, la raison de l'incohérence doit être identifiée et éliminée afin que les deux données correspondent exactement.

Une fois cette confirmation effectuée, le travail d'extraction des phrases qui n'ont pas les étiquettes identiques peut être réalisé. Le principe est similaire à celui du chapitre précédent (Chapitre 3). Un nouveau script est alors nécessaire pour comparer l'étiquette ligne par ligne, et si une étiquette différente est rencontrée, le programme enregistre une étoile « * » sur cette ligne dans la colonne suivant la dernière colonne de données. Si les données de la colonne marquée d'une étoile sont extraites séparément, un fichier enregistrant l'emplacement des différentes données dans l'étiquette est obtenu.

Enfin, afin d'obtenir un fichier qui fusionne les différentes étiquettes produites par LGeRM et l'analyseur syntaxique, les données des deux fichiers doivent être combinées et fusionnées en un seul fichier CoNLL (cf. Figure 15).

		Lemme(Lgerm)	Parser	Lgerm	Lgerm(originale)			étoile
1	Or	OR2	ADV	ADV	ADVgen	4	advmod	
2	s'	SE1	PRON	PRON	PROper	4	expl	
3	en	EN2	ADV	ADV	PROadv	4	obl	
4	vet	ALLER	VERB	VERB	VERcjg	0	root	
5	Galehout	GALEHAUT	PROPN	PROPN	NOMpro	4	nsubj	
6	entre	ENTRE	ADP	ADP	PRE	7	case	
7	lui	LUI	PRON	PRON	PROper	4	obl	
8	et	ET	CCONJ	CCONJ	CONcoo	10	cc	
9	son	SON4	DET	DET	DETpos	10	det	
10	compaignon	COMPAGNON	NOUN	NOUN	NOMcom	7	conj	
11	liés	LIÉ	ADJ	ADJ	ADJqua	7	amod	
12	et	ET	CCONJ	CCONJ	CONcoo	13	cc	
13	dolens	DOLENT	ADJ	ADJ	ADJqua	14	conj	
14	liés	LIÉ	DET	ADJ	ADJqua	7	amod	*
15	de	DE	ADP	ADP	PRE	16	case	
16	ce	CE1	PRON	PRON	PROdem	14	nmod	
17	que	QUE	SCONJ	SCONJ	CONsub	22	mark	
18	ses	SON4	DET	DET	DETpos	19	det	
19	compains	COMPAGNON	NOUN	NOUN	NOMcom	22	nsubj	
20	s'	SE1	PRON	PRON	PROper	22	expl	

Figure 15. Le résultat de fusion

Cinq documents, à savoir, « *ArtusdeBretagne* », « *LancprM1* », « *LancprM2* », « *MerlinProse* » et « *PremiersFaitsArthur* » ont été analysés à cette étape de la recherche. Les résultats finaux sont présentés ci-dessus. Ces résultats joueront un rôle important pour guider les phases ultérieures de ce projet.

Par exemple, dans la phrase : « *Mais de sa mort ne fet pas a parler ci endroit kar mors a si preudome com Galehout estoit ne fet pas a ramentevoir devant le point.* » (Vient de *LancprM1)*, le mot « *preudome* » est étiqueté ADJ (Hops) et NOUN (LGeRM). On peut déterminer que l'étiquette LGeRM est incorrecte. A cela s'ajoutent de nombreuses erreurs d'étiquetage corrigées grâce à ces divergences. Après une petite analyse d'environ 500 erreurs, une idée approximative peut être tirée : globalement, la proportion de divergences causées par LGeRM est supérieure à celle causée par Hops.

3. Difficultés rencontrées

À cette étape du travail, l'attention se porte sur les étiquettes des tokens eux-mêmes, et la difficulté du travail tourne autour de la conversion des étiquettes. Cependant, comme nous devons fusionner un fichier de tokens et une d'étiquette provenant des deux outils, il est également important que les données des tokens correspondent exactement. Les

difficultés de cette phase tournent, donc, autour de ces deux aspects. Je présenterai donc la difficulté la plus représentative de chaque cas.

3.1. Problème des lacunes du tableau de conversion

Lors de la conversion d'étiquettes, j'ai constaté que le tableau utilisé pour mettre en œuvre la conversion n'incluait pas tous les cas de conversion d'étiquettes. En plus des relations de conversion mentionnées dans le tableau, il y avait en fait d'autres relations de conversion. De ce fait, j'ai réalisé qu'il y avait des lacunes dans ce tavleau de conversion. Par exemple, la sortie du résultat d'étiquette dans LGeRM contient « PONfbl% » et « PON », qui auraient dû appartenir à « PUNCT », mais cette relation ne figure pas dans ce tableau.

En outre, la conversion entre les étiquettes n'est pas tout à fait exacte. Après tout, dans une opération automatique, il est difficile de convertir chaque phrase après une analyse minutieuse de son contenu et de sa structure. D'ailleurs, il s'agit plutôt d'une substitution directe. Les étiquettes, elles-mêmes, peuvent être écrites différemment mais expriment en fait le même contenu. Par exemple, « AUX » et « VERB », dont le premier est une étiquette CATTEX et le second une UD-POS, sont étiquetés différemment mais font tous deux référence à des verbes. Il est donc difficile de dire qu'il s'agit d'une « différence entre les résultats de deux outils » (cf. Annexe 3).

Pour résoudre ce problème, j'ai essayé d'effectuer quelques substitutions, comme le remplacement direct de tous les « AUX » par des « VERB », et le résultat a permis de réduire l'occurrence de telles erreurs d'appréciation. Cependant, d'autres lacunes peuvent exister dans le tableau de conversion.

3.2. Problème de ponctuation lors de la fusion

Afin de s'assurer que les étiquettes correspondent exactement après la fusion, il est nécessaire de s'assurer que les données des tokens dans les deux fichiers sont identiques lors de l'exécution du travail de fusion. Toutefois, comme les données produites par l'analyseur syntaxique ne comprennent pas de ponctuation, il est nécessaire de supprimer la ponctuation du fichier LGeRM.

Cependant, comme les étiquettes CATTEX sont classées avec précision, les balises correspondant aux ponctuations incluent « PONfbl », « PONfrt », « PONpdr », « PONpga » et « PONpxx ». En pratique, je constate que le fichier comprend non

seulement ces étiquettes mais aussi « PONfbl% » ainsi que « PON ». Afin de supprimer les ponctuations ainsi que les données associées, un script doit être écrit pour faciliter son exécution. La partie principale du programme est la suivante :

```
for line in data.values:
    #supprimer les ponctuations avec les étiquettes
    if line[1]!='PONfbl' and line[1]!='PONfbl%' and line[0]!='PONpxx'and line[0]!='PONpga'and line[0]!='PONpdr'
    and line[0]!='PON'and line[1]!='PONpxx'and line[1]!='PONpga'and line[1]!='PONpdr'and line[1]!='PON':
        if line[1]!='PONfrt':
            f.write((str(i + 1) + '\t' + str(line[0]) +'\t' + str(line[1]) +'\t'+ str(line[2])+ '\n'))
            i=i+1
        else:
            f.write(('\n'))
            i=0
```

Figure 16. Une partie principale de « supprimer_les_ponctuations.py »

Ainsi, le problème est résolu en exécutant ce script – script qui a, d'ailleurs, rapidement tout identifié et tout réécrit sauf la ligne de ponctuation. Il faut savoir que le problème lié à la ponctuation est présent tout au long de ce travail de stage et qu'il est dû aux différentes exigences des outils en matière de données. Avec un entraînement continu de l'analyseur, ce problème peut être résolu.

