Lemmatisation et annotation des formes verbales du français médiéval

Clara Bringer

Projet encadré par Delphine Bernhard et Julie Glikman

Université de Strasbourg
Master 2 de Technologies des Langues

14 juillet 2019

Table des matières

Table des matières **1**

1 Introduction **5**

2 Description de la langue et difficultés existantes **7**

2.1 Études réalisées sur le français médiéval 7

2.2 Évolution de la langue 8

2.3 Tableaux de conjugaison 8

2.3.1 *Cent verbes conjugués en français médiéval* (Bragantini-Maillard et Denoyelle 2012) 9

2.3.2 *Morphologie historique des verbes français* (Lanly 1995) 9

2.3.3 TCAF – Tableau de Conjugaison de l'Ancien Français (Okada et Ogurisu 2007) 10

2.4 Résolution des problèmes 10

3 Outils existants **12**

3.1 LGeRM – Lemmes, Graphies et Règles Morphologiques 12

3.1.1 Présentation de l'outil 12

3.1.2 Les outils à disposition pour réaliser ce lemmatiseur 16

3.1.3 Fonctionnement 16

Principe 16

Les ressources utilisées 17

L'algorithme 19

3.1.4 Résultats 19

3.2 Flemm 20

3.2.1 Présentation et fonctionnement de l'outil 20

Module de transcodage 20

Module d'analyse 21
3.2.2 Résultats 22
Étalonnage 22
Expérience de segmentation et d'étiquetage 23
Lemmatisation des mots inconnus 24
Vérification manuelle 24
3.2.3 Utilité pour le français médiéval 25
3.3 TCAF – Tableau de Conjugaison de l'Ancien Français 25
3.4 Conclusion 25

4 Présentation des corpus **27**
4.1 Question du texte unique 27
4.2 Présentation du texte 27
4.3 Présentation du corpus TXM 28
4.3.1 Informations générales 28
Choix de l'édition 28
Enrichissements apportés 28
4.3.2 Données utilisées 31
Contenu de l'édition 31
Version utilisée 32
L'interface électronique 32
Les limites de cet outil 34
4.3.3 Obtention des annotations du corpus TXM 34
4.3.4 Enregistrement du corpus annoté 36
4.4 Corpus étiqueté par LGeRM 37
4.4.1 Présentation 37
4.4.2 Obtention des annotations du corpus étiqueté par LGeRM 37
4.5 Comparaison des deux corpus 38
4.5.1 Cas des formes présentes dans le texte du TXM, mais pas dans celui de LGeRM 39
4.5.2 Choix du corpus à utiliser 42

5 Constitution de la base de données des verbes du TCAF **44**
5.1 Choix du jeu d'étiquettes d'annotation 45
5.1.1 Présentation du jeu d'étiquettes Cattex09max 45

Adaptation du jeu d'étiquettes à nos données 45
5.1.2 Morphalou 46
Présentation et description du contenu 46
Intérêt pour notre travail 48
5.1.3 Choix du jeu d'étiquettes à utiliser 50
5.2 Enregistrement de la base du TCAF 50
5.2.1 Description des ressources et problèmes rencontrés 50
5.2.2 Extraction de la liste des verbes répertoriés 54
5.2.3 Extraction de la base du TCAF 54
Les différentes lexies du verbe 55
Les verbes similaires 57
Les verbes dérivés 57
5.2.4 Enregistrement de la base de données 59

6 Méthode utilisée pour réaliser l'outil d'annotation et de lemmatisation 60
6.1 Création du script d'étiquetage automatique et évaluation des résultats . . . 60
6.1.1 Première approche 60
6.1.2 Résultats 62
6.2 Amélioration des résultats 62
6.2.1 Deuxième approche 62
Amélioration sur les verbes à l'infinitif 63
Amélioration sur les verbes au participe passé et présent 63
Amélioration sur l'ordre des suggestions 64
6.2.2 Troisième approche 64
6.3 Enregistrement des résultats 65
6.3.1 Choix du format de sauvegarde des données 65
6.4 Résultats de cette méthode 68
6.4.1 Calcul de la précision 68
6.4.2 Erreurs et verbes non annotés 69

7 Conclusion 70

A Scripts développés et utilisés 72

Table des figures 79

Liste des tableaux **81**

Bibliographie **82**

Chapitre 1

Introduction

Le français médiéval est une langue qui fait l'objet de beaucoup d'études. Il existe notamment plusieurs outils numériques qui permettent de le traiter, de l'analyser, ou d'effectuer des recherches sur des textes ou des dictionnaires en ancien français.

Pour permettre cela, les textes en ancien français sont annotés et lemmatisés. D'après l'ATILF (ATILF - CNRS & Université de Lorraine), la lemmatisation consiste à « regrouper des formes sous les lemmes correspondants », un lemme étant une « forme graphique choisie conventionnellement comme adresse dans un lexique » (ATILF - CNRS & Université de Lorraine). L'annotation est l'action d'« accompagner un texte de remarques diverses généralement manuscrites » (ATILF - CNRS & Université de Lorraine). Ici, nous parlons d'annoter automatiquement les termes du texte selon leur catégorie grammaticale, ainsi qu'éventuellement diverses informations supplémentaires selon cette catégorie, comme le genre ou encore le nombre.

Dans le cas de formes verbales, la lemmatisation consiste à trouver la forme infinitive du verbe, et l'annotation consiste à déterminer le mode, le temps, la personne et le nombre du verbe.

Si le traitement (c'est-à-dire l'annotation et la lemmatisation comme précédemment définies) des formes non verbales est plutôt bien assuré, ce n'est pas le cas pour les formes verbales. En effet, on ne dispose pas de suffisamment d'éléments ou de connaissances de cette langue : « la connaissance [de cette langue] reste fragmentaire, fondée qu'elle est sur des sources uniquement écrites et pour l'essentiel littéraires » (Bragantini-Maillard et Denoyelle 2012).

Comme nous le verrons plus en détail au cours de ce travail, le français médiéval est une langue dont l'orthographe et la conjugaison ne sont pas stables. Ainsi, déterminer le lemme

d'une forme verbale et sa personne, son temps et son mode n'est pas chose facile, en raison du nombre de graphies possibles pour une même unité lexicale, et du nombre d'unités lexicales possibles pour une même graphie. De plus, la résolution de ces ambiguïtés ne peut s'appuyer sur une concordance des temps, qui n'existe pas encore à cet état de langue, ni sur l'analyse du sujet, souvent absent car il n'est pas nécessaire, comme c'est le cas pour le latin.

À ce jour, malgré les nombreux travaux qui ont été faits sur le français médiéval, il n'existe aucun outil permettant de lemmatiser les formes verbales d'un texte médiéval.

L'objectif de ce travail sera donc de pallier cette lacune, en proposant un outil permettant d'étiqueter les formes verbales du français médiéval, qui pourra indiquer le lemme, le temps et la personne de chaque verbe. Ce travail s'appuiera en particulier sur l'étude du texte de la *Queste del saint Graal*. Nous travaillerons donc sur le français médiéval du XIII$^{\text{ème}}$ siècle, siècle auquel ce roman a été écrit.

Pour cela, nous étudierons les variations graphiques du français médiéval, afin de trouver des solutions pour annoter et lemmatiser ces formes verbales.

Ensuite, nous étudierons LGeRM (ATILF - CNRS & Université de Lorraine 2019), Flemm (Namer 2005) et le TCAF (Okada et Ogurisu 2007), trois outils numériques qui existent déjà pour l'annotation de corpus ou la recherche dans un dictionnaire. Nous verrons ainsi quelles solutions ont été apportées pour lemmatiser et annoter les formes non verbales, et comment sont gérées les formes verbales.

Ensuite, nous présenterons le corpus étudié, ainsi que deux versions étiquetées par des outils différents. Nous devrons alors déterminer laquelle de ces versions utiliser pour la suite, lors de l'annotation et de la lemmatisation des formes verbales.

Puis, grâce aux recherches effectuées précédemment, nous déterminerons et décrirons dans le détail une méthode permettant d'annoter et de lemmatiser les formes verbales de l'édition du texte choisie.

Enfin, nous évaluerons la qualité du travail effectué, et parlerons des améliorations éventuelles pouvant être apportées.

Chapitre 2

Description de la langue et difficultés existantes

2.1 Études réalisées sur le français médiéval

Le français médiéval est une langue dont nous avons une connaissance incomplète, qui ne s'appuie que sur des sources écrites et souvent littéraires. De plus, la conjugaison de l'ancien français est un concept assez moderne et récent, puisque les textes étudiés écrits en français médiéval ne présentent pas de norme en ce qui concerne la conjugaison :

> « La conjugaison d'ancien français est assurément une fabrication moderne, le français médiéval transmis par les sources ne présentant pas l'unification normative du système moderne. Il n'en reste pas moins vrai que dès les plus anciens textes français se dégagent des régularités, caractéristiques d'un système verbal cohérent, en dépit de variations régionales ou autres. » (Bragantini-Maillard et Denoyelle 2012).

Des travaux ont été entrepris sur l'étude de cette langue, en particulier des études sur la conjugaison, qui ont menés à l'écriture de ressources telles que *Cent verbes conjugués en français médiéval*, (Bragantini-Maillard et Denoyelle 2012), *Morphologie historique des verbes français* (Lanly 1995), ou encore le site Tableau de Conjugaison de l'Ancien Français (Okada et Ogurisu 2007), que nous étudierons par la suite.

Les travaux qui ont été entrepris pour l'étude de cette langue avaient deux perspectives : diachronique, c'est-à-dire l'étude de l'évolution des faits dans le temps, et synchronique, c'est-à-dire l'étude des phénomènes linguistiques sur une même période. Ces travaux ont notamment servi à retracer les évolutions phonétiques et analogiques, ainsi que les phénomènes

d'uniformisation ou de divergence régionale qui se sont produits lors du passage du latin au français.

2.2 Évolution de la langue

Le français médiéval est une langue en pleine évolution. En effet, cette langue conserve beaucoup d'archaïsmes, comme décrit ci-dessous, mais en même temps des constructions plus modernes commencent à apparaître.

> « Le moyen français présente déjà des aspects modernes mais conserve souvent des archaïsmes (*amy* ou *ami* ; *congnaistre* ou *conaitre*), l'orthographe des mots n'est pas encore stabilisée (*conaistre*, *conaitre* ou *connaitre*) et le système flexionnel est en pleine évolution (pluriel en s, x ou z : *ciels*, *cielx*, *cielz*). C'est aussi une période où de nouveaux mots apparaissent, par exemple dans les traductions en langue française des classiques grecs et latins. De plus il n'y a pas encore de français standard, et les textes sont parfois très marqués dialectalement par l'atelier de saisie du manuscrit : textes picards dans lesquels on trouve *chiel* pour *ciel*, ou textes anglo-normands où l'on rencontre *bastoun* pour *baston*). Un autre phénomène à prendre en compte relève des pratiques de transcription des manuscrits qui ont évolué au cours du temps. » (Souvay et Pierrel 2009)

2.3 Tableaux de conjugaison

Ces études ont permis de confirmer une cohérence de la morphologie verbale, malgré les flottements, l'évolution rapide des formes, ou la forte variabilité d'écriture des formes communes, surtout si l'on prend en compte la variabilité régionale, présente dans tous les textes. Cette cohérence a permis d'établir des modèles de conjugaison, des reconstitutions de la conjugaison.

En effet, des tableaux de conjugaison de verbes en français médiéval ont pu être construits, comme pour le TCAF, mais également pour l'ouvrage *Cent verbes conjugués en français médiéval* (Bragantini-Maillard et Denoyelle 2012), ainsi que pour *Morphologie historique des verbes français* (Lanly 1995).

2.3.1 *Cent verbes conjugués en français médiéval* (Bragantini-Maillard et Denoyelle 2012)

Cet ouvrage se compose de 116 tableaux de conjugaison de verbes en français médiéval. Ce sont des verbes parmi les plus usuels (réguliers et irréguliers) qui ont été choisis, et un index est disponible, répertoriant les verbes courants qui n'ont pas été retenus pour les tableaux et renvoyant le lecteur au modèle de conjugaison adéquat.

Chaque fiche de verbe tient sur deux pages. Sur la première page se trouve le tableau de conjugaison en ancien français commun du verbe comme il a été reconstitué selon l'état de langue de la fin du XII^{ème} siècle. Sur la deuxième page se trouve « le recensement des formes secondes, médio-françaises et régionales » (Bragantini-Maillard et Denoyelle 2012).

Ce recensement n'est cependant pas exhaustif, il se limite aux phénomènes récurrents et notables.

> « Ainsi ne sont pas relevées les variantes qui ressortissent à la variation graphique (ex. graphies *o* et *ou*, *al* et *au*...) ou à l'évolution phonétique (ex. [ii] > [i] dans des désinences telles que *-iiens* ; la désinence de deuxième personne du pluriel notée *-és* après réduction de l'affriquée). Quant aux variations désinentielles dues à l'évolution phonétique, à l'analogie ou au traitement régional (ex. au Nord-Est, *-ièmes* pour *-ions* à la première personne du singulier du subjonctif imparfait), elles sont rassemblées pour l'essentiel en introduction, dans les encadrés intitulés *Régionalismes*. » (Bragantini-Maillard et Denoyelle 2012)

2.3.2 *Morphologie historique des verbes français* (Lanly 1995)

Cet ouvrage a pour but de retracer « l'histoire de la conjugaison des principaux verbes français, de présenter du moins l'origine et l'évolution de leurs principales formes » (Lanly 1995).

Chaque fiche de verbe comporte trois colonnes. La première contient « les formes originelles latines ou latinisées, classiques et vulgaires » (Lanly 1995). La deuxième contient « l'état ou les états successifs en ancien français et en moyen français (jusqu'à la fin du XVI^{ème} siècle en général) » (Lanly 1995). Enfin, la troisième contient les formes modernes.

2.3.3 TCAF – Tableau de Conjugaison de l'Ancien Français (Okada et Ogurisu 2007)

Le TCAF est un outil électronique permettant de rechercher une forme verbale, et fournissant le ou les lemmes correspondants, ainsi que le tableau de conjugaison de chacune de ces formes verbales.

La particularité de cet outil est qu'il propose différentes graphies pour chaque forme. Par exemple, lorsque l'on cherche la forme *cuit*, le TCAF fournit les lemmes *coillir*, *cuire* et *cuidier*, ainsi que leur tableau de conjugaison respectif.

Cependant, la liste des graphies possibles pour chaque forme ne peut être exhaustive, comme nous l'avons vu plus tôt.

2.4 Résolution des problèmes

Comme nous l'avons vu, beaucoup de formes existent pour un même mot dans le dictionnaire ; leur liste n'est pas exhaustive. Il est donc impossible d'établir une liste complète des formes verbales. Une bonne solution à cette lacune serait de construire une liste des formes verbales à partir de tableaux de conjugaison existants, puis de compléter cette liste avec un ensemble de règles permettant de retrouver une forme connue à partir de la forme à analyser.

Un autre problème réside dans la résolution des ambiguïtés. En effet, il existe beaucoup d'ambiguïtés au sein des formes verbales : des formes qui sont utilisées pour plusieurs verbes différents, à des personnes ou des temps différents. Par exemple, la forme *cuit* peut correspondre :

— au verbe *cuidier* (qui signifie « penser » en français moderne) à la première personne de l'indicatif présent ;
— au verbe *cuidier* à la première personne du subjonctif présent ;
— au verbe *cuidier* à la troisième personne du subjonctif présent ;
— au verbe *cuire* à la troisième personne du subjonctif présent ;
— au verbe *cuire* au participe passé ;
— au verbe *coillir* (qui signifie « cueillir ») à la troisième personne de l'indicatif présent.

Un moyen de résoudre ces ambiguïtés pourrait être la concordance des temps entre les autres verbes de la phrase ou du paragraphe ayant également été étiquetés, ou encore d'utiliser le sujet du verbe afin de déterminer la personne.

Or, la concordance des temps n'existe pas encore en français médiéval, et l'usage du sujet n'est pas courant, car pas nécessaire, comme en latin. Il est donc très difficile, à partir d'une

forme verbale et de ses lemmes possibles, de désambiguïser, et donc de retrouver le bon lemme et la bonne forme par des règles. Ces règles sont presque impossibles à définir.

La meilleure solution pour résoudre ces ambiguïtés sera alors de conserver les lemmes possibles, et de tous les proposer. L'utilisateur pourra ainsi choisir de manière logique celui qui correspond le mieux.

Chapitre 3

Outils existants

Nous étudierons ici trois outils : LGeRM (Lemmes, Graphies et Règles Morphologiques), Flemm et le TCAF (Tableau de Conjugaison de l'Ancien Français). En effet, LGeRM et le TCAF sont deux outils très utilisés pour étudier le français médiéval, et Flemm est très efficace pour analyser le français moderne.

3.1 LGeRM – Lemmes, Graphies et Règles Morphologiques

3.1.1 Présentation de l'outil

LGeRM est un lemmatiseur hors contexte, développé dans le cadre du projet du Dictionnaire du Moyen Français (DMF) (Martin 2015).

Un lemmatiseur est un outil électronique permettant de regrouper des formes sous les lemmes correspondants, c'est-à-dire toutes les formes possibles pour un même mot sous une même forme.

Le DMF est un dictionnaire électronique en ligne, composé à partir d'un corpus de 219 textes en ancien français, datés entre 1330 et 1500.

Ce lemmatiseur a été créé pour faciliter la consultation du DMF. En effet, il est ainsi possible de rechercher n'importe quelle forme d'un mot, qui sera alors lemmatisée. Le ou les lemmes trouvés seront ensuite cherchés dans le dictionnaire : « Il permet ainsi de faciliter la consultation d'un dictionnaire, l'interrogation et la lemmatisation de textes médiévaux et trouve des applications dans l'édition électronique de manuscrits et la construction automatique de glossaires. » (Souvay et Pierrel 2009)

Il s'adresse à des spécialistes de la langue médiévale et à des novices.

Il utilise une base de formes connues lemmatisées et un ensemble de règles.

Ainsi, l'idée de ce projet est de pouvoir consulter un dictionnaire sans connaître la forme exacte du mot, puisqu'il permet d'effectuer une recherche sur une forme quelconque d'un mot. Ce dictionnaire fournira alors la ou les entrées correspondantes à la forme fournie. Au contraire, les autres dictionnaires du français médiéval nécessitent une forme précise du mot pour trouver son entrée.

Cette idée est cependant difficile à réaliser pour le français médiéval, en raison de ses nombreuses spécificités. En effet, nous avons vu plus tôt que cette langue est en pleine évolution, et notamment que son orthographe n'est pas stabilisée.

De plus, plus le mot compte de syllabes, plus sa variation est importante : « le traitement informatique du moyen français devra tenir compte de ces deux faits : un grand nombre de possibilités pour une même forme morphosyntaxique et une liste complète des formes impossible à établir. » (Souvay et Pierrel 2009)

Il est facile de trouver des exemples de mots possédant différentes formes. Nous avons donc cherché les mots *connaissance*, *information*, et *cours* dans le DMF.

Les figures 3.1, 3.2 et 3.3 montrent quelques-uns de ces résultats. La définition du terme recherché est surlignée en jaune, et chaque forme de chaque mot est écrite en bleu.

B. - "Fait d'être mis au courant, de se mettre au courant ; renseignement ainsi porté à la connaissance de qqn" : Quant Agrevain l'oÿ, s'abaisse le menton ; Couvoittize le fist avoir opinion D'entendre la parolle dou traïttre felon, Et puis y tent l'oreille, lui vint temptation De mauvaise volente par *infourmation*. (*Flor. Rome* W., c.1330-1400, 188). ...la [aux noces] ouvri la vierge sa bouche (...) a parole de seure *informacion* quant elle dist aux menistres : Faites tout ce qu'il vous dira. (*Mir. Theod.*, 1357, 80). Je me voel ensonniier de l'ordonner et meltre en prose selonch le vraie *information* que j'ay eu des vaillants hommes, chevaliers et escuiers, qui les ont aidiés à acroistre. (FROISS., *Chron.* L., I, c.1375-1400, 1). Sus laquelle *informacion* du chevalier je me suis fondé et arrestez et ay escript ce qui s'ensieut. (FROISS., *Chron.* M., XIV, c.1375-1400, 13). ...sire Symon Susbery (...) se fonda et fourma en sa parolle bien et saigement et vivement du tout, sus le conseil et *information* que le duc de Glocestre leur avoit dit et donné. (FROISS., *Chron.* M., XIV, c.1375-1400, 29). ...ilz furent tous ensamble et conseilliet l'un parmy l'autre, parmy l'*information* que Geronnet leur avoit dit et fait et monstré à quelle heure ilz venroient (FROISS., *Chron.* M., XIV, c.1375-1400, 207). ...les gerres de France et d'Engleterre (...) je Jehans Froissars (...) me voel ensonniier de metre en prose et ordonner selonch la vraie *information* que je ay eu des vaillans honmes, chevaliers et esquiers qui les dittes armes ont aidiet a acroistre (FROISS., *Chron.* D., p.1400, 35). ...voeullés demander le fait et l'oquoison, Et puis en jugerons par *informatïon*. (*Huon Bordeaux* B., c.1400-1450, 12). ...on voet tesmoignier qu'il fist colation Par deviers le Saint-Père (...) Pour sentence jeter en France le roion Sus le roy et les siens sans *infourmation* (*Geste ducs Bourg.* K., c.1410-1419, 263). De Bourbon y fu li dus, et osi fu Berris Et maint autre princhе (...) S'en avoient enjoinct le prouvos de Paris Pour *information* raporter par escris Sur ce fait-ci en droit, qui ne fu mie petis. (*Geste ducs Bourg.* K., c.1410-1419, 307). Mon tresredoubté seigneur, encores pour tousdiz obeïr a vos prieres,

FIGURE 3.1 – Extrait de l'article *Information* du DMF

a) "Faculté de connaître, de discerner les choses" : L'AMANT. Je vous en pri ; car je ne trueve Riens en moy dont loër me doie, Fors dou bien et de la grant joie Qui me vient de vostre presence. ESPERENCE. C'est par deffaut de *congnoissance* ; Car se tu fusses bien apris, En ton cuer eüsses compris Qu'a l'issir dou ventre ta mere Elle ne te fu pas amere, Einsois te fu moult amiable... (MACH., *R. Fort.*, c.1341, 96). En celle foriest fu .XVI. ans en un tenant, Tant qu'il plot a Jhesu-Cris, (...) Que ly chine se furent du vivier departant Qui *congnissance* avoient et sentement d'enfant (*Chev. cygne* P., c.1356, 44). Quant il ot l'aage de IX. ans, Que de norrice fu exens, Et laissa l'estat d'innocence, Et prist à avoir *congnoissance*, Toutes ses inclinations Et ses ymaginations, Tuit si penser, tuit si desir Furent en faire le plaisir De dames et de damoiselles. (MACH., *P. Alex.*, p.1369, 9). Quant ma dame ne m'a recongneü, Je doy moult bien scens perdre et *congnoissance*. (MACH., *L. dames*, 1377, 141). Se quanque Diex en monde a fait Et quanque Nature a pourtrait Povoient avoir, par souhait, Sens, entendement, *congnoissance*, Plus certein que science n'ait, N'aroient il jamais retrait Dou bon et bel qu'aim sans retrait La bonté ne la grant vaillance. (MACH., *Les lays*, 1377, 362). ...tantost Qu'ilz (...) urent La *cognoiscence* de raison, Enquerre vouldrent l'achoison Des choses faites et requises (CHR. PIZ., *M.F.*, II, 1400-1403, 105). Nous scavons que une chose de petite *congnoissance* ne puet attaindre a ce que puet congnoistre la vertus, qui est de plus haulte *congnoissance*, comme l'oyeil corporel ne pourroit veoir ce que congnoist l'imaginacion par dedans, et l'imaginacion ne pourroit comprendre ce que je, Raison, congnois. (GERS., *Trin.*, 1402, 158). Quant de la savance chascun sceit que en ce royaume sont gens de hault sens et de clere *cognoissance*. (CHART., *Q. inv.*, 1422, 46). De la savance et *cognoissance* qui doit acompaigner la magesté des princes et des seigneurs et leurs chevances convertir en maintes oeuvres me puis je bien taire et en laissier les parolles a ceulx a qui sont les faiz et les dangiers, mais je m'arreste a l'obeissance et discipline de chevalerie dont nostre estat est a present reprouchiez et griefment reprins. (CHART., *Q. inv.*, 1422, 62). O tu, Entendement, fait au patron de la Trinité

b) "Fait de connaître qqc., fait d'être informé de qqc., de savoir qqc." : ...tant qu'il le maine En si hautaine *congnoissance* Que plus sert et plus a plaisance En servir la vierge Marie (*Mir. ev. N.D.*, c.1348, 62). .Que fera ta moullier qui tant est doulce et france, Qui s'en ira pour toi en estrange tenance ? (...) Ayés cy *connoissance* ! (*Renaut Mont. B.N.* V., c.1350-1400, 427). Lors regarda la belle et se douche samblance ; En lui véoir a pris d'amours le *connissance*, Et dist : "Il y aroit deduit par habondance." (*Hugues Capet* L., c.1358, 18). ...par quoy il appert que la consideracion et la *cognoissance* de telle fin appartient a ceste science civile. (ORESME, *E.A.*, c.1370, 105). Car a les entendre, il y convient *cognoissance* et estude ; et a ce est requis lonctemps. (ORESME, *E.A.*, c.1370, 373). ...ce roy Charle, si comme on list et treuve es croniques anciennes - car vous savez que toutte la *cognoissance* de ce monde retournent par l'escripture (...) fut en Espaingne par pluseurs foix (FROISS., *Chron.* M., XIV, c.1375-1400, 9). ...en l'eglise de Saint-Victor, il, pour la grant *cognoissance* qu'il avoit en icelle eglise et aus religieus dudit hostel, pour ce qu'il avoit plusieurs fois ouvré en icelle eglise, monta ou dortoir de ladite eglise, et illec, en la chambre et aus piez du lit dudit prieur, print un breviaire (*Reg. crim. Chât.*, I, 1389-1392, 218). Car par les escriptures puet on avoir la *congnissance* de toutes coses (FROISS., *Chron.* D., p.1400, 37). ...d'elle nous vient L'appercevence et *cognoiscence*, Qu'avons a la divine essence. (CHR. PIZ., *M.F.*, II, 1400-1403, 108). Et jasoit que encores n'eussent ilz senty ne gouté par vraye *congnoissance* la tresdoulce, tressainte et tresamoureuse grace de nostre vray Dieu le Saint Esperit, sy nous ont ilz tous adreschiez par leurs exemples et escriptures aux vrayes gloires de noz ames (LA SALE, *Salade*, c.1442-1444, 3). ...onques ne fuz si espoventée (...) que j'ay esté presentement de voz parolles, quand elles m'ont donné la *cognoissance* de ce qu'oncques je n'oiz (*C.N.N.*, c.1456-1467, 565). Ceste est une science surmontant les aultres sciences, eslevee par dessus toute speculation et consideration naturele, qui se dist phisique *congnoissance* ou science naturelle, et est ceste science theologique et divine preferee a toutes sciences par dignité et utilité (*Somme abr.*, c.1477-1481, 98).

