UCLouvain

FIAL
Un éventail de formations, une multitude d'horizons

Faculté de philosophie, arts et lettres

FreMLy: un outil de simplification pour les textes médicaux et de langue générale

Auteur : Eva Rolin
Promoteur(s) : Thomas François
Lecteur(s) : Patrick Watrin
Année académique 2019-2020
Master en Linguistique, finalité en traitement automatique du langage

Remerciements

Avant d'entamer le premier chapitre de ce mémoire, je tiens à remercier les personnes qui ont contribué de près ou de de loin à l'écriture de celui-ci. Tout d'abord, je tiens à remercier mon promoteur, Thomas François, pour avoir accepté de me suivre dans cette aventure et pour les judicieux conseils dont il m'a fait part. Je remercie également Madame Natalia Grabar, laquelle m'a inspirée la thématique de ce mémoire, pour la confiance qu'elle m'a accordée en m'autorisant à utiliser ses ressources. Je tiens également à remercier Monsieur Rémi Cardon pour son aide précieuse tout au long de l'écriture de mon mémoire. Je ne peux évidemment pas clotûrer cet avant-propos sans remercier mes parents qui m'ont encouragée tout au long de ma scolarité parsemée d'incertitudes, d'angoisses mais également d'éclats de joie. Ils ont toujours fait en sorte que je ne manque de rien et que je puisse m'épanouir autant dans ma scolarité que dans ma vie personnelle. Puisque les amis sont la famille que nous avons choisie, je tiens également à remercier tous mes amis mais surtout Clara Leclef laquelle m'a soutenue tout au long de l'écriture de mon mémoire, encouragée et changé les idées quand le moral était absent. Enfin je dédie mon mémoire à mon meilleur ami, Augustin Hengen, qui malheureusement n'est plus là pour le lire, lui qui a été présent à chaque étape de ma vie et qui a cru en moi.

Table des matières

Introduction

Durant l'été 2018, nous avons eu le privilège de participer temporairement au projet CLEAR dans le cadre de notre stage au CNRS. Ce projet, comme son nom l'indique, est destiné à faciliter la compréhension des textes médicaux en employant les méthodes de traitement automatique du langage. Une telle initative est bénéfique aux patients qui ont accès à des informations beaucoup plus claires sur leurs traitements et pathologies leur permettant une meilleure adhésion au processus de soins (Cardon, 2018). Mais le projet est également profitable aux médecins puisqu'une meilleure compréhension du langage médical implique de facto une meilleure interaction entre les professionels de la santé et leurs patients.

Il est vrai que la langue médicale, de la même manière que l'ensemble des langues techniques, est souvent considérée comme complexe en raison de multiples acronymes, sigles, mots composés, etc. D'un point de vue sémantique, celle-ci nous apparait également complexe parce que le sens des termes employés a été défini par les spécialistes. Dès lors, nous ne comptons plus le nombre de fois où nous éprouvons des difficultés lors de la lecture de notices de médicaments ou tout simplement lorsque le docteur pose un diagnostic sur nos maux. Cependant, comme le fait remarquer Sara et Sonia (2013), depuis quelques années, un effort de la part des organismes publics pour simplifier la langue médicale est perceptible à travers par exemple des conseils lors de l'élaboration de notices. Ceux-ci encouragent notamment les entreprises pharmaceutiques à clarifier leur style rédactionnel en évitant des phrases trop longues, en évitant plus que possible le jargon médical. Evidemment, ces recommendations sont idylliques et sont très difficiles à appliquer étant donné que simplifier des textes techniques demandent non seulement de cibler les mots qui font difficulté aux patients mais également de simplifier *réellement* les textes c'est-à-dire de fournir des mots qui sont réellement plus faciles que les termes techniques. Or, lorsque nous regardons les synonymes proposés sur Internet, ceux-ci sont aussi voire plus complexes que les termes qui posent difficulté.

C'est autour de cette dynamique que nous proposons d'asseoir notre mémoire. En effet, nous avons décidé de créer un outil pour la simplification de textes médicaux et pour la langue française. Nous avons choisi de ne pas nous limiter à la langue médicale parce que nous pensons qu'il est important de comprendre d'abord le fonctionnement de la simplification de la langue générale pour pouvoir prétendre simplifier des termes plus techniques. Nous ne créons pas cet outil à partir de rien puisque nous continuons notre objectif de stage : créer un outil de simplification pour les textes médicaux à partir de la bibliothèque de simplification anglophone *LEXenstein* (Paetzold et Specia, 2015). Effectivement, *FreMLy* s'inspire non seulement du fonctionnement modulaire de *LEXenstein* et de son format mais également de ses multiples classes proposées.

Pour créer notre outil, nous avons décidé suivre une heuristique rigoureuse et précise laquelle correspond aux différentes étapes de notre mémoire. Dans un premier temps, nous proposons un état de l'art sur les différentes techniques de simplification lexicale pour la langue générale et ensuite pour la langue médicale en abordant les différentes étapes du processus : l'identification des mots complexes, la génération des candidats, la sélection de ceux-ci et le classement des synonymes en fonction de leur facilité. Nous proposons également une section "plus" linguistique en abordant les difficultés de la langue médicale. Ensuite, la deuxième partie est réservée à notre méthodologie dans laquelle nous expliquons quelles classes de *LEXenstein* nous avons réutilisées et modifiées pour la conception de *FreMLy*. Nous expliquons également les implémentations que nous avons réalisées dans le langage python. Pour finir, nous proposons une section réservée à nos résultats et descriptions de ceux-ci. Nous terminons avec une conclusion et une discussion sur les recherches menées.

Première partie

Etat de l'art

Dans cette présente section, nous proposons un panorama des études qui ont été menées dans les domaines suivants : la simplification automatique textuelle, la simplification lexicale pour la langue générale, les textes de spécialité et la simplification lexicale des textes médicaux. Ces différents champs de recherche présentent un intérêt tout particulier étant donné le but poursuivi dans ce mémoire : créer un outil de simplification lexicale *FreMLy* pour les textes médicaux et pour le français sur base *LEXenstein* (Paetzold et Specia, 2015). Nous structurons l'état de l'art en réservant un chapitre par domaine recherche.

Nous nous situons du côté de la simplification automatique textuelle étant donné notre but principal : rendre accessibles les textes médicaux pour un public spécifique (les patients). Ce domaine de recherche peut être subdivisé en deux sous-domaines : la simplification lexicale et la simplification syntaxique. Néanmoins, nous nous intéresserons uniquement à la simplification des unités lexicales dans la mesure où *LEXenstein* ne propose pas de modules de simplification syntaxique. Mais nous pensons également que c'est un choix judicieux étant donné la forte présence de termes compliqués pour un public non savant. C'est d'ailleurs pour cette raison que nous avons choisi de porter un intérêt aux textes de spécialité. Nous sommes convaincus qu'une étude approfondie de ceux-ci permettra de mettre en exergue à la fois les différentes caractéristiques et difficultés que ceux-ci peuvent présenter. Pour finir, présenter des travaux dans le domaine de la simplification lexicale de textes médicaux nous semble inévitable étant donné la nature de notre recherche.

Cet état de l'art a différents buts. Dans un premier temps, il nous permettra de nous donner une vision des différents travaux qui se font dans le domaine de la simplification afin de nous soumettre des pistes pour la partie expérimentale. Il va nous permettre également, grâce à la partie linguistique sur les textes de spécialité, de cibler les difficultés que peuvent rencontrer les patients lors de la lecture de textes médicaux. Cette partie de l'état de l'art a pour finalité de nous donner toutes les ressources théoriques pour la création de *FreMLy*.

Concernant la structure de la section, nous présenterons les domaines en partant du plus général pour finir avec le plus spécifique. De cette manière, la première section permettra de donner un aperçu de ce qu'est la simplification textuelle grâce à une définition théorique et plusieurs travaux clés, la deuxième section portera sur la simplification lexicale, toujours accompagnée de travaux réalisés dans le domaine. Nous finirons avec les sections 3 et 4 qui se baseront sur les textes de spécialités et sur la simplification lexicale des textes médicaux.

1 La simplification automatique textuelle pour la langue générale

La simplification automatique de textes est un domaine du TAL et a pour but d'appliquer des transformations sur les phrases d'un texte afin de les rendre plus lisibles, tout en conservant leur grammaticalité et leur sens (Cardon, 2018). Dans cette recherche, nous nous intéressons à la simplification à destination d'humains et plus spécifiquement à destination de tout patient désireux de comprendre un compte-rendu médical, par exemple.

Il existe d'ailleurs plusieurs travaux destinés à des publics très divers : **aphasiques** (Carroll et al., 1999) , **dyslexiques** (Rello et al., 2013), **illettrés** (Candido et al., 2009), **patients** (Cardon, 2018), etc. Ces recherches sont très utiles dans la mesure où de nombreux textes sont caractérisés par des constructions syntaxiques qui ne suivent pas le modèle canonique ou utilisent des termes compliqués (Siddarthan, 2014). En ce sens, Siddharthan (2014) rapporte que certains aspects de la langue posent problème à certains groupes spécifiques de lecteurs : les mots techniques et peu fréquents peuvent faire difficulté aux aphasiques ou aux autistes tandis que les phrases longues avec des mots longs et peu fréquents troubleront davantage les dyslexiques étant donné leur forte sensibilité à la surcharge d'informations. La particularité de notre problématique est qu'elle ne s'adresse pas nécessairement à des personnes dont la faculté de lecture est déficiente. En effet, le vocabulaire médical comporte plusieurs difficultés tels que des technicismes collatéraux, des siglaisons, des acronymes, etc qui compliquent la compréhension des patients. Par conséquent, la simplification automatique textuelle peut apporter une solution à ce problème.

Généralement, deux sous-tâches sont distinguées dans la simplification automatique textuelle : la simplification syntaxique et la simplification lexicale (Ligozat et al., 2013). La première transforme des phénomènes syntaxiques pouvant poser problème à la lecture en des équivalents plus accessibles aux lecteurs. Par conséquent, les phrases relatives, subordonnées ou passives sont modifiées au profit de constructions considérées comme plus simples (Saggion et Graeme, 2017). Etant donné que ces deux tâches sont liées, c'est-à-dire que si nous choisissons de simplifier la structure syntaxique d'une phrase, il est également nécessaire d'appliquer des transformations lexicales pour conserver la grammaticalité (Saggion et Graeme, 2017), nous avons choisi de donner un bref aperçu de la simplification syntaxique. Ensuite, nous aborderons le principe de la simplification lexicale avant d'entamer la section 2 consacrée uniquement à ce type de simplification.

1.1 La simplification syntaxique

La simplification syntaxique prête attention aux structures de phrases qui peuvent poser problème telles que les subordonnées, les coordonnées, les phrases relatives et toutes autres phrases qui ne suivent pas l'ordre habituel de la phrase (sujet-verbe-complément) (Saggion et Graeme., 2017) . Dans ce cas, certains systèmes utilisent des approches à base de règles. Citons à cet égard le travail de Chandrasekar et al. (1996) qui ont développé une approche à base de règles écrites à la main. Pour l'exemplifier, nous proposons de prendre la règle suivante : W X :NP,RELPRON Y,Z. -> W X :NP Z. X : NP Y

Cette règle permet de transformer (1) en (2) :

> (1) Hu Jintao, *qui* est l'actuel chef suprême de la République populaire de Chine, était en visite en Espagne. (2) Hu Jintao était en visite en Espagne. Hu Jintao est l'actuel chef suprême de la République populaire de Chine.

Ce principe est interprêté par Chandrasekar (1997) de la manière suivante :

> si une phrase commence par un segment W et un groupe nominal (NP) suivi par une phrase de la forme (RELPRON Y) suivi de Z où Y et Z sont des séquences de mots arbitraires alors la phrase peut être simplifiée en deux phrases, W X suivi par Z et X suivi par Y.

Cette règle peut être appliquée à la phrase ci-dessus (Saggion et Graeme, 2017) :

> En considérant que les variables dans la partie gauche de la règle sont : W = "", X= "Hu Jintao", Y = "est l'actuel chef suprême de la République populaire de Chine" et, Z= "était en visite en Espagne", il suffit de réécrire la phrase en utilsant le côté droit de la règle.

Conséquemment, la proposition relative a été transformée afin que le lecteur n'ait pas à se préoccuper du pronom relatif et de son antécédent. Le sujt a également été repété pour une meilleure compréhension. Cependant, les systèmes à base de règles se révèlent le plus souvent laborieux et chronophages. C'est pour cette raison que Chandrasekar et Srinivas ont proposé en 1997 un système à base de règles d'apprentissage automatique fonctionnant en deux étapes : la description structurelle de la phrase et la simplification de celle-ci en utilisant la description réalisée à la première étape. Les règles, quant à elles, sont créées à partir d'un algorithme permettant d'induire automatiquement des règles à partir d'un corpus aligné annoté (Chandrasekar et al., 1997). D'autres travaux ont été réalisés dans le cadre de la simplification syntaxique comme la thèse de Mark Dras (1999).

1.2 La simplification lexicale

Même s'il est possible de simplifier la syntaxe sans le lexique ou inversément, ces deux "composantes" textuelles sont intrinsèquement liées. En simplifiant une construction complexe, il est fort à parier qu'une transformation lexicale devra être également opérée (Saggion et Graeme, 2017). Le but de la simplification lexicale est de remplacer les mots difficiles, qui peuvent potentiellement poser problème à des lecteurs cibles, tout en préservant le sens original de la phrase simplifiée. Siddharthan (2014) considère, à juste titre, que les recherches en simplification lexicale ont tendance à tomber dans l'une des deux catégories suivantes : ajouter des définitions dans le cas de mots techniques et difficiles ou remplacer les mots compliqués par des synonymes plus simples. La première est la plupart du temps utilisée pour la simplification de textes/phrases/mots techniques :

(1) Ils ont la *tuberculose*.

(2) Ils ont la tuberculose, *une maladie qui affecte le plus souvent les poumons*.

Tandis que la substitution lexicale aide davantage les personnes atteintes d'aphasie ou de dyslexie en donnant des termes simplifiés et connus de tous :

(1) Elle porte une *somptueuse* robe.

(2) Elle porte une *jolie* robe.

Contrairement à la simplification syntaxique, il n'est pas possible de recourir à des systèmes à base de règles : le lexique est un ensemble infini et il ne serait pas, par conséquent, envisageable de créer des règles. Il a fallu trouver d'autres techniques et méthodes pour un autre problème. Dans le chapitre suivant, nous allons nous concentrer plus en profondeur sur les étapes du processus de la simplification lexicale.

2 La simplification lexicale

La simplification lexicale peut être divisée en 4 étapes importantes. Ces mêmes étapes sont expliquées par Cardon (2018) dans ses recherches sur la simplification lexicale de textes médicaux. Les différentes étapes de la figure 2 correspondent à :

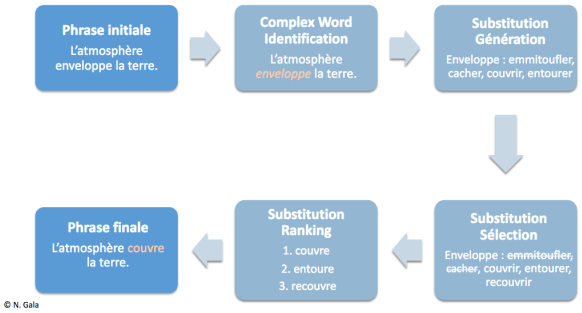


FIGURE 1 – Processus de simplification (Gala, 2019)

1. **L'identification de termes complexes** Pour simplifier un texte, il s'agit d'abord d'identifier les éléments lexicaux qui peuvent poser problème aux lecteurs. Il existe diverses manières de s'y prendre. Par exemple, il est d'usage d'utiliser un lexique de référence composé de termes simples : si le mot à identifier se retrouve dedans, alors le mot est simple. Dans le cas contraire, il est considéré comme complexe. Il est également possible d'identifier des termes complexes à l'aide d'une machine à vecteurs de support (ensemble de techniques d'apprentissage) : à partir d'un corpus annoté, la machine apprend au moyen de prescripteurs linguistiques (nombre de sens, la fréquence, le nombre de syllabes, etc) à déterminer si un terme est complexe ou simple.
2. **Génération de candidats** Après avoir déterminé les éléments qui font défaut, il s'agit de leur trouver des remplaçants. Ceux-ci doivent non seulement être plus faciles mais surtout, ils ne doivent pas altérer le sens original de la phrase de départ. Lors de cette étape, Caroll et al.(1998) et François et al.(2016) privilégient l'utilisation de ressources lexicales (thesaurus, antologie, dictionnaire). Plus récemment, Glavas et Stajner (2015) ont proposé l'utilisation de prolongements lexicaux pour générer des synonymes.
3. **Sélection des candidats** Cette étape est une étape de désambiguïsation puisqu'elle retient uniquement les candidats qui font sens dans la phrase à simplifier. Dans ce but, Paetzold et Specia (2017) utilisent des réseaux de neurones tandis que François et al. (2016) privilégient les machines à vecteurs de support (SVM).
4. **Classement des candidats** Pour finir, s'il existe plusieurs candidats, cette étape permet de les classer en fonction de leurs degrés de difficulté. Cardon (2018) rappelle les critères exploités lors de la compétition *SemEval2012* où

la tâche est justement l'ordonnancement de candidats en fonction de plusieurs caractéristiques : le nombre de ngrammes, le nombre de syllabes, fréquences, etc.