Chapitre 5. Construire un corpus pour le Lexicoscope

En tant qu'outil adapté à l'étude des motifs phraséologiques, le Lexicoscope peut être utilisé pour extraire des *word sketches* indiquant, pour un pivot donné, ses cooccurrents les plus fortement associés en fonction de la relation syntaxique mise en jeu. Par conséquent, afin de mener des études phraséologiques plus efficaces, après avoir lemmatisé et étiqueté le corpus, nous décidons d'intégrer ce corpus dans nos outils de fouille textuelle.

Dans ce chapitre, je montre comment compléter un fichier XML conforme au format d'entrée du corpus pour cette plate-forme. En même temps, j'explique comment consulter les résultats de ces données à l'aide du Lexicoscope.

1. Formatage en XML

Le Lexicoscope requiert un fichier au format XML contenant des métadonnées pour la création de la base de données. Et les données stockées dans le fichier XML doivent être enregistrées au format XML-CoNLL spécifique à ce logiciel (cf. Figure 17).

```
<?xml version='1.0' encoding='utf-8'?>
<corpus><doc><meta>authors  (c) Équipes SRCMF and BFM
bibl    (c) Équipes SRCMF and BFM, CNRS/ENS-LYON 2009-2015
date
discipline  littérature
firstPage
genre   roman
language    fr
languageId  fr
lastPage
monogr.title    SRCMF ArtusdeBretagne
publisher
title   Artus de Bretagne
volume
year
</meta><text><p id="p1"><s id="s1" newdoc_id="ArtusdeBretagne" sent_id="1">1    Apréz
2   la  LE  DET _   _   3   det _   _
3   mort    MORT1   NOUN    _   _   45  obl _   _
4   le  LE  DET _   _   6   det _   _
5   bon BON ADJ _   _   6   amod    _   _
6   roy ROI1    NOUN    _   _   3   appos   _   _
7   Artus   ARTHUR  PROPN   _   _   6   flat    _   _   ,
```

Figure 17. L'exemple de format XML

Après les précédentes étapes de travail, nous avons déjà obtenu un fichier CoNLL qui répond aux exigences. Cependant, afin d'adapter pleinement le fichier aux exigences du Lexicoscope, un traitement supplémentaire du fichier est nécessaire.

La première chose à faire est d'ajouter les données de la catégorie « lemme » au troisième champ de chaque ligne de données dans le fichier CoNLL produit par l'analyseur syntaxique. Ainsi, il est possible de remplir le fichier CoNLL avec l'information du lemme du fichier LGeRM directement bien que le fichier de l'analyseur ne porte pas le contenu du lemme, puisque nous avons établi une correspondance biunivoque entre les tokens des deux fichiers dans la phase de travail précédente.

De plus, comme le Lexicoscope permet de visualiser le contexte élargi de la phrase dans laquelle l'expression apparaît lorsqu'on l'interroge, la plateforme exige également que les données importées contiennent des informations sur la ponctuation afin d'assurer l'intégrité du texte contextuel. Cependant comme mentionné ci-dessus, les fichiers traités par l'analyseur syntaxique ne contiennent pas de données de ponctuation. Même si la ponctuation est incluse dans le fichier LGeRM d'origine, elle ne peut pas être ajoutée directement dans le fichier CoNLL, car les deux données ne correspondent pas. Pour résoudre ce problème, nous proposons d'ajouter la ponctuation à la fin de la ligne de données correspondant au mot de la phrase, c'est-à-dire au onzième champ de la ligne de token, réservé aux informations additionnelles. En prenant la phrase suivante comme exemple, le point doit être ajouté à la fin de la ligne pour le mot « vie ».

1	Si	SI4	ADV	_	_	3	advmod	_	_		
2	s'	SE1	PRON	_	_	3	obj	_	_		
3	entramerent		ENTRAIMER	VERB	_	_	0	root	_	_	
4	moult	MOULT	ADV	_	_	3	advmod	_	_		
5	tant	TANT	ADV	_	_	3	advmod	_	_		
6	comme	COMME	SCONJ	_	_	9	mark	_	_		
7	Dieux	DIEU	PROPN	_	_	9	nsubj	_	_		
8	lor	LEUR1	PRON	_	_	9	iobj	_	_		
9	donna	DONNER	VERB	_	_	3	advcl	_	_		
10	vie	VIE	NOUN	_	_	9	obj	_	_	.	

Figure 18. L’exemple d’une phrase avec le point

Afin d'extraire toute la ponctuation qui correspondrait aux données de CoNLL, j'ai écrit un script pour m'aider dans cette tâche. La partie principale du programme est la suivante :

```
for line in data.values:
    if line[1]!='PONfbl' and line[1]!='PONfrt' and line[1]!='PONfbl%'
    and line[0]!='PONpxx'and line[0]!='PONpga'and line[0]!='PONpdr'
    and line[0]!='PON'and line[1]!='PONpxx'and line[1]!='PONpga'
    and line[1]!='PONpdr'and line[1]!='PON':
        f.write(('&' +'\n'))
    else:
        f.write((str(line[0])))
```

Figure 19. Une partie principale de « trouve_ponc.py »

L'idée centrale est de supprimer manuellement la première ligne du fichier original, puis d'utiliser les étiquettes pour déterminer si la ligne est une ponctuation et l'écrire dans un autre fichier ou pas. S'il s'agit d'un mot, le symbole « & » est écrit - ce symbole a été choisi parce qu'il n'existe pas dans le fichier original et n'affecte donc pas les données originales lorsqu'il est supprimé. L'objectif est d'occuper la ligne afin qu'elle n'apparaisse pas comme une ligne vide. En effet, lors de l'écriture, s'il y a une ligne vide, le programme passera automatiquement à l'endroit suivant où il y a des données à écrire. Il en résulte un décalage des données de sortie. La raison pour laquelle je ne travaille pas directement sur le fichier CoNLL complet est la divergence des données entre les deux. Le fichier CoNLL n'est pas le même qu'un fichier original parce que toutes les ponctuations, sauf les points, sont supprimés, et la ligne de point est remplacée par une ligne vide séparant les phrases. Cependant, il n'y a pas de ligne vide dans le fichier original et toutes les ponctuations occupent une ligne. Il est donc difficile d'ajouter des ponctuations directement dans le fichier CoNLL.

Une fois que toute la ponctuation a été extraite, les données peuvent être écrites directement dans le fichier CoNLL et le fichier peut être parcouru pour enlever tous les symboles « & » et pour conserver toute la ponctuation à la fin de la ligne.