FIGURE 3.2 – Extrait de l'article *Connaissance* du DMF

a) Loc. verb.

- *Aller le cours/prendre le cours/se mettre au cours.* "Courir, s'élancer" : Or se sëoit, or se levoit, Or le *cours*, or le pas alloit. (*Renart contref.* R.L., t.1, 1328-1342, 215). Lors se mirrent au *corps* pour leur vie sauver. (*Renaut Mont. B.N.* V., c.1350-1400, 293). Le destrier meine joye et hennist haut et cler ; Bayart ce mest au *cours* tant qu'il puest randonner (*Renaut Mont. B.N.* V., c.1350-1400, 449). Ly pors fu fiers et orguilleux, et devoura pluseurs levriers et alans, et prinst son *cours* par la forest qui estoit haulte et droicte, et commenca grant la huee. (ARRAS, c.1392-1393, 18). Mais le porc tourne et se met au *cours* par telle maniere qu'il n'y ot chien ne chevalier ne homme qui n'en perdist la trace et veue (ARRAS, c.1392-1393, 19). Lors sy se sont appareillez si com pour jouster, et quant ilz orent pris leurs *cours*, si s'entreviennet l'un encontre l'autre de grant randonnee... (*Chev. papegau* H., c.1400-1500, 58). Sy s'eslongna un petit de l'huis, puis se mist au *cours* de toute sa force et y vint hurter de tel randon qu'il rompy les gons et les verroulz (*Percef. IV*, R., c.1450 [c.1340], 754).
- *Prendre à cours/prendre au cours.* "Attraper en courant" : Et en vne autre ylle y a gens qui ont pies de cheuaux, et sont fors et puissans et bien courans ; car il prennent au *cours* les bestes sauuages et les manguent. (MANDEVILLE, *Voy.* L., p.1360, 344). Qui tendroit la mer au deulx boulx A quatre cordes esthachee (...) L'om prandroit les lievres a *cors* (*Myst. st Sébast.* M., c.1450-1500, 125).

FIGURE 3.3 – Extrait de l'article *Cours* du DMF

Ainsi, le mot *connaissance* a comme formes différentes *congnoissance*, *congnissance*, *cognoiscence*, *cognoissance*, *connoissance*, *connissance* et *cognissance* ; le mot *information* a comme formes différentes *infourmation*, *informacion* et *information* ; enfin, le mot *cours* a comme formes différentes *cours*, *corps* et *cors*.

On constate que le substantif *cours* connaît seulement quelques variations, alors que le substantif *connaissance* en connaît beaucoup, au moins six rien que dans les extraits précédents.

Le DMF utilise plusieurs dictionnaires, dont le Godefroy (*D. Godefroy* 2008) et le Tobler-Lommatzsch (Blumenthal et Stein 2002). Ces deux dictionnaires sont des références en ce qui concerne l'époque médiévale, et ils utilisent chacun leurs propres règles de lemmatisation. Ces dictionnaires imposent à l'utilisateur de connaître l'entrée précise du terme qu'il recherche, c'est-à-dire souvent le lemme sans marque de nombre, de genre ou de flexion verbale. Un dictionnaire plus efficace devrait permettre un accès à toutes les formes graphiques du mot. Il devrait donc analyser la forme et la rattacher à une ou plusieurs entrées. Une méthode pour arriver à ce résultat est de s'appuyer sur les formes fléchies du mot (comme le font Morphalou (*Morphalou* 2012) et le Trésor de la Langue Française informatisé (TLFi) (ATILF - CNRS & Université de Lorraine)). Mais cette technique seule ne permettra pas d'obtenir des résultats concluants dans le cas du français médiéval, puisqu'il n'existe aucune liste exhaustive des formes fléchies, c'est pourquoi le Godefroy et le Tobler-Lommatzsch imposent à l'utilisateur de connaître l'entrée.

3.1.2 Les outils à disposition pour réaliser ce lemmatiseur

La première version de LGeRM a été réalisée grâce au DMF. Ce dernier est entièrement codé et exploité en XML (eXtensible Markup Language), qui est un langage informatique de balisage générique. Il s'écrit donc grâce à des balises, qui permettent de structurer et hiérarchiser un document. Il est donc particulièrement adapté ici, puisqu'il va permettre de construire et organiser facilement les articles. Un éditeur XML est ainsi utilisé par les rédacteurs pour construire leurs articles. Cet éditeur permet notamment de baliser l'entrée, le code grammatical, des définitions, des exemples, ainsi que l'occurrence de l'entrée dans l'exemple.

Le fait de sauvegarder les données du DMF en XML permet d'extraire deux listes sur lesquelles va s'appuyer le lemmatiseur : la liste des entrées du dictionnaire (les lemmes) et la liste des occurrences de l'entrée (les graphies).

Mais, comme nous l'avons vu plus haut, cela ne suffit pas. Il faut également un moyen d'analyser les formes inconnues. De nombreuses connaissances sur la morphologie de la langue médiévale sont alors nécessaires. Or, une étude commune avec l'Équipe de l'Unité de Recherche sur le Français Ancien avait été réalisée dans le cadre d'un DEA (Souvay 1986). Cette étude a permis d'avoir une première expérience en matière d'analyse de textes, d'élaborer un système d'analyseur, et de faire l'inventaire des connaissances à utiliser. Enfin, un premier prototype de lemmatiseur a pu être réalisé pour les mots invariables et le système nominal, grâce à une centaine de règles morphologiques.

3.1.3 Fonctionnement

Principe

Un algorithme fournit une liste d'hypothèses de lemmes à partir de la forme donnée en entrée. Il produit de nouvelles formes, en veillant à ne pas boucler ou produire trop de résultats.

Il a pour cela besoin d'utiliser plusieurs ressources : la liste des lemmes, la liste des graphies, et les règles de morphologie.

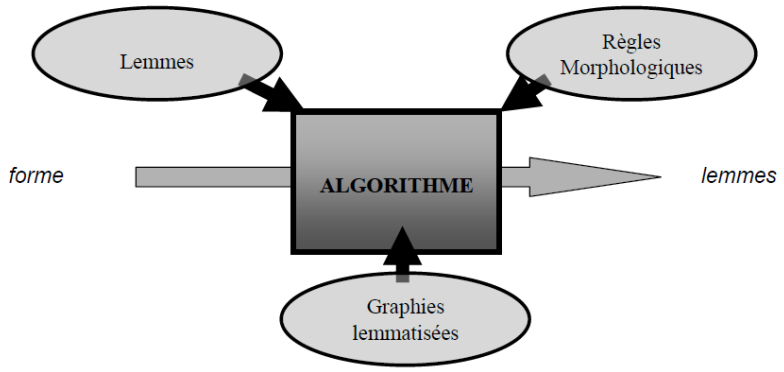


FIGURE 3.4 – Architecture du système (Souvay et Pierrel 2009)

Les ressources utilisées

Les lemmes La première version du lemmatiseur est basée sur une version initiale du DMF, qui comprenait 26 400 entrées. La liste des lemmes a été mise à jour au fur et à mesure de l'évolution du DMF ; ainsi, il comprend aujourd'hui plus de 60 000 entrées.

Les graphies 80 000 graphies ont été extraites grâce aux matériaux lexicographiques du DMF.

L'enrichissement s'est fait en partie par la génération automatique des formes au pluriel à partir des formes au singulier des substantifs et des adjectifs, et inversement ; par la génération automatique des formes féminines des adjectifs...

Il s'est également fait par l'extraction du DMF des références aux entrées des dictionnaires Tobler-Lommatzsch et Godefroy. Cela a permis d'enrichir la base en graphies plus anciennes que celles du DMF. Ainsi, la base contenait 150 000 graphies en 2004.

Une autre manière d'enrichir la liste des graphies est d'exploiter la nomenclature du TLFi, et sa liste de 400 000 formes fléchies. Cela sert particulièrement pour les formes flexionnelles verbales, qui sont mal représentées dans le DMF.

La génération de formes anciennes à partir des formes modernes par application de règles graphiques et morphologiques est également utilisée.

Enfin, le fonds de formes flexionnelles établi par Robert Martin (Martin 1960) est utilisé. Ce fonds est une base de formes verbales analysées, qui comprend 116 000 formes, dont de nombreuses formes irrégulières de verbes qui ne peuvent pas être générées automatiquement.

Les règles La définition des règles s'appuie au début sur des travaux réalisés en 1986 par l'Équipe de l'Unité de Recherche sur le Français Ancien. Les règles initiales ont été adaptées afin de produire des formes alternatives, et non plus ramener au lemme. Des règles pour la flexion verbale ont également été ajoutées.

Les règles sont des règles de réécriture sur les caractères de la forme en cours de traitement. Chaque règle a une pré-condition et une post-condition facultatives. La pré-condition porte sur l'initiale, la finale ou le contexte entourant un groupe de lettres. La post-condition porte sur le résultat.

La pré-condition peut être une fonction du type *entre*, *suivi de*, *suivi de sauf*, *non suivi de*..., en faisant référence à un caractère ou une suite de caractères. La post-condition va par exemple indiquer si la règle porte sur le système verbal, ou si la règle a conduit à un succès, c'est-à-dire que la forme recherchée est dans la base de graphies.

Il existe quatre catégories de règles :

1. règles morphologiques sur la flexion verbale ;
2. règles morphologiques sur la flexion nominale ou adjectivale ;
3. règles d'agglutination ;
4. règles générales purement orthographiques.

Voici quelques exemples de règles (Souvay et Pierrel 2009) :

— exemple de règle sur la flexion verbale :

Pré-condition :	en finale
Règle :	MECT → METTRE
Post-condition :	est verbal
Exemple :	*admect* → *admettre*

— exemple de règle sur la flexion nominale ou adjectivale :

Pré-condition :	en finale
Règle :	ALES → AL
Post-condition :	est nominal ou adjectival
Exemple :	*abbaciales* → *abbacial*

— exemple de règle orthographique :

Pré-condition :	en initiale et suivi de voyelle
Règle :	SÇ → S
Post-condition :	
Exemple :	*presçavoir* → *presavoir*

Les règles ne sont pas codées directement dans le programme informatique, afin de faciliter leur saisie, leur mise au point, leur relecture et leur portabilité. Elles sont représentées dans le format XML, et affichées grâce à une feuille de style. Elles sont de la forme « si une condition est remplie, alors on effectue une action » ; la condition correspond à la pré-condition et à la post-condition, et l'action correspond à la règle de réécriture.

Intérêts d'utiliser des graphies et des règles L'ajout de règles permet de résoudre les cas qui n'aboutissent pas, ou qui ont un résultat erroné.

L'ajout de graphies permet de répondre plus rapidement, notamment pour les mots courts.

L'algorithme

La lemmatisation classique consiste à trouver la forme normalisée du lemme.

Ici, il s'agit de trouver une forme connue dans la base de graphies la plus proche possible de la forme à lemmatiser.

Si la forme est dans la base, alors le lemmatiseur propose les formes correspondantes, et la lemmatisation est terminée.

Sinon, le lemmatiseur applique les règles morphologiques à cette forme. Si l'une des formes générées est connue, alors la lemmatisation est finie. Sinon, le processus est appliqué sur les nouvelles formes.

L'algorithme s'arrête si aucune nouvelle forme n'est générée, ou s'il a généré un trop grand nombre de formes (nous ne connaissons pas le seuil utilisé).

3.1.4 Résultats

Les résultats sont satisfaisants en mode consultation du dictionnaire, même s'il n'existe pas d'évaluation précise à notre disposition : l'utilisateur est guidé vers la bonne entrée. Les principales sources d'erreur sont les homographes, pour lesquels seul le lemme d'un des homographes est connu ; le lemmatiseur ne propose donc qu'une des possibilités.

Par exemple, pour l'homographe *alions*, le DMF ne propose que le lemme *aller*, et non pas *allier* qui est également valable.

En mode lemmatisation de textes, plus de problèmes sont constatés. Les hypothèses multiples font beaucoup de bruit. Les homographes rares sont systématiquement proposés, et l'ambiguïté entre verbe et préposition n'est pas levée. Mais il est malgré tout très pertinent pour la majorité des formes. Ainsi, une évaluation a été réalisée en 2006 sur un texte de

46 153 mots. Dans 60% des cas, un seul lemme est proposé, et il est correct. Dans 39% des cas, plusieurs lemmes, dont le correct, sont proposés.

Pour notre travail, le mode consultation de dictionnaire sera ainsi utile et intéressant, puisqu'il permet de donner de manière efficace un lemme possible associé à une forme verbale. Seulement, il devra être amélioré, puisqu'il n'en propose qu'un seul. Le mode lemmatisation de textes pourrait résoudre ce problème, mais il faudrait pour cela lever l'ambiguïté qui existe entre verbe et préposition.

Nous avons ainsi présenté un lemmatiseur hors contexte de l'ancien français, utilisé pour faciliter la consultation du Dictionnaire du Moyen Français. Par la suite, nous allons présenter Flemm (Namer 2014), un outil permettant de travailler sur des textes en français ayant déjà été étiqueté par un outil tel que TreeTagger ou Brill. Bien que dédié à des textes en français, il pourra nous être utile dans la création de notre outil.

3.2 Flemm

3.2.1 Présentation et fonctionnement de l'outil

Flemm est un ensemble de modules Perl5 qui effectue l'analyse flexionnelle de textes en français, préalablement étiquetés avec un catégoriseur, tel que TreeTagger ou Brill. Flemm opère sur un texte étiqueté, et fournit le lemme et les informations flexionnelles calculables hors contexte de chaque mot de ce texte.

> « Le programme prend en entrée un couple mot/étiquette grammaticale et renvoie en sortie la forme non fléchie du mot, ainsi que l'ensemble des traits flexionnels calculables hors-contexte. Il se décompose en deux modules. » (Namer 2014)

Module de transcodage

Le module de transcodage consiste en une interface entrée/sortie qui contrôle la validité de l'étiquette grammaticale fournie, et qui la corrigera éventuellement si besoin. Il convertit également « le format d'entrée dans une notation interne indépendante du système d'étiquetage » (Namer 2014), puis le format de départ est restitué en sortie.

L'interface permet de faire un transcodage catégoriel, qui apparie les jeux d'étiquettes en entrée à un jeu interne. Cela donne une grande indépendance à l'analyseur par rapport au

catégoriseur. Cela permet également d'avoir un système modulaire.

Pendant le transcodage, le programme examine la segmentation et la validité des étiquettes. En effet, une séquence mal segmentée est rédhibitoire pour les résultats de la lemmatisation.

Ensuite, la cohérence entre l'étiquette E fournie et la terminaison T du mot est contrôlée. Une règle compare le couple (E, T) avec l'ensemble des couples (E, T_i) possibles. Pour les mots outils, le programme consulte la liste finie de ces mots.

Toutes les erreurs ne sont pas déterminables avec cette méthode. De plus, l'approximation dans la correction peut entraîner de nouvelles erreurs. Ainsi, certaines erreurs seront remplacées par d'autres.

Module d'analyse

Le module d'analyse est un analyseur morphologique. Il calcule le lemme et les traits morphologiques, et traite les ambiguïtés. Trois opérations sont chargées de séparer la base et la terminaison du mot, et de calculer la forme neutre de la base, c'est-à-dire la forme de la base la plus courte et la plus simple.

Opération 1 Cette opération sert à détecter la terminaison du mot. Des règles de découpage s'effectuent en fonction de l'étiquette du mot, qui est alors examiné de droite à gauche. Toutes les terminaisons possibles sont prises en compte, puis la base résultante est examinée pour déterminer la bonne terminaison.

Opération 2 Cette opération consiste à réduire les allomorphes. Les allomorphes sont des variantes différentes d'un même morphème, comme *assoit* et *assied* dans les exemples « il s'assoit » et « il s'assied », ou *paye* et *paie* dans les exemples « je paye » et « je paie ».

Le calcul de la base neutre exploite les relations allomorphiques existantes entre les formes de base d'un même mot. Ces relations sont prises en compte par un ensemble de règles qui s'appliquent sur le couple (E, T) précédent. Si la base neutre est celle d'un verbe, le programme calcule la désinence de l'infinitif.

Opération 3 Cette opération calcule les informations flexionnelles. La partition base / terminaison déclenche la synthèse des traits accords, temps, cas... selon la partie du discours concernée. Le calcul de la terminaison, associé à celui de la forme neutre, permet de construire un modèle flexionnel. Celui-ci permet de classer les éléments d'une même catégorie grammaticale en fonction des règles flexionnelles.

3.2.2 Résultats

Pour les évaluations, le TLFnome a été utilisé. Le TLFnome est « un lexique de formes fléchies construit à l'INaLF par M. Papin et J. Maucourt, à partir de la nomenclature du Trésor de la Langue Française (TLF) » (Namer 2014). Il est utilisé comme témoin, car ses étiquettes et ses lemmes sont très fiables ; beaucoup de vérifications ont été réalisées.

Flemm a d'abord été étalonné en recalculant les lemmes des 412 081 formes étiquetées du TLFnome.

Ensuite, deux expériences ont été menées, afin d'évaluer la capacité de Flemm d'améliorer les performances de l'étiqueteur auquel il est associé.

La première expérience consiste à détecter et corriger les erreurs d'étiquetage et de segmentation. L'autre consiste à lemmatiser les mots inconnus.

Ces deux expériences ont été réalisées sur « trois corpus de contenus très différents, chacun étiqueté par Brill et par TreeTagger » (Namer 2014).

Le premier corpus est un extrait d'un roman d'Agatha Christie, avec un niveau de langue assez soutenu et de nombreux dialogues. Il comporte 73 000 mots.

Le deuxième corpus est une collection d'articles du journal *Le Monde* durant l'année 1992, avec un niveau de langue de type journalistique : du vocabulaire courant, beaucoup de noms propres, une syntaxe très riche et variée... Il comporte 1 534 600 mots.

Le troisième et dernier corpus est un recueil de notices bibliographiques pré-segmentées dans le domaine de l'agro-alimentaire, avec une syntaxe et un vocabulaire spécifiques des textes techniques et scientifiques. Il comporte 1 247 500 mots.

Enfin, une vérification manuelle des résultats de Flemm a été réalisée à l'IRIN (Nantes).

Étalonnage

Les résultats obtenus avec Flemm ont été comparés à ceux du TLFnome. Les taux d'erreurs obtenus sont décrits sur la figure 3.5.

Etiquette	**Taux d'Erreurs**	**Exemple → Lemme correct / Erreur** (FLEMM)
Participe Passé	0,08%	décrue → *décroire/décroître*
Participe Présent	0,16%	mouvant → *mouver/mouvoir*
Verbes Conjugués	1,04%	embatait → *embatre/embater*
Noms	3,8%	retrouvailles → *retrouvailles/retrouvaille*
		wagons-lits → *wagon-lit/wagons-lit*
Adjectifs	0,62%	aubergine → *aubergine/aubergin*

FIGURE 3.5 – Comparaison avec le lexique du TLFnome (Namer 2014)

On constate ainsi que les seules étiquettes à l'origine d'erreurs sont les verbes au participe passé, les verbes au participe présent, les verbes conjugués, les noms, et les adjectifs. Parmi celles-ci les noms qui sont la plus grande source d'erreur (avec un taux d'erreur de 3,8%).

Expérience de segmentation et d'étiquetage

Cette expérience a consisté « à répertorier les erreurs de segmentation et d'étiquetage détectées par Flemm, au moyen de l'option correspondante du programme, dresser la liste correspondante des unités lexicales dont les étiquettes sont "bien" corrigées » (Namer 2014). Les résultats obtenus sont décrits sur la figure 3.6.

Corpus (Nombre de mots)	Etiqueteur	Re-segmentation	Erreurs d'étiquetage	Ré-étiquetage Correct
AChristie (73 000)	BRILL TREETAGGER	0,06% 3,1%	0,85% 0,4%	77% 65%
LeMonde (1 534 592)	BRILL TREETAGGER	≈ 0% 0,02%	0,03% 0,12%	70% 33%
Agro (1 247 504)	BRILL TREETAGGER	0,75% 0,13%	3% 0,77%	90% 70%

FIGURE 3.6 – Re-segmentation et ré-étiquetage (Namer 2014)

On constate que la quantité d'erreurs d'étiquetage de chaque étiqueteur varie énormément selon le type et la taille de corpus. Pour les étiqueteurs Brill et TreeTagger, le corpus générant le moins d'erreurs est celui constitué d'articles de *Le Monde* de l'année 1992, composé de 1 534 600 mots, avec respectivement 0,03% et 0,12% d'erreurs.

En revanche, le corpus générant le plus d'erreurs d'étiquetage est celui composé de notices bibliographiques dans le domaine de l'agro-alimentaire, composé de 1 247 500 mots, avec 3% de taux d'erreurs pour Brill, et 0,77% pour TreeTagger.

L'étiqueteur à l'origine du plus grand nombre d'erreurs d'étiquetage est donc Brill.

Mais ce classement n'est pas respecté lorsque l'on regarde le taux d'erreurs de segmentation. En effet, pour TreeTagger, le corpus qui est à l'origine du plus grand nombre d'erreurs est le corpus composé d'un extrait d'un roman d'Agatha Christie, composé de 73 000 mots, avec un taux d'erreurs de 3,1%. Au contraire, pour Brill, il s'agit du corpus constitué de notices bibliographiques dans le domaine de l'agro-alimentaire, avec un taux d'erreurs de 0,75%.