Dans cette partie, nous allons abordé les différents travaux présentant diverses techniques qui ont été réalisés pour chaque étape de la simplification lexicale. Il ne s'agira donc plus d'expliquer *pourquoi* nous faisons ces étapes mais plutôt *comment* celles-ci fonctionnent.

2.1 Identification de termes complexes

Dans l'introduction, nous avons présenté le lexique et la machine de vecteurs à support pour l'étape d'identification. Nous avons choisi de présenter deux approches plus récentes. Dès lors, nous commençons par 2 travaux (Tack et al., 2016 ; Avdiu et al., 2019) qui ne relèvent pas directement de notre domaine d'expertise mais qui s'en rapprochent : la prédiction automatique de la compétence lexicale d'un apprenant. En effet, Avdiu et al. (2019) mentionnent dans leur article que cette tâche est étroitement liée à l'identification des termes complexes ou encore à la simplification automatique des textes. Selon eux, les différences se situent à la fois au niveau du sujet d'étude (est-ce que l'on s'intéresse aux mots ou aux phrases ?), à la nature de la tâche (est-ce que l'on veut faire de l'identification, de la prédiction ou encore de l'estimation ?) ou bien au niveau du groupe cible.

2.1.1 Prédiction automatique de la connaissance lexicale

Tack et al. (2016) envisagent la création d'un modèle qui prédit la compétence lexicale d'un apprenant individuel en y intégrant des traits de complexité lexicale sur base des travaux de Gala et al. (2014). Les chercheurs distinguent deux heuristisques très différentes pour déterminer la complexité : la première se base sur des critères de surface tels que le nombre de lettres dans un mot, la fréquence de ce mot, le nombre de syllabes, etc tandis qu'à l'inverse, la deuxième se concentre sur l'apprenant en modélisant son lexique et en supposant comme difficiles les mots qui ne sont pas reconnus par le modèle (Tack et al., 2016).

Dans le même ordre d'idée, nous proposons le travail d'Avdiu et al. (2019) dont le but est également d'élaborer un modèle de prédiction de la connaissance lexicale d'un apprenant. A la différence de Tack et al. (2016) qui utilisent la ressource lexicale FLELex, Avdiu et al. (2019) exploitent l'apprentissage automatique ("machine learning") à l'aide de forêts aléatoires et de réseaux de neurones. L'intérêt de cette recherche se trouve dans les prescripteurs utilisés dans le but de distinguer un apprenant d'un autre. Ceux-ci sont également au nombre de 2 et se situent au niveau de l'apprenant et du lexique. Comme indicateurs de complexité lexicale, nous retrouvons une fois de plus la fréquence ainsi que des variables psycholinguistiques déjà citées auparavant (le nombre de lettres, le degré d'abstractivité, etc). Concernant les indicateurs de complexité au niveau de l'apprenant, Avdiu et al. (2019) se différencient de Tack et al. (2016) en utilisant une liste de mots qui sont censés être connus dans différents genres et domaines. Leur postulat est le suivant : si l'apprenant connait de nombreux mots fréquents dans un domaine en particulier, il est probable que celui-ci connaisse d'autres mots à hautes fréquences dans ce domaine.

2.1.2 L'apprentissage automatique pour identifier les termes complexes

Pour compléter ces 2 recherches, nous rapportons une des quelques pistes qui ont été proposées dans diverses tâches partagées ("shared task") portant sur la tâche d'identification des mots complexes (Paetzold et Specia., 2016). La plupart des systèmes proposés lors de SemEval 2016 utilisent des techniques d'apprentissage automatique comme c'est le cas pour l'apprentissage par arbres de décision dont le fonctionnement peut être décrit de cette manière (Azencott, 2018) : chaque nœud de l'arborescence teste une condition sur une variable et chacun de ses enfants correspond à une réponse possible à cette condition. Les feuilles correspondent à une étiquette. Par exemple, l'équipe HMC effectue la phase d'identification par le biais d'un arbre de décision ayant une profondeur maximale de 4. Les forêts aléatoires, qui combinent plusieurs arbres afin que les erreurs de chacun puissent être compensées, sont également utilisées par l'équipe MELBOURNE et accompagnées de plusieurs variables sémantiques et lexicales.

Grâce à ces recherches, nous pouvons déjà mettre en lumière plusieurs prescripteurs de difficulté lexicale : la fréquence, le nombre de lettres et de syllabes par exemple et l'absence de termes dans un lexique donné. Ce constat est d'ailleurs en partie vérifié par Paetzold et Specia (2016) qui ont collecté des statistiques pour mettre en évidence les différences entre mots complexe et simple. En moyenne, ceux-ci ont remarqué que les mots complexes étaient moins ambigus, plus courts et moins fréquents dans le Wikipédia simple. Ils ont également remarqué que la plupart des mots considérés comme complexes par les annotateurs étaient des mots techniques.

2.2 La génération de substituts

Pour cette tâche, nous avons identifié 3 techniques différentes. Comme cela a été dit dans le point précédent, les approches classiques en substitution privilégient des ressources lexicales (Carroll et al., 1998 ; François et al., 2016). Mais nous pouvons également extraire des candidats à partir de corpus parallèles (Horn et al., 2014) ou encore généré des synonymes à partir de réseaux de neurones (Glavas et Stajner, 2015).

2.2.1 Les ressources lexicales

La première approche de simplification lexicale est attribuée à Carroll et al., (1998) et est destinée à un public de personnes atteintes d'aphasie. La simplification est réalisée à partir d'une base de données lexicales en ligne, Wordnet [1]. Cette ressource est un dictionnaire informatisé, dont l'unité de base est le concept. Pour définir le sens des mots, Wordnet se base sur deux "notions" : les synsets (ensemble de synonymes) et les relations entre les synsets. Ces relations de sens sont de différentes natures : hyponymie, synonymie, antonymie, etc [2].

Cette base de données représente, par conséquent, un outil intéressant pour la tâche de la génération de candidats étant donné qu'elle est non seulement exploitable par les ordinateurs mais qu'elle regroupe également les unités selon leur relations sémantiques et lexicales [3]. A partir de cette ressource, une liste de synonymes est extraite pour chaque mot complexe. Le meilleur candidat sera celui ayant la plus haute fréquence d'apparition dans la liste de Kucera-Francis (Carroll et al., 1998). Nous exemplifions cette approhe à l'aide des propos de Saggion (Saggion, 2017) que nous avons adapté en français :

> Si nous avons la phrase suivante "Mon automobile est en panne" et si le mot "automobile" a les synonymes et les fréquences suivantes dans la liste de Kucera-Francis voiture (278), auto (108), automobile (50). Nous choisirons de remplacer "automobile" par "voiture" étant donné sa haute fréquence d'apparition.

Cependant, François et al., (2016) considèrent que les distinctions entre les sens dans Wordnet sont parfois trop nombreuses et par conséquent difficiles à opérer pour des annotateurs humains (Edmonds et Kilgarriff, 2002). Partant de ce constat, ils ont proposé un autre système de simplification lexicale basé sur une ressource lexicale, ReSyf [4], dont la granularité sémantique est plus optimale.

1. https ://wordnet.princeton.edu/
2. https ://wordnet.princeton.edu
3. https ://linguistech.ca/WordNet$_{FE}XCERTT_I$
4. https ://cental.uclouvain.be/resyf/

Le système de simplification, réalisé par François et al., (2016), peut être comparé à celui de Caroll et al., (1998) étant donné que tous les deux utilisent une ressource lexicale pour la génération de candidats. Pourtant, deux différences importantes sont à souligner dans leurs méthodologies respectives :

1. *ReSyf* est une ressource lexicale en français avec des synonymes qui sont classés selon leur degré de difficulté et une opération de désambiguisation a été effectuée tandis que *Wordnet* est une base de données lexicales en anglais dont nous pouvons également extraire des synonymes mais ceux-ci n'ont été ni classés, ni désambiguïsés.
2. *Wordnet* est seulement utile pour la tâche de génération de candidats tandis que *ReSyf* remplit à la fois les tâches de génération, de sélection et de classement.

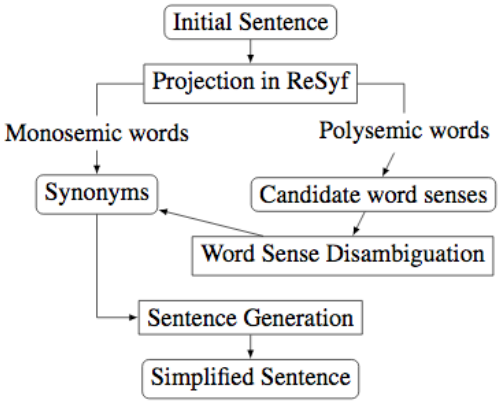


FIGURE 2 – Simplification lexicale à partir de Resyf, (Hmida et al., 2019)

La figure 2 représente le pipeline du système de simplification lexicale à partir de Resyf. Pour obtenir les synonymes, les mots complexes sont projetés dans Resyf. Lorque le système rencontre des mots monosémiques (= un seul sens), une liste de candidats classés en fonction de leur difficulté est obtenue tandis que dans le cas de mots polysémiques (= plusieurs sens), ceux-ci doivent passer par une étape de désambiguïsation afin de sélectionner le sens le plus approprié (Hmida et al., 2019). Nous verrons plus en détails les méthodologies adoptées pour sélectionner les sens et classer les candidats dans les pages suivantes.

2.2.2 Les corpus parallèles

Il n'est pas toujours aisé de trouver des bases de données lexicales dans une autre langue que l'anglais ou alors faut-il la créer comme c'est le cas pour Resyf. Horn et al., (2014) ont par ailleurs trouvé une solution pour pallier ce problème en utilisant des corpus parallèles : ils ont aligné des paires de phrases provenant du Wikipédia anglais et sa version simplifiée (English Wikipedia) afin d'en extraire des règles de

simplification. Biran et al. (2011) ont également exploité les articles de Wikipedia et du Wkipedia simple mais leur approche ne requiert pas d'alignement ou de correspondance spécifique. En alignant les phrases, Horn et al. (2014) espèrent apprendre un plus grand nombre de règles. Par conséquent, étant donné une phrase normale (régulière) et sa version simplifiée, les candidats à la substitution sont extraits lorsqu'un mot dans la phrase simple correspond à un mot différent dans la phrase normale. Cependant, tous les mots alignés ne sont pas nécessairement des variantes lexicales acceptables. Horn et al. (2014) utilisent pour cette raison des filtres en supprimant les paires où le mot normal apparait dans une "stop list" puisque les mots figurant dans celle-ci sont déjà considérés comme simples : en ne gardant que les mots ayant les mêmes parties du discours et en supprimant les candidats qui sont des noms propres puisqu'ils ne doivent pas être simplifiés.

2.2.3 Les plongements lexicaux

Glavas et Stajner (2015) proposent d'exploiter les *words embeddings* ou plongements lexicaux. Partant du principe que des ressources lexicales ou corpus parallèles ne sont pas toujours disponibles, ils prévilégient une approche qui requiert uniquement un assez large corpus de textes réguliers (en opposition aux textes simplifiés) : il serait possible de trouver des synonymes plus simples à condition d'avoir des techniques fiables pour prédire la complexité des mots et leur similarité sémantique . Ces dernières années, les words embeddings ont le vent en poupe en Traitement Automatique du Langage et pour cause : il est possible de faire des rapprochements sémantiques entre des mots qui ne se retrouvent pas forcément les uns à côté des autres dans le corpus d'entrainement.

> La technique des word embeddings consiste à projeter les mots d'une langue (contenu dans une fenêtre graphique définie) dans un espace de représentation vectorielle. Chaque mot est représenté par un vecteur , à n dimensions, qui correspond à une projection du mot dans un espace où les distances modélisent les relations inter-mots. Cette projection permet de tirer profit des mots selon leurs sens dans une région de l'espace sémantique proche

Pour leur méthode de simplification lexicale, Glavas et Stajner (2015) ont utilisé le modèle GloVe pour obtenir des vecteurs réprésentant les mots. La similarité sémantique des mots s'obtient en calculant le cosinus de l'angle entre chaque vecteur de ces mots.

Il est intéressant d'attirer l'attention sur les conclusions que Hmida et al. (2018) ont pu faire à propos des words embeddings et de la génération de synonymes : en se référant au contexte et à la distance sémantique, les plongements lexicaux peuvent produire des mots qui ne sont pas des synonymes provenant de relations non appropriées comme l'antonymie. D'ailleurs, Monsieur Tannier lors de sa venue au Cental a insisté sur cette distinction : deux mots dans l'espace vectoriel sont deux mots qui partagent un contexte similaire, mais ce ne sont pas nécessairement des mots qui partagent un sens

proche. Nous pouvons d'ailleurs exemplifier cet argument à l'aide de la figure suivante :

$$w_{vélo} \sim w_{bicyclette} \qquad w_{chat} \sim w_{tigre} \qquad w_{bon} \sim w_{mauvais}$$
$$w_{malade} \sim w_{malades} \qquad w_{Paris} \sim w_{Londres} \qquad w_{petit} \sim w_{grand}$$

FIGURE 3 – Mots proches dans l'espace vectoriel (Tannier, 2018)

Nous remarquons, dans la figure 3, que les couples de mots dont les vecteurs sont très proches ne peuvent se substituer les uns aux autres. Pourtant, nous observons que certaines paires de mots entretiennent une relation sémantique : *bon*, *mauvais*, *petit*, *grand* sont des antonymes (mot ayant un sens contraire à celui d'un autre). Nous pouvons également pousser l'analyse plus loin et considérer que *chat* et *tigre* sont des co-hyponymes (rapport d'inclusion entre des unités lexicales, considéré comme orienté du plus spécifique au plus général) de *félin*.

2.3 La sélection de candidats

Pour cette étape, nous avons retenu deux techniques principales : une basée sur les signatures sémantiques (Hmida et al., 2019 ; Bilami et Gala., 2017) et une autre qui exploite un modèle latent de la langue (LWLM) (De Belder et Moens., 2010).

2.3.1 Signatures sémantiques

Afin de choisir le substitut au sens le plus approprié, il est possible d'avoir recours à un algorithme de désambiguïsation utilisant les signatures sémantiques des mots et des sens des mots (Hmdi et al., 2019). Ces signatures sémantiques sont basées sur les propriétés individuelles de chaque noeud dans le réseau lexical JeuxDeMots (Bilami et Gala., 2017) puisque les sens de Resyf sont les mêmes que JeuxDeMots, ils ont simplement "subi" un rafinement sémantique. L'intérêt des signatures sémantiques est qu'elles intègrent les relations entre chaque entrée lexicale. Bilami et Gala. (2017) ont listé chaque relation sémantique possible : synonymye, acception, domaine, agent, patient, hyperonymie, hyponymie et idée associée. Grâce à ces signatures, il est possible d'appliquer un algorithme dont le fonctionnement est le suivant (Hmida et al., 2019) : l'algorithme compare chaque sens de chaque mot à simplifier avec chaque mot appartenant au contexte sur base de leurs signatures sémantiques. Ensuite, deux possibilités peuvent être envisagées en fonction de l'existence d'une relation ou pas : s'il existe une relation entre les deux éléments comparés, cela signifie une parfaite similarité sinon l'agorithme calcule la similarité entre les deux signatures sémantiques en déterminant

le cosinus de l'angle entre eux. Le sens avec le meilleur score c'est-à-dire celui avec le plus haut taux de similarité sera choisi comme candidat adéquat.

Cette méthodologie peut être comparée aux recherches réalisées par Bott et al. (2012). En effet, ceux-ci exploitent aussi la distance de cosinus pour sélectionner le meilleur candidat. La méthodologie rapportée par Saggion et Graeme (2017) est la suivante : pour chaque mot complexe, le système extrait un vecteur représentant le contexte du mot et est comparé à l'ensemble des vecteurs de sens disponibles pour ce mot dans la base de données lexicales, Spanish Open Thesaurus (SOT). Celui ayant la distance cosinus la plus faible sera sélectionné. Selon Bilami et Gala (2017), la différence majeure réside dans la manière dont les poids des représentations sont calculés : le poids des signatures sémantiques est calculé à partir des propriétés structurelles du réseau lexical tandis que le poids des vecteurs de Bott et al. (2012) sont calculés à partir du contexte (cooccurences).

2.3.2 Modèle de langue latent (LWLM)

De Belder et Moens (2010) proposent une toute autre manière d'effectuer la tâche de désambiguîsation en utilisant un LWLM (Latent Words Language Model) : il utilise les réseaux bayésiens pour représenter les mots et la signification contextuelle de ceux-ci (Siddhartan, 2014). Pour chaque mot, deux listes de synonymes sont générées : une provenant de dictionnaire de synonymes tel que WordNet (*synonyms*) et une autre issue du modèle (LWLM) suite à une phase d'apprentissage pendant laquelle le modèle apprend pour chaque mot un ensemble probabiliste de synonymes et de mots à partir d'un corpus qui n'a pas été annoté (apprentissage non supervisé) (De Belder et Moens., 2010). Pour chaque mot à l'intersection de ces deux ensembles, la probabilité que ce soit un bon substitut est calculée (cf. figure 4).