Enfin, pour convertir le fichier en XML, il est nécessaire d'ajouter les balises ainsi que les informations du corpus aux métadonnées. Cela complète le fichier en tant qu'entrée XML pour le Lexicoscope.

2. Visualiser les données à l'aide du Lexicoscope

Une fois les données importées avec succès, le nouveau corpus créé sous le nom de phraseoMedieval_frm, est disponible sur le site du Lexicoscope. Dans l'écran de sélection du corpus, nous pouvons voir les informations et la description du corpus sélectionné et

consulter librement les fonctions de « Vocabulaire du corpus », « Collocations et colligations » et « Métadonnées » concernant ce corpus.

Sur la page de « Vocabulaire du corpus », nous pouvons voir toutes les œuvres qui composent ce corpus et les statistiques globales qui s'y rapportent (cf. Figure 20). Il est facile de visualiser les informations de toutes les œuvres dans leur ensemble ou de visualiser les informations d'une œuvre en particulier.

Figure 20. Les statistiques globales pour le corpus phraseoMedieval_frm

Un exemple de l'analyse à l'aide du Lexicoscope se trouve dans l'étude de Denoyelle et Sorba (2020) sur la phraséologie du roman médiéval. Le verbe « *donner* » est utilisé comme un verbe de haute fréquence au sein de nombreuses unités phraséologiques. Denoyelle et Sorba ont choisi quatre constructions de « *donner* » pour leur analyse, et les résultats sont présentés ci-dessous :

	Doner + cop(s)	***Doner + conseil***	***Doner + congié***	***Diex + doner***
Lancelot T1&T2	57 occ.	9 occ.	2 occ.	28 occ.
Tristan T1&T3	53 occ.	8 occ.	6 occ.	46 occ.
Artus de Bretagne	7 occ.	1 occ.	12 occ.	35 occ.
Merlin (prose) (extrait)	0	0	1 occ.	2 occ.
Aucassin et Nicolette	0	1 occ.	0	2 occ.
Yvain	5 occ.	1 occ.	0	15 occ.
Jehan et Blonde	3 occ.	0	2 occ.	7 occ.
Roman de Silence	2 occ.	2 occ.	0	16 occ.
Tristan (Béroul)	1 occ.	6 occ.	1 occ.	5 occ.
Total	128 occ.	28 occ.	24 occ.	162 occ.

Tableau 7. La répartition des quatre constructions d'analyse

Nous pouvons remarquer que les constructions « *doner + cop* » et « *Diex + done* » sont beaucoup plus fréquentes que les autres, et que « *Diex + doner* » est la plus fréquente. « *Cela s'explique par le fait qu'il possède un patron syntaxique (N sujet + V) différent des trois autres (V + N complément) et il présente un caractère fortement idiomatique comme la forme de salutation recommandée* » (Denoyelle et Sorba, 2020).

Je vais utiliser « *Diex + doner* » comme exemple pour montrer comment visualiser les données dans le corpus phraseoMedieval_frm sur le Lexicoscope. Tout d'abord, afin de trouver l'expression, nous pouvons taper « *Diex doner* » dans la page de « Définition de la requête ». Le Lexicoscope nous donnera automatiquement les différentes suggestions de requête et la fréquence d'occurrence estimée (cf. Figure 21).

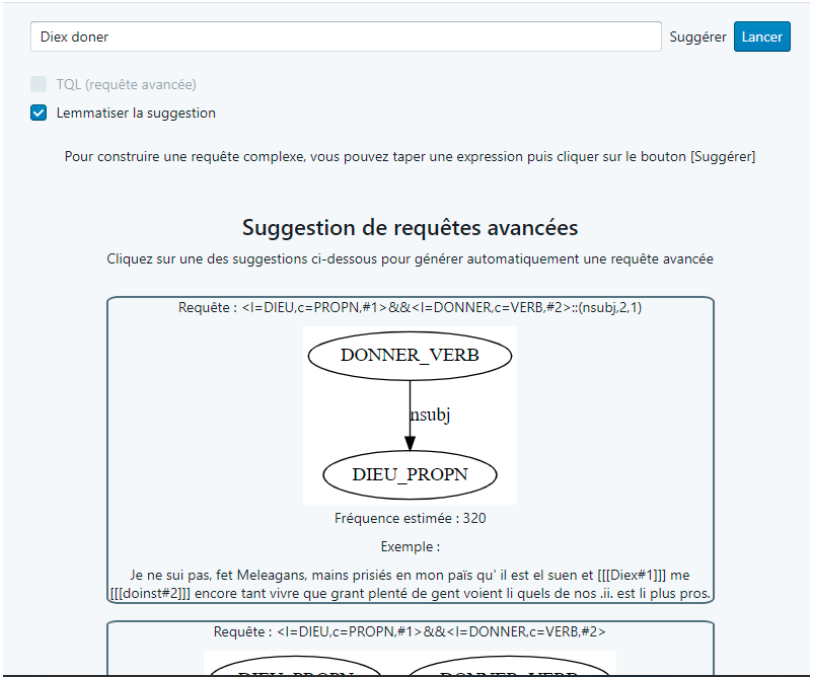


Figure 21. Les suggestions de requête et la fréquence estimée pour « *Diex doner* »

On clique sur l'image correspondante pour entrer automatiquement <l=DIEU, c=PROPN,#1>&&<l=DONNER,c=VERB,#2>::(nsubj,2,1) afin d'accéder à la page de recherche, où nous pouvons voir des informations plus complètes sur les résultats de l'étude, notamment les spécificités par corpus, la fréquence par réalisation, etc.

La page de la fonction de concordances montre toutes les phrases avec la construction « *Diex doner* » qui apparaissent dans l'œuvre (cf. Figure 22).

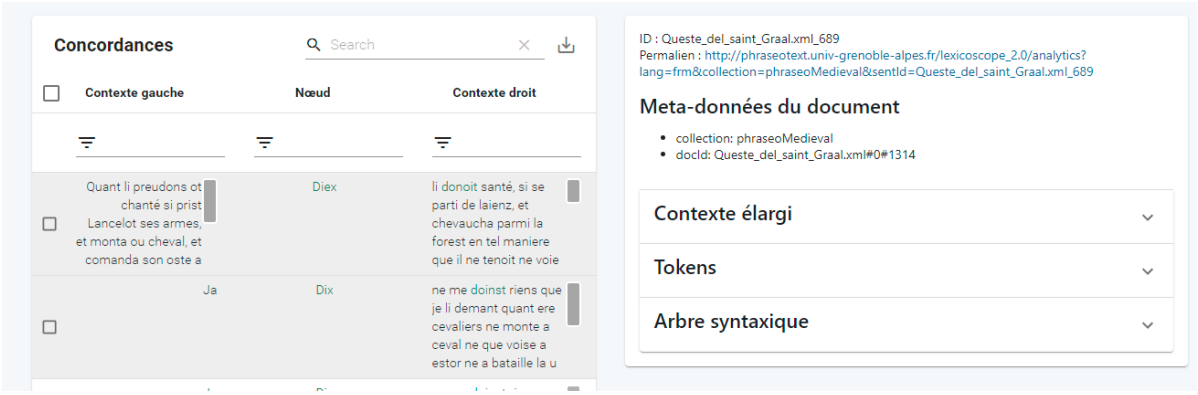


Figure 22. Le page de concordances pour « *Diex doner* »

Comme indiqué ci-dessus, cette page dispose d'autres fonctionnalités pratiques, dont l'affichage du contexte élargi de la phrase sélectionnée, l'affichage des tokens ou encore la génération automatique d'un arbre syntaxique.