On constate ainsi que l'étiqueteur à l'origine du plus grand nombre d'erreurs de segmentation est TreeTagger, au contraire des erreurs d'étiquetage.

Lemmatisation des mots inconnus

Cette expérience visait à compter le nombre de mots inconnus des étiqueteurs utilisés, puis à observer le comportement de Flemm face à ces mots. Les résultats sont représentés dans le tableau de la figure 3.7.

Corpus (nombre de mots)	Etiqueteur	Mots Inconnus	Lemmatisation correcte
AChristie (73 000)	BRILL	1,2%	70%
	TREETAGGER	2%	70%
LeMonde (1 534 592)	BRILL	7,5%	77%
	TREETAGGER	4,6%	90%
Agro (1 247 504)	BRILL	3,7%	80%
	TREETAGGER	6%	80%

FIGURE 3.7 – Lemmatisation des mots nouveaux (Namer 2014)

On constate ainsi que, pour les deux étiqueteurs Brill et TreeTagger, le corpus contenant le moins de mots inconnus est le corpus composé d'un extrait d'Agatha Christie, avec respectivement 1,2% et 2% de mots inconnus. Dans le cas des deux étiqueteurs, 70% de ces mots ont été correctement lemmatisés.

Pour les deux autres corpus, les résultats sont assez inégaux, mais le taux de mots inconnus correctement lemmatisés est à chaque fois supérieur à 77%, et monte même jusqu'à 90% dans le meilleur des cas.

Vérification manuelle

Cette expérience a été réalisée sur un corpus en français construit à partir de la revue *La Recherche*. Il comprend 450 articles sur différents thèmes scientifiques. Il a été étiqueté par Brill. L'expérience a été effectuée sur 5% des mots de ce corpus, soit 64 610. Les résultats portent sur deux points.

Tout d'abord, « le choix de la convention pour l'expression des lemmes des pronoms personnels : actuellement, et pour des motifs d'efficacité, Flemm affecte un seul lemme à toutes les formes pronominales clitiques, et les distingue par des traits de nombre, personne et cas » (Namer 2014). Par exemple, *lui* et *le* sont associés à la forme *il* avec des cas différents (respectivement le datif et l'accusatif).

Ensuite, « le pourcentage d'erreurs de lemmatisation en ne tenant compte que des mots étiquetés correctement : En intégrant la lemmatisation des pronoms qu'il juge erronée, l'évaluateur arrive à une moyenne de 2% d'erreurs de lemmatisation » (Namer 2014)

3.2.3 Utilité pour le français médiéval

Le module de transcodage ne nous sera pas utile, puisqu'il nécessite une liste finie des couples (E, T_i) précédemment définis possibles, ainsi qu'une liste finie des mots outils. Or, ces listes sont presque impossibles à définir pour le français médiéval, puisque, comme nous l'avons vu plus tôt, cette langue est en pleine évolution et son orthographe n'est pas fixe.

Au contraire, le module d'analyse pourra nous être utile, grâce aux trois opérations. Adapté au français médiéval, ces opérations peuvent permettre d'améliorer le mode consultation de dictionnaire de LGeRM, et ainsi permettre de trouver tous les lemmes possibles d'une forme verbale.

3.3 TCAF – Tableau de Conjugaison de l'Ancien Français

Le TCAF est un outil sur le site du DicFro (*DicFro* 2008) qui fournit des tableaux de conjugaison de l'ancien français, ainsi que des dictionnaires pour le français, le français médiéval et le latin. Le TCAF a été réalisé par Machio Okada et Hitoshi Ogurisu, entre 2007 et 2012. Une version hors ligne existe également.

Il contient les tableaux de conjugaison de 266 verbes en français médiéval différents.

Nous avons vu dans la partie sur les variations graphiques du français médiéval que cet outil permet, pour une forme verbale donnée, de fournir le ou les lemmes correspondants. Bien que la liste des graphies de chaque forme verbale ne puisse être exhaustive, cette base de données peut constituer un élément important pour la lemmatisation verbale complète de textes.

3.4 Conclusion

Après examen de différents outils, nous constatons que l'outil Flemm ne nous sera en lui-même pas utile, puisqu'il n'est pas adapté au français médiéval (seulement au français moderne), notamment le module de transcodage. En revanche, le fonctionnement du module d'analyse peut être intéressant. En effet, ce module utilise des règles de découpage base / terminaison, des règles de calcul de la base neutre, ainsi que des règles flexionnelles. Une telle approche à base de règles a permis d'obtenir de bon résultats, qui sont décrits plus haut. Une approche par règles pour le français médiéval a donc également du sens.

Nous chercherons donc à utiliser les règles morphologiques sur la flexion verbale de LGeRM, décrites plus haut, afin d'implémenter ces idées.

Enfin, nous utiliserons comme support la base de données du site du TCAF.

Chapitre 4

Présentation des corpus

4.1 Question du texte unique

Nous allons travailler sur un texte unique, afin de réduire les risques de variations graphiques, diachroniques ou dialectales. Des variations seront tout de même présentes, mais bien moins qu'entre plusieurs corpus.

Nous allons présenter, dans ce chapitre, le texte et l'édition sur lesquels nous avons décidé de travailler, ainsi que deux corpus annotés de cette édition du texte, afin de pouvoir les comparer et déterminer lequel des deux nous allons utiliser pour la suite.

4.2 Présentation du texte

Le corpus sur lequel nous travaillerons, sur lequel nous allons construire notre lemmatiseur, est le texte de la *Queste del saint Graal*. Il a été écrit vers 1225-1230 dans le Nord-Est de la France. Son auteur n'est pas connu. C'est l'un des plus anciens romans en prose, et l'un des plus connus du Moyen-Âge.

Il existe 53 manuscrits de ce texte, qui ont donné lieu à plusieurs éditions. L'édition de 1923 d'Albert Pauphilet a été utilisée pour réaliser ce corpus, à partir d'un manuscrit du XIII^{ème} siècle. Nous reviendrons plus tard plus en détail sur cette version.

La *Queste del saint Graal* appartient à un ensemble de récits consacrés au roi Arthur, aux Chevaliers de la Table Ronde, à l'enchanteur Merlin, ainsi qu'à cette quête du Graal, qui précède et annonce la fin du monde arthurien.

4.3 Présentation du corpus TXM

4.3.1 Informations générales

Choix de l'édition

Nous travaillerons sur l'édition électronique de ce roman mise à disposition pour le logiciel TXM (*Manuel de TXM 0.7 FR* 2018), librement téléchargeable, et le site du TXM (*BFM - Base de Français Médiéval* 2016). Ces plateformes sont développées dans le cadre du projet Textométrie (Heiden, Magué et Pincemin 2010). La textométrie est l'analyse de données textuelles appliquée au texte.

Cette édition utilise celle de 1923 d'Albert Pauphilet, basée sur un manuscrit du XIII^{ème} siècle, qui se trouve à la Bibliothèque municipale de Lyon. Mais il existe plus de 20 000 différences entre ces deux éditions, alors que le texte comporte environ 110 000 mots. Cette différence provient du fait que l'équipe du TXM a choisi de suivre le manuscrit le plus fidèlement possible, ce qui était moins le cas chez l'ancien éditeur.

Le souhait de l'équipe du TXM était ainsi de ne pas reconstituer un texte proche du supposé original, mais bien de restituer une version de la *Queste del saint Graal* telle qu'elle a circulé au XIII^{ème} siècle, « telle qu'elle a circulé environ un demi-siècle après la date supposée de sa composition, telle que les contemporains du manuscrit y ont eu accès, telle qu'ils l'ont lue et comprise » (Marchello-Nizia 2013), ce qu'ils appellent une « version usagée ». Le texte se veut ainsi très fidèle, mis à part un petit nombre de corrections précises.

Enrichissements apportés

En plus des différentes versions du textes, de sa traduction ou des photos numérisées du manuscrit, une étude succincte de la langue est fournie avec cet outil. Elle permet de mieux situer le texte dans le temps et l'espace.

Premier enrichissement Le premier enrichissement apporté à ce dossier est l'étiquetage morphologique du texte. Ainsi, une étiquette morphologique est attachée à chacun des mots du texte. Cette étiquette s'affiche automatiquement, lorsque l'utilisateur passe sa souris sur le mot. Cet enrichissement est illustré sur la figure 4.1.

En effet, tous les textes présents sur le site de la BFM (*BFM - Base de Français Médiéval* 2016) ont fait l'objet d'un étiquetage morphosyntaxique automatique. Cet étiquetage a été vérifié manuellement dans beaucoup de ces textes. Cet étiquetage a été fait en respectant le format Cattex09.

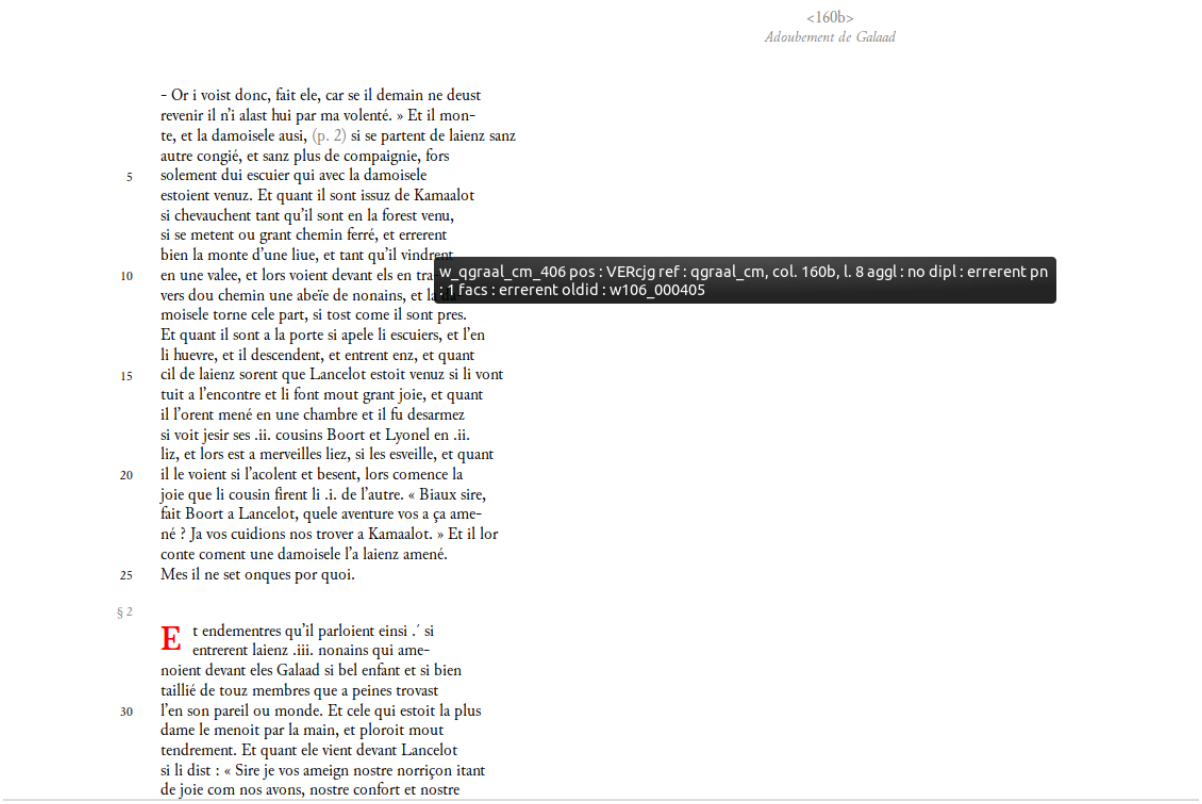


FIGURE 4.1 – Premier enrichissement au texte

Présentation du jeu d'étiquettes utilisé Le jeu d'étiquettes utilisé pour l'annotation de ce corpus est le jeu Cattex09 (Guillot et al. 2013a). Ce jeu a été créé par l'équipe de la BFM (*BFM - Base de Français Médiéval* 2016), afin d'annoter le contenu de la Base de Français Médiéval.

Il existe deux versions de ce jeu d'étiquettes : une version complète Cattex09max, qui comprend des informations morphologiques complètes telles que le nombre ou le temps ; et une version minimale Cattex09min, qui comprend uniquement des informations sur les catégories et leurs types. La version utilisée par les textes de la BFM, et par les annotations du corpus étiqueté par LGeRM, est la version minimale, désignée par défaut par « Cattex09 » (Guillot et al. 2013b).

Les étiquettes de Cattex09 sont composées d'un champ catégorie et d'un champ type, chaque valeur de champ étant composée de trois lettres, en majuscule pour la catégorie, en minuscule pour le type. Les catégories correspondent majoritairement aux parties du discours, et les types aux sous-classes des catégories.

Les catégories sont les suivantes : *VER* (verbe), *NOM* (nom), *ADJ* (adjectif), *PRO* (pronom), *DET* (déterminant), *ADV* (adverbe), *PRE* (préposition), *CON* (conjonction), *INJ* (interjection), *PON* (ponctuation), *ETR* (mot étranger), *ABR* (abréviation), *RED* (mot redondant), *OUT* (catégorie temporaire).

Pour donner quelques exemples de types, les types des verbes sont les suivants : *inf* (infinitif), *cjg* (conjugué), *ppa* (participe présent), *ppe* (participe passé) ; et les types des noms sont les suivants : *com* (commun), *pro* (propre).

Deuxième enrichissement Un second enrichissement au dossier est le balisage particulier des passages en discours direct, déjà mis en évidence par un surlignage coloré, de trois nuances de couleur pour différencier les trois niveaux de discours directs présents. Cet enrichissement est illustré sur la figure 4.2.

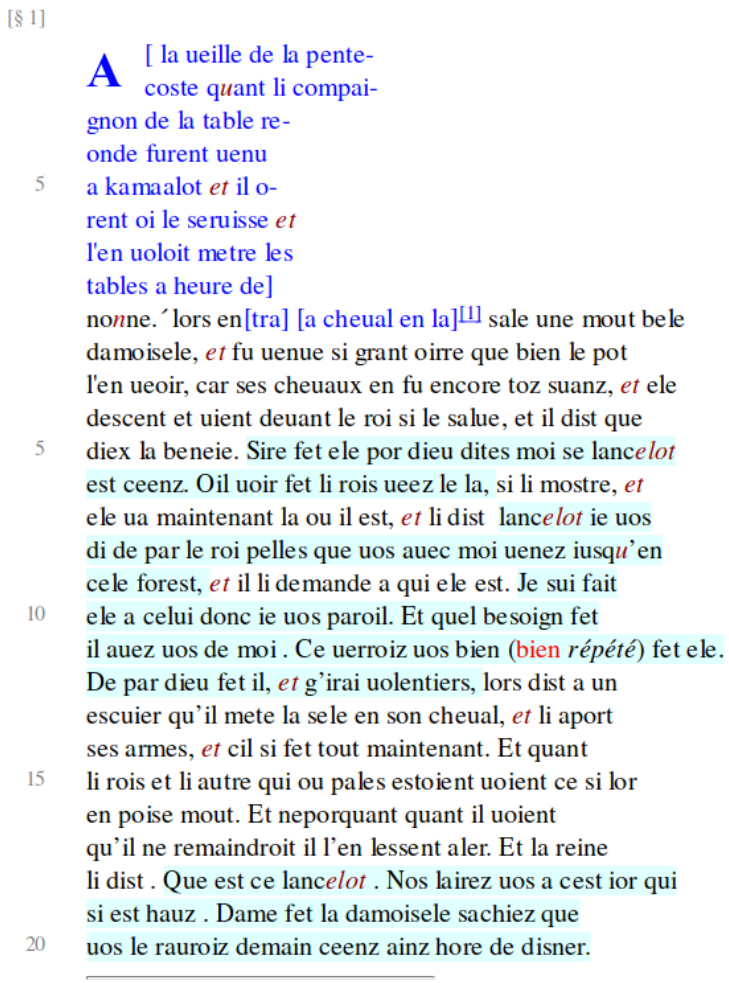


FIGURE 4.2 – Deuxième enrichissement au texte

Ces enrichissements permettent de faire beaucoup de recherches différentes, comme explicité plus bas dans la partie consacrée au moteur de recherche.

4.3.2 Données utilisées

Contenu de l'édition

Cette édition contient :

— 50 pages de l'introduction (Marchello-Nizia 2013) ;

— les photos numérisées de 65 folios recto et verso du manuscrit de Lyon, ce qui correspond à 130 pages de deux colonnes chacune, c'est-à-dire 260 colonnes de texte, ainsi

que les photos numérisées de quelques folios du manuscrit de Paris, qui a permis de combler les lacunes du premier ;
— l'édition du texte médiéval en trois niveaux de transcription, que nous présenterons plus en détail dans la suite : la version courante (ou normalisée), la version diplomatique, et la version fac-similaire ;
— la traduction en français moderne ;
— les notes et commentaires ajoutés au texte ;
— un index des noms propres et un glossaire ;
— des études menées sur cette édition et sur ce texte.

Version utilisée

Ainsi, le texte du manuscrit a bénéficié d'une triple transcription, c'est-à-dire de trois niveaux de représentation plus ou moins proches du manuscrit.

La version courante Cette version permet une lecture normale. Elle est fidèle au manuscrit, et est adaptée aux normes modernes. C'est la version qui s'affiche par défaut à l'ouverture des interfaces, et qui est annotée.

La version diplomatique Cette version, en cours d'élaboration, est encore plus fidèle que la version courante. À terme, sa ponctuation sera adaptée à la typographie moderne, aucune diacritique moderne ne sera utilisée, et on ne distinguera pas le *-i-* et le *-j-*, ni le *-u-* et le *-v-*.

La version fac-similaire Cette version restitue les principales particularités graphiques du manuscrit, telles que les marques d'abréviation, les variantes calligraphiques, ou les signes de ponctuation médiévaux. De plus, elle respecte le groupement des mots du copiste.

La version courante est celle que nous utiliserons dans le cadre de ce travail, puisqu'elle est annotée.

L'interface électronique

Cette édition est disponible en consultation libre sur le logiciel TXM et le site du TXM (*BFM - Base de Français Médiéval* 2016).

Ces deux interfaces offrent les mêmes possibilités ; seule l'interface diffère légèrement. Elles permettent un accès libre, un texte fiable et facile d'accès, pour des usages diversifiés. Elles

disposent de beaucoup de ressources, telles que « l'affichage multifacette », ou la possibilité de réaliser de nombreuses recherches complexes, grâce à un balisage riche et profond de type XML de type TEI (Text Encoding Initiative), qui est proposé par une communauté académique internationale visant à définir des recommandations pour l'encodage de documents textuels.

L'affichage Les interfaces du TXM permettent un « affichage multifacette », c'est-à-dire qu'un affichage en plusieurs colonnes est possible. On peut ainsi visionner côte à côte plusieurs versions différentes du texte (comme l'un ou plusieurs des niveaux de transcription, la traduction en langue moderne, ou les photos numérisées du texte).

La figure 4.3 montre un exemple de « l'affichage multifacette », avec la version courante dans la colonne de gauche, la version fac-similaire dans la colonne du milieu, et la traduction dans la colonne de droite.

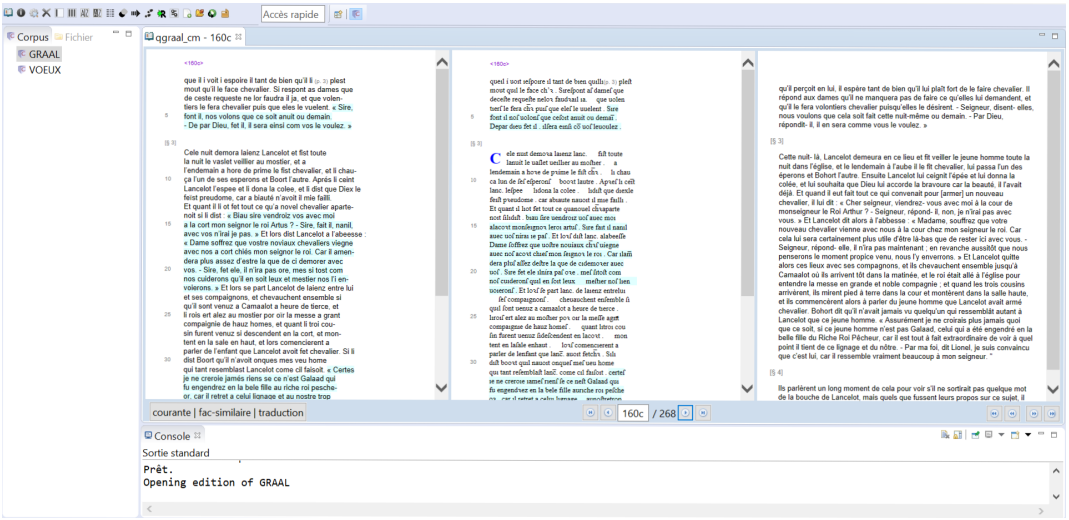


FIGURE 4.3 – Exemple d'affichage multifacette du logiciel TXM

Le moteur de recherche Le moteur de recherche intégré aux interfaces permet de retrouver facilement des mots, des constructions, des catégories grammaticales, et permet de les trier, de les combiner, d'en faire des concordances... Il fonctionne grâce au balisage XML et à l'étiquetage morphologique complet.

Plus précisément, ce moteur de recherche et d'analyse textométrique permet de demander, en langage CQL, la liste d'occurrences, ou encore la concordance d'une forme grammaticale ou lexicale, selon un contexte que l'utilisateur peut définir. Des requêtes complexes peuvent

également prendre en compte la forme des mots, ou leur catégorie grammaticale.

Le langage CQL (Corpus Query Language) est un langage de recherche permettant de faire des recherches avancées sur des modèles grammaticaux et lexicaux complexes (*CQL - Corpus Query Language* 2015). Il utilise des crochets « [] » pour noter les critères voulus sur chaque unité lexicale. Ces critères contiennent en premier à quoi doit s'appliquer la condition (par exemple au lemme ou à l'étiquette du mot) ainsi que la condition, écrite grâce à des expressions régulières.

Quelques exemples de requête CQL sont les requêtes suivantes : `[attribute="value"]`, qui permet de rechercher tous les mots dont l'attribut a pour valeur *value*, et `[lemma="confus.*"]`, qui permet de rechercher tous les mots dont le lemme commence par *confus*.

Le balisage XML Ces interfaces utilisent un balisage XML-TEI, afin d'encoder le texte et les analyses. Grâce à ce balisage, il nous sera facile d'avoir accès au texte et à ses différents éléments.

Les limites de cet outil

TXM offre ainsi beaucoup d'avantages, et permet des recherches efficaces grâce, nous l'avons vu, à un étiquetage morphologique complet et à un encodage précis.

En effet, 53 étiquettes morphologiques sont enregistrées et utilisées. Elles permettent de donner des informations précises, sauf les étiquettes concernant les verbes. En effet, seules quatre étiquettes existent, qui correspondent aux verbes conjugués, aux verbes à l'infinitif, aux verbes au participe présent et ceux au participe passé. Aucune précision n'est donnée quant au temps du verbe conjugué, ni sa personne ou son mode.

L'étiquetage des verbes est donc très limité, ce qui ne permet pas d'effectuer des recherches efficaces ou pertinentes sur eux. Cette lacune vient du fait que le français médiéval est une langue dont l'orthographe et la conjugaison ne sont pas fixées, comme nous l'avons vu dans le chapitre sur les variations graphiques.

Notre travail de lemmatisation des formes verbales permettrait donc de compléter cet outil, puisqu'il donnerait des étiquettes beaucoup plus précises aux verbes : il fournirait ainsi toutes les possibilités de lemmes, de temps, de personnes et de modes.

4.3.3 Obtention des annotations du corpus TXM

Pour extraire les annotations du corpus TXM, nous disposons de la version hors ligne, téléchargeable depuis la plateforme du TXM (*BFM - Base de Français Médiéval* 2016), dans

l'onglet *GRAAL*.

Cette version hors ligne est un répertoire contenant cinq sous-répertoires, et un fichier XML. L'un de ces sous-répertoires, nommé *HTML*, contient les fichiers HTML correspondant à chaque page de l'œuvre pour chaque version (à savoir la courante, la diplomatique, la fac-similaire, la ms-colonne, la ms-page, et enfin la traduction).