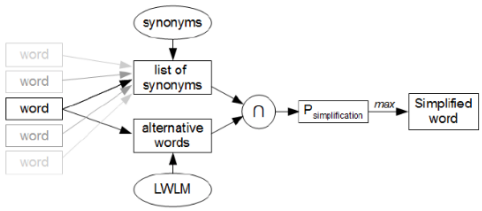


FIGURE 4 – Représentation schématique de la méthode (De Belder et Moens, 2010)

2.4 Le classement des candidats

Pour finir, nous proposons 2 recherches très diverses : l'une portant sur les machines à vecteurs de support (SVM) (Hmida et al., 2019 ; François et al., 2016) et l'autre sur un modèle de régression neural (Paetzold et Specia., 2017).

2.4.1 Machines à vecteurs de support (SVM)

Les machines à vecteurs de support sont de puissants algorithmes d'apprentissage automatique qui se basent sur un algorithme linéaire imaginé par Vladimir Vapnik et Aleksandr Lerner en 1963 (Azencott, 2018). Pour le problème de classement des candidats, François et al (2016) ont opté pour une approche appelée *pairwise* : le classement est tranformé en classification par paires (Li, 2014). Cette approche inclut plusieurs méthodes d'apprentissage dont celle de Herbrich et al. (2000) : ils ont dans un premier temps formalisé le problème de classement sous la forme d'une classification par paires pour ensuite utiliser l'algorithme SVMRank.

Puisque l'algorithme d'apprentissage demande à la fois des données d'entrainement déjà triées et des vecteurs de représentations des mots, François et al. (2016) ont utilisé la liste Manulex regroupant des mots et leur niveau de difficulté. Ensuite, chacun des lemmes de Manulex ont été représentés par un vecteur rassemblant 69 caractéristiques autant linguistiques que psycholinguistiques. A partir de ces données, la phase de réalisation de paires d'entraînement a pu être réalisée de la façon suivante (François et al., 2016) : nous choisissons deux mots de difficulté différente et nous fusionnons leurs vecteurs de caractéristiques en les soustrayant. Pour finir, comme c'est le cas pour tout modèle, il a fallu trouver les meilleurs paramètres en sélectionnant les meilleures variables linguistiques et psycholinguistiques à partir du test de corrélation de Spearman et en entrainant les SVM sur des jeux de données.

2.4.2 Modèle de régression neural

Pour cloturer ce chapitre sur les techniques, nous proposons une technique de classement à partir de l'utilisation de réseaux de neurones et plus particulièrement des *perceptrons multi-couche*. A la différence du premier réseau de neurones appelé *perceptron*, le perceptron multi-couche (pmc) possède des couches intermédiaires (*hidden layers*) entre la couche d'entrée, qui est composée de p neurones n'ayant pas de connexions entre eux correspondant chacun à une variable d'entrée, et la couche de sortie (Azencott, 2018). Etant donné que le pmc est un modèle non linéaire, des fonctions d'activation non linéaires seront également utilsées telles que la fonction tangente hyperbolique.

Paetzold et Specia (2017) ont utilisé les réseaux de neurones lors de la phase de classement des candidats à la simplification lexicale. Pour ce faire, leur méthodologie se divise en 3 étapes : la régression, l'ordonnancement et le test de confiance ("check confidence"). Pour la première partie donc, ils utilisent un pmc (cf. Figure 5) : le réseau prend en entrée une série de caractéristiques de deux candidats et produit en sortie une valeur, si celle-ci est positive alors le candidat 2 est plus simple que le 1 et si celle-ci est négative alors le candidat 1 est plus simple que le 2 (Paetzold et Specia, 2017).

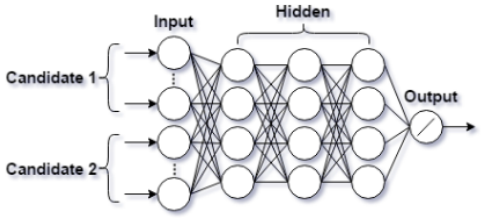


FIGURE 5 – Architecture du pmc (Paetzold et Specia, 2017)

Celui-ci a été entrainé sur des données d'apprentissage qui suivent le format suivant : une phrase, le mot à simplifier et les candidats déjà classés. Les couches intermédiaires utilisent la fonction d'activation tanh (tangente hyperbolique) et le noeud de sortie utilise une fonction linéaire. Après avoir entrainé leur modèle, Paetzold et Specia (2017) entament la deuxième étape en calculant le score final des candidats en considérant que plus le score est bas, plus le candidat est considéré comme simple. Le score est obtenu sur base de l'équation suivante (cf. Figure 5) :

$$R(c_i) = \sum_{c_j \neq c_i \in C} M(c_i, c_j)$$

FIGURE 6 – Formule d'ordonnancement (Paetzold et Specia, 2017)

Cela signifie que $M_{(c_i,c_j)}$ est la valeur estimée par le modèle pour une paire de candidats c_i, c_j générée par C. Durant l'ordonnancement, Paetzold et Specia ont calculé le score finale $R(c_i)$ de tous les candidats c_i. Pour finir, au lieu de remplacer le mot complexe par le candidat le plus simple, ils ont choisi de comparer le candidat et le mot d'origine dans son contexte en utilisant un modèle de langue. C'est pour cette raison que Paetzold et Specia ont appelé cette étape *test de confiance*.

2.5 Conclusion de la section

L'étape d'identification des mots complexes est une étape qui n'est pas toujours respectée étant donné que certaines approches considèrent chaque mot comme étant un candidat potentiel à la simplification et celui-ci est remplacé seulement s'il possède un synonyme plus simple. Par ailleurs, grâce aux modèles de prédiction automatique de compétence lexicale proposés par Tack et al. (2016) et Avdiu et al. 2019), nous avons pû mettre en évidence que les prescripteurs tels que la fréquence, le nombre de lettres ou de syllabes pouvaient être utilisés pour prédire la complexité d'un mot. Nous avons également rencontré des méthodes d'apprentissage tels que les machines à vecteurs de support ou les arbres de décision qui nécessitent des corpus annotés.

Pour la génération de substituts, nous avons retenu 3 techniques : la première utilise une ressource lexicale (François et al., 2016 ; Carroll et al., 1998), la deuxième exploite des corpus parallèles (Horn et al., 2014) et la dernière génère ses substituts grâce aux prolongements lexicaux (Glavas et Stajner., 2015). Nous avons mis en lumière l'avantage des prolongements lexicaux puisqu'ils ont la particularité de ne dépendre ni d'une base de données lexicales ni de corpus parallèles. Cependant, il a également été rapporté par Hmida et al (2019) que les prolongements lexicaux produisent parfois du bruit en raison de relations sémantiques non appropriées telles que l'antonymie. Ce constat avait également été constaté et exprimé par Tannier lors de son séminaire sur les comptes-rendus médicaux : deux mots qui sont proches dans l'espace vectoriel partagent certes un contexte similaire mais ne sont pas pour autant des mots qui partagent un sens proche. Il faut savoir opérer la distinction entre la similarité et la parenté. Des mots peuvent partager des liens de parenté (antonyme, hyperonyme, hyponyme) mais ne pas partager un sens commun.

Après avoir généré des candidats, le processus de simplication détermine quels candidats sont cohérents en contexte, pour ce faire nous avons retenu 2 techniques, l'une basée sur les signatures sémantiques et l'autre basée sur un modèle de langue latent. La première représente les mots sous forme de vecteurs et exploite la similarité sémantique entre les vecteurs des mots et les vecteurs des sens des mots dans le but de déterminer un score basé sur le plus haut taux de similarité entre les candidats. Tandis que De Belder et Moens ont choisi d'exploiter les réseaux bayésiens pour représenter les mots et la signification contextuelle de ceux-ci. La dernière phase appelée classement des candidats qui consiste à classer les candidats à la simplification peut être réalisée grâce à des machines à vecteurs de support (SVM) ou à un modèle de régression neural. Dès lors, le problème est envisagé selon deux points de vue différents soit comme un problème de classification soit comme un problème de régression.

3 La problématique des textes médicaux

Dans ce chapitre, nous tentons de mettre en exergue la problématique des textes médicaux en donnant une définition de travail des textes de spécialité et en énumérant les caractéristiques complexes et propres aux textes de santé. La simplification de textes pour les aphasiques, les dyslexiques ou bien les apprenants semble être pertinente étant donné que tous, pour des raisons différentes, présentent des difficultés de lecture. En effet, si nous prenons le cas des aphasiques, la lésion des aires de Broca ou de Wernicke provoque chez la personne atteinte, différents troubles langagiers tels que la difficulté à trouver ses mots, la réduction de la communication, l'invention de mots ainsi qu'une incompréhension à la lecture. Quant aux dyslexiques, un rapide coup d'oeil sur l'image ci-dessous suffit à nous convaincre de l'utilité d'un outil de simplification.

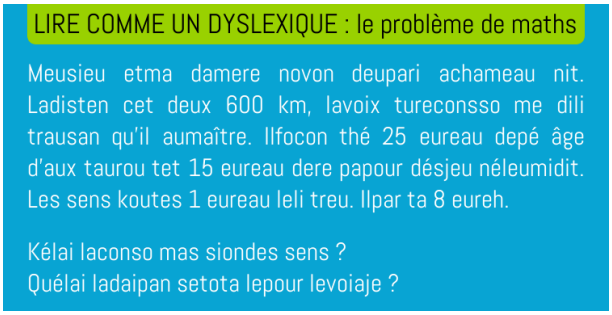


FIGURE 7 – Lire comme un dyslexique (ABC aider [5])

Mais qu'en est-il des patients pour qui nous voulons simplifier les textes de santé ? Notre but est d'interroger la littérature et de répondre à la fin de ce chapitre à la question suivante : pourquoi simplifier les textes médicaux si les lecteurs ne présentent pas de difficultés de lecture ?

Dans la section suivante, nous allons nous concentrer essentiellement sur les aspects linguistiques qui peuvent affecter la compréhension des patients. Pour ce faire, nous tenons à intégrer, dans cette introduction, un exemple de compte-rendu médical donné par Grabar et al. (2018) afin d'amorcer la section suivante :

> Un patient âgé de 77 ans consulte pour une asthénie, un amaigrissement de 8kg en 6 mois et une toux persistante depuis 2 mois. Il est né et vit au Bénin. Il a comme antécédents un diabète de type 2 traité par Metformine. Il n'a aucun antécédent d'asthme ni d'atopie. Absence de signe digestif. Un bilan biologique réalisé il y a 2 mois retrouve une hyperéosinophilie à 23 036/mm3. Le reste de la NFS est normal. L'examen clinique est normal. Il n'y a pas d'hyperréactivité bronchique aux EFR. Un traitement par corticoïdes inhalés est inefficace. Les sérologies parasitaires et examen des selles sont négatifs. Les PNE restent élevés à 28 000/mm3 après traitement antiparasitaire. (...)

Nous pouvons déjà mettre en lumière plusieurs particularités : beaucoup de chiffres, d'acronymes/sigles (NFS, EFR, PNE), de noms de médicaments (Metformine), de termes qui semblent appartenir à la langue médicale (atopie, asthme, corticoïdes, etc). Cependant, certains sont faciles à comprendre tels que *diabète* ou *asthme* tandis que d'autres sont opaques comme *hyperéosinophilie* ou *sérologie*. Pour rendre compte de ces différentes caractéristiques, nous allons procéder à un classement des différents types de lexèmes appartenant à la langue médicale. Mais avant, nous tenons à faire un bref état de l'art sur la question des textes de spécialité étant donné qu'un texte médical est avant tout un texte de spécialité.

3.1 Les textes de spécialité : généralités

Ce n'est un secret pour personne, et c'est d'ailleurs ce qui vient certainement à l'esprit de tout un chacun lors de l'évocation des textes de spécialité, ceux-ci riment avec complexité. C'est également un des arguments avancé par Thoiron (1991) : les textes de spécialité ont la réputation d'être difficiles à lire par le novice. Cette difficulté est loin d'être attribuée à tort et nous verrons plus tard plusieurs raisons pour lesquelles ces textes sont jugés compliqués. Par ailleurs, cette réflexion de Thoiron est intéressante parce qu'elle implique que l'on se questionne sur ce qu'il appelle la *compétence conceptuelle* de la part du lecteur. Cette compétence est la première raison de la difficulté des textes spécialisés selon lui. Pour l'illustrer, Thoiron(1991) utilise cet exemple suivant :

> Un article consacré à la transmission du VIH est probablement plus accessible à un médecin qu'à un spécialiste du théâtre brechtien. Mais la situation serait inversée avec une publication traitant du Verfremdungseffekt.

Cet exemple met en lumière le fait même que les textes techniques demandent une mobilisation importante de savoirs et concepts du domaine en question pour comprendre le lexique spécialisé de ceux-ci. Par ailleurs, ces lexiques sont composés de **termes**, que nous définissons à l'aide de la définition suivante que nous avons trouvé dans un article de Mortureux (1995) :

> Le terme est une unité lexicale définie dans les textes de spécialité. L'ensemble des termes s'appelle la terminologie [...] Les termes sont [...] des unités lexicales **dont le sens est défini par les spécialistes** dans les textes de spécialité. (Kocourek 1982, p. 77)

Les termes techniques sont créés par des experts, pour désigner de nouveaux concepts qui sont liés à un domaine particulier. Mais, il s'agit de nuancer, toujours à l'aide des propos de Mortureux (1995) :

> terme scientifique : milieu homogène de spécialistes de même culture, de même formation [...] communication à un haut niveau d'élaboration conceptuelle et de rigueur dans l'analyse [...] énonciation écrite [...] monovalence

> [...] Les **termes employés ne s'étendent pas à la communauté linguistique tout entière**, sauf si tel concept vient à jouir d'une extension considérable, auquel cas il cesse d'être proprement un terme scientifique. **Les techniciens représentent une catégorie de locuteurs non homogène [...]**

Ces propos mettent en lumière la particularité et la complexité des textes de spécialité : un terme usité par un communauté d'experts peut jouir d'une extension et cesser d'être opaque pour un public non expert. Par conséquent, l'opacité d'un terme n'est pas une condition sine qua non pour être considéré comme tel. Autrement dit, le lexique spécialisé n'est pas composé **uniquement** de termes qui sont compris seulement par un public expert. C'est d'ailleurs un constat que nous avions fait dans l'introduction de ce chapitre : nous comprenons parfaitement ce que signifie les termes *diabète* et *asthme* alors que nous avons du mal à saisir les sens de *hyperéosinophilie* et *sérologie*. Cette difficulté devra être prise en compte lors de la tâche d'identification de termes complexes.

Conséquemment, notre souhait est d'arriver à une vulgarisation des textes médicaux en les rendant accessibles et compréhensibles par des non spécialistes, par une communauté linguistique toute entière, en remplaçant un terme technique par un terme issu de la langue générale ou par un autre terme technique qui a profité d'une extension. Dans ce cas, la simplification de textes spécialisés nous paraît légitime et permet d'apporter un élément de réponse à la question suivante :"*Pourquoi simplifier les textes médicaux si les lecteurs ne présentent pas de difficultés de lecture ?*"

3.2 Les textes médicaux

Grâce à la section précédente, nous avons pu mettre en lumière que les textes de spécialité possèdent des termes techniques qui peuvent poser problème lors de la lecture de ceux-ci. Par conséquent, puisque notre recherche a pour thème les textes médicaux, il s'agit d'étudier de plus près leur terminologie. Est-elle nécessairement synonyme de complexité ? Afin de répondre aux mieux à nos interrogations, nous avons puisé dans la littérature scientifique en vue d'étudier plus en détails la terminologie médicale et les problèmes que cela implique.

Lorsque nous lisons les conseils pour l'élaboration des notices destinées aux patients réalisée par l'ansm [7] (Agence nationale de sécurité du médicament et des produits de santé), nous pouvons trouver une série de conseils concernant le style rédactionnel des notices, ils encouragent notamment à :

- éviter tout langage complexe et, a fortiori, le jargon médical :
- transcrire toutes les informations en langage courant non spécialisé ;

7. https ://ansm.sante.fr/

— utiliser des phrases courtes ;
— accompagner les noms chimiques des substances (en particulier dans la section concernant les interactions médicamenteuses), d'une explication compréhensible de leur effet thérapeutique ;
— exprimer les effets indésirables avec un libellé compréhensible par les patients, suivi du terme médical correspondant entre parenthèses.