Sur la page de la fonction de cooccurrences, il montre les unités qui apparaissent le plus fréquemment avec « *Diex doner* » (cf. Figure 23). Cela peut grandement faciliter la recherche sur les unités phraséologiques.

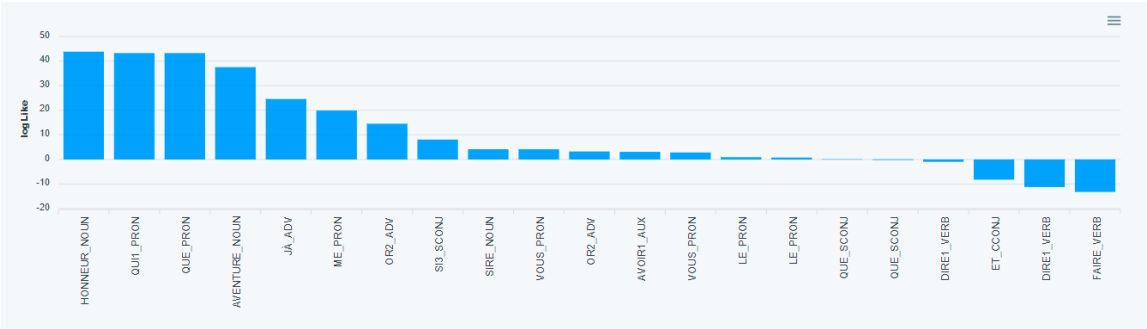


Figure 23. Les mots avec la relation de cooccurrences avec « *Diex doner* »

Le Lexicoscope possède bien d'autres fonctions non mentionnées ci-dessus, et ces fonctions puissantes et variées aident les chercheurs dans leur travail. Ainsi, on constate que, grâce à ses fonctionnalités, il s'agit effectivement d'un outil approprié pour étudier les motifs phraséologiques. Dans notre projet, l'utilisation du Lexicoscope a été plus que nécessaire.

Partie 3

-

Résultats

Chapitre 6. Analyse des résultats du travail

Après les trois étapes de la tâche, ce stage arrive à son terme. Afin d'indiquer la valeur et la signification de ce stage, j'analyse brièvement dans ce chapitre tout le travail qui a été fait.

Pour ce faire, je résume et j'explique d'abord ce que j'ai réalisé dans chacune des trois étapes du stage, et si elles ont eu un effet positif sur la recherche et le projet. Je poursuis avec une brève discussion sur ce stage. La problématique et l'objectif de ce stage sont également repris dans ce chapitre.

1. Résultats obtenus

Comme ce stage a été divisé en trois parties et que chaque partie a son propre travail défini, je présente les résultats de chaque étape du travail en trois parties.

1.1. Résultats de l'analyse syntaxique automatique des relations entre les tokens

Cette étape du travail est centrée sur l'analyse des relations entre les tokens dans les phrases. Après analyse et traitement, les données des deux outils sont converties dans un nouveau format. En même temps, toutes les phrases avec une segmentation différente sont extraites. A ce stade, nous nous intéressons aux données obtenues par l'analyseur syntaxique après le traitement de ces phrases, et en particulier aux différences dans les relations entre les tokens dues à la tokenisation différente. Afin de visualiser cette différence, les fichiers CoNLL de sortie sont convertis en graphiques pour être analysés.

La plupart des différences dans la segmentation des phrases dans cette analyse est due à des différences dans la division de « l'en ». Sinon, il n'y a pas d'erreurs évidentes dans le processus de tokenisation. Nous pouvons conclure, assez sommairement, qu'il n'y a pas de différences majeures entre les deux outils en termes de segmentation des phrases et de tokenisation.

1.2. Résultats de l'analyse syntaxique automatique des étiquettes des tokens

À cette étape, le travail se concentre sur les étiquettes des tokens. Après conversion des deux formats différents d'étiquettes, je fusionne les données pour m'assurer que les tokens sont exactement les mêmes. Afin de savoir quelles étiquettes différentes d'un même token sont correctes et lesquelles sont mal classées, nous avons besoin de l'aide d'un expert

en grammaire française médiévale. Dans cette étude, Mme Corinne Denoyelle a corrigé manuellement un grand nombre de phrases analysées dans le « *LancprM1* » qui contiennent des étiquettes différentes. Grâce à sa contribution, il est possible d'avoir une brève analyse des résultats de cette étape.

Étiquette	Analyseur syntaxique	LGeRM
ADJ	11	27
ADP	10	7
ADV	25	36
CCONJ	10	4
DET	10	4
NOUN	17	93
NUM	21	0
PRON	44	16
SCONJ	38	7
VERB	10	119
Répartition des erreurs identifiées grâce aux divergences (Répartition des erreurs = total/ (196+313))	38.5%	61.5%

Tableau 8. La fréquence des erreurs pour les étiquettes dans le cas des divergences entre les deux analyseurs

Le nombre total d'étiquettes (pour toute la phrase) à analyser est de 61 296, dont 509 cas de divergence sont analysés manuellement. Les résultats sont présentés dans le tableau ci-dessus. Nous pouvons voir que pour les cas de divergences d'étiquetage, l'analyseur syntaxique présente un taux de correction plus élevé que le LGeRM. Pour les étiquettes incorrectes spécifiques, l'analyseur syntaxique semble avoir fait plus fréquemment des erreurs dans l'attribution de PRON et SCONJ. Pour LGeRM, en revanche, il y a un grand nombre d'erreurs pour VERB et NOUN. Ces résultats peuvent peut-être constituer une indication ou une suggestion utile pour une orientation future de la recherche.

Étant donné que la quantité de données n'est pas encore assez importante et que les données proviennent toutes d'un seul texte, les résultats peuvent être soumis à un certain aléa. Une plus grande quantité de données est nécessaire pour obtenir des résultats plus fiables.

1.3. Résultats de l'intégration de corpus dans le Lexicoscope

Cette étape du projet a consisté à traiter à nouveau les données obtenues dans l'étape précédente et à les insérer avec succès dans le Lexicoscope, créant ainsi un corpus librement accessible. Cette tâche permet de faciliter l'accès aux résultats, et les fonctions pratiques du Lexicoscope sont une aide nécessaire pour l'étude des unités phraséologiques des romans français médiévaux.

Je pense que la création de ce corpus peut servir de base à d'autres recherches et études dans le cadre de ce même projet.

Pour mieux comprendre l'ensemble de la chaîne de traitement du travail, le schéma ci-dessous représente une vue globale de la chaîne de traitement, ainsi que le détail des noms des scripts qui interviennent tout au long de la chaîne de traitement (cf. Figure 24).