Ces fichiers HTML contiennent le contenu de chaque page pour chaque version, c'est-à-dire le texte de la page, les numéros de lignes, le numéro de la page, l'étiquette de chaque mot, ainsi que des informations d'affichage, telles que la couleur d'arrière plan ou du texte. La figure 4.4 présente un extrait de l'un de ces fichiers.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns:tei="http://www.tei-c.org/ns/1.0" xmlns:bfm="http://bfm.ens-lsh.fr/ns/1.0" xmlns:me="http://www.menota.org/ns/1.0">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Queste del saint Graal</title>
    </head>
    <body style="margin-left:2%;margin-right:2%">
        <div class="text" style="font-size:13px;padding:2%">
            <div class="p" style="margin-left:10%;position:relative;white-space:nowrap">
                <span style="color:darkviolet;font-family:arial;font-size:80%;page-break-before:always">&lt;160a&gt;</span>
                <br/>
            </div>
            <p style="text-align:justify;text-indent:50px">
                <span style="color:blue;font-family:Times New Roman;font-style:italic">Ici commence la version de la
Queste del saint Graal donnée par le manuscrit K (Bibliothèque Municipale de Lyon,
Palais des Arts n° 77), folios 160 recto à 224 verso. Tout le début du texte a été
mutilé : la première grande lettre a été découpée, comme on le voit sur la reproduction
du manuscrit, et quelques lignes du texte manquent, que nous donnons ici entre crochets,
en bleu, d'après le manuscrit Z (Paris, BNF n. acq. fr. 1119, folio 138 recto, colonne a)
qui est un manuscrit proche de celui que nous éditons ici.</span>
            </p>
            <span style="color:gray;font-size:80%">(p. 1)</span>
            <div class="p" style="margin-left:10%;position:relative;font-family:arial;line-height:1.25;white-space:nowrap">
                <br/>
                <span style="position:absolute;left:-8%;color:gray;font-size:80%">[§ 1]</span>
                <br/>
                <span style="color:blue">
                    [
                    <span class="word" title="PRE" id="w_106_000001">A</span>
                    <span class="word" title="DETdef" id="w_106_000002">la</span>
                    <span class="word" title="NOMcom" id="w_106_000003">veille</span>
                    <span class="word" title="PRE" id="w_106_000004">de</span>
                    <span class="word" title="DETdef" id="w_106_000005">la</span>
                    <span class="word" title="NOMpro" id="w_106_000006">Pente-<br/>coste</span>
                    <span class="word" title="CONsub" id="w_106_000007">quant</span>
                    <span class="word" title="DETdef" id="w_106_000008">li</span>
                    <span class="word" title="NOMcom" id="w_106_000009">compai-<br/>gnon</span>
                    <span class="word" title="PRE" id="w_106_000010">de</span>
                    <span class="word" title="DETdef" id="w_106_000011">la</span>
                    <span class="word" title="NOMcom" id="w_106_000012">table</span>
                    <span class="word" title="ADJqua" id="w_106_000013">re-<br/>onde</span>
                    <span class="word" title="VERcjg" id="w_106_000014">furent</span>
                    <span class="word" title="VERppe" id="w_106_000015">venu</span>
                    <br/>
                    <span style="position:absolute;left:-8%;font-size:80%;color:gray">5</span>
                    <span class="word" title="PRE" id="w_106_000016">a</span>
                    <span class="word" title="NOMpro" id="w_106_000017">Kamaalot</span>
                    <span class="word" title="CONcoo" id="w_106_000018">et</span>
                    <span class="word" title="PROper" id="w_106_000019">il</span>
                    <span class="word" title="VERcjg" id="w_106_000020">o-<br/>rent</span>
                    <span class="word" title="VERppe" id="w_106_000021">oï</span>
                    <span class="word" title="DETdef" id="w_106_000022">le</span>
                    <span class="word" title="NOMcom" id="w_106_000023">servisse</span>
                    <span class="word" title="CONcoo" id="w_106_000024">et</span>
                    <br/>
```

FIGURE 4.4 – Extrait de la première page de texte de la version courante

Les éléments qui nous intéressent sont ainsi le texte et les étiquettes de mot. Nous avons également remarqué que les balises correspondant aux mots du texte, et comprenant les étiquettes, sont des balises `span` ayant comme classe *word*. Nous avons donc pu rechercher les balises `<span class="word">` dans tous les documents, grâce à la librairie python `BeautifulSoup`, qui permet d'accéder facilement au contenu d'une page HTML. À partir de

ces balises, il a été possible d'accéder au terme, ainsi qu'à l'étiquette correspondant à chaque balise. Ces étiquettes respectent le jeu Cattex09, comme pour le corpus étiqueté par LGeRM.

Ce texte comprenait malgré tout quelques difficultés. En effet, il comprenait notamment des termes coupé en leur milieu, par un tiret ou une balise, écrits sur deux lignes différentes, qui parfois se trouvaient elles-mêmes sur des pages différentes. Nous avons donc dû nettoyer ces mots séparés en les regroupant. Voici un exemple de ce cas, présent sur la figure 4.4 : `<span class="word"title="ADJqua"id="w_106_000013">re-<br/>onde</span>`.

Le texte contenait également des mots dont une partie était écrite entre crochets, comme dans l'exemple suivant :

```
<span class="word" title="VERcjg" id="w_106_000036" >
en<span style="color:blue" >[tra]</span>
</span>
```

Nous avons dans ce cas gardé la version complète du mot, c'est-à-dire comprenant la partie entre crochets, soit *entra* dans l'exemple précédant. En effet, ces crochets symbolisent des extraits du texte manquants dans l'édition, d'après le manuscrit Z, proche de celui utilisé par le TXM.

4.3.4 Enregistrement du corpus annoté

Pour enregistrer ces données, et afin de pouvoir les comparer facilement au corpus étiqueté par LGeRM, nous avons décidé de sauvegarder ces données dans un fichier CSV à deux colonnes. Les termes du texte seront enregistrés dans la première colonne, et leurs étiquettes dans la deuxième.

Le script ayant permis d'extraire les données des pages HTML du corpus et de les enregistrer au format CSV est présenté sur la figure A.1 (page 72).

Nous obtenons ainsi un fichier au format CSV à deux la colonnes, la première contenant le terme du texte, et la deuxième contenant son étiquette au format Cattex09.

En procédant de cette manière, nous obtenons un fichier contenant 118 278 mots en tout, dont 22 522 étiquetés comme des verbes. Parmi eux, il y a 3 900 formes différentes. 16 820 des verbes sont annotés comme verbes conjugués, 3 164 sont annotés comme participes passés, 227 sont annotés comme participe présent, et enfin 2 313 sont annotés comme infinitifs.

4.4 Corpus étiqueté par LGeRM

4.4.1 Présentation

Ce corpus a été composé à partir du texte et des étiquettes du corpus TXM (de l'édition courante), et lemmatisé et annoté grâce à LGeRM (Souvay et Pierrel 2009). Cette version est disponible sur le site de l'ATILF (ATILF - CNRS & Université de Lorraine).

Ce texte contient 118 490 mots au total, dont 22 502 ayant été étiqueté comme verbe. Parmi ces verbes, il y a 3 863 formes différentes. Il y a également 16 815 formes annotées comme étant un verbe conjugué, 3 148 formes annotées comme participe passé, 226 formes annotées comme participe présent, et enfin 2 313 formes annotées comme infinitif.

4.4.2 Obtention des annotations du corpus étiqueté par LGeRM

Pour obtenir les annotations du corpus étiqueté par LGeRM, nous avons contacté Gilles Souvay, qui est ingénieur de recherche au CNRS, et qui participe au projet LGeRM. Nous lui avons fait part de notre projet, et il a accepté de nous fournir un document CSV contenant les annotations du texte. Ce document est séparé en quatre colonnes. La première correspond au mot présent dans le texte du corpus TXM. La deuxième correspond au(x) lemme(s) du mot du texte, trouvés par LGeRM ; ce sont des lemmes provenant du DMF (Martin 2015). La troisième correspond aux étiquettes TreeTagger ajoutées par LGeRM de ce lemme. Enfin, la quatrième colonne correspond à l'étiquette Cattex09 du mot, provenant de TXM. Un extrait de ce fichier est présenté sur la figure 4.5.

1	A	À	prép.	PRE
2	la	LE	art. déf.	DETdef
3	veille	VEILLE1|VEILLE2|VEILLE3|VOILE2	subst. fém.|subst. fém.|subst. fém.|subst. masc. et fém.	NOMcom
4	de	DE	prép.	PRE
5	la	LE	art. déf.	DETdef
6	Pentecoste	µPENTECÔTE	nom propre	NOMpro
7	quant	QUAND	adv. et conj.	CONsub
8	li	LE	art. déf.	DETdef
9	compaignon	COMPAGNON	subst. masc.	NOMcom
10	de	DE	prép.	PRE
11	la	LE	art. déf.	DETdef
12	table	TABLE	subst. fém.	NOMcom
13	reonde	ROND	adj. et subst. masc.	ADJqua
14	furent	ÊTRE1	verbe	VERcjg
15	venu	VENIR	verbe	VERppe
16	a	À	prép.	PRE
17	Kamaalot	µCAMELOT	nom propre	NOMpro
18	et	ET	conj. de coord.	CONcoo
19	il	IL	pron. pers.	PROper
20	orent	AVOIR1|ORER	verbe|verbe	VERcjg
21	oï	OUÏR	verbe	VERppe
22	le	LE	art. déf.	DETdef
23	servisse	SERVICE	subst. masc.	NOMcom
24	et	ET	conj. de coord.	CONcoo
25	l'en	ON	pron. pers.	PROind
26	voloit	VOLER|VOULOIR	verbe|verbe	VERcjg
27	metre	METTRE	verbe	VERinf
28	les	LE	art. déf.	DETdef
29	tables	TABLE	subst. fém.	NOMcom
30	a	À	prép.	PRE
31	heure	HEURE|HOIR	subst. fém.|subst. masc.	NOMcom
32	de	DE	prép.	PRE
33	nonne	NONE|NONNE	subst. fém.|subst. fém.	NOMcom
34	'	PONfbl	PONfbl	PONfbl
35	lors	LORS	adv.	ADVgen

FIGURE 4.5 – Extrait du corpus étiqueté par LGeRM

Ce corpus contient des ambiguïtés sur les lemmes et les étiquettes TreeTagger, symbolisées par le caractère « | » qui sépare chaque possibilité. L'étiquette Cattex09 est utilisée pour désambiguïser, même si ce n'est pas toujours possible. Une limite de cette annotation est que, lorsqu'une ambiguïté a été levée, les autres hypothèses ne sont pas présentées.

Ainsi, nous avons un fichier CSV à quatre colonnes, que nous pouvons parcourir simplement, et dont nous pouvons trouver les verbes en vérifiant que la valeur de la quatrième colonne de chaque ligne commence par la chaîne *VER*.

4.5 Comparaison des deux corpus

Afin de comparer efficacement ces deux corpus, nous avons développé un script en Python, visible sur la figure A.2 (page 73), afin de calculer quelques statistiques. Ce script nous a notamment servi à calculer le nombre de mots, de verbes, de formes verbales différentes, de verbes conjugués, de verbes au participe passé, de verbes au participe présent, et de verbes à l'infinitif de chacun des textes.

Le tableau 4.1 récapitule les chiffres que nous avons obtenus pour chaque texte, qui ont été donnés précédemment.

	Corpus TXM	Corpus étiqueté par LGeRM
Nombre de mots	118 278	118 490
Nombre de verbes	22 522	22 502
Nombre de verbes conjugués	16 820	16 815
Nombre de verbes à l'infinitif	2 313	2 313
Nombre de verbes au participe passé	3 164	3 148
Nombre de verbes au participe présent	227	226

TABLE 4.1 – Comparaison des chiffres des deux corpus

Nous constatons que ces chiffres ne sont pas les mêmes, ce qui n'est pas logique puisque le corpus étiqueté par LGeRM provient de celui de TXM. Nous allons par la suite essayer de comprendre ces différences, afin de déterminer le corpus annoté comprenant des erreurs, afin d'utiliser le texte n'en comprenant pas.

Nous constatons en premier lieu que le corpus étiqueté par LGeRM contient 212 mots de plus que le corpus TXM. Cependant, le corpus étiqueté par LGeRM contient 52 formes étiquetées comme verbes en moins que le corpus TXM, dont 5 verbes conjugués, 16 participes passés et un participe présent.

Parmi les formes verbales présentes dans le corpus étiqueté par LGeRM, toutes sont présentes parmi les formes verbales du corpus TXM.

Au contraire, parmi les formes verbales présentes dans le corpus TXM, neuf de ces formes ne sont pas présentes parmi les formes verbales du corpus étiqueté par LGeRM ; il s'agit des formes « ferrees », « hardiz », « solde », « gait », « tes », « apa », « reilliez », « puanz », et « mont ».

4.5.1 Cas des formes présentes dans le texte du TXM, mais pas dans celui de LGeRM

En ce qui concerne les formes verbales présentes dans le corpus TXM mais pas dans celui de LGeRM, nous pouvons séparer deux catégories concernant leurs différences.

La première catégorie concerne les formes « hardiz », « solde », « ferrees », « gait », « puanz », et « mont ».

Ces formes, bien qu'identifiées comme des verbes dans le corpus TXM, ne sont pas annotées comme étant des verbes par LGeRM. Ce sont des formes qui présentent des ambiguïtés, et dont l'étiquette Cattex09 a pour valeur « 0 ».

Voici ci-dessous le tableau 4.2 associant chacune de ces formes à son étiquette TXM, et à ses étiquettes et lemmes déterminés par LGeRM.

Forme	Etiquette du TXM	Etiquettes TreeTagger par LGeRM	Lemmes par LGeRM
hardiz	VERppe	adj. | subst. masc.	HARDI | HARDIT
solde	VERcjg	subst. fém. | adj. fém.	SOLDE1 | SOLDE2
ferrees	VERppe	verbe | adj. | subst. fém.	FERRER | FERRÉ | FERRÉE2
gait	VERcjg	subst. masc.| subst. masc.	GUET | GUET-APENS
puanz	VERppa	adj. | adv.	PUANT | PUER
mont	VERcjg	adv. | subst. masc. | subst. masc. | quantif.	MON2 | MONDE1 | MONT | MOULT

TABLE 4.2 – Tableau de correspondances entre les étiquettes du TXM et celles de LGeRM pour les formes présentant des ambiguïtés

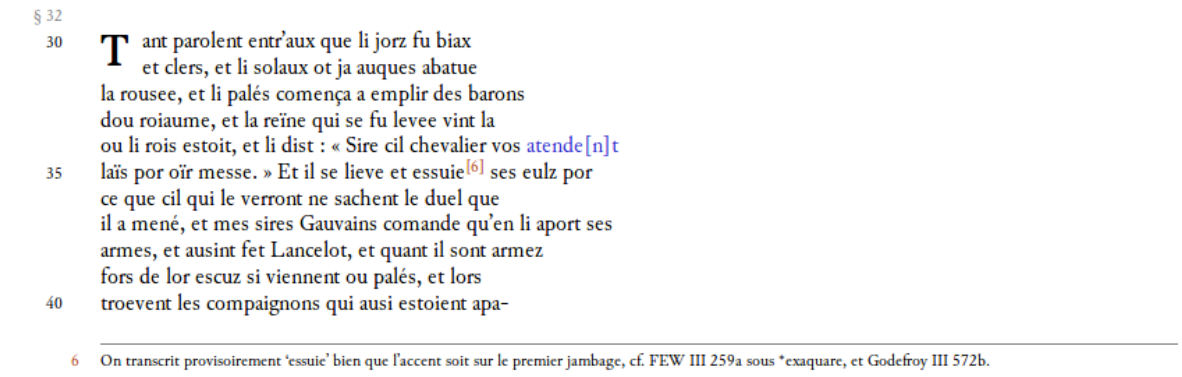
La deuxième catégorie concerne les formes « apa », « reilliez », et « tes ».

Ces formes correspondent en réalité à des erreurs provenant de notre méthode d'extraction des données depuis les fichiers HTML de la version téléchargée de TXM. En effet, ces documents comportent des exceptions imprévisibles dans les données, et également dans les balises `<span class="word">` qui nous ont servi à extraire les annotations.

Nous remarquons donc que certaines formes, comme « tes », « apa », et « reilliez », sont mal extraites. En effet, « tes » correspond en réalité au suffixe de la forme « dites », qui est effectivement un verbe conjugué. Mais les exceptions présentes dans les fichiers ne nous ont pas permis de regrouper les deux parties du mots « di » et « tes », séparées par un tiret dans deux balises `span` différentes. De la même façon, « apa » et « reilliez » sont respectivement le préfixe et le suffixe du verbe « apareilliez ». C'est pour cela que ces formes ne sont pas présentes dans le corpus étiqueté par LGeRM, puisqu'elle ne constituent pas des mots.

Dans le cas de « apa » et « reilliez », ils se trouvent sur deux pages différentes. Celle qui contient « apa » se termine de la façon présentée à la figure 4.6, et celle qui contient

« reilliez » se termine comme présenté sur la figure 4.7.


§ 32

Tant parolent entr'aux que li jorz fu biax
et clers, et li solaux ot ja auques abatue
la rousee, et li palés comença a emplir des barons
dou roiaume, et la reïne qui se fu levee vint la
ou li rois estoit, et li dist : « Sire cil chevalier vos atende[n]t
laïs por oïr messe. » Et il se lieve et essuie[6] ses eulz por
ce que cil qui le verront ne sachent le duel que
il a mené, et mes sires Gauvains comande qu'en li aport ses
armes, et ausint fet Lancelot, et quant il sont armez
fors de lor escuz si viennent ou palés, et lors
troevent les compaignons qui ausi estoient apa-

6 On transcrit provisoirement 'essuie' bien que l'accent soit sur le premier jambage, cf. FEW III 259a sous *exaquare, et Godefroy III 572b.

FIGURE 4.6 – Extrait de la page se terminant par « apa »

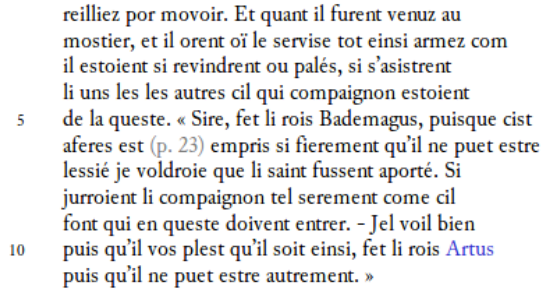

reilliez por movoir. Et quant il furent venuz au
mostier, et il orent oï le servise tot einsi armez com
il estoient si revindrent ou palés, si s'asistrent
li uns les les autres cil qui compaignon estoient
de la queste. « Sire, fet li rois Bademagus, puisque cist
aferes est (p. 23) empris si fierement qu'il ne puet estre
lessié je voldroie que li saint fussent aporté. Si
jurroient li compaignon tel serement come cil
font qui en queste doivent entrer. - Jel voil bien
puis qu'il vos plest qu'il soit einsi, fet li rois Artus
puis qu'il ne puet estre autrement. »

FIGURE 4.7 – Extrait de la page commençant par « reilliez »

De plus, la figure 4.8 montre un extrait du fichier HTML correspondant à la figure 4.6.