Conséquemment, nous remarquons que les fabriquants de produits pharmaceutiques sont encouragés à éviter un " langage complexe" qui est assimilé au jargon médical au profit d'un "langage plus simple", le langage courant non spécialisé. Ils sont également encouragés à acompagner les noms chimiques (=termes) et les effets indésibrables d'une explication **compréhensible**. Ces conseils permettent de souligner le fait que le langage médical est assimilé à un langage compliqué/opaque qui ne peut être compris par les profanes que par le biais d'explications ou de substituts de la langue courante. Cela permet également d'étayer les propos tenus auparavant selon lesquels les termes scientifiques ne sont compris que par une population de spécialistes et de nous conforter dans l'idée que le langage médical est complexe. Puisque nous savons que la complexité de la langue médicale est due aux termes utilisés, il convient de les étudier plus en profondeur.

3.2.1 L'emprunt

L'emprunt est très fréquent dans la langue médicale : nous retrouvons des termes empruntés au grec ancien (oesophage), à l'arabe (nuque), au latin(alvéole), à l'anglais (stroke center), mais également des composés hybrides c'est-à-dire des termes construits à partir de plusieurs langues (Sara et Sonia, 2013). Des solides connaissances en langues anciennes/modernes ains que des compétences pour décomposer les termes sont indispensables pour deviner le sens de ceux-ci. Nous mettons d'ailleurs au défi le lecteur de comprendre les composés hybrides suivants proposés par Serge (2001) : scanographie (provenant de l'anglais "to scan" + du grec "graphein"), alcolose (de l'arabe "al-qilyi" et du suffixe grec "-osis") ou encore tularémie (de l'éponyme "tulare" et du grec "haima"). Nous comprenons rapidement à l'aide de ces exemples que chaque patient ne possède pas nécessairement les aptitudes linguistiques pour déchiffrer et comprendre.

3.2.2 La siglaison

Les langues de spécialité, et la langue médicale n'échappe pas à la règle, contiennent des sigles (abréviation formée par une suite de lettres qui sont les initiales d'un groupe de mots) et des acronymes (substantif dont l'origine est un sigle, mais qui se prononce comme un mot ordinaire [...]). Comme le soulignent Sara et Sonia (2013), certains sigles

et acronymes ne posent pas de problèmes tant leurs formes abrégées sont connues du grand public, comme c'est le cas pour ADN, SIDA ou UV. Mais cela suppose néanmoins que le lecteur ait une connaissance des sigles internationaux et de domaines spécifiques (Sylvie, 1993), étant donné que leur sens n'est déductible d'aucune autre manière. A cela, nous pouvons également ajouter le problème de la polysémie des sigles et des acronymes illustré par Sara et Sonia (2013) :

— BAV baisse d'acuité visuelle **ou** bloc auriculo-ventriculaire
— IVG interruption volontaire de grossesse **ou** insuffisance ventriculaire gauche

Enfin, la langue médicale emprunte également des sigles anglais qui peuvent prêter à confusion. Illustrons cette confusion à l'aide des exemples de Sylvie (1993) où les deux premiers sont des sigles anglais et les deux autres français :

OS : Opening Snap (claquement d'ouverture de la mitrale)

O. S., o. s. : Oculus Sinister ; left eye ; (l'œil gauche)

OS : bouche, ouverture (mouth, opening - pas d'abréviation en anglais)

OS : Occipito-sacrée (occipito-posterior position) (position)

3.2.3 Les éponymes

Si l'on se réfère à la définition proposée par le site Larousse [8], l'adjectif *éponyme* signifie *qui donne son nom à quelque chose*. Par exemple, l'unité de mesure *volt* provient du physicien italien Alessandro Volta. Les éponymes ne sont pas nombreux dans le lexique médical puisque Sylvie (1996) chiffre leurs fréquences à moins de dix pourcents : les médecins francophones préfèrent éviter leurs emplois à cause de leur opacité. Celle-ci est toutefois relative étant donné que nous n'éprouvons aucune difficulté de compréhension à la lecture des éponymes suivants : la maladie d'Alzeihmer, la maladie de Parkinson, etc (Sara et Sonia, 2013). Nous les comprenons parce que ces maladies sont malheureusement répandues et sont rentrées dans notre vocabulaire quotidien. Par contre, lorsque nous lisons les termes tels que la maladie de Guillain-barré, le signe de Kernig et le syndrôme d'Adams-Stokes (Sara et Sonia, 2013) nous ne pouvons nous empêcher de sourciller. Par conséquent, le constat reste le même que pour les emprunts et les signes : le terme ne nous offre aucune information sur son sens si celui-ci nous est inconnu.

3.2.4 Technicismes collatéraux

Nous retrouvons cette appelation dans l'article de Sara et Sonia (2013). Selon elles, il existe deux sortes de technicismes : les spécifiques et les collatéraux. Elles désignent les

8. https ://www.larousse.fr/dictionnaires/francais/éponyme/30582

premiers comme les termes qui sont stricitement liés au domaine d'expertise comme "transaminase" renvoyant à un "enzyme sous l'action duquel s'effectue la transamination" ou "trismus" renvoyant à la "constriction des maxiliaires provoquée par la contraction des muscles masticateurs". Mais le langage médical, en plus de sa propre terminologie, emprunte également au lexique du langage courant : nous retrouverons "fièvre modérée" pour signifier "un peu de température" et "toux productive" au lieu de "toux grasse" (Sonia et Sara, 2013). Cet emprunt à la langue courante peut être également une source de difficulté pour le lecteur étant donné qu'il pourrait être tenté de comprendre les technicismes collatéraux comme des mots appartenant au langage courant, ce qui a pour conséquence de donner des phrases farfelues et incompréhensibles. En ce sens, des technicismes spécifiques ont plus de chances d'être compris étant donné qu'ils sont directement reconnaissables par leurs morphologies et qu'ils sont monosémiques.

3.2.5 La morphologie

Toujours selon Sara et Sonia, le langage médical privilégie les structures du type N Adj : adénome toxique au lieu de la phrase prédicative "l'adénome qui est toxique" ; atrophie splénique au lieu de "atrophie de la rate" ; insuffisance rénale au lieu d'une structure N de N " insuffisance des reins". Ce genre de construction peut également poser problème à un lecteur non averti puisque la structure N Adj peut être dérivée soit d'une phrase prédicative soit d'une structure de N de N, ce qui prête à confusion. De plus, certains composés sont construits en hypallage c'est-à-dire que l'adjectif s'applique à un terme absent (Sara et Sonia, 2013) comme c'est le cas pour "diabète inspide" où "inspide" se réfère aux urines et non pas au diabète.

La longueur des phrases Après avoir démontré la complexité du vocabulaire médical, il ne s'agit plus de se situer au niveau des lexèmes, mais au niveau de la phrase. Selon Maurice (2006), la phrase spécialisée est considérée comme complexe à cause de sa longueur. Il est vrai que lors de nos recherches sur la lisibilité et sur ses formules de lisibilité classiques, nous avons pu observer que la complexité est très souvent assimilée à la longueur de la phrase. En ce sens, Flesch considère que plus un mot et une phrase sont longs, plus ils seront complexes. Maurice (2006) considère les phrases scientifiques plus complexes parce qu'elles seraient plus longues que les phrases générales. Mais cette observation s'applique-t-elle également à la phrase médicale ? La réponse est non, il aurait été observé que les phrases médicales et générales sont de tailles égales. Maurice (2006) a d'ailleurs démontré que les phrases médicales contenaient moins de verbes conjugués que les phrases générales, moins de 30% de propositions que les phrases de la langue courante. Grâce à cette recherche, nous comprenons que nous devons laisser de côté la croyance selon laquelle la longueur et complexité sont intimement liées, dans le domaine médical en tout cas.

3.3 Discussion

Grâce à cette section, nous avons pu mettre en avant les difficultés auxquelles nous sommes confrontés lorsque nous lisons des textes médicaux. Ces difficultés, nous devons également les prendre en compte dans notre processus de simplification. Nous pensons que la meilleure réponse à l'opocacité des termes est de fournir des informations supplémentaires. Finalement, ce qui rend la compréhension difficile c'est le manque de savoir encyclopédique du lecteur dans le domaine médical. Pour combler ce manque, nous pourrions proposer des paraphrases, par exemple. Comme l'ont souligné les chercheuses Sara et Sonia, la morphologie de l'emprunt peut être exploitée afin de proposer un synonyme plus simple. Selon elles, ces emprunts s'accompagnent parfois de synonymes locaux construits comme des composés :

> cardiopathie – maladie du cœur ; céphalée – mal de tête ; érythrocyte – globule rouge ; myalgie – douleur musculaire

3.4 Conclusion

En analysant le lexique médical, nous avons pu constater, contre toute attente, que le langage médical est caractérisé par une extrême concision puisqu'il va privilégier des formes raccourcies comme des sigles, des acronymes ou encore des abréviations ; l'utilsation d'éponymes en vue d'éviter une myriade d'explications ; l'utilisation de structures N Adj au détriment de phrases prédicatives. Le lexique médical est compliqué parce que, comme nous l'avions supposé au départ, le sens de ses termes n'est pas déductible. Pour comprendre un texte médical,le lecteur doit mobiliser de nombreuses connaissances à la fois en langues modernes/anciennes pour déchiffrer les emprunts, en sigles internationaux pour comprendre les sigles empruntés, en médecine pour comprendre les éponymes médicaux, etc. Bref, le lecteur doit posséder un savoir encyclopédique solide qu'il ne peut acquérir seulement au cours de longues années d'étude et de pratique dans le domaine médical. Cependant, lors de notre recherche nous avons rencontré certains termes médicaux qui n'ont posé aucun problème à la lecture alors que nous ne possédons pas de connaissances dans le domaine. Pour illustrer cette observation, nous reprenons la définition de termes de Guilbert repris par Mortureux (1995) que nous avions déjà utilisée :

> Les termes employés ne s'étendent pas à la communauté linguistique tout entière, **sauf si tel concept vient à jouir d'une extension considérable, auquel cas il cesse d'être proprement un terme scientifique.**

Certaines expressions ou certains termes médicaux ont tellement été usités et entendus qu'ils ont joui d'une extension, jusqu'à rentrer dans notre langage courant. Au moyen

de cette citation, nous tenions à montrer que la frontière entre complexité et simplicité est tout à fait relative et mince : alors que deux termes paraissent semblables au point de vue morphologique (maladie d'Alzeihmer vs maladie de Guillain-Baré), l'un posera des problèmes de compréhension à cause de la non connaissance de la pathologie (maladie de Guillain-Baré) tandis que l'autre sera considéré comme simple (maladie d'Alzeihmer) puisqu'il est connu et entré dans le vocabulaire courant.

4 La simplification automatique lexicale des textes médicaux

Grâce au chapitre précédent, nous avons pu répondre à la question suivante : "*Pourquoi simplifier les textes médicaux si les lecteurs ne présentent pas de difficultés de lecture ?*" De manière générale, nous avons établi que pour comprendre un texte technique, il faut mobiliser de nombreux concepts et savoirs, qui sont les plus souvent connus uniquement des techniciens. Suivant le même raisonnement, nous avons également établi que les textes médicaux étaient caractérisés par une terminologie opaque (emprunt, siglaison, éponymes, technicismes, etc) pour les non experts. Ce chapitre est donc l'occasion de réaliser un panorama des études en simplification des textes médicaux afin d'observer les adaptations qui ont été entreprises pour traiter les spécificités de la terminologie médicale.

Pour ce faire, de la même manière que pour le chapitre portant sur la simplification de la langue générale, nous réservons une section pour chaque étape de la simplification. Pour rappel, celles-ci sont au nombre de 4 : identification des termes complexes, génération des candidats, sélection des candidats et classement des candidats.

4.1 L'identifcation des termes complexes

Durant l'état de l'art sur la simplification lexicale de la langue générale : nous avions relevé plusieurs manières d'identifier des mots complexes soit à l'aide de prescriteurs linguistiques, soit à l'aide d'un lexique ou encore à l'aide de techniques supervisées comme des machines à vecteurs de supports ou encore un arbre de décision. Qu'en est-il pour le domaine médical ?

4.1.1 Les prescripteurs classiques adaptés pour la langue médicale

Elhadad (2006) prend en compte les différentes caractéristiques du genre médical et adapte le prescripteur classique basé sur la fréquence. Celle-ci met également de côté le

mot "complexité" pour le remplacer par le mot "familiarité" qui convient davantage. Sa première hypotèse donc est d'utiliser la fréquence comme prescripteur linguistique. Cette mesure a déjà été exploitée notamment par Carroll et al. dans les années 98 : plus un mot est fréquent, plus celui-ci a des chances d'être compris/reconnu par le lecteur. Elhadad (2006) argumente en faveur de ce prédicteur de complexité au moyen du raisonnement suivant :

> Knowing that the *ReutersHealth* articles are targeted at a lay audience, we conclude that frequent terms in the ReutersHealth corpus are likely to be understood by a lay reader.

Les termes fréquents dans une ressource vulgarisée, ici *ReutersHealth*, ont beaucoup de chances d'être compris par un lecteur profane. Dans ce cas, à partir du moment où la fréquence d'un terme, calculée à partir de la somme de l'ensemble des fréquences des variantes morphologiques de ce terme, est au-dessus d'un seuil prédéterminé, il est considéré comme familier (Elhadad, 2006).

Nous avions déjà établi quelques caractéristiques du terme (défini par un spécialiste à destination d'un autre spécialiste) mais nous n'avions pas mis en lumière la caractéristique principale de celui-ci : sa monosémie. C'est par le biais de cette caractéristique que Elhadad (2006) a émis sa deuxième hypothèse. Cependant, ce prédicteur a dû être abandonné parce que des mots reconnus comme universellement compréhensibles comme "adulte" étaient, dans cette optique, considérés comme non familiers en dépit d'un nombre faible de sens (Elhadad, 2006). Celle-ci a également mis en place une heuristique très intéressante à nos yeux, laquelle considère automatiquement chaque abréviation comme étant incompréhensible. Si nous avions privilégié des formules traditonnelles, les abréviations, sigles et acronymes auraient certainement été considérés comme familiers/simples en vue de leur caractère monosyllabique d'où l'importance d'adapter les critères de lisibilité au domaine des textes à mesurer.

4.1.2 Ressources lexicales

Van den Bercken (2019) propose plusieurs travaux (Chen et Yu., 2017 ; Chen et al., 2016 ; Quenam et al., 2017) qui utilisent MetaMap, un programme qui donne un accès aux concepts du métathesaurus UMLS (Unified Medical Language) qui est lui-même une synthèse de différentes terminologies. Par exemple, si le mot à identifier se retrouve dans MetaMap alors il est considéré comme complexe.

4.1.3 Annotations d'experts

Même s'il est certain que nous nous situons du côté de la simplification **automatique** lexicale, nous jugeons utile de présenter deux techniques qui requièrent l'aide

d'humains. En effet, Chen et al. (2016) ont choisi une approche basée sur l'annotation d'experts pour deux raisons. Premièrement, les termes médicaux demandent beaucoup de compréhension même pour des personnes ayant un niveau d'étude élevé. Il faut donc des personnes qualifiées. Deuxièmement, les experts ont l'habitude de traiter avec les patients et comprennent leurs besoins.

Dans le même ordre d'idée, nous pouvons citer le travail de Zeng-Treitler et al. (2005) qui ont créé un instrument pour évaluer la connaissance du vocabulaire clinique dans le cadre du projet Consumer Health Vocabulary Initiative. Cet instrument est un questionnaire contenant 34 questions à choix multiples dont la première moitié (version A) contient des termes spécifiques et l'autre moitié (version B), la version plus connue de ces termes (cf. figures 8 et 9). Dans ce cas-ci, le questionnaire est adressé à un public non-expert. Sur base de cette enquête, les chercheurs ont pu établir des scores de familiarité des termes : une note de 1 est attribuée si le terme est correct sinon, une note de 0. Ensuite, Zeng-Treitler et al. (2005) ont estimé le score de familiarité de chaque terme de la population en faisant la moyenne de tous les scores de chaque terme.

Version A

1. A geriatric person is one who is _____.
   A. Very old
   B. lanky and good looking
   C. well groomed
   D. aggressive and loud

FIGURE 8 – Exemple A du questionnaire proposé par Zeng et al(2005)

Version B

1. An elderly person is one who is _____.
   A. Very old
   B. lanky and good looking
   C. well groomed
   D. aggressive and loud

FIGURE 9 – Exemple B du questionnaire proposé par Zeng et al(2005)

4.1.4 Machine à vecteur de support (SVM)

Pour clore cette section, nous terminerons avec une autre technique proposée également par Zeng-Treitler et al. (2005) : l'utilisation de machines de vecteurs de support. Celles-ci ont déjà été expliquées en détails à la section précédente. Le principe général est le suivant : à partir de variables linguistiques et psycholinguistiques, le classifieur détermine le score de familiarité moyen des termes basé sur des corpus de textes. Zeng-Treitler et al. (2005) utilisent la fréquence des termes dans les trois corpus de

référence (Medline, Medline plus, MedlinePlus logs), le pourcentage de mots simples de la liste de Dale-Chall et la longueur moyenne des mots.