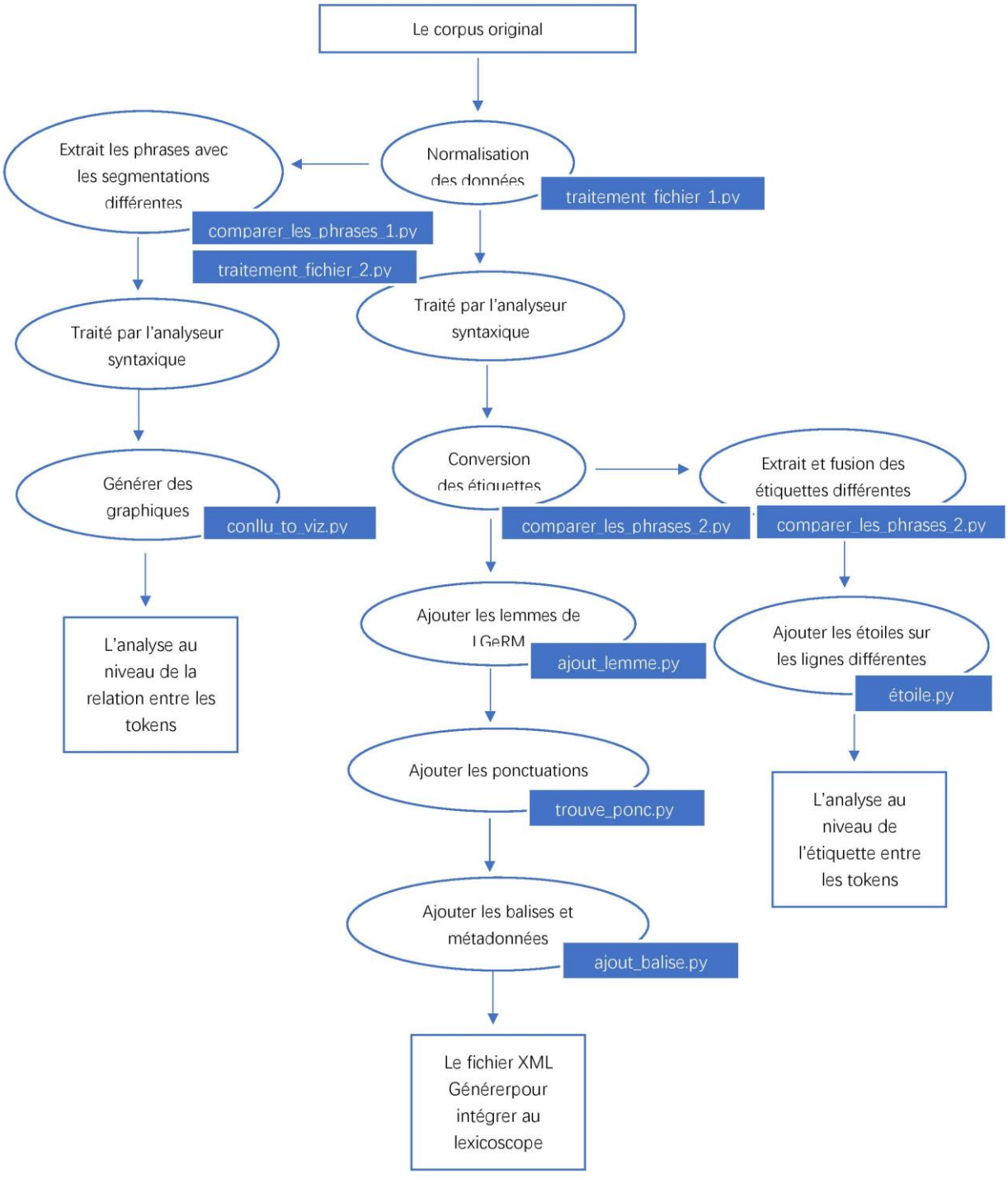


Figure 24. Le schéma global de la chaîne de traitement

2. Discussion

La tâche principale de ce stage était de réaliser une analyse syntaxique automatique d'un corpus de français médiéval. Le contenu du stage était très proche de notre expertise et les tâches ont été très bien organisées.

Pendant mon stage, d'une part, j'ai été exposée à beaucoup de nouvelles connaissances, telles que le fonctionnement du système Linux, la connaissance du format CoNLL, la connaissance des étiquettes variées, etc. D'autre part, ce travail a requis un usage intensif des connaissances de base en programmation, des connaissances liées au TAL et certaines connaissances linguistiques, ce qui m'a aidé à consolider et à améliorer mes compétences professionnelles.

Au cours de mon travail, j'ai acquis une grande quantité de données utiles à l'étude syntaxique du corpus en français médiéval, et dans la phase finale de mon travail, j'ai réussi à créer un corpus pertinent sur le Lexicoscope.

Cependant, la plupart des résultats produits au cours de ce travail nécessitent une certaine connaissance de la grammaire française médiévale afin de les évaluer et de les juger. Dans la plupart des cas, en raison de ce manque de connaissances, j'ai eu besoin de l'aide d'un expert pour pouvoir comprendre clairement les résultats des données. Cela m’a influencée dans la façon d’analyser les résultats des données.

Par ailleurs, à cause du travail à distance, la communication avec les encadrants de mon stage a été affectée négativement. Cela entraînait parfois une mauvaise compréhension de ce sur quoi je travaillais et ralentissait un peu le stage.

Personnellement, ce stage m'a ouvert les yeux sur un plus grand champ. Il m'a appris beaucoup de nouvelles choses et m'a permis de m'intéresser davantage au sujet de ce projet. Je considère le travail fourni durant ce stage comme une passerelle et une préparation à la recherche qui va suivre. Malheureusement, en raison des contraintes de temps liées au stage, je ne serai pas en mesure de participer à la prochaine phase de la recherche. Cependant, je pense que les résultats de mon travail peuvent servir de base à des recherches ultérieures.

Conclusion et perspective

1. Conclusion

Ce stage a été réalisé dans le cadre d'un projet qui propose de repérer et d'étudier les unités phraséologiques structurant précisément le genre romanesque médiéval. Les tâches du stage sont liées à l'analyse syntaxique automatique du corpus du français médiéval et sont divisées en trois parties principales : l'analyse syntaxique en dépendances ; l'analyse morphosyntaxique au niveau des étiquettes des tokens, et la construction du corpus pour le Lexicoscope.

Pour effectuer le travail, le corpus original doit être lemmatisé, étiqueté et analysé en dépendances syntaxiques en utilisant deux outils complémentaires : LGeRM, une plate-forme de lemmatisation du français médiéval, et l'analyseur syntaxique Hops qui s'appuie sur le modèle de langue pré-entraîné FlauBERT. Après avoir obtenu les sorties de ces outils, il faut les fusionner en considérant les différences éventuelles de segmentations en phrase, tokenisation et ponctuation, afin de procéder à l'intégration dans le Lexicoscope.

Avec la méthodologie adoptée, après une étude et une réflexion continue, et sous la direction des encadrants, les tâches du stage ont été réalisées en respectant les délais et avec les résultats correspondants. Et les résultats ont donné les bases des recherches ultérieures.