```
115 <span class="word" title="NOMcom" id="w_106_009312">armes</span>
116 <span class="word" title="PONfbl" id="w_106_009313">,</span>
117 <span class="word" title="CONcoo" id="w_106_009314">et</span>
118 <span class="word" title="ADVgen" id="w_106_009315">ausint</span>
119 <span class="word" title="VERcjg" id="w_106_009316">fet</span>
120 <span class="word" title="NOMpro" id="w_106_009317">Lancelot</span>
121 <span class="word" title="PONfbl" id="w_106_009318">,</span>
122 <span class="word" title="CONcoo" id="w_106_009319">et</span>
123 <span class="word" title="CONsub" id="w_106_009320">quant</span>
124 <span class="word" title="PROper" id="w_106_009321">il</span>
125 <span class="word" title="VERcjg" id="w_106_009322">sont</span>
126 <span class="word" title="VERppe" id="w_106_009323">armez</span>
127 <br/>
128 <span class="word" title="ADVgen" id="w_106_009324">fors</span>
129 <span class="word" title="PRE" id="w_106_009325">de</span>
130 <span class="word" title="DETpos" id="w_106_009326">lor</span>
131 <span class="word" title="NOMcom" id="w_106_009327">escuz</span>
132 <span class="word" title="ADVgen" id="w_106_009328">si</span>
133 <span class="word" title="VERcjg" id="w_106_009329">viennent</span>
134 <span class="word" title="PRE.DETdef" id="w_106_009330">ou</span>
135 <span class="word" title="NOMcom" id="w_106_009331">palés</span>
136 <span class="word" title="PONfbl" id="w_106_009332">,</span>
137 <span class="word" title="CONcoo" id="w_106_009333">et</span>
138 <span class="word" title="ADVgen" id="w_106_009334">lors</span>
139 <br/>
140 <span style="position:absolute;left:-8%;font-size:80%;color:gray">40</span>
141 <span class="word" title="VERcjg" id="w_106_009335">troevent</span>
142 <span class="word" title="DETdef" id="w_106_009336">les</span>
143 <span class="word" title="NOMcom" id="w_106_009337">compaignons</span>
144 <span class="word" title="PROrel" id="w_106_009338">qui</span>
145 <span class="word" title="ADVgen" id="w_106_009339">ausi</span>
146 <span class="word" title="VERcjg" id="w_106_009340">estoient</span>
147 <span class="word" title="VERppe" id="w_106_009341">apa-
148     <br/>
149     <hr/>
150     <p style="font-size:70%;text-align:justify">
151         <span style="position:absolute;left:-30px">
152             <a href="#noteref_6" name="note_6">6</a>
153             .
154         </span>
155         On transcrit provisoirement ‘essuie’ bien que l’accent soit sur le premier jambage, cf. FEW III 259a sous *exaquare, et Godefroy III 572b.
156     </p>
157     <br/>
158     <br/>
159 </span>
```

FIGURE 4.8 – Extrait du fichier HTML se terminant par « apa »

Nous constatons ainsi que la dernière balise `<span class="word">` de cet extrait est celle contenant la forme « apa- », séparée de son suffixe « reilliez » présent sur la page suivante.

Or, cette balise contient également les balises permettant de mettre en page la note de bas de page, également visible sur la figure 4.6. Ce sont ces balises qui sont à l'origine du problème, puisque notre script utilise le texte des balises `<span class="word">`, qui correspond dans ce cas-là à « apa-On transcrit provisoirement 'essuie' bien que l'accent soit sur le premier jambage, cf. FEW III 259a sous *exaquare, et Godefroy III 572b. ».

Il ne nous a cependant pas été possible d'identifier un pattern afin de régler automatiquement ce problème, c'est-à-dire d'intégrer dans le script des règles pour résoudre ce problème. Il nous serait en revanche possible de résoudre manuellement ce problème pour ces deux cas précis, mais nous n'avons pas moyen de savoir si cette erreur s'est produite à d'autres endroits dans le corpus, et donc aucun moyen de résoudre toutes les erreurs causées par ce problème.

4.5.2 Choix du corpus à utiliser

Nous avons ainsi constaté que le corpus du TXM, que nous avons extrait depuis les fichiers HTML de la version téléchargée de TXM, comporte des erreurs d'extraction. Ces erreurs sont liées à la façon dont sont construits ces fichiers HTML, en ce qui concerne les notes de bas

de page, comme nous l'avons vu dans la section précédente. Nous avons pu identifier deux cas d'erreurs, mais d'autres existent certainement, sans que nous ayons pu les identifier.

De plus, certaines formes identifiées comme des verbes par TXM ont été étiquetées autrement par LGeRM, générant des ambiguïtés dans le corpus étiqueté par LGeRM.

Ces erreurs et différences entre les deux corpus expliquent les différences de nombre de mots et de verbes entre les deux que nous avons vus plus tôt.

Il est donc plus judicieux d'utiliser le corpus étiqueté par LGeRM, puisque ce corpus ne contient pas les erreurs liées à notre système d'extraction des données du TXM.

Un autre avantage d'utiliser ce corpus est la présence des annotations de LGeRM, en plus des annotations de TXM.

En effet, nous allons ainsi pouvoir utiliser les annotations de LGeRM, et celles de TXM pour les désambiguïser si nécessaire, puisque les annotations de TXM ont été vérifiées manuellement.

Chapitre 5

Constitution de la base de données des verbes du TCAF

Afin d'annoter le corpus précédemment choisi, nous allons commencer par utiliser la base de données de verbes mise à disposition sur le site du TCAF. Afin de pouvoir utiliser cette base de données de manière efficace et rapide, nous allons avoir besoin de la sauvegarder.

En effet, il sera plus rapide d'enregistrer cette base, puis de l'utiliser au moment de l'annotation du texte, plutôt que de la parcourir directement au moment de l'annotation, notamment parce que nous allons pouvoir apporter des améliorations à la base existante, comme cela sera expliqué plus loin.

Pour enregistrer la base de données du TCAF, nous disposons de deux ressources :

— le site du TCAF (Okada et Ogurisu 2007), qui contient 266 pages correspondant aux 266 verbes recensés ; chaque page contient le tableau de conjugaison du verbe concerné, ainsi qu'une liste de verbes dérivés ;

— le document au format PDF *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012), qui contient la liste exhaustive des 266 verbes enregistrés dans cette base, ainsi que la liste, pour chacun de ces verbes, de toutes les formes fléchies connues qui ont pu être recensées, et enfin les verbes dérivés existants. Ces formes fléchies sont réparties par personne, temps et mode.

Nous avons choisi d'enregistrer les données au format XML. En effet, ce format présente beaucoup d'avantages par rapport à une base de données relationnelle (Hiller et Lipson 2009).

Premièrement, ce format est lisible directement, et donc facile à exploiter et à débugger.

Ensuite, le format XML est très flexible, il nous permettra beaucoup de libertés pour enregistrer les données sous la forme que l'on juge la plus adéquate.

5.1 Choix du jeu d'étiquettes d'annotation

Afin de pouvoir sauvegarder la base de données dans un fichier XML, il est important de commencer par choisir un jeu d'étiquettes d'annotation.

Nous nous sommes intéressés à deux jeux d'étiquettes différents : le jeu Cattex09max, et la version 2.0 de Morphalou.

5.1.1 Présentation du jeu d'étiquettes Cattex09max

Les étiquettes de Cattex09max sont composées d'un champ catégorie, d'un champ type (de la même façon que pour la version minimale de Cattex09), ainsi éventuellement que des champs mode, temps, personne, nombre, genre, cas ou degré. Ces champs ne s'appliquent pas tous à toutes les catégories.

Dans le cas des verbes, seuls les champs mode, temps, personne et nombre nous intéressent.

Les modes des verbes disponibles pour ce format sont les suivants : *ind* pour l'indicatif, *imp* pour l'imparfait, *con* pour le conditionnel, et *sub* pour le subjonctif.

Les temps disponibles sont les suivants : *pst* pour le présent, *ipf* pour l'imparfait, *fut* pour le futur, *psp* pour le passé simple.

Les personnes disponibles sont les suivantes : *1*, *2*, et *3* pour les première, deuxième et troisième personnes respectivement.

Enfin, les nombres disponibles sont les suivants : *s* pour le singulier, et *p* pour le pluriel.

Adaptation du jeu d'étiquettes à nos données

Nous remarquons cependant que deux temps vont manquer par rapport aux temps existants dans le document XML que nous avons construit, contenant la base de données du TCAF : le passé et le passé archaïque. Nous allons donc personnaliser ce jeu d'étiquettes, en ajoutant comme temps *pas* pour le passé, et *arp* pour le passé archaïque.

Ainsi, les types des étiquettes qui nous intéressent, ainsi que leurs correspondances avec les temps, les modes et les personnes du site du TCAF, sont décrites dans les tableaux 5.1 et 5.2 ci-dessous.

Temps et modes du TCAF	Temps et modes de Cattex09max
Indicatif Présent	ind pst
Indicatif Passé simple	ind psp
Indicatif Imparfait	ind ipf
Inficatif Passé archaïque	ind arp
Inficatif Futur	ind fut
Subjonctif Présent	sub pst
Subjonctif imparfait	sub ipf
Conditionnel	con
Impératif	imp

TABLE 5.1 – Tableau de correspondances entre les annotations de temps et de mode du TCAF et celles de Cattex09max

Personnes et nombres du TCAF	Personnes et nombres de Cattex09max
je	1 s
tu	2 s
il	3 s
nous	1 p
vous	2 p
ils	3 p

TABLE 5.2 – Tableau de correspondances entre les annotations de personnes et de nombres du TCAF et celles de Cattex09max

5.1.2 Morphalou

Présentation et description du contenu

Morphalou est « un lexique ouvert des formes fléchies du français » (*Morphalou* 2012), maintenu par l'ATILF. C'est un lexique très large, qui couvre environ 540 000 formes fléchies, « linguistiquement valide (sous la responsabilité d'un comité territorial) et formellement en accord avec les propositions de normalisation pour les ressources lexicales du TAL à l'ISO » (*Morphalou* 2012).

Plus précisément, il comporte 68 075 lemmes, et 539 413 formes fléchies ; il liste notamment toutes les formes fléchies d'un lemme. Il est disponible au format XML et est encodé en UTF-8.

La figure 5.1 présente un échantillon du lexique.

```
<lexicalEntry id="championne_1">
        <feminineVariantOf target="champion_1">champion</feminineVariantOf>
        <formSet>
                <lemmatizedForm>
                        <orthography>championne</orthography>
                        <grammaticalCategory>commonNoun</grammaticalCategory>
                        <grammaticalGender>feminine</grammaticalGender>
                </lemmatizedForm>
                <inflectedForm>
                        <orthography>championne</orthography>
                        <grammaticalNumber>singular</grammaticalNumber>
                </inflectedForm>
                <inflectedForm>
                        <orthography>championnes</orthography>
                        <grammaticalNumber>plural</grammaticalNumber>
                </inflectedForm>
        </formSet>
        <originatingEntry target="TLF">CHAMPION, ONNE, subst.</originatingEntry>
</lexicalEntry>
```

FIGURE 5.1 – Échantillon du lexique Morphalou 2.0 (*Morphalou* 2012)

Le lexique de Morphalou contient des entrées lexicales `<lexicalEntry>`, composées d'informations sur la forme de chaque lexie, sous la balise `<formSet>`. Celle-ci contient deux balises : la première, `<lemmatizedForm>`, contient le lemme de la lexie ; la deuxième, `<inflectedForm>`, contient les formes fléchies correspondantes. Ces deux dernières balises sont constituées de l'orthographe, de la catégorie grammaticale, et du genre grammatical dans le cas des noms. De plus, la balise `<inflectedForm>` comporte également le mode, le temps, le genre, la personne ou encore le nombre, selon la catégorie grammaticale de la forme.

Chaque forme différente pour un lemme sera ainsi enregistrée comme une nouvelle entrée lexicale, notamment les variantes orthographiques et morphologiques, les abréviations, ou les troncations.

Ce lexique est représenté sous le format Lexical Markup Framework (LMF) (Francopoulo et al. 2006). LMF est un modèle qui fournit un système commun standardisé pour la construction de lexiques de TAL (Francopoulo et al. 2006).

La figure 5.2 ci-dessous est une représentation du LMF implémenté pour le Morphalou.

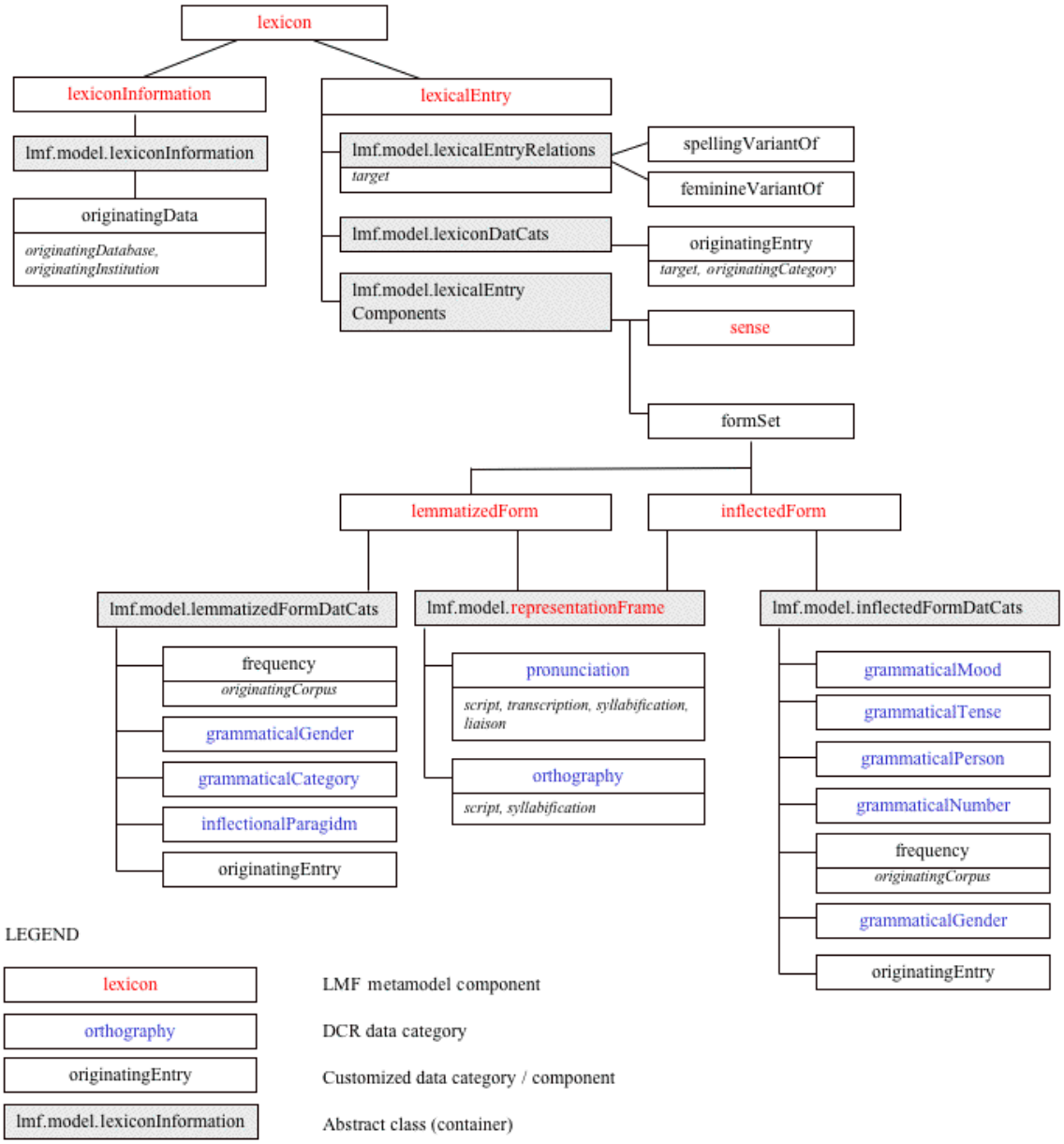


FIGURE 5.2 – Représentation du LMF implémenté pour Morphalou (*Morphalou* 2012)

Intérêt pour notre travail

Ce modèle est intéressant puisqu'il permet d'enregistrer toutes les formes (dont toutes les graphies) d'un lemme. Or, l'orthographe, la grammaire et la conjugaison n'étant pas figées en français médiéval, il est nécessaire de lister toutes les graphies possibles de chaque forme,

afin de constituer une base, non pas complète, mais s'en approchant le plus.

Nous pourrons ainsi nous servir des mots-clés existants du LMF :

— *subjunctive*, *indicative*, *conditional*, *imperative*, *participle* et *infinitive* pour annoter le mode de chaque forme verbale ;

— *present*, *past*, *simplePast*, *imperfect* et *future* pour annoter le temps de chaque forme verbale ;

— *firstPerson*, *secondPerson* et *thirdPerson* pour annoter la personne de chaque lexie ;

— *singular* et *plural* pour le nombre de chaque lexie.

Ce schéma ne permet cependant pas de couvrir toutes les formes verbales médiévales. En effet, le passé archaïque est un temps qui n'existe pas de nos jours, mais qui était utilisé en français médiéval ; nous avons donc ajouté un mot-clé *archaicPast* pour annoter ce temps.

Les tableaux 5.3 et 5.4 montrent les correspondances entre les étiquettes de Morphalou et les tiroirs verbaux du site du TCAF, ainsi que les étiquettes du jeu Cattex09max précédemment décrit.

Temps et modes du site du TCAF	Temps et modes de Morphalou	Temps et modes de Cattex09max
Infinitif	infinitive	
Participe Présent	participle present	
Participe Passé	participle past	
Indicatif Présent	indicative present	ind pst
Indicatif Passé simple	indicative simplePast	ind psp
Indicatif Imparfait	indicative imperfect	ind ipf
Indicatif Passé archaïque	indicative archaicPast	ind arp
Indicatif Futur	indicative future	inf fut
Subjonctif Présent	subjunctive present	sub pst
Subjonctif Imparfait	subjunctive imperfect	sub ipf
Conditionnel	conditional	con
Impératif	imperative	imp

TABLE 5.3 – Tableau de correspondances entre les annotations de temps et de mode du site du TCAF, celles de Morphalou et celles de Cattex09max

Personnes du site du TCAF	Personnes et nombres de Morphalou	Personnes et nombres de Cattex09max
je	firstPerson singular	1 s
tu	secondPerson singular	2 s
il	thirdPerson singular	3 s
nous	firstPerson plural	1 p
vous	secondPerson plural	2 p
ils	thirdPerson plural	3 p

TABLE 5.4 – Tableau de correspondances entre les annotations de personne du site du TCAF, celles de Morphalou et celles de Cattex09max

5.1.3 Choix du jeu d'étiquettes à utiliser

Il nous paraît plus judicieux d'utiliser le format de Morphalou, car ce jeu est beaucoup plus complet que Cattex09max. De plus, il a une hiérarchie de balises définies très précises, ce qui n'est pas le cas du format Cattex09max. Le format Morphalou sera donc très facile à utiliser, contrairement au format Cattex09max, qu'il faudrait beaucoup compléter afin de construire un document XML bien hiérarchisé.

5.2 Enregistrement de la base du TCAF

5.2.1 Description des ressources et problèmes rencontrés

Le but ici est d'enregistrer l'ensemble de la base des verbes du TCAF (Okada et Ogurisu 2007) dans un fichier XML au format précédemment défini.

Nous avons choisi d'utiliser en premier lieu le document *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012) pour extraire les données du TCAF. En effet, nous avons également la possibilité de faire des recherches sur le site du TCAF afin d'en extraire les données recherchées. Mais, pour cela, nous avons besoin de la liste exhaustive des verbes recensés, afin de pouvoir faire une recherche par verbe sur le site. Or, cette liste n'est pas disponible sur le site, elle est uniquement disponible dans le document *Tableaux de conjugaison de l'ancien français*. Nous devons donc enregistrer cette liste de verbes à partir de ce document, afin de pouvoir effectuer les recherches nécessaires sur le site. Nous avons donc décidé de tenter, en même temps, de récupérer l'ensemble des données depuis ce document.

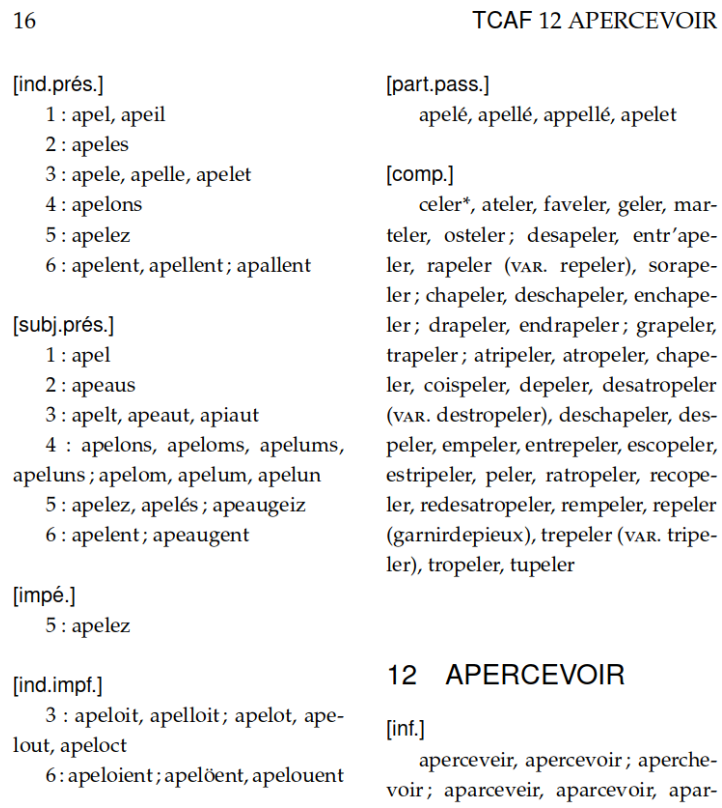
Mais cette démarche va poser plusieurs problèmes, que nous allons expliquer ensuite. Pour pallier ces problèmes, nous utiliserons le site du TCAF.

Nous avons donc tout d'abord voulu extraire les données du document *Tableaux de conjugaison de l'ancien français*. Nous avons essayé d'extraire automatiquement les données de ce document grâce à un script en Python, en utilisant notamment des librairies telles que `PDFMiner` (Shinyama 2014), qui sont des librairies permettant justement d'extraire le contenu de documents PDF vers d'autres formats, tels que le format texte ou le format HTML notamment.

Cependant, l'encodage de ce document ne nous a pas permis d'en extraire automatiquement des informations exploitables. En effet, les caractères accentués ne le restaient pas ; par exemple, le caractère « é » devenait « ´e ». De plus, ce mauvais encodage avait pour résultat entre autres de concaténer deux mots distincts, normalement séparés par un retour à la ligne ; de plus, les mots coupés sur deux lignes (et donc séparés par un tiret) restaient coupés.

Nous avons donc essayé de plus simplement copier le contenu du document, puis de le coller dans un fichier texte. Mais les problèmes décrits restaient présents. De plus, le document PDF étant présenté sous forme de colonnes, un problème supplémentaire fut donc que les données copiées ne respectaient pas la présentation initiale, comportant notamment les retours à la ligne.

La figure 5.3 montre un extrait de ce document, la figure 5.4 montre les données correspondantes qui ont été extraites automatiquement grâce à la librairie `PDFMiner`, et la figure 5.5 montre les données correspondantes qui ont été copiées et collées dans un éditeur de texte.



16 TCAF 12 APERCEVOIR

[ind.prés.]
1 : apel, apeil
2 : apeles
3 : apele, apelle, apelet
4 : apelons
5 : apelez
6 : apelent, apellent ; apallent

[subj.prés.]
1 : apel
2 : apeaus
3 : apelt, apeaut, apiaut
4 : apelons, apeloms, apelums, apeluns ; apelom, apelum, apelun
5 : apelez, apelés ; apeaugeiz
6 : apelent ; apeaugent

[impé.]
5 : apelez

[ind.impf.]
3 : apeloit, apelloit ; apelot, apelout, apeloct
6 : apeloient ; apelöent, apelouent

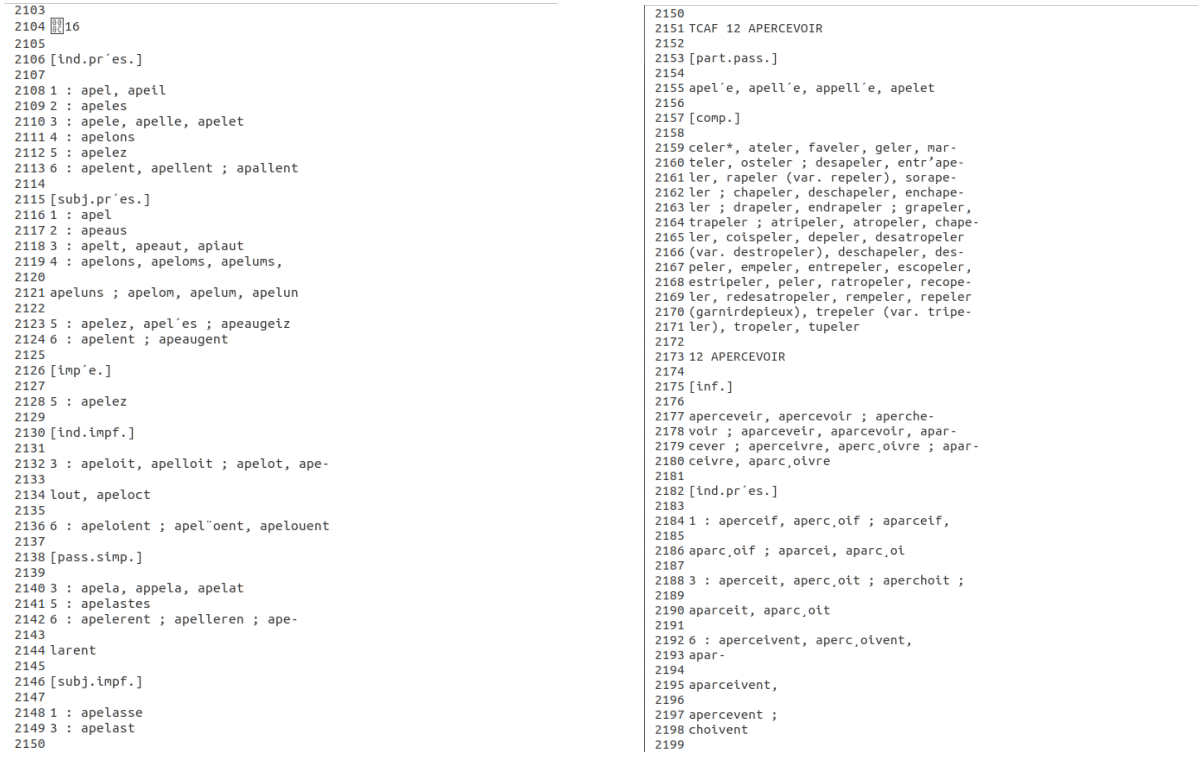
[part.pass.]
apelé, apellé, appellé, apelet

[comp.]
celer*, ateler, faveler, geler, marteler, osteler ; desapeler, entr'apeler, rapeler (VAR. repeler), sorapeler ; chapeler, deschapeler, enchapeler ; drapeler, endrapeler ; grapeler, trapeler ; atripeler, atropeler, chapeler, coispeler, depeler, desatropeler (VAR. destropeler), deschapeler, despeler, empeler, entrepeler, escopeler, estripeler, peler, ratropeler, recopeler, redesatropeler, rempeler, repeler (garnirdepieux), trepeler (VAR. tripeler), tropeler, tupeler

12 APERCEVOIR

[inf.]
aperceveir, apercevoir ; aperchevoir ; aparceveir, aparcevoir, apar-

FIGURE 5.3 – Extrait de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012)



```
2103
2104 16
2105
2106 [ind.pr´es.]
2107
2108 1 : apel, apeil
2109 2 : apeles
2110 3 : apele, apelle, apelet
2111 4 : apelons
2112 5 : apelez
2113 6 : apelent, apellent ; apallent
2114
2115 [subj.pr´es.]
2116 1 : apel
2117 2 : apeaus
2118 3 : apelt, apeaut, apiaut
2119 4 : apelons, apeloms, apelums,
2120
2121 apeluns ; apelom, apelum, apelun
2122
2123 5 : apelez, apel´es ; apeaugeiz
2124 6 : apelent ; apeaugent
2125
2126 [imp´e.]
2127
2128 5 : apelez
2129
2130 [ind.impf.]
2131
2132 3 : apeloit, apelloit ; apelot, ape-
2133
2134 lout, apeloct
2135
2136 6 : apeloient ; apel¨oent, apelouent
2137
2138 [pass.simp.]
2139
2140 3 : apela, appela, apelat
2141 5 : apelastes
2142 6 : apelerent ; apelleren ; ape-
2143
2144 larent
2145
2146 [subj.impf.]
2147
2148 1 : apelasse
2149 3 : apelast
2150
```