4.2 La génération de candidats

Cette étape est fondamentale mais délicate puisqu'elle nécessite non seulement de trouver des synonymes cohérents mais ceux-ci doivent être également plus faciles et plus accessibles que les mots de départ. Cette étape est d'autant plus ardue avec les termes médicaux étant donné leur caractère technique. Dès lors, *comment les auteurs parviennent-ils à vulgariser ceux-ci ?* Pour répondre à cette question, nous proposons deux types d'approches, l'une à partir de prolongements lexicaux ("*words embeddings*") et l'autre basée sur des ressources dédiées représentées sous forme de lexiques où les termes techniques sont mis en correspondance avec les expressions non spécialisées équivalentes. (Grabar et Hamon, 2016)

4.2.1 Les plongements lexicaux ou words embeddings

Segura-Bedmar et Martinez (2017), dont le but est de remplacer les termes décrivant les effets indésirables des médicaments par des synonymes plus faciles à comprendre par les patients, utilisent comme candidats à la substitution, les mots avec les vecteurs les plus proches du terme à remplacer. Cependant, cette approche peut ne pas fonctionner avec les mots polysémiques. Les chercheurs ont donc ajouté une deuxième condition pour l'obtention des candidats à la substitution : en plus de partager un espace sémantique similaire, le candidat synonyme et le mot original doivent également partager un contexte semblable d'un point de vue sémantique. Pour ce faire, Segura-Bedmar et Martinez (2017) proposent d'utiliser la formule de Glavas et Stajner (2015) pour calculer la similarité entre un candidat à la substitution et les mots contextuels du mot original à simplifier en calculant la moyenne de la distance cosinus entre tous les mots contextuels et le candidat synonyme :

$$csim\left(s, w\right) = \frac{1}{|C(w)|} \sum_{w' \in C(w)} cos\left(v_s, v_{w'}\right)$$

FIGURE 10 – Formule calculant la moyenne de la distance cosinus (Glavas et Stajner (2015)

où s est le candidat synonyme, w est le mot original, C(w) est l'ensemble des mots du contexte de w(fenêtre de taille 3 autour de w) et vx désigne le mot vecteur d'un mot x. La limite principale rapportée par les chercheurs est qu'elle ne peut proposer que des synonymes d'un mot unique.

4.2.2 Ressources lexicales

Wikipédia peut être considéré comme étant une ressource lexicale : le wikipédia médical contient à la fois les termes techniques mais également des équivalent plus simples. Selon Vinod et al. (2014), ces termes sont reliés par des relations sémantiques comme dans l'exemple suivant : *Le rhinopharynx,* ***dit également nasopharynx ou cavum***[9]*, est la partie supérieure du pharynx (...).* Ils recensent notamment une série de propositions reliant termes et équivalents vulgarisés :

also called	commonly called	sometimes called	also termed
also known as	commonly known as	sometimes known as	previously known as
also referred to as	commonly referred to as	sometimes referred to as	colloquially known as

FIGURE 11 – Phrases reliantes ("common linking phrases")(Vinod et al, 2014)

Leur but est, donc, d'identifier et d'extraire les entités médicales ainsi que leurs synonymes à partir de phrases reliantes mais également à l'aide d'indices textuels tels que les parenthèses qui introduisent des ortographes, des abréviations et des synonymes alternatifs ; une typologie différente (gras ou en italique) ; des hyperliens renvoyant à des pages similaires (Vinod et al., 2014). Cependant, l'extraction n'est pas suffisante étant donné qu'extaire ces relations ne donne aucune information concernant la complexité des termes puisque nous ignorons lesquels sont simples et lesquels sont complexes. Pour pallier à ce problème, les chercheurs proposent d'observer la fréquence des termes : si un terme est utilisé de manière plus fréquente dans un texte professionnel, il peut être considéré comme professionel et inversément (Vinod et al., 2014). Nous pouvons obtenir ces fréquences à partir des formules suivantes :

- CHV_propensity(T) = count(T apparait dans un corpus de textes grand public)/(taille du corpus de textes grand public)
- PROF_propensity(T) = count(T apparait dans un corpus de textes professionnel)/(taille du corpus de textes professionnel)

De cette manière, un terme a plus de chance d'être compris par le grand public si et seulement si sa propension CHV est plus elevée que sa propension PROF et inversément.

9. https ://fr.wikipedia.org/wiki/Rhinopharynx

4.2.3 Exploitation de la morphologie

Hamon et Grabar (2016) proposent également une approche exploitant Wikipédia comme ressource. La raison de ce choix est la même que pour Vinod et al., (2014) : Wikipédia est une encyclopédie universelle qui a pour but de rendre accessibles des concepts techniques à des personnes non-spécialistes. Leur approche consiste à exploiter la morphologie des termes médicaux pour la création de paraphrases. Nous avions mis en exergue dans la section précédente que le langage médical était formé de termes empruntés au grec, à l'arabe, au latin mais également des composés hybrides. En les décomposant, il est possible de deviner le sens de ceux-ci comme le précisent Grabar et Hamon (2016) :

> Une des particularités de ces termes est qu'ils impliquent souvent des bases venant du latin ou du grec (comme myocardiaque formé avec une base latine myo (muscle) et une base grecque cardia (cœur), ou cholécystectomie formé avec deux bases grecques chole (bile) et ectomy (ablation chirurgicale), et une base latine cystis (vessie))(...)

Pour ce faire, trois types de données on été exploitées : les termes médicaux à expliquer provenant de la Snomed International et de la partie française d'UMLS ; les articles de Wikipédia du Portail de médecine en guise de corpus et des ressources linguistiques (morphologiques, supplétives et synonymes) pour permettre principalement de faire le lien entre les bases des composés et la langue française (Grabar et Hamon., 2016). A partir de ces données, la méthodologie suivante a été adoptée :

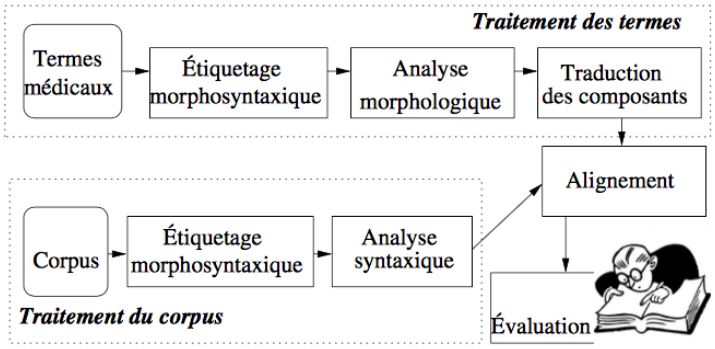


FIGURE 12 – Méthode d'extraction de paraphrases pour termes médicaux (Grabar et Hamon, 2016)

La première phase revient à étiqueter morphosyntaxiquement en contexte [10] et à lemmatiser les termes médicaux à l'aide de Cordial qui a été adapté pour ce genre de textes. Au terme de cette étape, les lemmes accompagnés de leurs parties du discours

10. S'il existe plusieurs étiquettes pour un même terme, la plus fréquente est sélectionnée

sont obtenus. Ensuite, l'analyse morphologique se fait à l'aide de l'outil DériF qui effectue une une analyse des lemmes afin de calculer leur structure morphologique, de les décomposer en leurs composants (bases et affixes), et de les analyser sémantiquement (Grabar et Hamon, 2016). Par conséquent, chaque base et chaque affixe sont associés à une étiquette syntaxique (NOM, ADJ ou V). Si la base n'existe pas en français moderne, l'étiquette la plus probable lui est attribuée (N* (noms), A* (adjectif.)) Pour illustrer cette étape, nous reprenons les explications de Grabar et Hamon (2016) pour l'adjectif myocardique étiqueté [[[myo N*] [carde N*] NOM] ique ADJ] :

> Ainsi, l'analyse de myocardique/A indique que ce mot contient deux bases supplétives nominales myo N* (muscle) et carde N* (cœur) et un affixe adjectival -ique/ADJ.

Pour finir, grâce aux ressources linguistiques récoltées, les morphèmes sont associés à leurs équivalents français : les mots correspondants sont projetés dans Wikipédia afin d'en extraire les syntagmes contenant les paraphrases. Dans ce cas, le résultat final pour myocardique donne *muscle du coeur*.

4.3 La sélection de candidats

Pour souligner l'importance de l'étape de sélection, nous reprenons les deux limites principales du travail de Segura-Bedmar et Martinez (2017) qui proposent la génération de substituts à partir de plongements lexicaux. Non seulement cette approche ne convient pas pour les mots polysémiques mais elle peut également donner des synonymes qui ne possèdent pas la même catégorie de discours que le mot à remplacer. C'est pour cette raison que nous devons passer par une étape de sélection des substituts.

Pour rappel, pour pallier aux problèmes énoncés précédement, Segura-Bedmar et Martinez (2017) ont proposé de comparer les contextes des candidats à la simplification et du mot original afin de minimiser le risque de sélectionner des synonymes hors contexte. Paetzold et Specia (2016), quant à eux, ont présenté une solution à la problématique des parties du discours, en utilisant un parseur : 10 candidats pour chaque mot cible sont récupérés à l'aide du modèles de vecteurs Word2vec sur un corpus qui a été analysé par le parseur *Stanford Parser*.

Les recherches concernant cette étape de la simplification des textes médicaux est lacunaire. Notre solution serait de réutiliser les technologies dédiées à la sélection de candidats pour la langue générale, en adaptant les ressources utilisées. Par exemple, le modèle de langue latent de De Belder et Moens (2010) utilise, pour rappel, deux listes de synonymes : une provenant d'un dictionnaire de synonymes et une autre issue du modèle suite à une phase d'apprentissage non supervisé. Dès lors, nous pourrions utiliser le métathésaurus UMLS adapté à la langue médicale proposant des paires de syonymes comme embolie ;thrombose (Grabar et Hamon, 2016).

4.4 Le classement des candidats

A cette étape du processus de simplification, nous nous retrouvons avec un ou plusieurs candidats qui sont aptes à substituer le mot original parce qu'ils vérifient deux conditions : ils sont synonymes et possèdent le même sens que le mot original étant donné l'étape précédente. Dans le cas où plusieurs candidats sont proposés, nous effectuons un classement allant du synonyme le plus facile au plus compliqué grâce à diverses méthodes.

4.4.1 La fréquence

A l'image de l'étape précédente, peu de recherches ont été réalisées pour le classement des termes médicaux. Finalement, cette étape est l'inverse de la première puisque le but est non pas d'identifier le terme complexe mais de déterminer le candidat le plus facile parmi un ensemble de termes sélectionnés. Conséquemment, la fréquence peut être également utilisée pour cette phase comme c'était le cas pour Elhadad (2006) dont le postulat est le suivant : plus un mot est fréquent, plus celui-ci a des chances d'être compris par le lecteur. Nous retrouvons cette même logique dans les travaux de Vinod et al. (2014) déjà cités auparavant : si le candidat est plus fréquent dans un corpus grand public (général), alors ce candidat est considéré comme facile. De cette manière, nous pouvons envisager de comparer chaque fréquence de chaque candidat et sélectionner le candidat ayant la fréquence la plus elevée. De leurs côtés, Biran et al. (2011) employent la mesure suivante :

$$M(c) = \frac{F(c, Wikipedia)}{F(c, SimpleWikipedia)} \times ||c||$$

où F(c,C) est la fréquence du candidat c dans le corpus C et ||c||, sa longueur. (Paetzold et Specia., 2016)

4.4.2 Adaptation de technologies dédiées à la langue générale

Comme nous l'avions exprimé dans la section précédente, nous sommes convaincus qu'il est possible de réutiliser les technologies dédiées au classement de candidats pour la langue générale, en identifiant les ressources à modifier. Par exemple, pour utiliser une machine à vecteurs de support (SVM) à la manière de François et al. (2016), nous avons besoin de données d'entrainement déjà triées et des vecteurs rassemblant des caractéristiques linguistiques et psycholinguistiques. Tandis que, le modèle de régression neural proposé par Paetzold et Specia (2017) demande des données d'apprentissage au format suivant : une phrase, le mot à simplificer et les candidats déjà classés.

4.5 Résumé

Pour achever cette section, nous proposons un résumé de chaque recherche qui a été menée dans le domaine de la simplification lexicale des textes médicaux. Concernant la tâche d'identification des mots complexes, nous avions conclu qu'il fallait adapter les prescripteurs classiques (longueur du mot, nombre de syllabes, etc) étant donné les caractéristiques du langage médical (emprunt, siglaison, technicismes, etc). Dès lors, nous avons déterminé 3 groupes différents de techniques qui se basent soit sur des prescripteurs classiques adaptés à la langue médicale, soit sur des ressources lexicales ou bien encore des annotations d'experts. Pour illustrer la première "typologie", nous avons repris les travaux de Elhadad (2006). Sa première hypothèse est d'utiliser la fréquence comme prescripteur linguistique : les termes fréquents dans une ressource vulgarisée ont beaucoup de chance d'être compris par un lecteur profane. Elhadad exploite également le caractère monosémique des termes : si le mot à identifier ne possède qu'un sens, il est à priori complexe. Enfin, la chercheuse propose une dernière heuristique laquelle considère automatiquement chaque abréviation comme étant complexe. Le deuxième type d'approche utilise des ressources dédiées à la simplification. A cet égard, nous pouvons citer Chen et Yu (2017), Chen et al. (2016) qui utilisent MetaMap donnant accès aux concepts du méthasaurus UMLS. Nous avons terminé cette étape avec les approches utilisant les annotations d'experts : Chen et al. (2016) privilégient une approche humaine étant donné le caractère technique des termes médicaux tandis que Zeng-Treitler et al (2005) utilisent des questionnaires pour faire évaluer la connaissance du vocabulaire clinique dans le but d'attribuer des score de familiarité à chacun des termes évalués.

Après avoir identifié les mots complexes, il s'agit de générer des mots candidats pour les remplacer. Nous avons distingué 3 manières de produire des substituts : utiliser des prolongements lexicaux (Segura-Bedmar et Martinez., 2017), exploiter Wikipédia et ses indices textuels (Vinod et al., 2014) ou bien créer des paraphrases à partir de la morphologie des termes (Grabar et Hamon, 2016). Nous avons mis en garde les lecteurs contre les limites des prolongements lexicaux puisque cette approche ne peut pas fonctionner avec les mots polysémiques et qu'elle peut également proposer des candidats qui n'ont pas la même partie du discours que le mot à remplacer. Nous avons également mis en lumière qu'il était plus efficace de choisir des prolongements lexicaux issus de la langue générale puisqu'ils recensent des pages médicales mais pas uniquement, contrairement à des prolongements lexicaux spécifiques tels que PUBMED. Par la suite, nous avons été interessé par les recherches de Vinod et al. (2014) qui utilisent le wikipédia médical puisque cette ressource lexicale contient à la fois des termes techniques et des termes accessibles au grand public. A partir de ce constat, ceux-ci ont imaginé une heuristique selon laquelle il était possible d'extraire des candidats à la substitution à partir de phrases reliantes ("*common linking phrases*"), de parenthèses, de typologies différentes ou encore des hyperliens qui sont présents sur l'encyclopédie en ligne. Pour finir, Grabar et Hamon (2016) ont choisi de créer des paraphrases à

partir de la morphologie des termes médicaux étant donné qu'ils impliquent souvent des bases venant du latin ou grec et une base grecque (voire deux).

Après la génération de candidats, le processus de simplification choisit quels candidats sont cohérents en fonction du contexte. Nous avons retenu le travail de Segura-Bedmar et Martinez (2017) étant donné qu'ils proposent pour les mots polysémiques de comparer les contextes des mots à remplacer et des candidats. Tandis que Paetzold et Specia (2016) proposent d'utiliser le *Stanford Parser* comme solution à la désambiguisation. Nous avons également proposer l'idée de réutiliser les technologies dédiées à la langue générale en identifiant les ressources à changer et en proposant des équivalents pour la langue médicale.

Pour finir, la dernière phase consiste à classifier les candidats à la simplification. Pour celle-ci, nous avons réutilisé un prescripteur linguistique que nous avions déjà rencontré lors de la première phase : la fréquence. Celui-ci est exploité par Elhadad (2006) et Vinod et al.(2014) puisque tous s'appuient sur le même postulat : plus un mot est fréquent dans un corpus général, plus celui-ci a des chances d'être considéré comme facile. Finalement, à l'image de l'avant-dernière phase, nous avons également proposé de modifier des technologies qui ont été proposées dans le cadre de la langue générale, telles que les machines à vecteurs de support (SVM) et un modèle de régression neural, en choisissant des données d'aprentissage adaptées.

4.6 Conclusion de cette partie

Durant cette première partie, nous avons réalisé un panaroma des recherches réalisées dans les domaines suivants : la simplification automatique textuelle, la simplification lexicale pour la langue générale, les textes techniques et médicaux et la simplfication lexicale des textes médicaux. Ce panorama nous a démontré qu'il existait de nombreuses techniques de simplification lexicale alliant différentes sortes de technologies (prolongements lexicaux, machines à vecteurs de supports, ressources lexicales, réseaux de neurones, etc) mais il a surtout permis de mettre en lumière la complexité de la langue médicale. Tout au long de ce chapitre, nous avons appris à nous familiariser avec les particularités de cette langue opaque et concise, pleine de technicismes, d'abréviations et de sigles. C'est d'ailleurs en prenant conscience des caractéristiques de celle-ci que nous pourrons créer un outil efficient. Dès lors, la prochaine étape consiste à produire un outil exploitaible de simplification lexicale pour les textes médicaux et de langue générale à partir de *LEX*enstein (Paetzold et Specia, 2015).