2. Perspectives

Les ressources et les outils utilisés pour ce stage sont excellents et la méthodologie utilisée pour guider la réalisation des tâches est très solide. Cependant, il est encore peut-être possible d'améliorer le travail.

Tout d'abord, ce projet est un projet de collaboration multidisciplinaire et il est normal que des connaissances de plusieurs disciplines soient impliquées. Toutefois, si une grande partie des données est analysée en collaboration par des experts d'autres disciplines, le travail est considérablement ralenti. Il serait probablement plus efficace de séparer les tâches en fonction des domaines de spécialisation dans les travaux futurs.

Deuxièmement, dans ce travail, nous utilisons deux outils afin d'analyser les différences dans les données. Cependant, il est possible que les sorties des deux outils

soient erronées en même temps, même si le résultat des deux outils est le même. Je pense qu'à l'avenir, pour l'analyse des données et l'entrainement des modèles, un nouvel outil pourrait être ajouté pour analyser l'ensemble des données.

Enfin, le corpus de ce projet est fondé sur un genre littéraire du XIIIe siècle qui contient des romans en prose et en vers, ainsi que des chroniques. Est-il possible d'inclure d'autres nouveaux genres textuels comme les lettres ? Je pense qu'il serait intéressant de pouvoir comparer et analyser des phrases provenant d'un corpus plus large dans le cadre de travaux futurs.

3. Bilan personnel

Le travail effectué au cours de mon stage m'a montré que dans un projet collaboratif, c'est l'organisation rationnelle des tâches et la coopération mutuelle entre les équipes qui assurent une progression constante. Une communication insuffisante peut entraîner des problèmes majeurs et un ralentissement du travail.

Au cours de ce stage, j'ai pu appliquer pleinement la théorie et la pratique de mon cours de maîtrise, y compris la théorie et les connaissances en matière de programmation de TAL, pour résoudre les difficultés rencontrées pendant le stage. Cependant, mes capacités et mes connaissances étaient trop faibles par rapport à celles d'un chercheur professionnel. Beaucoup de choses qui semblent simples pour les experts sont difficiles et incompréhensibles pour moi. Il faut dire que l'apprentissage est sans fin, tant à l'université que sur le lieu de travail, et qu'il faut continuer à apprendre pour améliorer ses compétences et ses qualifications personnelles.

Grâce à ce stage, j'ai appris l'utilisation de nouvelles technologies, comme la façon de travailler à distance sur des systèmes Linux, l'utilisation d'analyseurs syntaxiques, etc. Ce projet de collaboration m'a également permis de comprendre que les technologies TAL peuvent réellement être appliquées à différents domaines. Il s'agit de technologies à la fois adaptées à la recherche et à des besoins pratiques. Je pense que je vais continuer à me former sur le TAL et en faire mon plan de carrière.

La principale difficulté que j'ai rencontrée au cours de mon stage a été le problème de la gestion du planning dans le travail à distance. De plus, en tant que francophone non native, je me suis parfois retrouvée face à un manque de communication ou face à des malentendus qui ont engendré des complications. Je m'en souviendrai toujours dans mes

prochaines études et mon futur travail afin d'éviter, autant que possible, que les mêmes problèmes se reproduisent.

Bibliographie

Bazin-Tacchella, S. & Souvay, G. (2020). Lemmatisation et construction automatique de ressources lexicographiques : les développements du lemmatiseur LGeRM. *Diachroniques. Revue de Linguistique française diachronique, Presses de l'Université Paris-Sorbonne (PUPS)* ⟨ halshs-02485461⟩

Benson M. (1989). The Structure of the Collocational Dictionary. *International Journal of Lexicography. vol. 2, 1*. pp. 1-14.

Cavalla, C. & Sorba, J. (2018). Étude diachronique du figement : collocations verbo-nominales. *In O. Soutet, I. Sfar & S. Mejri (éd.), Phraséologie et discours* (p.131-142*)*. Paris : Honoré Champion.

Curado, F. A. (2001). Lexical Behaviour in Academic and Technical Corpora: implications for ESP development. *Language Learning & Technology, vol. 5, 3.* pp. 106-129.

Denoyelle, C. & Sorba, J. (2020). L'approche phraséologique du roman médiéval : une voie de caractérisation générique ?. *7e Congrès Mondial de Linguistique Française, 78, SHS Web Conf., pp.05005,* ff10.1051/shsconf/20207805005ff. ffhal-02931259f

Devlin, J., Chang, M.W., Lee, K. & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 4171–4186.

E, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., Lecouteux, B., Allauzen, A., Crabbé, B., Besacier, L. & Schwab, D. (2020). FlauBERT: Unsupervised language model pre-training for French. *In Proceedings of the 12th Language Resources and Evaluation Conference*, p. 2479–2490, Marseille, France: European Language Resources Association.

Gledhill, Ch. J. (2000). Collocations in Science Writing. *Language in performance: 22. Tuebingen: Gunter Narr Verlag.* pp. 270.

Guillot, C., Heiden, S. & Lavrentiev, A. (2017). « Base de français médiéval : une base de référence de sources médiévales ouverte et libre au service de la communauté scientifique », *Diachroniques : Revue de Linguistique française diachronique–7*, p. 168-184, url : https://halshs.archivesouvertes.fr/halshs/018 09581

Guillot, C., Prévost, S. & Lavrentiev, A. (2013). Manuel de référence du jeu Cattex09. *École normale supérieure de Lyon*, Lyon, 2013a.

González-Rey I. (2002). *La phraséologie du français*. Presses universitaires du Mirail, Toulouse, coll. Interlangues, linguistique et didactique, 268 p.

Grobol, L. & Crabbé, B. (2021). Analyse en dépendances du français avec des plongements contextualisés. *Actes de la 28e Conférence TALN*.https://hal.archives-ouvertes.fr/hal-03223424/file/HOPS_final.pdf

Grobol, L. & Crabbé, B. (2021). Analyse en dépendances du français avec des plongements contextualisés. *28e Conférence sur le Traitement Automatique des Langues Naturelles, Lille (virtuel)*, France. ffhal-03223424f

Holgado, C., Lavrentev, A. & Constant, M. (2021). Évaluation de méthodes et d'outils pour la lemmatisation automatique du français médiéval. *Actes de la 28e Conférence TALN*, Lille, France. <talnarchives.atala.org/TALN/TALN-2021/44.pdf>

Kraif, O. & Diwersy, S. (2012). Le Lexicoscope : un outil pour l'étude de profils combinatoires et l'extraction de constructions lexico-syntaxiques. *Actes de la conférence TALN 2012*, Grenoble, France (p.399-406)

Kilgarriff, A., P. Rychlý, P. Smrž & Tugwell, D. (2004). The Sketch Engine. In: Williams, G. & S. Vessier (eds.). *Proceedings of the Eleventh EURALEX International Congress, EURALEX. Lorient: Université De Bretagne Sud*. 105–116.