```
2150
2151 TCAF 12 APERCEVOIR
2152
2153 [part.pass.]
2154
2155 apel´e, apell´e, appell´e, apelet
2156
2157 [comp.]
2158
2159 celer*, ateler, faveler, geler, mar-
2160 teler, osteler ; desapeler, entr'ape-
2161 ler, rapeler (var. repeler), sorape-
2162 ler ; chapeler, deschapeler, enchape-
2163 ler ; drapeler, endrapeler ; grapeler,
2164 trapeler ; atripeler, atropeler, chape-
2165 ler, coispeler, depeler, desatropeler
2166 (var. destropeler), deschapeler, des-
2167 peler, empeler, entrepeler, escopeler,
2168 estripeler, peler, ratropeler, recope-
2169 ler, redesatropeler, rempeler, repeler
2170 (garnirdepieux), trepeler (var. tripe-
2171 ler), tropeler, tupeler
2172
2173 12 APERCEVOIR
2174
2175 [inf.]
2176
2177 aperceveir, apercevoir ; aperche-
2178 voir ; aparceveir, aparcevoir, apar-
2179 cever ; aperceivre, aperc¸oivre ; apar-
2180 celvre, aparc¸oivre
2181
2182 [ind.pr´es.]
2183
2184 1 : aperceif, aperc¸oif ; aparceif,
2185
2186 aparc¸oif ; aparcei, aparc¸oi
2187
2188 3 : aperceit, aperc¸oit ; aperchoit ;
2189
2190 aparceit, aparc¸oit
2191
2192 6 : apercelvent, aperc¸oivent,
2193 apar-
2194
2195 aparcelvent,
2196
2197 apercevent ;
2198 choivent
2199
```

FIGURE 5.4 – Extrait automatique de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012)

```
2
3 16TCAF12 APERCEVOIR[ind.pr´es.]1 : apel, apeil2 : apeles3 : apele, apelle, apelet4 : apelons5 : apelez6 : apelent, apellent ;
  apallent[subj.pr´es.]1 : apel2 : apeaus3 : apelt, apeaut, apiaut4 : apelons, apeloms, apelums,apeluns ; apelom, apelum, apelun5 :
  apelez, apel´es ; apeaugeiz6 : apelent ; apeaugent[imp´e.]5 : apelez[ind.impf.]3 : apeloit, apelloit ; apelot, ape-lout,
  apeloct6 : apeloient ; apel¨oent, apelouent[pass.simp.]3 : apela, appela, apelat5 : apelastes6 : apelerent ; apelleren ; ape-
  larent[subj.impf.]1 : apelasse3 : apelast[part.pass.]apel´e, apell´e, appell´e, apelet[comp.]celer*, ateler, faveler, geler, mar-
  teler, osteler ; desapeler, entr'ape-ler, rapeler (var. repeler), sorape-ler ; chapeler, deschapeler, enchape-ler ; drapeler,
  endrapeler ; grapeler,trapeler ; atripeler, atropeler, chape-ler, coispeler, depeler, desatropeler(var. destropeler), deschapeler,
  des-peler, empeler, entrepeler, escopeler,estripeler, peler, ratropeler, recope-ler, redesatropeler, rempeler,
  repeler(garnirdepieux), trepeler (var. tripe-ler), tropeler, tupeler12  APERCEVOIR[inf.]aperceveir, apercevoir ; aperche-voir ;
  aparceveir, aparcevoir, apar-
```

FIGURE 5.5 – Extrait copié de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012)

On constate bien que la mise en forme n'est pas respectée, ni les caractères accentués.

Il ne nous sera donc pas possible d'extraire une liste correcte, c'est-à-dire qui respecte la mise en forme et l'encodage du document, des formes de chaque verbe de ce document.

5.2.2 Extraction de la liste des verbes répertoriés

Ce document contient également la liste des 266 verbes recensés dans la base de données du TCAF, que nous allons extraire, afin de pouvoir les rechercher sur le site du TCAF directement.

Cette liste présentait évidemment les mêmes problèmes que le reste du document, mais respectait un format très précis. Ainsi, il a été simple de la nettoyer automatiquement, contrairement à la suite du document dont le comportement était parfois imprévisible.

Le script ayant permis ce nettoyage automatique est celui présenté sur la figure A.3 (page 74).

Pour ce faire, nous avons donc copié la table des matières de ce document dans un document texte. Nous avons immédiatement observé des patterns très précis dans le format des données. En effet, sur une même ligne, chaque élément était séparé par un espace ; par exemple, une ligne peut comprendre un verbe en majuscule, un numéro de ligne ou de page, ou des points. Comme chacun de ces éléments sont séparés par un espace, nous avons séparé chaque ligne selon les espaces. Ensuite, pour chaque élément ainsi séparé, nous avons vérifié s'il correspondait à l'un des éléments cités (un point, un nombre, ou un mot). Les mots trouvés étaient ensuite écrits en minuscule dans un nouveau fichier.

5.2.3 Extraction de la base du TCAF

Nous disposons ainsi de la liste exhaustive des verbes enregistrées dans la base de données du TCAF, et nous savons que le document PDF *Tableaux de conjugaison de l'ancien français* n'est pas exploitable pour extraire les formes verbales. Nous allons donc désormais utiliser le site du TCAF.

L'idée, pour extraire la base de données complète, est de faire une recherche sur le site du TCAF pour chaque verbe, puis d'extraire automatiquement les données de chaque page trouvée pour en sauvegarder les informations utiles (à savoir toutes les formes du verbe).

Pour cela, nous aurons besoin de deux librairies python principales : la librairie `requests`, qui permet de faire des requêtes selon une URL donnée ; et la librairie `BeautifulSoup`, qui permet de lire une page web et d'en extraire facilement des données.

Pour chaque verbe, le contenu de la page du TCAF correspondante est divisé en trois parties.

Les différentes lexies du verbe

Elles sont groupées par mode et par temps, en prenant en compte toutes les graphies connues. Le format des balises contenant ces lexies est celui décrit par la figure 5.6, qui est une capture d'écran du code source de la page du site du verbe *abregier*.

```
<div id="middle">
<div id="content">
      <div class="verb">ABREGIER</div>
            <div class="tense">Infinitif</div>
        <div class='conjugation'>abregier</div>
            <div class="tense">Indicatif Présent</div>
        <div class='conjugation'>(il) abriege<br />(vous) abregiez<br />(ils) abregent</div>
            <div class="tense">Indicatif Passé simple</div>
        <div class='conjugation'>(il) abreja</div>
            <div class="tense">Indicatif Futur</div>
        <div class='conjugation'>(nous) abrigerum</div>
            <div class="tense">Subjonctif Imparfait</div>
        <div class='conjugation'>(il) abregeast</div>
            <div class="tense">Participe Passé</div>
        <div class='conjugation'>abregié, abregied, abrejé; abrigié</div>
</div>
<div id="errata" style="display:none;">
</div>
<div id="debug" style="display:none;">
  </div>
</div>
```

FIGURE 5.6 – Format des lexies du verbe *abregier*

Nous avons donc utilisé la librairie `BeautifulSoup` pour accéder à la balise `<div id="content">`. Il nous a ensuite suffi d'accéder à toutes les balises contenues dans celle-ci.

Il a fallu respecter une contrainte : certaines pages, comme celle du verbe *bruire*, contiennent plusieurs verbes conjugués, sous la forme suivante présentée par la figure 5.7.

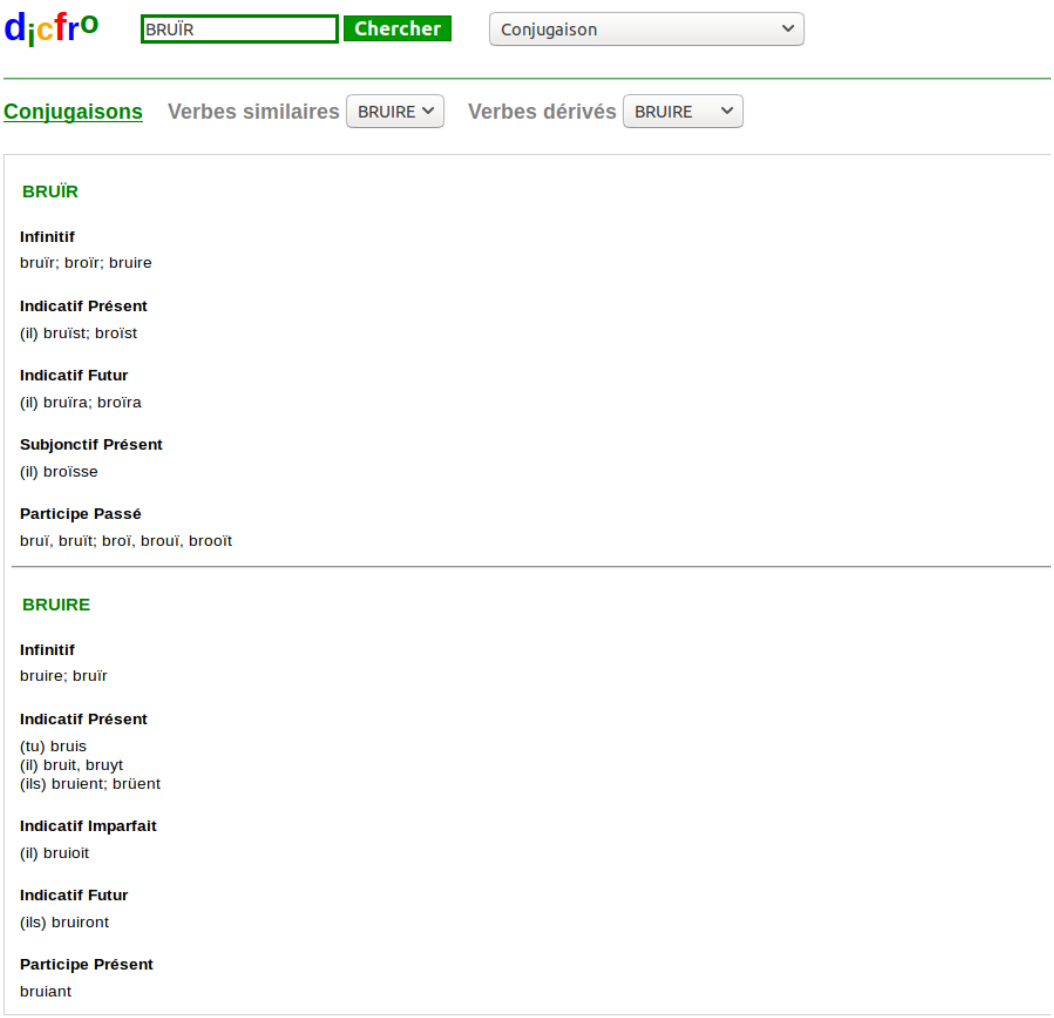


FIGURE 5.7 – Page du TCAF du verbe *bruire*

Ces verbes conjugués correspondent en réalité au même verbe, mais à des graphies différentes. Il a donc fallu enregistrer toutes ces graphies pour un même verbe.

Pour chaque verbe, chaque graphie a été enregistrée sous la forme d'un dictionnaire Python, la clé étant la forme de la graphie, la valeur étant elle-même un dictionnaire contenant le temps, et la personne dans les cas concernés. Les valeurs de temps et de personne correspondent aux mots-clés de Morphalou, décrits plus haut. Voici un exemple pour le verbe *abregier*, pour la forme *abriege*, qui est la troisième personne au singulier de l'indicatif présent :

```
{
  "abriege": {
```

```
    "tense": {"mood": "indicative", "tense": "present"},
    "person": {"person": "thirdPerson", "number": "singular"}
  }
}
```

Les verbes similaires

Cette partie est optionnelle. Ce sont des verbes qui ont généralement une terminaison commune avec le verbe, et qui ont eux-même leur propre page sur le TCAF ; ils ne sont donc pas intéressants pour nous car ils sont alors listés parmi les verbes que nous avons précédemment extraits. Le format des balises du TCAF pour ces verbes est celui présenté sur la figure 5.8, avec pour exemple la page du verbe *abregier*.

```
330
331   <a class="tab-selected" id="content-tab"
332      title="Définitions des mots"
333      href="javascript: updateTabContents('content-tab');" >Conjugaisons</a>
334
335
336 <span class="identified"
337       title="Tableaux de conjugaison de l'ancien français">
338   Verbes similaires</span>
339
340 <select class="identified"
341         onchange="searchWord('http://micmap.org/dicfro/search/tableaux-de-conjugaison/%s', this.value)">
342
343   <option>ABREGIER</option>
344
345
346     <optgroup label="ABREGIER">
347
348           <option value="assegier" >
349         assegier      </option>
350
351     </optgroup>
352
353
354 </select>
355
```

FIGURE 5.8 – Format des verbes similaires au verbe *abregier*

Nous avons donc ignoré ces balises lors de l'extraction de formes.

Les verbes dérivés

Cette partie est optionnelle. Ce sont des verbes qui ont une racine commune avec le verbe. Ils en diffèrent donc généralement par le préfixe, et éventuellement l'orthographe. Les différences d'orthographe sont déjà incluses parmi les différentes lexies, mais pas les différents préfixes. Il sera donc intéressant de connaître tous les préfixes possibles pour un même verbe, afin de reconstruire toutes les lexies possibles selon chaque préfixe. Le format des balises

concernant les verbes dérivés d'une page est celui de la figure 5.9, en prenant comme page d'exemple celle du verbe *abregier*.

```
355
356 <span class="identified"
357       title="Tableaux de conjugaison de l'ancien français">
358   Verbes dérivés</span>
359
360 <select class="identified"
361         onchange="searchWord('http://micmap.org/dicfro/search/tableaux-de-conjugaison/%s', this.value)">
362
363   <option>ABREGIER</option>
364
365
366     <optgroup label="ABREGIER">
367
368           <option value="gregier" >
369         gregier      </option>
370
371     </optgroup>
372
373
374 </select>
375
```

FIGURE 5.9 – Format des verbes dérivés du verbe *abregier*

Nous avons donc accédé à la balise `<span class="identified">` correspondant aux verbes dérivés (en omettant donc les verbes similaires, enregistrés avec les mêmes balises que les verbes dérivés), grâce à la librairie `BeautifulSoup`. Une fois cette balise trouvée, il a suffi d'accéder à la prochaine balise `<optgroup label="ABREGIER">` (dans le case du verbe *abregier*), qui contient toutes les balises `<option>` contenant chaque verbe dérivé.

Comme nous l'avons évoqué plus haut, il va être intéressant d'utiliser ces verbes dérivés afin de créer de nouvelles formes à partir de celles existantes.

En effet, autant les différences de graphies sont prises en compte dans la liste de graphies de la page, autant ce n'est pas le cas des différents préfixes. L'idée sera donc, pour chaque verbe dérivé, de lui associer des lexies construites à partir des lexies existantes pour le verbe de base. Par exemple, pour le verbe *abregier*, qui a pour dérivé le verbe *gregier*, le but sera de construire des formes correspondant au verbe *gregier* à partir des formes du verbe *abregier*.

Pour cela, nous avons construit une liste contenant les préfixes à remplacer, sous la forme de tuples Python, dont le premier élément est le préfixe à supprimer, et le deuxième élément est le préfixe à ajouter. C'est-à-dire, toujours avec l'exemple du verbe *abregier*, qui a pour seul verbe dérivé *gregier*, cette liste sera `[("ab", "g")]`, puisque leur racine commune est *regier*.

5.2.4 Enregistrement de la base de données

Une fois les données extraites du site du TCAF, il faut les enregistrer au format XML, avec comme jeu d'étiquettes celui de Morphalou que nous avons décrit plus haut. Pour cela, nous avons utilisé la librairie python `xml.etree.ElementTree` (*xml.etree.ElementTree — The ElementTree XML API — Python 3.7.3 documentation* 2019), qui permet d'écrire et mettre en forme des documents XML.

```
<lexicalEntry id="abregier">
    <formSet>
        <lemmatizedForm>
            <orthography>
                abregier
            </orthography>
            <grammaticalCategory>
                verb
            </grammaticalCategory>
        </lemmatizedForm>
        <inflectedForm>
            <orthography>
                abregier
            </orthography>
            <grammaticalMood>
                infinitive
            </grammaticalMood>
        </inflectedForm>
        <inflectedForm>
            <orthography>
                abriege
            </orthography>
            <grammaticalMood>
                indicative
            </grammaticalMood>
            <grammaticalTense>
                present
            </grammaticalTense>
            <grammaticalNumber>
                singular
            </grammaticalNumber>
            <grammaticalPerson>
                thirdPerson
            </grammaticalPerson>
        </inflectedForm>
        <inflectedForm>
            <orthography>
                abregiez
            </orthography>
            <grammaticalMood>
                indicative
            </grammaticalMood>
            <grammaticalTense>
                present
            </grammaticalTense>
            <grammaticalNumber>
                plural
            </grammaticalNumber>
            <grammaticalPerson>
                secondPerson
            </grammaticalPerson>
        </inflectedForm>
```

FIGURE 5.10 – Représentation XML du verbe *abregier*

```
<lexicalEntry id="gregier">
    <formSet>
        <lemmatizedForm>
            <orthography>
                gregier
            </orthography>
            <grammaticalCategory>
                verb
            </grammaticalCategory>
        </lemmatizedForm>
        <inflectedForm>
            <orthography>
                gregier
            </orthography>
            <grammaticalMood>
                infinitive
            </grammaticalMood>
        </inflectedForm>
        <inflectedForm>
            <orthography>
                griege
            </orthography>
            <grammaticalMood>
                indicative
            </grammaticalMood>
            <grammaticalTense>
                present
            </grammaticalTense>
            <grammaticalNumber>
                singular
            </grammaticalNumber>
            <grammaticalPerson>
                thirdPerson
            </grammaticalPerson>
        </inflectedForm>
        <inflectedForm>
            <orthography>
                gregiez
            </orthography>
            <grammaticalMood>
                indicative
            </grammaticalMood>
            <grammaticalTense>
                present
            </grammaticalTense>
            <grammaticalNumber>
                plural
            </grammaticalNumber>
            <grammaticalPerson>
                secondPerson
            </grammaticalPerson>
        </inflectedForm>
```

FIGURE 5.11 – Représentation XML du verbe *gregier*

Nous obtenons ainsi un fichier XML de 2 939 129 lignes, contenant 1 855 verbes différents et leurs différentes lexies, construits à partir des 266 verbes recensés sur le site du TCAF. Cela représente en tout 189 186 lexies.

Nous allons ainsi pouvoir nous servir de ces données afin d'annoter le texte.

Chapitre 6

Méthode utilisée pour réaliser l'outil d'annotation et de lemmatisation

Dans le chapitre suivant, nous allons expliquer notre démarche pour réaliser l'outil de lemmatisation et d'annotation du texte *La Queste del saint Graal*, à partir des données et outils décrits dans les parties précédentes.

6.1 Création du script d'étiquetage automatique et évaluation des résultats

À partir du fichier XML précédemment créé contenant les verbes, il va nous être possible d'annoter automatiquement les verbes du corpus étiqueté par LGeRM.

6.1.1 Première approche

Pour cela, nous allons nous servir de la librairie `xml.etree.ElementTree`, qui permet également de parcourir des documents XML et de faire des recherches dans des documents XML.

Le principe va donc être de parcourir le texte entier. Pour chaque mot du texte, si l'étiquette correspondante est une étiquette de verbe, c'est-à-dire *VERcjg*, *VERppe*, *VERppa*, ou *VERing*, alors on extraira la ou les formes correspondantes trouvées dans le fichier XML. Puis, pour chaque forme trouvée, il faudra également extraire le lemme (c'est-à-dire la forme infinitive correspondante), le temps, le mode et la personne de cette forme.

Pour cela, nous allons utiliser les expressions XPath, pour lesquelles `xml.etree.ElementTree` offre un support.

XPath permet de représenter un document XML comme une arborescence de nœuds. Il existe sept types de nœuds :

- root, qui correspond à la balise racine du document, dans notre cas `<lexicon>` ;
- element, qui correspondent aux balises autres que la racine, dans notre cas `<formSet>` ou `<lemmatizedForm>` par exemple ;
- attribute, qui correspondent aux attributs des balises, dans notre cas, l'attribut `id` de la balise `<lexicalEntry id="">` par exemple ;
- text, qui correspond au texte des balises ; dans notre cas, nous avons par exemple la balise `<grammaticalCategory>verb</grammaticalCategory>` ;
- comment, qui correspond à un commentaire ;
- processing instruction, qui correspond à un type de nœuds précis, qui porte des instructions pour l'application ;
- namespace, qui correspond à un espace de nom, qui contient l'ensemble des noms des balises pour ce document ; ces noms doivent être uniques.

Dans notre cas, seuls la racine, les éléments et les textes nous seront utiles.

Pour correspondre au format de Morphalou, cela signifie rechercher la forme verbale parmi les balises `<orthography>` en sauvegardant sa balise parent `<inflectedForm>` ou `<lemmatizedForm>`, et la balise parent de celle-ci `<formSet>`.

S'il s'agit d'une balise `<inflectedForm>`, nous pouvons accéder aux balises `<grammaticalMood>`, `<grammaticalTense>` et `<grammaticalPerson>`, positionnées au même niveau que la balise `<orthography>` trouvée. Ensuite, à partir de la balise supérieure `<formSet>`, nous pouvons accéder à la balise `<lemmatizedForm>`, qui nous permettra d'accéder à la forme infinitive, correspondant à la balise enfant `<orthography>`.

S'il s'agit d'une balise `<lemmatizedForm>`, nous accédons à la balise `<orthography>` trouvée.

La méthode qui permet d'obtenir toutes ces informations est celle présentée sur la figure A.4 (page 74).

Ces données sont ensuite enregistrées dans un dictionnaire python, appelé `inflection` dans le script, de la forme suivante :