Deuxième partie

Méthodologie

Après avoir réalisé un bref état de l'art sur l'ensemble des techniques de simplification pour les textes médicaux et de la langue française, il s'agit de mettre tout ce savoir en pratique. Cette deuxième partie est l'occasion d'expliquer pas à pas l'implémentation de notre outil de simplification, *FreMLy*.

FreMLy signifie "French Medicine Library", cet outil de simplification fonctionne de la même manière que la bibliotèque de Paetzold et Specia (2015) : il est composé de plusieurs modules contenant chacun une implémentation différente et fonctionne à l'aide d'un fichier au format VICTOR (modifié). Pour chaque module, nous motivons nos choix, faisons un parallèle avec les classes de *LEXenstein* et nous abordons des points théoriques quand cela est nécessaire. Nous terminons par la présentation des ressources utilisées et les détails de nos implémentations. Concernant les annexes de nos implémentations, suis aux recommandations de secrétation, nous avons mis l'ensemble de nos annexes de scripts sur clé usb. Nous avons également envoyé celles-ci à notre promoteur et notre lecteur.

Dès lors, dans un premier temps, nous présentons les formats utilisés par notre outil ainsi que les corpus test que nous avons réalisés pour tester chacun de nos modules. Par la suite, nous proposons plusieurs sections, chacune portant le nom d'un processus de la simplification lexicale : identification des mots complexes, génération de substituts, sélection de substitus et classement des substituts. Enfin, nous présentions deux sous-sections supplémentaires, l'une abordant le module final qui génère les phrases simplifiées, l'autre proposant une syntèse reprenant tous les points importants de cette section.

5 Présentation des formats

Afin de réaliser la simplification lexicale, nous avons opté pour le format VICTOR proposé par Paetzold et Specia (2015) dans LEXenstein. Nous avons dû également ajouter le format CBERT pour les besoins d'un module de génération de substituts. A l'image de LEXenstein, il faut impérativement respecter ces formats pour un bon déroulement du processus.

5.1 Le format VICTOR amélioré

Le format VICTOR a été conçu pour les tâches de substitution, sélection et de classement. Chaque ligne est structurée comme dans la figure 13 ci-dessous. Nous retrouvons dans cet ordre : la phrase, le mot complexe assorti de sa position dans la phrase, le candidat de substitution et son classement.

FIGURE 13 – Structure du format VICTOR (Paetzold, 2016)

$$\langle \mathbf{S}_i \rangle \ \langle \mathbf{w}_i \rangle \ \langle \mathbf{h}_i \rangle \ \langle \mathbf{r}_i^1{:}\mathbf{c}_i^1 \rangle \ \langle \mathbf{r}_i^2{:}\mathbf{c}_i^2 \rangle \cdots \langle \mathbf{r}_i^{n-1}{:}\mathbf{c}_i^{n-1} \rangle, \langle \mathbf{r}_i^n{:}\mathbf{c}_i^n \rangle$$

Nous tenons à souligner un changement important par rapport à *LEXenstein*. Pour la tache de génération de candidats, à l'exception du module utilisant BERT, nous demandons seulement un fichier où chaque ligne est structurée de la manière suivante : la phrase, le candidat, l'index du candidat. Nous pensons qu'il est inconfortable pour les utilisateurs de commencer à chercher des synonymes et le classement de ceux-ci pour l'élaboration de leurs fichiers VICTOR. Par ailleurs grâce aux modules *ReSyf generator, DEM generator et FastText generator*, l'utilisateur obtient en sortie un nouveau fichier VICTOR avec les phrases, les candidats, les index des candidats et leurs synonymes. C'est également pareil pour la tâche de classement.

5.2 Le format CBERT

Ce format sert à la tâche de génération de candidats et est utilisé par le module *BERT generator*. Chaque ligne, doit être structurée de la même manière que pour le VICTOR format sauf que nous attendons pas une mais deux phrases : deux phrases séparées par un "/", le mot complexe et sa position dans la phrase, le candidat de substitution et son classement.

6 Présentation des fichiers test

Dans cette section, nous présentons les différents fichiers utilisés pour tester nos différents modules. Nous avons choisi d'en faire un premier pour la langue française générale et un deuxième pour le language médical étant donné que le but poursuivi est de créer une librairie pour le français **ET** pour le langage médical. Nous avons essayé de suivre un protocole rigoureux pour la création de ces fichiers en récupérant soit des données dont nous connaissons la qualité soit en réalisant un genre d'accord inter-juge dans le but de réaliser des fichiers test de la manière la plus arbitraire possible.

6.1 Fichier test pour la langue française générale

Pour établir un fichier test pour la langue française générale, nous avons réutilisé des jeux de données que Thomas François nous a donnés. Ceux-ci contiennent des phrases, les mots potentiels à substituer et leurs synonymes assortis d'un score de 0 ou de 1 selon si le mot est jugé plus facile ou plus difficile que le mot cible par Nuria Gala et Thomas François. Nous avons réutilisé ces données non seulement parce qu'elles sont dans un format similiaire au format VICTOR mais également parce que les mots à substituer sont déjà identifiés, ce qui est un réel avantage puisque nous avons décidé de ne pas réaliser l'étape d'identification de mots complexes pour diverses raisons énoncées dans la section suivante. Par la suite, nous avons, à l'aide d'un script, modifié le format des données pour qu'il soit adéquat au format VICTOR. Nous avons également dû réaliser à partir de ces données un autre fichier au format CBERT.

Cependant, étant donné que nous avons besoin d'un minimum de 200 phrases pour tester nos modules, nous avons du modifier notre mode opératoire. Nous avons pour chaque phrase du jeu de données de Thomas François identifié -quand c'était possible -d'autres mots à substituer. Nous nous sommes concertés avec un membre de la famille, travaillant dans l'enseignement technique, pour en discuter. Nous avons réitéré ce mode opératoire avec un autre corpus qui nous avait été donné lors d'un cours en linguistique statistique. Il s'agit d'un corpus en lisibilité constitué de 13 colonnes : le nom du texte, l'ordre de la phrase dans le texte dont elle est extraite, le niveau du texte dont la phrase est tirée, la phrase, le nombre de mots, le nombre de caractères, le score moyen de lecture, le temps moyen des sujets ayant lu la phrase, la fréquence médiane des mots dans la phrase, la fréquence moyennne des mots de la phrase, le 90ème perentile et 4 autres variables dont les détails n'ont pas été fournis.

Nous sommes conscients que pour plus de rigueur scientifique nous aurions du interroger un plus grand nombre de personnes afin d'arriver à un consensus. Nous aurions pu également exploiter la machine de vecteurs que nous avons créée pour le classement pour l'identification des mots complexes. Cependant, nous pensons qu'avoir réalisé cette annotation avec une enseignante travaillant dans le technique est plus

enrichissant étant donné que celle-ci est confrontée tous les jours à des élèves ayant des difficultés de compréhension et vivant dans l'insécurité linguistique.

6.2 Fichier test pour le langage médical

Pour la consitution de ce fichier aux formats VICTOR et CBERT, nous avons choisi d'exploité CAS (Grabar et al., 2018), un petit corpus de cas cliniques, qui nous a été fourni par N. Grabar, contenant des cas cliniques tels que ceux publiés dans la littérature scientifique ou utilisés pour l'éducation et la formation des étudiants en médecine (Grabar et al., 2018). Nous avons décidé d'utiliser le modèle random de python proposant plusieurs fonctions simulant le hasard afin d'obtenir des phrases au hasard. A partir de ces phrases, à l'image de Zeng-Treitler et al. (2005) qui ont plébiscité l'aide d'annotateurs, nous avons demandé à notre entourage d'identifier les mots complexes à l'aide d'un formulaire composé par nos soins. Nous avons choisi des annotateurs non experts (qui n'ont aucune familiarité de près ou de loin avec le milieu médicaux) étant donné que notre librairie s'adresse à un public néophyte. Mais nous avons également demandé à des annotateurs experts (exerçant dans le milieu hospitalier) afin de confronter leurs questionnaires et idéalement légitimer notre recherche c'est-à-dire démontrer que, contrairement au public expert, les patients ont besoin d'une simplification des termes médicaux.

Le questionnaire que nous avons créé est composé d'une trentaine de phrases dans lesquelles nous demandons à nos 15 annotateurs- dont nous-mêmes- de surligner les mots qu'ils considèrent complexes. Ensuite, nous avons rentré toutes les informations dans un fichier. De cette manière, nous considérons comme complexes, les mêmes mots qui ont été surlignés par plus de 5 annotateurs (1/3 des participants) différents. Malgré notre minuscule échantillon, nous avons pu dégager plusieurs tendances dans nos recherches :

- Ceux qui sont dans le domaine médical : ils identifient entre 0 et 5 termes complexes parmi les 30 phrases ;
- Ceux qui ont des "connaissances médicales" parce qu'ils ont des parents dans le milieu ou qu'ils regardent des séries telles que *Dr.House* ou *Grey's Anatomy* permettant une immersion dans le domaine hospitalier : ils identifient entre 5 et 15 termes complexes ;
- Ceux qui appartiennent à un public non expert (la majorité des gens interrogés) : ils identifient plus de 15 termes complexes ;

Conséquemment, nos pouvons sans difficulté affirmer qu'une simplification des textes médicaux est utile pour des personnes n'ayant aucune connaissance médicale.

Cependant, comme c'était le cas pour le corpus de test pour la langue française, nous n'avons pas assez de phrases pour tester nos modules. Pour rappel, un corpus test minimum doit contenir envirion 200 phrases. Dès lors, nous avons réitéré l'expérience

avec un corpus trouvé sur la page personne de Natalia Grabar [11] contenant des résumés de Cochrane à destination d'un public expert ou non-expert. Cette fois-ci nous avons pris le parti d'annoter nous-mêmes : nous n'avons aucune connaissance dans le domaine médical et au moment de répondre au questionnaire, nous avions identifié plus ou moins les mêmes mots complexes que les annotateurs non experts. Nous pensons donc être un candidat objectif et idéal pour cette tâche.

7 L'identification des mots complexes

La première étape du processus de simplification est, comme nous l'avons déjà mentionné auparavant, l'identification des termes complexes. A la différence de Paetzold et Specia et de leur librairie, nous avons décidé de ne pas réaliser cette étape de manière automatique. En effet, nous avons jugé avec notre promoteur, Thomas François, qu'il était plus judicieux de s'intéresser aux autres parties du processus. Cependant, puisque nous avons plébiscité l'aide d'annotateurs, nous considérons cette étape comme réalisée manuellement.

8 Modules de génération de substituts

Après avoir identifié les termes à remplacer, il s'agit de générer des candidats potentiels pour les remplacer. Dans leur librairie, Paetzold et Specia proposent 7 façons de générer des candidats : à l'aide de plongements lexicaux (Paetzold Generator, Glavas Generator), de corpus parallèles (Kauchak Generator) de ressources lexicales (Yamamoto Generator, Merriam Generator, Wordnet Generator) ou bien grâce à une approche plus mathématique (Biran Generator).

Lors de la lecture des différents articles et codes consacrés au module de génération de LEXenstein, nous avons choisi de nous intéresser plus en détails aux plongements lexicaux ("*word embeddings*') et ressources lexicales. Nous avons choisi de ne pas réaliser de génération à partir de corpus parallèles étant donné que des recherches sont en train d'être menées à ce sujet (Cardon, 2018). Conséquemment, nous avons décidé d'exploiter, contrairement à Paetzold qui utilise Word2Vec, le modèle de représentation de langage basé sur un réseau neuronal BERT et la librairie d'apprentissage FastText qui génère des prolongements lexicaux. Nous les proposons à la place de Word2vec parce nous pensons qu'il est intéressant d'exploiter des technologies qui sont dans l'ère du temps et de mesurer leurs résultats. Concernant les ressources lexicales, nous avons été dans l'obligation de trouver des ressources françaises et spécialisées. Nous avons

11. http ://natalia.grabar.free.fr/

opté pour 2 ressources très différentes : une ressource synonymique graduelle (ReSyf) et un dictionnaire électronique (DEM).

Dès lors cette section est divisée en deux parties : chacune est basée sur la technique de génération que nous avons retenue. La première partie traite des plongements lexicaux, de manière théorique d'abord et ensuite de manière pratique en expliquant l'implémentation des différents modèles. La deuxième partie, quant à elle, aborde les 2 ressources lexicales choisies en présentant leurs compositions et leurs implémentations dans la librairie.

8.1 Les plongements lexicaux comme générateurs de candidats

Cette technique a déjà été abordée dans la partie théorique. En guise de rappel, nous reprenons les propos de Bilami et Gala (2017) illustrant le principe des prolongements lexicaux : cette technique consiste à projeter les mots d'une langue dans un espace de représentation vectoriel de sorte que chaque mot soit représenté par un vecteur à n dimensions. Grâce à cette méthode, il est possible de trouver des mots ayant des sens proches grâce à leur proximité dans l'espace sémantique. Cette similarité sémantique s'obtient en calculant le cosinus de l'angle entre chaque vecteur de mot.

8.1.1 Les plongements lexicaux selon Glavas et Stajner (2015)

Théorie Paetzold et Specia se sont basés sur l'article de Glavas et Stajner (2015) pour la création de la classe *GlavasGenerator*. Conséquemment, il s'agit de prendre connaissance de la méthodologie adoptée par les chercheurs. D'après Glavas et Stajner, il ne serait pas nécessaire d'aligner des phrases assorties de leurs versions simplifiées pour la génération de candidats à la substitution. En effet, ceux-ci postulent qu'il est possible de trouver des synonymes plus simples de mots complexes dans des corpus réguliers (en opposition à des corpus techniques) en exploitant les prolongements lexicaux. Pour obtenir les représentations vectorielles des mots, Glavas et Stajner ont utilisé GloVe et ont entrainé leur modèle sur le *Wikipédia* anglais ainsi que sur la ressource anglaise *English Gigaword*. Pour chaque mot w, 10 candidats- en excluant les dérivations de w -sont sélectionnés sur base des critères suivants :

1. **La similarité sémantique** : Comme cela a été mis en évidence ci-dessus, la similarité se mesure à l'aide du cosinus de l'angle entre le vecteur du mot original et celui du candidat.
2. **La similarité contextuelle** : Etant donné que GloVe est un modèle qui ne prend pas en compte le contexte, il ne peut pas traiter les mots polysémiques. Pour cette raison, Glavas et Stajner proposent une formule qui prend en compte

le contexte en calculant la moyenne de la distance cosinus entre le candidat à la substitution et le contexte du mot à simplifier. Cette formule a déjà été explicitée dans la section précédente.

3. **Le contenu de l'information** : Partant du constat de Devlin et Unthank (2006) selon lequel informativité et complexité sont liées, Glavas et Stajner considèrent qu'un candidat simple est un candidat qui est peu informatif. Ils obtiennent cette mesure à partir de la fréquence, comme en témoigne la formule suivante : La mesure finale s'obtient en faisant la différence entre les contenus informationnels du mot original et du candidat.
4. **Un modèle de langue** Pour finir, un modèle de langue est utilisé pour la raison suivante : un candidat est susceptible d'être un bon candidat s'il se situe dans une séquence de mots précédant et suivant le mot original. Cette idée est illustrée à l'aide des propos suivants : Si $w_{-2}w_{-1}ww1w2$ représente le contexte du mot original w, alors c est un bon candidat à la substitution pour w si $w_{-2}w_{-1}ww1w2$ est une séquence probable selon le modèle de langue. Concernant la ressource, Glavas et Stajner ont opté pour le modèle de langue Berkeley (Pauls et Klein, 2011).

8.1.2 Quels outils pour notre bibliothèque ?

Les outils et ressources Après s'être imprégné de la méthodologie de Glavas et Stajner, il s'agit de s'intéresser aux ressources et outils utilisés par *LEXenstein* pour sa classe *GlavasGenerator*, pour la mise en place de notre bibliothèque. Nous avons pu mettre en lumière que l'unique ressource dont nous avons besoin est un modèle entrainé sur n'importe quel type de corpus mais dont le format doit être supporté par Word2Vec. Cependant, nous considérons qu'il existe d'autres techniques pour apprendre des prolongements lexicaux qui sont plus récentes et qui, à la différence de word2Vec, capturent le contexte des mots comme BERT et ELMo. Nous remarquons également, qu'à la différence de Glavas et Stajner (2015), Paetzold n'utilise pas le contenu de l'information et n'utilise pas de modèle de la langue pour la génération de candidats. Autrement dit, le système repose uniquement sur les mesures de similarités sémantique et contextuelle entre vecteurs. Par ailleurs, celui-ci utilise aussi des outils tels qu'un stemmeur et un lemmatiseur afin de supprimer les candidats qui seraient des dérivations du mot à substituer.