Le, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., et al.. FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français. *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 2 : Traitement Automatique des Langues Naturelles*, Jun 2020, Nancy, France. pp.268-278. ffhal-02784776v3f

Leech, G. & Wilson, A. (1996). Eagles: Recommendations for the morphosyntactic annotation of corpora. *Technical report, Expert Advisory Group on Language Engineering Standards*, 1996.

Legallois, D., Charnois, T. & Poibeau, T. (2016). Repérer les clichés dans les romans sentimentaux grâce à la méthode des "motifs". *Lidil*,53, p.95-117

Manjavacas, E., Kádár, A. & Kestemont, M. (2019). Improving Lemmatization of Non.Standard Languages with Joint Learning. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 1493–1503, Minneapolis, Minnesota: Association for Computational Linguistics. DOI: 10.18653/v1/N19-1153.

Martin, L., Muller, B., Ortiz, Suárez, J., Dupont, Y., Romary, L., Villemonte De La Clergerie, É., Seddah, D. & Sagot, B. (2019). CamemBERT : a Tasty French Language Model. *arXiv preprint arXiv*:1911.03894.

Nivre, J., De Marneffe, M.C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D.,

Mcdonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R. & Zeman, D. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. I*n Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16),* p. 1659–1666, Portorož, Slovenia: European Language Resources Association (ELRA)

Pecman, M. (2005), Les apports possibles de la phraséologie à la didactique des langues étrangères. *Apprentissage des Langues et Systèmes d'Information et de Communication*, 2005, 08 (1), pp.109-122. ffedutice-00109615f

Petrov, S., Das, D. & McDonald, R. (2011). A universal part-of-speech tagset. *CoRR,* abs/1104.2086, 2011.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. *In Proceedings of International Conference on New Methods in Language Processing,* p. 44–49, Manchester, UK: Routledge.

Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, vol. 19, 1. pp. 143-177.

Straka, M. & Straková, J. (2017). Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. *In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, p. 88–99, Vancouver, Canada: Association for Computational Linguistics. DOI : 10.18653/v1/K17-3009.

Viprey, J.-M. (2006). Structure non séquentielle des textes. *Langages,163*, p.71-85

Sitographie

Touts les documents et toutes les informations accessibles via les adresses URL regroupées sur cette page ont été consulté pour la dernière fois le 31/08/2021.

https://lidilem.univ-grenoble-alpes.fr/node/16/axes-recherche/phraseologie-et-genres-textuels-cas-roman-medieval

https://lidilem.univ-grenoble-alpes.fr/node/16

https://www.liglab.fr/fr/presentation/presentation-du-laboratoire

https://sourceforge.net/projects/txm/

http://phraseotext.univ-grenoble-alpes.fr/lexicoscope_2.0/

https://fr.wikipedia.org/wiki/Phras%C3%A9ologie_(linguistique)

http://www.phraseonet.com/fr/la-phraseologie

http://stella.atilf.fr/LGeRM/

https://universaldependencies.org/format.html#conll-u-format

https://universaldependencies.org/tools.html#arborator

https://fr.wikipedia.org/wiki/%C3%89tiquetage_morpho-syntaxique

Glossaire

Phraséologiqie : Une combinaison plus ou moins figée de mots qui s'oppose par cette caractéristique à l'association libre de mots

Corpus : Collection de documents, de données, regroupés dans un certain but de recherche

Lemmatisation : Un traitement lexical apporté à un texte en vue de son analyse

Token : Pour désigner une entité (ou unité) lexicale, dans le cadre de l'analyse lexicale

Lemme : Un lemme est une unité autonome constituante du lexique d'une langue

Cooccurrence : Est la présence simultanée de deux ou de plusieurs mots ou autres unités linguistiques dans le même énoncé, par exemple la phrase, le paragraphe, l'extrait

Parseur : Analyseur syntaxique, interpréteur

Tokenisation : Parsage d'un texte en tokens, par exemple en mots et symboles (les mots, phrases et paragraphes sont détectés et regroupés)

Sigles et abréviations utilisés

LIDILEM : Le Laboratoire de Linguistique et Didactique des Langues Étrangères et Maternelles

LIG : Laboratoire d'Informatique de Grenoble

LGeRM : Lemmes Graphies et Règles Morphologiques

TAL : Traitement Automatique de la Langue

CNRS : Centre National de la Recherche Scientifique

ELAN : Équipe Littératures et Arts Numériques

XML : Extensible Markup Language

TEI : Text Encoding Initiative

SRCMF : Syntactic Reference Corpus of Medieval French

FlauBERT : French Language Understanding via Bidirectional Encoder Representations fromTransformers

UD : Universal Dependencies

POS : Part-of-speech tagging

Table des illustrations

Table des tableaux

Table des annexes

Annexe 1
Les corpus des textes littéraires en français médiévale

[1] GRE_ArtusdeBretagne
[2] GRE_Coinci1_A138
[3] GRE_HistAnc
[4] GRE_HistAncReg
[5] GRE_JehanBlonde
[6] GRE_LancprM1
[7] GRE_LancPrM2
[8] GRE_MeraugisPortlesguez
[9] GRE_Merlin_prose
[10] GRE_Merlin_prose_39-63
[11] GRE_Merlin_prose_64-91
[12] GRE_Merlin_vers
[13] GRE_PremiersFaitsArthur
[14] GRE_PremiersFaitsArthur2
[15] GRE_RomanSilence
[16] GRE_TristanPRCurtisT1
[17] GRE_TristanPRCurtisT2
[18] GRE_TristanPRCurtisT3
[19] GRE_Tristan_Menard_tome_1
[20]
[21] SRCMF_Aucassin
[22] SRCMF_Beroul
[23] SRCMF_clari
[24] SRCMF_coinci_nca
[25] SRCMF_gcoin1T
[26] SRCMF_lapidfp
[27] SRCMF_qgraalcm
[28] SRCMF_roland
[29] SRCMF_rosem2
[30] SRCMF_slethgier
[31] SRCMF_yvainku

Annexe 2
Les champs complets de CoNLL-U

This page pertains to UD version 2.

CoNLL-U Format

Quick links: [Word segmentation] [Morphology] [Syntax] [Miscellaneous] [Extensions]

We use a revised version of the CoNLL-X format called CoNLL-U. Annotations are encoded in plain text files (UTF-8, normalized to NFC, using only the LF character as line break, including an LF character at the end of file) with three types of lines:

1. Word lines containing the annotation of a word/token in 10 fields separated by single tab characters; see below.
2. Blank lines marking sentence boundaries.
3. Comment lines starting with hash (#).

Sentences consist of one or more word lines, and word lines contain the following fields:

1. ID: Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes (decimal numbers can be lower than 1 but must be greater than 0).
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOS: Universal part-of-speech tag.
5. XPOS: Language-specific part-of-speech tag; underscore if not available.
6. FEATS: List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.
7. HEAD: Head of the current word, which is either a value of ID or zero (0).
8. DEPREL: Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.
9. DEPS: Enhanced dependency graph in the form of a list of head-deprel pairs.
10. MISC: Any other annotation.