```
{
  "grammaticalMood": "",
  "grammaticalTense": "",
```

```
  "grammaticalPerson": "",
  "lemmatizedOrthography": ""
}
```

Le but de l'algorithme sera alors d'appeler cette méthode pour chaque forme fléchie trouvée dans le fichier XML, et d'enregistrer tous les résultats dans une liste python, que nous avons appelée `found`, que nous trions ensuite pour en enlever les éléments vides. Cette méthode est visible sur la figure A.5 (page 75).

La méthode principale est montrée à la figure A.6 (page 76). Cette méthode parcourt entièrement le texte, et, pour chaque forme verbale rencontrée, cherche toutes ses flexions dans le fichier XML, grâce aux méthodes `get_inflections` et `get_inflection` décrites dans les figures A.4 (page 74) et A.5 (page 75).

6.1.2 Résultats

Cette approche très simple a permis de faire correspondre 17 543 verbes à au moins une forme dans le fichier XML des verbes, ce qui correspond à 77,96% des verbes du texte (le texte contient en tout 22 502 verbes). Les autres verbes n'ont aucune correspondance dans le fichier XML.

Parmi ces 17 543 verbes trouvés :

— 14 369 sont annotés comme des verbes conjugués, soit 81,91% des verbes annotés, 85,45% des verbes conjugués, et 63,86% de l'ensemble des verbes du texte ;

— 1 759 sont annotés comme des verbes à l'infinitif, soit 10,03% des verbes annotés, 76,05% des verbes à l'infinitif, et 7,82% de l'ensemble des verbes du texte ;

— 118 sont annotés comme des verbes au participe présent, soit 0,67% des verbes annotés, 52,21% des verbes au participe présent, et 0,52% de l'ensemble des verbes du texte ;

— 1 297 sont annotés comme des verbes au participe passé, soit 7,39% des verbes annotés, 41,2% des verbes au participe passé, et 5,76% de l'ensemble des verbes du texte.

Ce score est plutôt élevé, mais il existe des moyens de le faire augmenter.

6.2 Amélioration des résultats

6.2.1 Deuxième approche

Il va nous être possible d'améliorer un peu ces résultats, en nous basant pour l'instant uniquement sur les données contenues dans le fichier du texte, c'est-à-dire sur les données

déjà travaillées par l'équipe de LGeRM.

En effet, nous avons pour l'instant utilisé comme seule information la catégorie de l'étiquette de TXM, c'est-à-dire, lors du parcours du texte, nous vérifiions seulement si l'étiquette Cattex09 de chaque mot rencontré commençait par la chaîne de caractères *VER*. Nous ne nous servions pas du type associé à cette catégorie, qui peut être *inf*, *cjg*, *ppe* ou *ppa*, comme nous l'avons vu lors de la présentation du jeu d'étiquettes Cattex09. Nous allons ainsi pouvoir nous servir de ces informations pour améliorer les résultats.

De plus, le ou les lemmes trouvés par LGeRM correspondant à chaque forme verbale pourront nous être utiles, en complément du type de l'étiquette de TXM.

Nous allons donc pouvoir nous baser là-dessus afin d'améliorer les résultats de notre première approche.

Amélioration sur les verbes à l'infinitif

Ainsi, pour les verbes étiquetés à l'infinitif, nous pouvons ajouter comme possibilités aux annotations le verbe du texte, ainsi que son ou ses lemmes proposés par LGeRM.

Par exemple, pour la forme *essaier* présente dans le texte, correspondant à la ligne `essaier; ASSAYER|ESSAYER; verbe|verbe; VERinf` du fichier CSV, nous pouvons ajouter les infinitifs *essaier*, *assayer* et *essayer* aux annotations trouvées grâce au fichier XML.

Cela permettra notamment d'obtenir un résultat si jamais le verbe en question n'a pas été trouvé dans le fichier XML.

Amélioration sur les verbes au participe passé et présent

Pour les verbes au participe passé et au participe présent, comme les annotations obtenues à partir du fichier XML créé à partir du TCAF n'apportent pas d'information supplémentaire que le lemme, nous pouvons alors ajouter comme possibilité le et les lemmes proposés par LGeRM.

Ainsi, dans le cas de la forme *venu* présente dans le texte, et correspondant à la ligne `venu; VENIR; verbe; VERppe` du fichier CSV, nous pouvons ajouter le participe passé du verbe *venir* comme possibilité d'annotation, en plus des annotations obtenues par le fichier XML.

Le script implémentant ces deux améliorations est visible sur l'image A.7 (page 77). Cela a permis d'améliorer l'annotation de 5 687 verbes du texte, dont 2 513 qui n'avaient pas

été trouvés dans le fichier XML. Ces 2 513 représentent 50,67% des verbes non trouvés à la première étape, et 11,17% de la totalité des verbes du texte.

En tout, on atteint les 89,13% de verbes annotés dans le texte, en comptant les verbes qui ont été trouvés directement dans le fichier XML et ceux qui ont pu être améliorés.

Amélioration sur l'ordre des suggestions

Nous allons également nous servir des annotations existantes pour déterminer un ordre dans les suggestions d'annotations des verbes. En effet, nous allons proposer en premier les annotations qui correspondent à l'étiquette existante proposée par LGeRM.

Cette amélioration n'augmentera évidemment pas le nombre de verbes annotés, mais permettra une meilleure suggestions des annotations déjà trouvées.

Nous avons décidé de garder également les annotations dont l'annotation ne correspond pas à l'étiquette, afin de garder en mémoire toutes les possiblités, mais celles-ci seront proposées en dernier, après les annotations correspondant à l"étiquette.

Par exemple, si un verbe a comme étiquette *VERppa*, et si les annotations trouvées à partir du fichier XML correspondent respectivement à un verbe conjugué et à un verbe au participe présent, alors la première annotation qui sera proposée sera celle qui correspond au participe présent, comme indiqué par l'étiquette.

Le script permettant de réaliser cet ordre est disponible sur la figure A.8 (page 77).

6.2.2 Troisième approche

Nous nous étions précédemment intéressés à l'outil LGeRM (Souvay et Pierrel 2009), en ligne sur le site de l'ATILF (ATILF - CNRS & Université de Lorraine 2019), qui permet de proposer des lemmes possibles pour n'importe quelle forme médiévale. Cet outil, comme expliqué plus haut, utilise des règles afin d'essayer de retrouver des formes connues, et ainsi de mettre en relation la forme recherchée avec un lemme. Il utilise en particulier des règles morphologiques sur la flexion verbale. Ces règles sont très intéressantes dans le cas de notre travail, puisqu'elles pourraient être utilisées sur les formes qui n'ont pas été annotées grâce aux deux approches précédentes. Cela permettrait de faire correspondre certaines graphies inconnues de notre fichier XML contenant la base de données du TCAF à des graphies connues, et ainsi améliorer encore les résultats.

Cependant, nous n'avons pas réussi à nous procurer ces règles. Ce travail n'est donc pas possible. Il ne nous est pas non plus possible de faire une requête sur l'outil en ligne

directement, puisque cet outil permet uniquement de chercher le lemme de la forme, et non pas le temps, le mode ou la personne, comme nous voulons le faire.

6.3 Enregistrement des résultats

Nous avons ainsi obtenu les annotations en mode, temps, personne et nombre des formes verbales du corpus étiqueté par LGeRM. Nous devons donc à présent sauvegarder ces annotations.

6.3.1 Choix du format de sauvegarde des données

Comme nous l'avons détaillé plus haut, les étiquettes pré-existantes du corpus étiqueté par LGeRM était enregistrées au format Cattex09. Il nous a donc semblé logique d'utiliser le format Cattex09max (Guillot et al. 2013b) afin de sauvegarder nos nouvelles étiquettes.

Disposant de peu d'informations sur ce format d'étiquettes, et n'ayant trouvé aucun exemple, nous n'avons pas su comment ces différents champs doivent être écrits, c'est-à-dire que nous n'avons pas su si tous ces champs doivent être simplement concaténés lors de l'écriture de l'étiquette. Cela nous a cependant paru très difficile à lire ; nous avons donc décidé de séparer tous les champs par le caractère « / », afin de faciliter la lecture.

Nous devons également intégrer le lemme à nos annotations, mais aucun champ du jeu Cattex09max ne permet cela ; nous avons donc décidé de simplement ajouter le lemme à la fin de l'étiquette, en le séparant de la même façon avec le caractère « / ».

Nous avons donc dû adapter le format de Morphalou, utilisé jusqu'ici, au format Cattex09max. Nous avons représenté les correspondances entre ces deux formats en Python par des dictionnaires, un pour le mode, un pour le temps, un pour la personne et un pour le nombre, qui ont pour clés l'étiquette de Morphalou, et pour valeurs le type de l'étiquette Cattex09max correspondante.

Enfin, nous avons décidé de sauvegarder ces étiquettes dans un fichier CSV contenant cinq colonnes. Les quatre premières colonnes contiennent les mêmes données que celles du corpus étiqueté par LGeRM, c'est-à-dire dans l'ordre, le terme du texte, les lemmes trouvés par l'outil LGeRM, les étiquettes TreeTagger trouvées par LGeRM, et l'étiquette Cattex09 du terme provenant du TXM. Enfin, la cinquième colonne contient le travail que nous avons effectué, c'est-à-dire les différentes étiquettes au format Cattex09max décrit, séparées par le caractère « | » pour décrire l'ambiguité de la même façon qui est faite pour les colonnes 2 et 3.

Par exemple, pour la forme *voloit*, nous obtenons la ligne `voloit; VOLER|VOULOIR; verbe|verbe; VERcjg; VERcjg/ind/ipf/3/s/voldre|VERcjg/ind/ipf/3/s/voloir`. Cette ligne signifie donc que nous avons deux annotations possibles. La première indique que c'est un verbe conjugué à l'indicatif imparfait à la troisième personne du singulier, avec comme lemme l'infinitif *voldre*. La deuxième indique que c'est un verbe conjugué à l'indicatif imparfait à la troisième personne du singulier, avec comme lemme l'infinitif *voloir*.

Cet exemple nous permet de nous rendre compte de plusieurs problèmes.

Tout d'abord, on remarque que les formes que nous obtenons par notre méthode correspondent aux formes infinitives « voldre » et « voloir ». « Voloir » correspond effectivement à l'un des lemmes possibles déterminés par LGeRM (« vouloir »). Cependant, le lemme « voler », déterminé par LGeRM, n'a pas été trouvé. En recherchant « voler » sur le site du TCAF, nous remarquons que le résultat correspond à la page de la forme « voloir », dont l'une des formes infinitives est effectivement « voler », comme montré sur la figure 6.1. Cette erreur vient donc du fait que les différents tableaux de conjugaison recensés sur le site du TCAF ne sont peut-être pas assez distincts ou séparés les uns des autres ; dans le cas de cet exemple, un même tableau de conjugaison recense en réalité plusieurs verbes différents, ainsi que beaucoup de leurs graphies possibles. Il nous est donc impossible de faire la séparation de ces verbes par nous-mêmes.

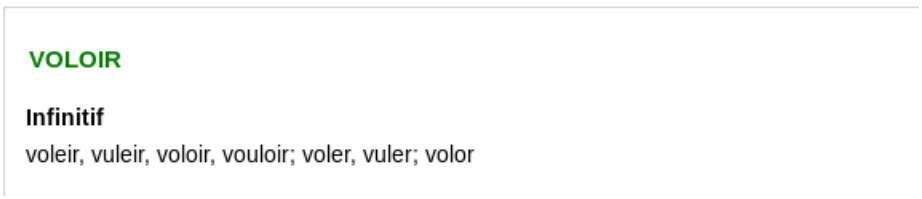


FIGURE 6.1 – Résultat de la recherche de « voloir » sur le site du TCAF

Ensuite, nous remarquons que la forme « voldre » que nous obtenons n'est pas présente parmi les suggestions de LGeRM. En recherchant « voldre » sur le site du TCAF, nous constatons que la page de résultats contient le tableau de conjugaison de « voldre », ainsi que celui de « voloir », comme montré sur la figure 6.2.

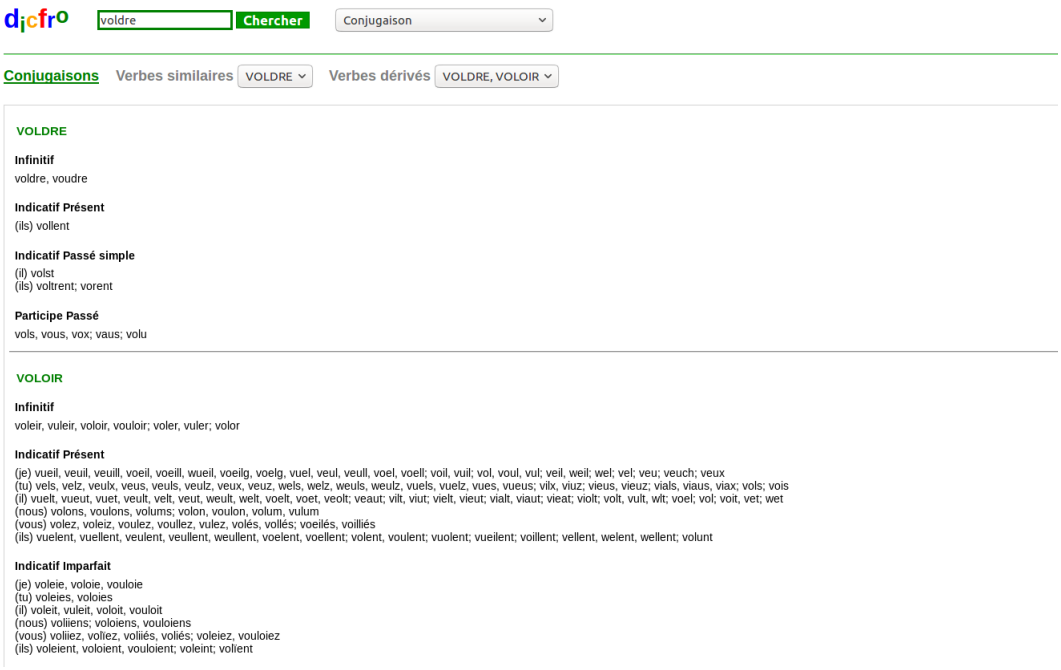


FIGURE 6.2 – Résultat de la recherche de « voldre » sur le site du TCAF

Or, comme nous l'avons expliqué plus haut, nous avons regroupé les tableaux de conjugaison des verbes se trouvant sur la même page dans notre fichier XML. Notre système ne fait donc pas de distinction entre « voldre » et « voloir », ce qui est une erreur dans ce cas.

Nous avons utilisé le script de la figure A.9 (page 78) afin de réaliser ces étiquettes.

Nous obtenons ainsi un fichier CSV au format présenté sur la figure 6.3.

55	le	LE	pron. pers.	PROper	
56	pot	POUVOIR2	verbe	VERcjg	VERcjg/ind/psp/3/s/paistre|VERcjg/ind/pst/3/s/pöoir|VERcjg/ind/psp/3/s/pöoir|VERppe/pondre
57	l'en	ON	pron. pers.	PROind	
58	veoir	VOIR1	verbe	VERinf	VERinf/veoir|VERinf/voir
59	,	_	ponctuation	(null)	
60	car	CAR	conj.	CONcoo	
61	ses	SON4	poss.	DETpos	
62	chevaux	CHEVAL	subst. masc.	NOMcom	
63	en	EN2	pron. adv.	PROadv	
64	fu	ÊTRE1	verbe	VERcjg	VERcjg/ind/psp/1/s/estre|VERcjg/ind/psp/3/s/estre|VERcjg/sub/ipf/3/s/estre
65	encore	ENCORE	adv.	ADVgen	
66	toz	TOUS|TOUT2	indéf.|indéf., adv. d'intensité	ADVgen	
67	suanz	SUER1	verbe	VERppa	VERppa/suer
68	,	_	ponctuation	(null)	
69	et	ET	conj. de coord.	CONcoo	
70	ele	ELLE1	pron. pers.	PROper	
71	descent	DESCENDRE	verbe	VERcjg	VERcjg/ind/pst/1/s/descendre|VERcjg/ind/pst/3/s/descendre
72	et	ET	conj. de coord.	CONcoo	
73	vient	VENIR	verbe	VERcjg	VERcjg/ind/pst/3/s/venir
74	devant	DEVANT	adv., prép. et subst. masc.	PRE	
75	le	LE	art. déf.	DETdef	
76	roi	RETS|ROI1	subst. masc. et fém.|subst. masc.	NOMcom	
77	si	SI4	adv.	ADVgen	
78	le	LE	pron. pers.	PROper	
79	salue	SALUER	verbe	VERcjg	
80	,	_	ponctuation	(null)	
81	et	ET	conj. de coord.	CONcoo	
82	il	IL	pron. pers.	PROper	
83	dist	DIRE1	verbe	VERcjg	VERcjg/ind/pst/3/s/dire|VERcjg/ind/psp/3/s/dire|VERcjg/sub/ipf/3/s/dire|VERppe/dire
84	que	QUE	conj., rel. interr.	CONsub	
85	Diex	µDIEU	nom propre	NOMpro	
86	la	LE	pron. pers.	PROper	
87	beneïe	BÉNIR	verbe	VERcjg	VERcjg/sub/pst/1/s/beneïr|VERcjg/sub/pst/3/s/beneïr|VERcjg/imp/2/s/beneïr
88	.	_	ponctuation	(null)	
89	«	_	ponctuation	(null)	
90	Sire	CIRE|SIRE	subst. fém.|subst. masc.	NOMcom	
91	,	_	ponctuation	(null)	
92	fet	FAIRE	verbe	VERcjg	VERcjg/ind/pst/3/s/faire|VERppe/faire
93	ele	ELLE1	pron. pers.	PROper	
94	,	_	ponctuation	(null)	
95	por	POUR	prép.	PRE	
96	Dieu	µDIEU	nom propre	NOMpro	
97	dites	DIRE1	verbe	VERcjg	VERcjg/ind/pst/2/p/dire|VERcjg/imp/2/p/dire
98	moi	MOI	pron. pers.	PROper	

FIGURE 6.3 – Corpus annoté au format Cattex09max

6.4 Résultats de cette méthode

6.4.1 Calcul de la précision

Pour évaluer cette méthode, nous avons calculé la précision au rang n sur un échantillon de 2 000 verbes ayant été annotés par notre méthode.

Ce calcul consiste, pour chaque verbe que nous avons annoté automatiquement, à déterminer, pour chaque annotation ordonnée possible, si elle est correcte ou non, et à appliquer la formule $P(N) = \frac{n}{N}$, avec N le rang auquel on veut calculer la précision, et n le nombre d'annotations correctes à ce rang.

Nous considérerons qu'une annotation est correcte selon deux critères.

En premier lieu, la catégorie et le type de l'annotation doivent correspondre à ceux de l'étiquette provenant de TXM. En effet, nous pouvons considérer que cette étiquette est correcte, puisqu'elle a été corrigée à la main.

Ensuite, le lemme trouvé par notre méthode doit correspondre à l'un des lemmes proposés par LGeRM, à l'orthographe près, en utilisant le site du TCAF comme support afin de vérifier l'orthographe.

Prenons la ligne suivante comme exemple : `oï; OUÏR; verbe; VERppe; VERppe/oïr|`

`VERppe/ouïr| VERcjg/ind/psp/1/s/oïr| VERcjg/ind/psp/3/s/oïr`. Nous pouvons calculer la précision au rang n de la façon suivante.

Quatre annotations ont été trouvées, qui sont dans l'ordre : `VERppe/oïr`, `VERppe/ouïr`, `VERcjg/ind/psp/1/s/oïr`, et `VERcjg/ind/psp/3/s/oïr`.

En ce qui concerne la première, la catégorie et le type (*VERppe*) correspondent bien à l'étiquette venant du TXM. De plus, le lemme *oïr* trouvé correspond bien, selon le TCAF, au lemme *ouïr* trouvé par LGeRM. La première annotation est donc correcte. On a donc $n = 1$ (une réponse correcte au rang actuel), et $N = 1$ (le rang actuel). Ainsi, $P(1) = 1$.

De la même façon, nous obtenons $P(2) = 1$, puisque $n = 2$ et $N = 2$.

En ce qui concerne la troisième annotation, la catégorie et le type (*VERcjg*) ne correspondent pas à ceux provenant de TXM (*VERppe*). Cette annotation est donc fausse. On a alors $N = 3$ et $n = 2$, ce qui donne $P(3) = \frac{2}{3}$.

Enfin, de la même façon, nous obtenons $P(4) = \frac{1}{2}$, puisque $n = 2$ et $N = 4$.

Ainsi, sur l'échantillon de 2 000 verbes, nous obtenons les précisions suivantes :

$P(1) = 0.95$, $P(2) = 0.79$, $P(3) = 0.75$, $P(4) = 0.74$, $P(5) = 0.78$, $P(6) = 0.73$, $P(7) = 0.73$, $P(8) = 0.75$

De plus, la précision sur l'ensemble des résultats est de 0.85.

Nous constatons que la précision au rang 1 est très élevée, et qu'elle diminue et est à peu près constante à partir du deuxième rang.

6.4.2 Erreurs et verbes non annotés

Comme nous l'avons vu précédemment, il existe deux types d'erreurs d'annotation possibles parmi les verbes annotés : une erreur de type, en se référant au type de TXM ; une erreur de lemme, selon les lemmes proposés par LGeRM. Cela signifie que les verbes présentant ces erreurs d'annotation possèdent des homographes.

En ce qui concerne les verbes n'ayant pas été annotés par notre méthode, il en existe deux types :

— les verbes dont la forme est absente de la base de données du TCAF, mais dont le tableau de conjugaison est présent ;

— les verbes dont la forme ainsi que le tableau de conjugaison sont absents du TCAF.

On constate donc que les annotations manquantes à notre méthode sont entièrement dues au fait que la base de verbes du TCAF est incomplète. L'utilisation de règles sur la flexion verbale pourrait donc permettre d'étendre cette base, et ainsi d'augmenter le nombre de verbes annotés.

Chapitre 7

Conclusion

Nous avons donc étudié le fonctionnement de plusieurs outils numériques permettant l'annotation de corpus ou la recherche dans un dictionnaire. Nous constatons ainsi qu'ils ne peuvent pas se contenter d'utiliser une base de graphies, puisqu'il est impossible d'avoir une base exhaustive, du fait des variations graphiques du français médiéval. Une solution pour pallier ce problème est donc d'utiliser des règles, afin de construire des graphies supplémentaires. C'est ce que font LGeRM, et Flemm dans le cas du français moderne.

Pour construire un outil permettant d'étiqueter efficacement les formes verbales du français médiéval, il a donc fallu réaliser une base de graphies, contenant les conjugaisons connues de verbes, ainsi que déterminer les règles nécessaires pour retrouver toutes les formes rencontrées dans un texte.

La base de graphies du TCAF peut servir de support à cette base de graphies, puisqu'elle est assez longue.

Nous avons donc pu annoter 89,13% des formes verbales du texte de *La Queste del saint Graal*, dont tous les verbes à l'infinitif, tous les verbes au participe présent et au participe passé, et 85,45% des verbes conjugués.

Ce score est assez élevé, mais il pourrait encore être augmenté, comme nous l'avons dit plus haut, par l'utilisation des règles morphologiques sur la flexion verbale utilisées par l'outil LGeRM. Il faudrait pour cela se procurer ces règles, ou bien en déterminer d'autres qui seraient adaptées, et appliquer ces règles aux formes verbales qui n'ont pas été trouvées dans le fichier XML contenant la base de données du TCAF.

L'algorithme serait alors de vérifier si les formes générées sont connues. Si c'était le cas, alors cela voudrait dire que des annotations possibles ont été trouvées. Sinon, il faudrait réappliquer les règles aux nouvelles formes créées, et ainsi de suite, jusqu'à ce que plus

aucune nouvelle forme ne soit générée, ou jusqu'à ce que trop de formes aient été générées, en déterminant un seuil maximum acceptable de nombre de formes à générer.

De plus, les annotations de formes extraites de notre fichier XML pourraient encore être améliorées en corrigeant l'erreur évoquée dans la partie précédente, c'est-à-dire en séparant les verbes présents sur une même page, comme dans le cas de « voloir » et « voldre ».

Nous avons enregistré tous les fichiers et documents utilisés lors de ce travail dans le dépôt GitLab à l'adresse `https://gitlab.com/clara.bringer96/memoire_m2`.

Ce dépôt contient deux dossiers.

Le premier intitulé « redaction » contient la bibliographie au format BibTex, utilisée dans ce document, un dossier contenant les images utilisées dans ce document, ce document au format tex et au format PDF.

Le deuxième intitulé « developpement » contient un dossier correspondant aux fichiers et scripts utilisés pour la partie sur la constitution et le choix des corpus, un dossier correspondant aux fichiers et scripts utilisés pour enregistrer la base de données du site du TCAF, et enfin un dossier correspondant au fichiers et scripts utilisés pour annoter le corpus choisi.

Annexe A

Scripts développés et utilisés

get_corpus_TXM.py

```python
import csv
import re
from pathlib import Path

from bs4 import BeautifulSoup

tag_re = re.compile(r"<\w*>(\w+)")
compiler = re.compile(r"([a-zA-Zéçï]*)[\-0-9]+([a-zA-Zéçï]*)")
end_cut = re.compile(r".+-$")
brackets = re.compile(r"(.*)\[(.*)\](.*)")


def clean_span(text):
    b = brackets.search(text)
    text = f"{b.group(1)}{b.group(2)}{b.group(3)}" if b else text

    c = compiler.search(text)
    text = f"{c.group(1)}{c.group(2)}" if c else text

    return text


def read_html(file, writer, start):
    with open(file, "r", encoding="utf8") as html:
        soup = BeautifulSoup(html, "html.parser")
        span_class = soup.find_all("span", attrs={"class": "word"})

        for span in span_class:
            s = tag_re.search(span.text)
            text = s.group(1) if s else span.text

            if len(start) > 0:
                text = start + text
                start = ""
            if end_cut.match(text):
                start = text[:-1]
            else:
                row = [clean_span(text), span["title"]]
                writer.writerow(row)

    return start


def read_rep(rep, output_file):
    with open(str(output_file), "w", newline="") as output:
        writer = csv.writer(output, quotechar="\"", delimiter=";")
        files_path = Path(rep)
        if files_path.exists() and files_path.is_dir():
            start = ""
            for f in sorted(files_path.iterdir()):
                print(f)
                start = read_html(f, writer, start)

```

FIGURE A.1 – Script d'extraction et d'enregistrement des données du corpus TXM

statistics.py