Les données d'entrainement Par conséquent, notre souhait est d'exploiter les prolongements lexicaux pour la générations de candidats en utilisant les modèles FastText et BERT. Outre l'implémentation de ceux-ci, il s'agit également de se poser des questions sur les données d'entrainement. Nous pouvons soit choisir de créer nos propres données ou alors d'utiliser des modèles préentrainés. Puisque pour avoir de bons résultats, les plongements lexicaux demandent des données entrainées sur d'immenses

et de nombreux corpus, nous avons pris le parti d'utiliser les modèles préentrainés qui sont proposés. D'un point de vue linguistique, nous avons établi, contrairement à notre première idée, qu'il était préférable d'utiliser des corpus généraux tels que Wikipidéa et non des corpus spécialisés : en optant pour des corpus spécialisés, les candidats seront également spécialisés. Or, il ne faut pas perdre de vue que notre objectif est de simplifier, nous devons être capable de trouver dans les corpus des synonymes plus simples.

8.1.3 BERT comme générateur de candidats

Présentation BERT (Bidirectional Encoder Representations from Transformers) est un modèle de représentation du langage basé sur un réseau neuronal qui a été créé par Google en 2018. BERT est utilisé à des fins très diverses que ce soit pour la traduction, pour la génération automatique de résumés de textes ou bien comme parseur/taggueur [12]. Concernant notre recherche, nous voulons exploiter ce modèle pour la création de représentations vectorielles et expérimenter les mesures décrites par Glavas et Stajner (2015) : la similarité sémantique et la similarité contextuelle.

Son architecture En choisissant BERT, nous devons inévitablement parler du *Transformer*, une architecture de réseaux de neurones récente. L'objectif n'est pas de comprendre celle-ci de manière profonde mais plutôt d'avoir un bref aperçu sur ce qui constitue l'innovation principale de BERT. Pour ce faire, nous avons choisi d'interroger des blogues spécialisés pour avoir accès à des explications vulgarisées. [13]. Le but du Transformer est de transformer une suite de tokens en une suite de longueur différentes de vecteurs. Pour ce faire, son architecture est composée d' :

1. Un encodeur (utilisé en phase de préentrainement) : celui-ci est composé lui-même de plusieurs petits encodeurs empilés qui contiennent chacun des couches *self-attention*. Le rôle de celles-ci est de détecter après entrainement les relations sémantiques entre les vecteurs reçus en entrée, lesquelles sont ajoutées aux vecteurs de sortie. Ceux-ci alimente un simple réseau de neurones dense qui ajoute une couche d'abstraction à l'encodeur [14].
2. Un décodeur : De la même manière que l'encodeur, le décodeur est composé de plusieurs petits décodeurs.

Il est également important de souligner la différence entre un réseau de neurones récurrents et l'architecture *Transfomer*, le premier définit l'ordre des tokens en entrée tandis que le deuxième doit recevoir explicitement une information de position concernant chaque token dans son prolongement lexical.

12. https ://lesdieuxducode.com/blog/2019/4/bert–le-transformer-model-qui-sentraine-et-qui-represente
13. https ://weave.eu/deep-transfer-learning-nlp-revolution/
14. https ://weave.eu/deep-transfer-learning-nlp-revolution/

Préentrainement bidirectionnel Cette architecture permet de réaliser un pré-entrainement bidirectionnel profond. Contrairement aux autres modèles (ELMo, par exemple) qui définissent le contexte d'un mot par les mots voisins se trouvant soit à droite soit à gauche de celui-ci, BERT propose de prendre en compte à la fois les voisins de droite et de gauche. Pour illustrer ces propos, nous reprenons l'exemple proposé dans la présentation du github officiel de BERT [15] traduit et adapté en français. La phrase qui fait office d'exemple est la suivante : "J'ai fait un dépôt au guichet." :

> Les modèles précédents : La représentation unidirectionnelle de "**dépôt**" est seulement basée sur "**J'ai fait un**" mais pas sur "**au guichet**".
>
> BERT : La représentation bidirectionnelle utilise à la fois le contexte gauche "**J'ai fait un** " et le contexte droite "**au guichet**" de **dépôt**.

Pour arriver à un tel résultat, les créateurs de BERT, Devlin et al. (2019) ont choisi de masquer 15% des mots en entrée, d'exécuter la séquence entière par le biais du Transformer et de prédire les mots masqués. Une fois de plus, les chercheurs de chez Google expliquent ce procédé à l'aide de l'exemple suivant que nous avons également traduit en français :

> Entrée : l'homme est allé au [MASK1]. il a acheté un [MASK2] de lait.
>
> Propositions : [MASK1] = magasin ; [MASK2] = gallon

A partir de cet exemple, nous avons été convaincu de l'utilité de cette architecture pour notre système de simplification et nous avons interrogé la littérature scientifique pour trouver une méthodologie exploitant le Transformer. Pour leur système, Quiang et al (2019) ont choisi de tirer parti de cette architecture bidirectionnelle et de ces mots masqués. Leur idée, que nous résumons très sommairement, est la suivante : utiliser BERT et son architecture pour cacher le mot compliqué et utiliser le mot prédit comme synonyme. Conséquemment, nous avons décidé de tirer parti des innovations de BERT en nous inspirant de la méthodologie proposée par Quiang et al (2019) et de nous éloigner de la méthodologie proposée par Glavas et Stajner (2015) utilisée par Paetzold.

BERT Generator Nous avons opté pour la bibliothèque python Pytorch qui propose notamment une réimplémentation de BERT et des modèles préentrainés [16]. Celle-ci est, selon Devlin et al. (2019), capable de reproduire leurs résultats et est donc fiable. Nous avons pris le code de base proposé par la librairie et nous l'avons ensuite adapté à nos besoins. Nous avons également du créer un nouveau format, en plus des 2 déjà existants, pour se prêter au mieux au modèle BERT.

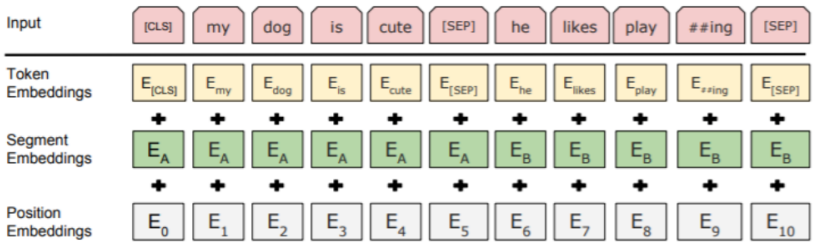
L'implémentation se déroule en 2 étapes : la phase de tokénisation en utilisant le tokéniseur du modèle multilingue pré-entrainé et l'utilisation du *Transformer* avec le langage masqué pré-entrainé. Pour que le tokénizeur fonctionne, celui-ci doit prendre

15. https ://github.com/google-research/bert/blob/master/README.md
16. https ://pypi.org/project/pytorch-pretrained-bert/

en entrée des phrases selon un format particulier. Par conséquent, 3 "traitements" sont opérés :

1. La première phrase doit commencer avec le jeton **[CLS]** et chaque fin de phrase doit se terminer par le jeton **[SEP]**
2. Masquer le token que que nous voulons prédire grâce à son index
3. Définir les indices de phrases A et B associés aux 1ère et 2ème phrases

FIGURE 14 – Représentation de l'entrée du tokenizeur BERT [17]



Ensuite, il ne reste plus qu'à pré-entrainer le modèle multilingue pour la classe *BertForMaskedLM* dans le but de prédire le token masqué dont nous avons renseigné l'indice dans la première partie du code.

Limitations Le problème, auquel nous avons été confronté, est le vocabulaire limité du modèle BERT multilingue. Pour information, celui-ci est composé de 104 langues et a été entrainé sur l'ensemble de chaque Wikipédia (entier) correspondant à chacune des langue [18]. Ce modèle a titillé notre curiosité : comment un modèle multilingue composé d'autant de langues peut-il donner de bons résultats ? Pour rappel, nous avons priviligié un modèle préentrainé puisque pour avoir de bons résultats, il nous faudrait un nombre beaucoup trop important de corpus. Pour illustrer notre problème, nous prenons les phrases suivantes : *Que penses-tu de mon médecin ? J'espère qu'il saura m'aider pour mon traitement et ma convalescence.*

BertTokenizer fonctionne selon le principe de *Word Piece Tokenization* : chaque mot est divisé en morceaux qui font partie du vocabulaire du modèle. Si le mot en entier n'est pas présent dans le vocabulaire comme le terme *convalescence*, alors le tokenizeur recherche le préfixe qui est le plus long dans le vocabulaire tel que '*con*', '*##vale*', '*##sce*', '*##nce*' où "#" signifie la continuation du mot. Mais le problème est le suivant, si nous voulons prédire le mot *convalescence* dans les phrases suivantes : "[CLS] Que penses-tu de mon médecin ? [SEP] J'espère qu'il saura m'aider pour mon traitement et ma convalescence [SEP]", le mot à prédire est à la 25 ème position-

18. https ://github.com/google-research/bert/blob/master/multilingual.md

puisque nous comptons à partir de 0- mais si nous regardons la sortie du tokeniseur, nous nous retrouvons avec le découpage suivant :

FIGURE 15 – Exemple de sortie du module BERT_generators

```
['[CLS]', 'Que', 'pense', '##s', '-', 'tu', 'de', 'mon', 'médecin', '?', '[SEP]', 'J', "'", 'es', '##p', '##ère', 'qu',
"'", 'il', 'sau', '##ra', 'm', "'", 'aider', 'pour', 'mon', 'traitement', 'et', 'ma', 'con', '##vale', '##sce', '##nce',
'[SEP]']
```

Par conséquent, BERT prédira un mot pour *mon* et non pour *convalescence*. Puisque celui-ci ne se trouve pas dans le vocabulaire, il est décomposé en plusieurs sous-mots qui se trouvent respectivement aux indices 29, 30, 31 et 32. Si nous faisons l'hypothèse de prédire l'ensemble des sous-mots composant le terme *convalescence* et d'ensuite reformer un mot à partir des sous-mots prédits, nous n'aurons pas un mot valide puisque BERT va devoir prédire à partir de sous-mots *"con","vale","sce" et "nce"* qui n'appartiennent pas à la langue française.

Notre première solution, certes intuitive mais naïve, a été d'utiliser un autre tokeniseur. Cependant, même si celui-ci ne découpe pas *convalescence* en plusieurs sous-mots, le token reste toujours absent du vocabulaire. Le problème ne se trouve donc pas au niveau du découpage mais au niveau du vocabulaire. Conséquemment, la solution serait de mieux gérer le cas où le tokenizeur rencontre des tokens inconnus en étendant le vocabulaire. Dans cette optique, nous avons épluché le web à la recherche de diverses solutions. Nous avons pû mettre en lumière que Pytorch et ses modèles préentrainés ne laissaient pas la possibilité d'étendre le vocabulaire déjà existant. Dans ce cas, nous nous sommes dirigés vers le GitHub implémentant BERT dans Pytorch [19] qui donne la possibilité de préparer son corpus, construire son vocabulaire à partir de son corpus et construire son propre modèle. Cependant, nous n'avons pas en notre possession un corpus assez volumineux pouvant prétrendre à de meilleurs résultats que ceux du modèle préentrainé multilingue. Nous pouvons donc en conclure que pour améliorer notre librairie, nous devrions préentrainer notre modèle sur un corpus volumineux exclusivement français (contrairement au modèle multilingue utilisé), composé à la fois de textes de spécialité et de textes issus de la langue générale.

8.1.4 FastText

Présentation FastText est une librairie d'apprentissage qui génère des prolongements lexicaux mais qui permet également la classification de documents. Dans ce mémoire, nous l'utilisons pour sa première fonctionnalité. L'avantage de FastText, en

19. https ://github.com/codertimo/BERT-pytorch

comparaison avec Word2vec, est qu'il fonctionne selon un principe d'embeddings de n-grams c'est-à-dire qu'un mot est représenté par la somme des vecteurs de ses n-grams (Pierrejean, 2019). L'intérêt d'une telle technique est double : nous pouvons obtenir une représentation vectorielle pour un mot qui n'existe pas dans le corpus d'entrainement ou pour un mot rare. Concernant les aspects plus techniques, FastText utilise 2 algorithmes : CBOW et Skip-Gram. Le premier prédit la probabilité qu'un mot apparaisse étant donné les mots qui l'entourent tandis que le deuxième prédit des contextes à partir du mot cible. Cependant, puisque FastText est programmé en C++, nous avons opté pour son implémentation dans *Gensim*.

Gensim et FastText Pour ce paragraphe, nous avons repris et traduit les informations qui se trouvent sur le site de *Gensim*[20] :

> Gensim contient une implémentation C rapide et native de Fasttext avec des interfaces Python. Ce module prend en charge les modèles de chargement formés avec l'implémentation fastText de Facebook. (...) [Nous avons téléchargé] des vecteurs de mots pré-entrainés pour le français qui ont été formés avec CBOW avec des poids de position, de dimension 300, avec des n-grammes de caractère de longueur 5, une fenêtre de taille 5 et 10 négatifs[21].

FastText generators (Annexe) Nous avons donc décidé d'utiliser la librarie *gensim*[22] pour apprendre la représentation des mots via FastText puisque celle-ci en offre une manipulation aisée. Dans un premier temps, nous avons exploité un module de Gensim appelé **KeyedVectors** dont nous avons traduit la description en français[23], celui-ci implémente les vecteurs de mots et leurs recherches de similarité :

> Les vecteurs de mots formés étant indépendants de la manière dont ils ont été formés (Word2Vec, FastText, WordRank, VarEmbed, etc.), ils peuvent être représentés par une structure autonome, telle que mise en œuvre dans ce module. La structure s'appelle «KeyedVectors» et consiste essentiellement en un mappage entre entités et vecteurs. Chaque entité est identifiée par son identifiant de chaîne. Il s'agit donc d'un mappage entre str => 1D numpy array.

Grâce à ce module, il est facile d'obtenir les mots correspondant à ces entités. Cependant, notre but est d'obtenir les mots qui sont les plus proches dans l'espace sémantique du mot complexe. Pour ce faire, nous avons également utilisé la fonction *most_similar* de Gensim prenant en argument le mot cible et le nombre de mots que nous voulons retenir. A cette étape du processus, nous nous retrouvons avec des mots assortis de

20. https ://radimrehurek.com/gensim/models/fasttext.html
21. https ://fasttext.cc/docs/en/crawl-vectors.html
22. https ://radimrehurek.com/gensim/models/fasttext.html
23. https ://radimrehurek.com/gensim/models/keyedvectors.html

leur score de similarité. En réponse à ce mauvais format, nous avons créé une fonction nettoyant la sortie afin de récupérer dans une liste les mots générés. Par la suite, nous avons récupéré exclusivement les candidats partageant la même partie du discours que notre mot complexe, bien que le modèle propose la plupart du temps des candidats de même nature que mot complexe. Pour étiqueter nos mots en fonction de leurs parties du discours, nous avons opté pour la librairie *spaCy*[24]. Enfin, et ça sera le cas pour tous les modules, nous transformons chaque liste de candidats de chaque mot complexe sous forme de chaînes de caractères de sorte à pouvoir les imprimer dans le format VICTOR adéquat. En sortie, nous obtenons un fichier où nous lisons pour chaque ligne : une phrase, un mot complexe, la position du mot et les candidats potentiels à la simplification.

Limitation La principale limitation est d'ordre "matérielle" c'est-à-dire que le temps pour charger le modèle préentrainé est extrêmement long. Outre une utilisation volumineuse du processeur, charger un modèle demande également beaucoup de mémoire. Dans ce cas, nous avons pris le parti de choisir un modèle au format texte étant donné que celui-ci contient uniquement pour chaque ligne un mot suivi de son vecteur.

24. https ://spacy.io/

8.2 Les ressources lexicales comme générateurs de candidat

La librairie *LEXEnstein* propose également de générer des substituts à partir de ressources lexicales. Celle-ci offre plusieurs classes qui sont toutes articulées autour d'un type de ressources lexicales : YamamotoGenerator (qui utilise comme input une clé pour l'API du dictionnaire Merriam), MerriamGenerator (qui utilise comme entrée une clé pour l'API du thésaurus Merriam) et le WordnetGenerator (qui extrait un dictionnaire reliant les mots à leurs synonymes). L'objectif est de rechercher une ressource lexicale en français, exhaustive et assez spécifique pour contenir des termes issus du domaine médical.

Concernant la méthodologie, à l'instar des prolongements lexicaux, nous avons décidé d'établir notre propre heuristique, laquelle sera expliquée en même temps que la présentation des ressources lexicales que nous avons utilisées. Etant donné que chaque classe a été codée en fonction des spécifités des ressources choisies et qu'aucune ressource ne convient pour le français et pour la langue médicale, nous sommes obligés de créer nos codes nous-mêmes.

8.2.1 Resyf, une ressource lexicale en français

Motivations L'enjeu pour ce module est, comme nous l'avons expliqué ci-dessus, de trouver une ressource en français qui puisse générer des synonymes **plus faciles**. Dès lors, notre première intuition a été de se diriger vers une ressource de synonymes, laquelle nous semble la plus cohérente pour cette tâche. Notre choix s'est porté vers *ReSyf*, que nous avons déjà rencontré à plusieurs reprises dans ces recherches, non seulement parce que celle-ci nous met à disposition un fichier xml mais également parce qu'elle a la particularité de proposer des synonymes gradués : nous trouvons les synonymes classés du plus facile au plus compliqué en fonction de caractéristiques de surface telles que que la longueur, le nombre de sens, etc.