Annexe 3
Table de conversion

Cattex2009	upos	ud-feats
ABR	X	Abbr=Yes
ADJcar	ADJ	NumType=Card
ADJind	ADJ	PronType=Ind
ADJord	ADJ	NumType=Ord
ADJpos	ADJ	Poss=Yes
ADJqua	ADJ	_
ADVgen	ADV	_
ADVgen.PROadv	ADV.ADV	_._
ADVgen.PROper	ADV.PRON	_.PronType=Prs
ADVing	ADV	PronType=Int
ADVint	ADV	PronType=Int
ADVneg	ADV	PronType=Neg
ADVneg.PROadv	ADV.ADV	PronType=Neg._
ADVneg.PROper	ADV.PRON	PronType=Neg.PronType=Prs
ADVsub	ADV	_
CONcoo	CCONJ	_
CONsub	SCONJ	_
CONsub.PROper	SCONJ.PRON	_.PronType=Prs
DETcar	DET	NumType=Card
DETcom	DET	Definite=Com
DETdef	DET	PronType=Art|Definite=Def
DETdem	DET	PronType=Dem
DETind	DET	PronType=Ind
DETint	DET	PronType=Int
DETndf	DET	PronType=Art|Definite=Ind
DETord	DET	NumType=Ord
DETpos	DET	Poss=Yes
DETrel	DET	PronType=Rel
ETR	X	Foreign=Yes
INJ	INTJ	_
NOMcom	NOUN	_
NOMpro	PROPN	_
OUT	X	_
PONfbl	PUNCT	_
PONfrt	PUNCT	_
PONpdr	PUNCT	_
PONpga	PUNCT	_
PONpxx	PUNCT	_
PRE	ADP	_

PRE.DETcom	ADP.DET	_.Definite=Yes
PRE.DETdef	ADP.DET	_.PronType=Art|Definite=Yes
PRE.DETrel	ADP.DET	_.PronType=Rel
PRE.PROper	ADP.PRON	_.PronType=Prs
PRE.PROrel	ADP.PRON	_.PronType=Rel
PROadv	ADV	_
PROcar	PRON	NumType=Card
PROcom	PRON	Definite=Com
PROdem	PRON	PronType=Dem
PROimp	PRON	PronType=Prs
PROind	PRON	PronType=Ind
PROint	PRON	PronType=Int
PROint.PROper	PRON.PRON	PronType=Int.PronType=Prs
PROord	PRON	NumType=Ord
PROper	PRON	PronType=Prs
PROper.PROadv	PRON.ADV	PronType=Prs._
PROper.PROper	PRON.PRON	PronType=Prs.PronType=Prs
PROpos	PRON	Poss=Yes
PROrel	PRON	PronType=Rel
PROrel.PROadv	PRON.ADV	PronType=Rel._
PROrel.PROper	PRON.PRON	PronType=Rel.PronType=Prs
RED	X	_
RES	X	_
VERcjg	VERB	VerbForm=Fin
VERinf	VERB	VerbForm=Inf
VERppa	VERB	VerbForm=Part|Tense=Pres
VERppe	VERB	VerbForm=Part|Tense=Past

Annexe 4
Jeu d'étiquettes CATTEX 2009

DETndf	DÉTERMINANT non défini
DETpos	DÉTERMINANT possessif
DETrel	DÉTERMINANT relatif
ETR	MOT ÉTRANGER
INJ	INTERJECTION
NOMcom	NOM commun
NOMpro	NOM propre
PONfbl	PONCTUATION faible
PONfrt	PONCTUATION forte
PONpdr	PONCTUATION parenthèse droite
PONpga	PONCTUATION parenthèse gauche
PONpxx	PONCTUATION parenthèse (gauche ou droite)
PRE	PRÉPOSITION
PRE.DETdef	contraction PRÉPOSITION + DÉTERMINANT défini
PRE.DETrel	contraction PRÉPOSITION + DÉTERMINANT relatif
PRE.PROper	contraction PRÉPOSITION + PRONOM personnel
PROadv	PRONOM adverbial
PROcar	PRONOM cardinal
PROdem	PRONOM démonstratif
PROimp	PRONOM impersonnel
PROind	PRONOM indéfini
PROint	PRONOM interrogatif
PROord	PRONOM ordinal
PROper	PRONOM personnel
PROper.PROper	contraction PRONOM personnel + PRONOM personnel
PROpos	PRONOM possessif
PROrel	PRONOM relatif
RED	REDONDANT
RES	RÉSIDU
VERcjg	VERBE conjugué
VERinf	VERBE infinitif
VERppa	VERBE participe présent
VERppe	VERBE participe passé

ABR	ABRÉVIATION
ADJcar	ADJECTIF cardinal
ADJind	ADJECTIF indéfini
ADJord	ADJECTIF ordinal
ADJpos	ADJECTIF possessif
ADJqua	ADJECTIF qualificatif
ADVgen	ADVERBE (général)
ADVgen.PROper	contraction ADVERBE + PRONOM personnel
ADVint	ADVERBE interrogatif
ADVneg	ADVERBE de négation
ADVneg.PROper	contraction ADVERBE de négation + PRONOM personnel
ADVsub	ADVERBE subordonnant
CONcoo	CONJONCTION de coordination
CONsub	CONJONCTION de subordination
DETcar	DÉTERMINANT cardinal
DETcom	DÉTERMINANT défini composé
DETdef	DÉTERMINANT défini
DETdem	DÉTERMINANT démonstratif
DETind	DÉTERMINANT indéfini
DETint	DÉTERMINANT interrogatif

Table des matières

MOTS-CLÉS : TAL, analyse syntaxique, unités phraséologiques, étiquetage morphosyntaxique

RÉSUMÉ

Pour explorer le rôle de la phraséologie étendue dans la structuration des genres textuels littéraires en français médiéval, les techniques de TAL et la linguistique de corpus prennent une part décisive, car les corpus informatisés et les programmes qui les utilisent permettent d'identifier les unités phraséologiques. Dans cet article, nous présentons un travail de traitement du corpus à l'aide de LGeRM, une plateforme de lemmatisation du français médiéval et de l'analyseur syntaxique Hops, incluant le traitement de lemmatiser, étiqueter et analyser en dépendances syntaxiques. Cet article détaille également comment le corpus traité peut être intégré dans le Lexicoscope, l'outil de fouille textuelle. L'objectif est de pouvoir constituer le corpus sur le Lexicoscope et d'utiliser les fonctionnalités de cet outil pour mener des recherches sur l'exploration des liens entre la phraséologie étendue et les genres textuels en français médiéval.

KEYWORDS: NLP, parsing, phraseological units, part-of-speech tagging

ABSTRACT

For explore the role of extended phraseology in the structuring of literary textual genres in medieval French, NLP techniques and corpus linguistics play a decisive role, as computerized corpus and the programs allow the identification of phraseological units. In this article, we present a corpus processing work using LGeRM, a lemmatization platform for medieval French, and the syntactic parser Hops, including lemmatization, tagging and syntactic dependency parsing. This article also details how the processed corpus can be integrated into the Lexicoscope, tool of text mining. The aim is to be able to build the corpus on the Lexicoscope and to use the functionalities of this tool to conduct research on the exploration of the links between extended phraseology and textual genres in medieval French.