```python
import csv
import re
from pathlib import Path

current = Path.cwd()
file_txm_csv = current / "TXM" / "annotated_corpus.csv"
file_lgerm_csv = current / "LGeRM" / "utf8_qgraal_LGeRM.csv"

is_verb = re.compile(r"^VER")


def get_numbers_text(file, index_tag):
    nb_words = 0
    verbs = dict()
    nb_verbs = 0
    other_tags = set()

    with open(str(file), newline="") as csv_file:
        reader = csv.reader(csv_file, delimiter=";", quotechar=None)

        for line in reader:
            if len(line[0]) > 0:
                nb_words += 1
                tag = line[index_tag]
                if is_verb.match(tag) is not None:
                    nb_verbs += 1
                    if tag not in verbs:
                        verbs[tag] = [0, set()]
                    verbs[tag][0] += 1
                    verbs[tag][1].add(line[0].lower())
                else:
                    other_tags.add(line[index_tag])

    return nb_words, verbs


def get_numbers_txm():
    return get_numbers_text(file_txm_csv, 1)


def get_numbers_lgerm():
    return get_numbers_text(file_lgerm_csv, 3)


def get_number_verbs(verbs):
    return sum([v[0] for t, v in verbs.items()])


def get_different_verbs(verbs):
    return {f for t, v in verbs.items() for f in v[1]}


def get_number_verbs_tag(verbs):
    return {tag: [v[0], len(v[1])] for tag, v in verbs.items()}


def get_differences(verbs_lgerm, verbs_txm):
    set_verbs_lgerm = get_different_verbs(verbs_lgerm)
    set_verbs_txm = get_different_verbs(verbs_txm)

    diff_lgerm_txm = set_verbs_lgerm - set_verbs_txm
    diff_txm_lgerm = set_verbs_txm - set_verbs_lgerm

    return diff_lgerm_txm, diff_txm_lgerm
```

FIGURE A.2 – Script permettant d'obtenir des chiffres sur le contenu des corpus

table_contents.py

```python
def read_table_contents():
    """
    Read the table of content copied from the PDF document, format it and rewrite it in a new text document
    :return: None
    """
    # File containing the table of content copied from the PDF document
    with open("tcaf_table_contents_to_format.txt", "r", encoding="utf8") as f:
        # File to write the formatted verion of the table of content
        with open("tcaf_table_contents_formated.txt", "w", encoding="utf8") as output:
            lines = f.readlines()
            for i in range(len(lines)):
                line = lines[i].split(" ")
                new_line = ""
                for j in range(len(line)):
                    is_number = False
                    is_parenthesis = False
                    for char in line[j]:
                        if '0' <= char <= '9':
                            is_number = True
                        if char == '(' or char == ')':
                            is_parenthesis = True
                    is_dot = line[j] == '.'

                    if not is_dot and not is_number and not is_parenthesis:
                        for char in line[j]:
                            if char != '\n' and char != '.':
                                new_line += char

                output.write(new_line.lower() + ("\n" if new_line != "" else ""))

```

FIGURE A.3 – Script utilisé pour formatter la table des matières de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012)

```python

current = Path.cwd()
morphalou_like_file = current / ".." / "bdd_tcaf" / "morphalou_like_bdd.xml"
verbs_tree = ET.parse(str(morphalou_like_file))

is_verb = re.compile(r"^VER")
get_inf = re.compile(r"(.+)\d+$")

```

```python

def get_inflection(form_set: Element, inflected_form: Element) -> Dict[str, str]:
    """
    Get the information for one inflection of a verb
    :param form_set: formSet tag found in the XML database
    :param inflected_form: inflectedForm tag found in the XML database
    :return: dictionary containing the mood, the tense, the person and the lemma of the inflection
    """
    lemmatized_form_tag = form_set.find(lemmatized_form)

    inflection = dict()

    mood_found = inflected_form.find(grammatical_mood)
    if mood_found is not None:
        inflection[grammatical_mood] = mood_found.text.strip()
    tense_found = inflected_form.find(grammatical_tense)
    if tense_found is not None:
        inflection[grammatical_tense] = tense_found.text.strip()
    person_found = inflected_form.find(grammatical_person)
    if person_found is not None:
        inflection[grammatical_person] = person_found.text.strip()
    number_found = inflected_form.find(grammatical_number)
    if number_found is not None:
        inflection[grammatical_number] = number_found.text.strip()

    if len(inflection) > 0:
        inflection[lemmatized_orthography] = lemmatized_form_tag.find(orthography).text.strip()

    return inflection

```

FIGURE A.4 – Méthode permettant d'accéder aux informations de chaque flexion

```python

def get_inflections(to_find: str) -> List[Dict[str, str]]:
    """
    Get the information for all inflections of a verb
    :param to_find: verb to find in the XML database
    :return: list of all inflections
    """
    found = [(form_set, inflected_form) for form_set in verbs_tree.findall("lexicalEntry/formSet")
             for inflected_form in form_set.findall("*")
             if inflected_form.find(orthography) is not None and
             inflected_form.find(orthography).text.strip().lower() == to_find.lower()
             ]

    inflections = []
    for form_set, inflected_form in found:
        inflection = get_inflection(form_set, inflected_form)
        if inflection is not None and inflection != dict():
            inflections.append(inflection)

    return inflections

```

FIGURE A.5 – Méthode permettant d'accéder aux informations de toutes les formes fléchies

```python
def extraction(csv_reader: reader, csv_writer: writer, stat_file: str, improve: bool = False):
    """
    Extract all the inflections for all verbs in the text, and store statistics
    :param csv_reader: object to read the CSV file
    :param csv_writer: object to write in the CSV file
    :param stat_file: file to store statistics in
    :param improve: flag to include improvments or not
    :return: None
    """
    # Number of verbs in the text
    nb_verbs = 0

    # Number of verbs that are annotated, with 1st method, with 2nd method, with both methods
    nb_verbs_annotated = [0, 0, 0]
    # Number of conjugated verbs that are annotated, with 1st method, with 2nd method, with both methods
    nb_verbs_cjg_annotated = [0, 0, 0]
    # Number of infinitive verbs that are annotated, with 1st method, with 2nd method, with both methods
    nb_verbs_inf_annotated = [0, 0, 0]
    # Number of past participle verbs that are annotated, with 1st method, with 2nd method, with both methods
    nb_verbs_ppe_annotated = [0, 0, 0]
    # Number of present participle verbs that are annotated, with 1st method, with 2nd method, with both methods
    nb_verbs_ppa_annotated = [0, 0, 0]

    i = 0

    for i, row in enumerate(csv_reader):
        # Printer to check the running of the script
        if i % 100 == 0:
            print(i)
        tag = row[3]

        if is_verb.match(tag):
            nb_verbs += 1
            found = False

            # Verb to find in the XML database
            to_find = row[0]
            # All the inflections found for this verb
            inflections = get_inflections(to_find)

            if len(inflections) > 0:
                found = True
                nb_verbs_annotated[0] += 1

                if tag == cattex09_verb_cjg:
                    nb_verbs_cjg_annotated[0] += 1
                elif tag == cattex09_verb_inf:
                    nb_verbs_inf_annotated[0] += 1
                elif tag == cattex09_verb_ppe:
                    nb_verbs_ppe_annotated[0] += 1
                elif tag == cattex09_verb_ppa:
                    nb_verbs_ppa_annotated[0] += 1
```

```python
                row.append("|".join([cattex09max_inflection(inflection)
                                     for inflection in get_set_inflections(inflections)]))

                # Ordering the inflections
                row = order_row(row)

        csv_writer.writerow(row)

    # Saving the statistics of this extraction
    with open(stat_file, "w", encoding="utf8") as saving:
        saving.write(f"{i} mots en tout.\n")
        saving.write(f"{nb_verbs} verbes en tout.\n")
        saving.write(f"Nombre de verbes annotés à la première étape : {nb_verbs_annotated[0]} ; "
                     f"à la deuxième étape : {nb_verbs_annotated[1]} ; "
                     f"aux deux étapes : {nb_verbs_annotated[2]}\n")
        saving.write(f"Nombre de verbes conjugués annotés à la première étape : {nb_verbs_cjg_annotated[0]} ; "
                     f"à la deuxième étape : {nb_verbs_cjg_annotated[1]} ; "
                     f"aux deux étapes : {nb_verbs_cjg_annotated[2]}\n")
        saving.write(f"Nombre de verbes à l'infinitif annotés à la première étape : {nb_verbs_inf_annotated[0]} ; "
                     f"à la deuxième étape : {nb_verbs_inf_annotated[1]} ; "
                     f"aux deux étapes : {nb_verbs_inf_annotated[2]}\n")
        saving.write(f"Nombre de verbes au participe passé annotés à la première étape : {nb_verbs_ppe_annotated[0]} ; "
                     f"à la deuxième étape : {nb_verbs_ppe_annotated[1]} ; "
                     f"aux deux étapes : {nb_verbs_ppe_annotated[2]}\n")
        saving.write(f"Nombre de verbes au participe présent annotés à la première étape : {nb_verbs_ppa_annotated[0]} "
                     f"; à la deuxième étape : {nb_verbs_ppa_annotated[1]} ; "
                     f"aux deux étapes : {nb_verbs_ppa_annotated[2]}")
```

FIGURE A.6 – Méthode permettant d'extraire les informations de chaque verbe du texte

```
            # Improvments
            if improve:
                infs = row[1].split("|")
                infinitives = []
                for inf in infs:
                    m = get_inf.match(inf)
                    if m:
                        infinitives.append(m.group(1).lower())
                    else:
                        infinitives.append(inf.lower())

                if tag == cattex09_verb_inf:
                    inflections.append({grammatical_mood: "infinitive", lemmatized_orthography: to_find})
                    for infinitive in infinitives:
                        if infinitive != to_find:
                            inflections.append({grammatical_mood: "infinitive",
                                                lemmatized_orthography: infinitive})

                    nb_verbs_annotated[1] += 1
                    nb_verbs_inf_annotated[1] += 1

                    if found:
                        nb_verbs_annotated[2] += 1
                        nb_verbs_inf_annotated[2] += 1

                if tag == cattex09_verb_ppa:
                    improved = False
                    for infinitive in infinitives:
                        inflections.append({grammatical_mood: "participle", grammatical_tense: "present",
                                            lemmatized_orthography: infinitive})
                        improved = True

                    if improved:
                        nb_verbs_annotated[1] += 1
                        nb_verbs_ppa_annotated[1] += 1

                    if found and improved:
                        nb_verbs_annotated[2] += 1
                        nb_verbs_ppa_annotated[2] += 1

                if tag == cattex09_verb_ppe:
                    improved = False
                    for infinitive in infinitives:
                        inflections.append({grammatical_mood: "participle", grammatical_tense: "past",
                                            lemmatized_orthography: infinitive})
                        improved = True

                    if improved:
                        nb_verbs_annotated[1] += 1
                        nb_verbs_ppe_annotated[1] += 1

                    if found and improved:
                        nb_verbs_annotated[2] += 1
                        nb_verbs_ppe_annotated[2] += 1
```

FIGURE A.7 – Améliorations du script

```
def order_row(row: List[str]) -> List[str]:
    """
    Get the row ordered by the tag of the verb
    :param row: row of the file to order
    :return: same row with order
    """
    ordered_row = row[:4]
    tag = ordered_row[3]
    inflection = row[4]

    like_tag = [r for r in inflection.split("|") if tag in r]
    tmp_row = [r for r in inflection.split("|") if r not in like_tag]
    tags = like_tag + tmp_row

    ordered_row.append("|".join(tags))

    return ordered_row
```

FIGURE A.8 – Script permettant d'ordonner les suggestions

```python

def cattex09max_inflection(inflection: Dict[str, str]) -> str:
    """
    Convert the inflection into Cattex09max format
    :param inflection: inflection to convert
    :return: string containing the Cattex09max formatted inflection
    """
    cattex_inflection = ""
    lemma = inflection[lemmatized_orthography] if lemmatized_orthography in inflection else ""
    mood = inflection[grammatical_mood] if grammatical_mood in inflection else ""
    if mood == "infinitive":
        cattex_inflection = f"{cattex09_verb_inf}/{lemma}"
    elif mood == "participle":
        tense = inflection[grammatical_tense] if "" in inflection else ""
        if tense == "past":
            cattex_inflection = f"{cattex09_verb_ppe}/{lemma}"
        elif tense == "present":
            cattex_inflection = f"{cattex09_verb_ppa}/{lemma}"
    else:
        cattex_inflection = cattex09_verb_cjg

        tense = inflection[grammatical_tense] if grammatical_tense in inflection else ""
        person = inflection[grammatical_person] if grammatical_person in inflection else ""
        number = inflection[grammatical_number] if grammatical_number in inflection else ""

        cattex_mood = cattex09max_moods[mood] if mood in cattex09max_moods else None
        cattex_tense = cattex09max_tenses[tense] if tense in cattex09max_tenses else None
        cattex_person = cattex09max_persons[person] if person in cattex09max_persons else None
        cattex_number = cattex09max_numbers[number] if number in cattex09max_numbers else None

        if cattex_mood is not None:
            cattex_inflection += f"/{cattex09max_moods[mood]}"
        if cattex_tense is not None:
            cattex_inflection += f"/{cattex09max_tenses[tense]}"
        if cattex_person is not None:
            cattex_inflection += f"/{cattex09max_persons[person]}"
        if cattex_number is not None:
            cattex_inflection += f"/{cattex09max_numbers[number]}"
        cattex_inflection += f"/{lemma}"

    return cattex_inflection

```

FIGURE A.9 – Script de mise en forme des flexions des verbes

Table des figures

3.1 Extrait de l'article *Information* du DMF 13
3.2 Extrait de l'article *Connaissance* du DMF 14
3.3 Extrait de l'article *Cours* du DMF 15
3.4 Architecture du système (Souvay et Pierrel 2009) 17
3.5 Comparaison avec le lexique du TLFnome (Namer 2014) 22
3.6 Re-segmentation et ré-étiquetage (Namer 2014) 23
3.7 Lemmatisation des mots nouveaux (Namer 2014) 24

4.1 Premier enrichissement au texte 29
4.2 Deuxième enrichissement au texte 31
4.3 Exemple d'affichage multifacette du logiciel TXM 33
4.4 Extrait de la première page de texte de la version courante 35
4.5 Extrait du corpus étiqueté par LGeRM 38
4.6 Extrait de la page se terminant par « apa » 41
4.7 Extrait de la page commençant par « reilliez » 41
4.8 Extrait du fichier HTML se terminant par « apa » 42

5.1 Échantillon du lexique Morphalou 2.0 (*Morphalou* 2012) 47
5.2 Représentation du LMF implémenté pour Morphalou (*Morphalou* 2012) . . . 48
5.3 Extrait de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012) 52
5.4 Extrait automatique de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012) 53
5.5 Extrait copié de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012) 53
5.6 Format des lexies du verbe *abregier* 55
5.7 Page du TCAF du verbe *bruire* 56
5.8 Format des verbes similaires au verbe *abregier* 57

5.9 Format des verbes dérivés du verbe *abregier* 58
5.10 Représentation XML du verbe *abregier* 59
5.11 Représentation XML du verbe *gregier* 59

6.1 Résultat de la recherche de « voloir » sur le site du TCAF 66
6.2 Résultat de la recherche de « voldre » sur le site du TCAF 67
6.3 Corpus annoté au format Cattex09max 68

A.1 Script d'extraction et d'enregistrement des données du corpus TXM 72
A.2 Script permettant d'obtenir des chiffres sur le contenu des corpus 73
A.3 Script utilisé pour formatter la table des matières de *Tableaux de conjugaison de l'ancien français* (Okada et Ogurisu 2012) 74
A.4 Méthode permettant d'accéder aux informations de chaque flexion 74
A.5 Méthode permettant d'accéder aux informations de toutes les formes fléchies 75
A.6 Méthode permettant d'extraire les informations de chaque verbe du texte . . 76
A.7 Améliorations du script 77
A.8 Script permettant d'ordonner les suggestions 77
A.9 Script de mise en forme des flexions des verbes 78

Liste des tableaux

4.1 Comparaison des chiffres des deux corpus 39
4.2 Tableau de correspondances entre les étiquettes du TXM et celles de LGeRM pour les formes présentant des ambiguïtés 40

5.1 Tableau de correspondances entre les annotations de temps et de mode du TCAF et celles de Cattex09max 46
5.2 Tableau de correspondances entre les annotations de personnes et de nombres du TCAF et celles de Cattex09max 46
5.3 Tableau de correspondances entre les annotations de temps et de mode du site du TCAF, celles de Morphalou et celles de Cattex09max 49
5.4 Tableau de correspondances entre les annotations de personne du site du TCAF, celles de Morphalou et celles de Cattex09max 50

Bibliographie

Apache Cassandra. 2016. https://cassandra.apache.org/.

ATILF - CNRS & Université de Lorraine. 2019. *LGeRM : plateforme de lemmatisation* [en français]. Visité le 25 mai. http://www.atilf.fr/LGeRM/plateforme/.

———. *LGeRM : lemmatisation de la variation graphique des états anciens du français et lexiques morphologiques* [en français]. http://www.atilf.fr/LGeRM/.

———. *TLFi* [en français]. http://atilf.atilf.fr/.

ATILF - CNRS & Université de Lorraine ; LFA - Université d'Ottawa. *Base de Graphies Verbales* [en français]. http://www.atilf.fr/bgv/.

BFM - Base de Français Médiéval [en français]. 2016. http://txm.bfm-corpus.org/.

Blumenthal, Peter, et Achim Stein. 2002. *Tobler-Lommatzsch : Altfranzösisches Wörterbuch* [en français]. http://www.uni-stuttgart.de/lingrom/stein/tl/index.htm.

Bragantini-Maillard, Nathalie, et Corinne Denoyelle. 2012. *Cent verbes conjugués en français médiéval* [en français]. Lettres. Armand Colin. ISBN : 978-2-200-27442-9.

Brill, Eric. 1992. « A Simple Rule-Based Part of Speech Tagger » [en anglais] : 4.

Buridant, Claude. 2000. *Grammaire nouvelle de l'ancien français* [en français]. Sedes.

Cambridge University Press. 2008. *Evaluation of ranked retrieval results* [en anglais]. https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html.

Catalogue de la Base de français médiéval [en français]. 2013. http://catalog.bfm-corpus.org/qgraal_cm.

Morphalou [en français]. 2012. http://www.cnrtl.fr/lexiques/morphalou/.

D. Godefroy [en français]. 2008. http://micmap.org/dicfro/search/dictionnaire-godefroy/.

DicFro [en français]. 2008. http://www.micmap.org/dicfro/.

Fouché, Pierre. 1945. *Morphologie historique du français - Le verbe* [en français].

Francopoulo, Gil, Monte George, Nicoletta Calzolari, Monica Monachini, Nuria Bel, Mandy Pet et Claudia Soria. 2006. « Lexical Markup Framework (LMF) » [en anglais] : 5.

Godefroy, Frédéric Eugène. 1881. *Dictionnaire de l'ancienne langue française et de tous ses dialectes du 9e au 15e siècle* [en français]. F. Vieweg (Paris).

Guillot, Céline, Celine Guillot, Sophie Prévost et Alexei Lavrentiev. 2013a. « Manuel de référence du jeu Cattex09 » [en français] : 33.

———. 2013b. « Principes d'annotation Cattex09 » [en français] : 10.

Heiden, Serge, Jean-Philippe Magué et Bénédicte Pincemin. 2010. « TXM : Une plateforme logicielle open-source pour la textométrie - conception et développement » [en français], 2 : 1021-1032. Edizioni Universitarie di Lettere Economia Diritto, juin. https://halshs.archives-ouvertes.fr/halshs-00549779/document.

Hiller, Jonathan D, et Hod Lipson. 2009. « STL 2.0 : A PROPOSAL FOR A UNIVERSAL MULTI-MATERIAL ADDITIVE MANUFACTURING FILE FORMAT » [en anglais] : 13.

Lanly, André. 1995. *Morphologie historique des verbes français. Notions générales, conjugaisons régulières, verbes irréguliers* [en français]. Unichamp. Champion. ISBN : 2-7453-0822-X. http://www.jstor.org/stable/27938615.

Manuel de TXM 0.7 FR [en français]. 2018, février. http://textometrie.ens-lyon.fr/files/documentation/Manuel%20de%20TXM%200.7%20FR.pdf.

Marchello-Nizia, Christiane. 2013. « Queste del saint Graal » [en français] (juillet) : 788. http://txm.ish-lyon.cnrs.fr/bfm/pdf/qgraal_cm_2013-07.pdf.

Martin, Robert. 1960. *Fonds de formes flexionnelles* [en français].

———. 2015. *DMF* [en français]. http://www.atilf.fr/dmf/.

Namer, Fiammetta. 2005. *CNRTL : Centre National de Ressources Textuelles et Lexicales - Flemm* [en français]. http://www.cnrtl.fr/outils/flemm/.

Namer, Fiammetta. 2014. « FLEMM : Un analyseur flexionnel du français à base de règles » [en français] : 24. https://www.researchgate.net/profile/Fiammetta_Namer/publication/238789437_FLEMM_UN_ANALYSEUR_FLEXIONNEL_DU_FRAN_CAIS_A_BASE_DE_R_EGLES/links/0046352cbbe4a37562000000/FLEMM-UN-ANALYSEUR-FLEXIONNEL-DU-FRAN-CAIS-A-BASE-DE-R-EGLES.pdf.

Okada, Machio, et Hitoshi Ogurisu. 2007. *Tableaux de conjugaison de l'ancien français (TCAF)* [en français]. http://www.micmap.org/dicfro/introduction/tableaux-de-conjugaison.

———. 2012. *Tableaux de conjugaison de l'ancien francais* [en français]. Rapport technique. Avril. https://juliettebourdier.files.wordpress.com/2013/02/tableaux-de-conjugaison-de-lancien-franc3a7ais.pdf.

xml.etree.ElementTree — The ElementTree XML API — Python 3.7.3 documentation [en anglais]. 2019. Documentation. Visité le 15 avril 2019. https://docs.python.org/3.7/library/xml.etree.elementtree.html.

Romary, Laurent, Susanne Salmon-Alt et Gil Francopoulo. 2004. « Standards going concrete : from LMF to Morphalou » [en anglais]. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries - ElectricDict '04,* 22. Geneva, Switzerland : Association for Computational Linguistics. Visité le 9 mars 2019. doi :10.3115/1610042.1610047. http://portal.acm.org/citation.cfm?doid=1610042.1610047.

Schmid, Helmut. 1994. *Probabilistic Part-of-Speech Tagging Using Decision Trees* [en anglais]. Rapport technique. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf.

Shinyama, Yusuke. 2014. *pdfminer : PDF parser and analyzer.* Visité le 19 mai 2019. http://euske.github.io/pdfminer/index.html.

CQL - Corpus Query Language [en anglais]. 2015, mai. Visité le 14 juillet 2019. https://www.sketchengine.eu/documentation/corpus-querying/.

Souvay, Gilles. 1986. *Analyse de textes de Moyen-Français* [en français]. Rapport de DEA. Centre de Recherche en Informatique de Nancy, Université de Nancy I.

Souvay, Gilles, et Jean-Marie Pierrel. 2009. « LGeRM Lemmatisation des mots en Moyen Français » [en français]. *Traitement Automatique des Langues* 50 (2) : 21. https://halshs.archives-ouvertes.fr/halshs-00396452/document.

python - Custom indent width for BeautifulSoup .prettify() [en anglais]. 2013. Visité le 14 avril 2019. https://stackoverflow.com/questions/15509397/custom-indent-width-for-beautifulsoup-prettify/15513483.

Tobler, Adolf, et Erhard Lommatzsch. 1925. *Altfranzösisches Wörterbuch* [en français]. Berlin : Weidmann.

Treetagger [en français]. http://cental.fltr.ucl.ac.be/treetagger/index.html.

Zink, Gaston. 1989. *Morphologie du français médiéval* [en français]. Linguistique nouvelle. Google-Books-ID : rtwJCwAAQBAJ. Presses Universitaires de France. ISBN : 978-2-13-063907-7. http://www.jstor.org/stable/395166?Search=yes&resultItemClick=true&searchText=morphologie&searchText=du&searchText=fran%C3%A7ais&searchText=m%C3%A9di%C3%A9val&searchText=zink&searchUri=%2Faction%2FdoBasicSearch%3Fgroup%3Dnone%26amp%3Bacc%3Doff%26amp%3Bfc%3Doff%26amp%3BQuery%3Dmorphologie%2Bdu%2Bfran%25C3%25A7ais%2Bm%25C3%25A9di%25C3%25A9val%2Bzink%26amp%3Bwc%3Don&seq=1#page_scan_tab_contents.