Présentation ReSyf contient 57 000 lemmes désambiguïsés issus de JeuxDeMots [25], étiquetés avec TreeTagger et filtrés avec Lexique 3 et le Trésor de la langue française informatisé (TLFi). Une entrée dans ReSyf correspond à la figure 18 se trouvant à la page suivante. Chaque entrée se trouve entre des balises du type <LexicalEntry>. Ensuite, nous pouvons trouver des informations concernant le lexème telles que la partie du discours et sa valeur et ensuite une série de synonymes assortis de leur rang en lien avec cette entrée.

25. http ://www.jeuxdemots.org/jdm-accueil.php

FIGURE 16 – Entrée lexicale du mot "accélérateur" dans ReSyf

```
<LexicalEntry>
    <feat att="ambiguity" val="1"/>
    <Lemma type="Form">
        <feat att="lexeme" val="accélérateur"/>
        <feat att="partOfSpeech" val="NC"/>
    </Lemma>
    <Sense id="accélérateur_NC_1">
        <feat att="poids" val="1.0"/>
        <feat att="usage" val="accélérateur (pédale)"/>
        <SenseExample id="accélérateur_NC_1">
            <feat att="type" val="synonyms"/>
            <feat att="annotation" val="manually"/>
            <feat att="score_lemma" val="--"/>
            <feat att="score_sense" val="--"/>
            <feat att="word" val="accélérateur"/>
            <feat att="rank" val="3"/>
        </SenseExample>
        <SenseExample id="accélérateur_NC_1">
            <feat att="type" val="synonyms"/>
```

API ReSyf Dans un souci d'optimisation et de praticité, nous avions décidé d'exploiter l'API disponible pour ReSyf pour la création de notre module Resyf generator API. Si notre programme fonctionne très rapidement avec la version échantillon de la base de données, nous n'obtenons pas de résultats lorsque nous voulons charger l'entiereté de la base de données. Après de multiples tentatives, dont la sérialisation de notre fichier, nous avons abandonné l'idée. Effectivement, même avec la version sérialisée, notre ordinateur ne semble pas assez puissant. Dès lors, nous nous sommes contentés du fichier XML.

ReSyf generator Le but principal de ce module est de rechercher si le mot à simplifier se trouve dans la base de données et obtenir les synonymes qui lui sont apparentés. Pour ce faire, nous avons utilisé le module xml.etree.ElementTree [26]. Au préalable, nous avons exploitié le lemmatiseur français de la librairie SpaCy étant donné que ReSyf contient uniquement les mots sous leurs formes lématisées. Ensuite, nous avons pu commencer à rechercher dans la base de données. Pour obtenir les synomymes d'un mot complexe, la seule condition est l'existence du lemme de celui-ci dans la ressource lexicale *ReSyf*. Dans le cas contraire, une liste vide ([]) est renvoyée.

Par après, nous avons transformé chacune des listes de candidats en chaines de caractères afin de les imprimer dans le bon format VICTOR. Parmi les candidats générés par la ressource lexicale, nous retenons uniquement les candidats partageant la même partie du discours que le mot cible.

26. https ://docs.python.org/2/library/xml.etree.elementtree.html

Limites La première limite est généralisable aux ressources lexicales : les résultats dépendent de la couverture de la ressource. C'est-à-dire qu'il faut trouver une base de données exhaustive comprenant autant de mots appartenant au registre courant qu'au registre technique dans notre cas.

La deuxième contrainte n'est pas liée à la technique utilisée mais plutôt aux outils employés pour le traitement de la langue. Dans certaines situations, il est possible que le lemmatiseur de *SpaCy* ne réussise pas à trouver le lemme de certaines formes. Par conséquent, celui-ci peut soit proposer une mauvaise lemmatisation soit retourner le mot tel qu'il est au départ (dans sa forme fléchie). Dans les deux cas, il ne sera pas possible de sortir des synonymes.

8.2.2 Dictionnaire Electronique des mots - DEM (Dubois et Dubois-Charlier, 2014)

Motivations Après avoir testé le site *ReSyf* avec des mots appartenant au domaine médical, nous avons constaté 2 types de problèmes : soit le mot ne figure pas dans la ressource, soit les synonymes proposés sont compliqués ou ne donnent pas assez d'informations pour un public non expert. Par exemple, lorsque nous recherchons le mot *diabète*, les synonymes retenus sont *glycosurie* et *hyperglycémie* tandis que pour *polypnée*, nous obtenons le terme très compliqué *tachypnée*. Nous nous retrouvons avec le même type de problème que lors de la génération de substituts par prolongements lexicaux. Par conséquent, comme Hamon et Grabar (2016) l'ont déjà suggéré, nous proposons d'annexer aux mots compliqués, des paraphrases les expliquant. Nous pouvons illustrer notre proposition à l'aide d'un exemple de résultat tiré de notre module DEM Generator. Nous avons souligné et mis en gras le terme complexe.

Le samu constate un coma avec une **polypnée** à 40mn, (...)
Le samu constate un coma avec une **polypnée** *(respiration accélérée)*, (...)

Naturellement, nous nous sommes tournés vers un dictionnaire électronique dans le but d'utiliser les définitions des mots en guise de paraphrases. La version XML [27] du DEM, en plus d'être exhaustive, est dans ce cas très appréciable puisqu'elle nous permet de manipuler facilement les mots et leurs définitions. Nous proposons de la voir plus en détails.

Présentation Le DEM est une base de données qui comprend 145 198 entrées qui sont chacune définie par 7 catégories. Pour les présenter, nous reprenons les informa-

27. http ://rali.iro.umontreal.ca/rali/ ?q=fr/dem

tions qui figurent sur le site du DEM [28]. De cette manière, chaque entrée est caractérisée par :

1. **M** : mot d'entrée avec des informations supplémentaires si nécessaire (no (différenciation de sens), réflexif, exclamatif, interrogatif et h-aspiré)
2. **CA** : catégorie grammaticale completée par des précisions sur le référent (humain, chose, animal, masculin, etc)
3. **DOM** : indication du domaine ainsi que le niveau de langue et emploi de la majuscule
4. **SENS** : la définition voire parfois un synonyme
5. **CONT** : contexte du contenu
6. **OP** : opérateur
7. **OP1** : autre type d'opérateur

FIGURE 17 – Entrée du mot "linguistique" dans le DEM (Dubos et Dubois-Charlier, 2014)

```
<entree ligne="80713">
   <M mot="linguistique" mot-initial="linguistique" no="2"/>
   <CONT>versé en N</CONT>
   <DOM nom="linguistique">LIN</DOM>
   <OP>étud</OP>
   <SENS>langue,langage</SENS>
   <OP1>P3b1</OP1>
   <CA categorie="N" type="non-anime" genre="F">-2</CA>
</entree>
```

L'objectif consiste, dès lors, à chercher dans la base de données le mot qui fait défaut au lecteur et d'en extraire sa définition pour faciliter la compréhension du lecteur. Cette technique ne nécessite donc pas d'étapes de sélection et de classement. En effet, les termes médicaux ont l'avantage d'être monosémique, il existe donc une seule et unique définition à ces termes.

DEM generator Pour implémenter ce module, nous avons dû rechercher dans la base de données si le mot à paraphraser était présent grâce au parseur *xml.etree.Element.tree* [29]. Nous avons également utilisé la bibliothèque SpaCy pour lemmatiser et étiqueter les parties du discours. Le premier procédé est une étape nécessaire au prétraitement de nos données puisque nous sommes dans un dictionnaire, les mots sont donc représentés sous leur forme canonique. L'étiquetage, quant à lui, est la réponse à un problème que nous avons rencontré. Lorsque nous voulions paraphraser un adjectif, nous avions en sortie, une paraphrase qui n'apportait pas d'informations sur le mot compliqué comme en témoigne l'exemple ci-dessous :

28. http ://rali.iro.umontreal.ca/rali/ ?q=fr/dem
29. https ://docs.python.org/2/library/xml.etree.elementtree.html

L'échocardiographie (...) a montré un épanchement péricardique **kystique**.
L'échocardiographie(...) a montré un épanchement péricardique **kystique** *(d kyste)*.

Nous observons que lorsque le mot est un adjectif, la définition de celui-ci n'est autre que le nom dont il provient. Pour pallier ce problème, nous avons décidé d'étiqueter les mots afin de distinguer les adjectifs des noms, des verbes, etc. Quand nous rencontrons ceux-ci, nous regardons de quels noms, les adjectifs sont issus pour en extraire le sens. De cette manière nous obtenons le résultat suivant :

L'échocardiographie (...) a montré un épanchement péricardique **kystique**.
L'échocardiographie(...) a montré un épanchement péricardique **kystique** *excroissance)*.

Nous tenons également à souligner un problème auquel nous avons du faire face : la polysémie. Il est vrai que les termes médicaux, comme nous l'avons dit plus haut, ont l'avantage d'être monosémiques. Cependant, le langage médical emprunte également au lexique du langage courant. Par conséquent, lorsque nous avons voulu trouver une paraphrase expliquant le sens du mot *rémission*, DEM generator a donné ce résultat en sortie :

Le traitement classique par ciclophosphamide obtient une **rémission** stable.
Le traitement classique par ciclophosphamide obtient une **rémission**(*absolution d péchés*) stable. .

Effectivement, le mot *rémission* n'appartient pas exclusivement au domaine médical pour signifier une "*diminution temporaire*" puisqu'on peut le retrouver dans les domaines religieux (comme c'est le cas dans l'exemple) et économique pour parler du *fait de surémettre*. Pour pallier ce problème, nous sélectionnons uniquement le sens des termes qui appartiennent aux domaines suivants **anatomie, pathologie, biologie, médecine, pharmacie, bactérie et virus, chirurgie, corps humain**. De cette manière, nous avons le résultat suivant (cf. ci-dessous). Cette amélioration rend ce module uniquement reservé aux termes médicaux.

Le traitement classique par ciclophosphamide obtient une **rémission** stable.
Le traitement classique par ciclophosphamide obtient une **rémission**(*diminution temporaire*) stable. .

Limites Les limites sont les mêmes que pour ReSyf. C'est-à-dire que même si le dictionnaire électronique que nous avons choisi contient le double du nombre de lemmes de ReSyf et qu'il possède une large couverture donc, nous ne pouvons pas empêcher l'absence de certains termes. Ensuite, puisque nous avons également utilisé un lemmatiseur pour rechercher les mots complexes, il est possible que celui-ci ne réussisse pas à trouver le lemme de certaines formes. Conséquemment, il est impossible pour notre module de rechercher une forme non canonique ou encore un mot qui n'existe pas suite à une mauvaise lemmatisation.

9 Modules de sélection de substituts

Les modules de sélection ont pour but de désigner les candidats les plus proches sémantiquement et contextuellement du mot cible pour ne pas nuire à la grammaticalité et à la cohérence des phrases. Paetzold et Spcia (2015) proposent dans *LEXenstein*, 9 manières de sélectionner des candidats pertinents : en exploitant les prolongements lexicaux (WordVectorSelector), un modèle de co-occurences par mot (BiranSelector), un module de désambiguïsation (WSDSelector), les parties du discours (AlusioSelector), le partitionnement de données (BelderSelector), un classificateur formé sur un ensemble de données caractéristiques (BoundarySelector), une machine à vecteurs de support (SVMBoundarySelector, SVMRankSelector) ou encore en sélectionnant toutes les substitutions possibles générées pour un mot cible donné (VoidSelector).

Parmi ces 9 approches, nous avons choisi d'en expérimenter 3 d'entre elles : la désambiguïsation par plongements lexicaux à l'aide de FastText, par parties du discours et à l'aide d'un module de désambiguïsation proposant plusieurs algorithmes classiques (PYWSD [30]). Nous avons opté pour ces techniques parce qu'elles ne demandent pas d'apprentisssages supervisés à la différence des classificateurs et machines à vecteurs de support. En effet, la plus grande contrainte dans cette recherche est la disponibilité des données : il est très difficile de trouver des corpus annotés adaptés à nos besoins et il serait trop chronophage d'en réaliser nous-mêmes. Dès lors, pour pallier ce problème, nous tentons de trouver au maximum des approches non-supervisées.

Puisque la sélection de substituts dépend directement des candidats générés par les modules précédents, nous tenons à les passer en revue afin de déterminer la pertinence d'implémenter des modules de désambiguïsation.

BERT generator L'incroyable innovation de BERT est, comme nous l'avons explicité plus haut, de prendre compte à la fois les voisins/contextes de droite et de gauche. Lors de notre implémentation, nous avons donné à BERT la phrase complète et l'indice

30. https ://pypi.org/project/pywsd/

du mot qu'il doit prédire. Grâce à ces informations et à son préentrainement bidirectionnel, le modèle est en mesure de deviner le nature du mot à prédire sans nuire à la sémantique de la phrase.

FastText generator FasText génère des candidats avec des parties du discours ou flexion différentes. Conséquemment, il est important de faire un choix parmi tous ces candidats. Dans le module de génération, nous avons déjà intégré une étape de sélection puisque, pour rappel, nous avons réalisé une fonction qui choisit uniquement les candidats possédant les mêmes parties du discours.

DEM generator Ce module est plus spécifique puisque nous avons établi que pour la simplification médicale, il était préférable de générer des paraphrases plutôt que des synonymes. Par conséquent, nous avons nul besoin de sélectionner des candidats : nous attachons uniquement une définition aux mots complexes. Néanmoins, nous avons choisi d'intégrer une étape de désambiguïsation en sélectionnant uniquement les sens provenant de domaines cliniques. Cette étape est expliquée, exemplifiée et motivée dans la section précédente.

ReSyf generator Ce dernier module propose **plusieurs** candidats sur base d'une ressource de synonymes. De la même manière que pour le module *FastText generator*, nous avons jugé qu'il était pertinent d'intégrer des fonctions de désambiguïsation dans celui-ci et de créer des modules de substitutions à partir des candidats qu'il aura généré.

9.1 Les parties du discours pour sélectionner des candidats

En premier lieu, nous proposons de nous intéresser à une approche sélectionnant uniquement les candidats qui partagent les mêmes parties du discours que le candidat cible.

9.1.1 Théorie

Pour la réalisation de la classe *AluisioSelector*, Paetzold et Specia se sont inspirés des recherches de Aluísio et Gasperin (2010) proposant le projet PorSimples pour la simplification de textes portuguais. Pour identifier la difficulté des mots, les chercheurs utilisent un dictionnaire de mots simples : ceux-ci regardent si les lemmes des tokens présents dans les textes à simplifier sont également présents dans le dictionnaire. Cependant, Aluísio et Gasperin (2010) ont été confrontés à un situation problématique :

lorsque l'on recherche un lemme dans un dictionnaire, il faut faire face aux différents sens des mots. Conséquemment, l'utilisation d'une étiqueteur en parties du discours est suggérée. Paetzold a exploité cette idée pour la sélection des candidats en proposant la méthode suivante : en utilisant le StanfordPOSTagger, le chercheur détermine d'abord la partie de discours (POS) du mot cible pour ensuite la comparer à celles des mots candidats.

Application Nous avons décidé qu'il était cohérent d'intégrer une fonction à l'intérieur des modules ReSyf generator et FastText generator vérifiant si les candidats générés et le mot à remplacer possédent les mêmes parties du discours. Pour ce faire, nous avons utilisé l'étiqueteur de spaCy. De cette manière, si les candidats et le mot cible ne partagent pas les mêmes parties du discours, ceux-ci ne sont pas sélectionnés.

9.2 Les vecteurs pour sélectionner des candidats

Nous avons déjà exploité les vecteurs des prolongements lexicaux pour générer des substituts à l'aide de modèles tels que BERT et ELMo. Nous proposons, de la même manière que pour la classe WordVectorSelector, d'exploiter les vecteurs pour la tâche de sélection : nous déterminons les candidats à sélectionner grâce à la proximité sémantique qu'ils partagent avec le candidat cible. Puisque Paetzold et Specia (2015) n'ont renseigné aucun article théorique concernant cette approche, nous avons décidé d'inspecter le code afin de déterminer l'heuristique que nous allons suivre.

9.2.1 Quelle approche adopter ?

Dans un premier temps, il s'agit d'opérer un *preprocessing* sur les données issues du corpus de test en initialisant une liste de mots vides (donne de meilleurs résultats lors de la vectorisation) et en utilisant un parseur dans le but d'analyser chaque phrase du corpus. Subséquemment, Paetzold détermine le vecteur du mot cible selon plusieurs paramètres. Par exemple, l'utilisateur peut choisir s'il veut ou pas un contexte autour du mot cible lors du calcul du vecteur du mot cible. Celui-ci peut également choisir de retenir uniquement des mots considérés comme informatifs (nom, verbe, adjectifs, adverbes) grâce au parsing réalisé auparavant. Par la suite, Paetzold détermine les vecteurs de chaque candidat et met à jour un dictionnaire ayant pour clé le nom du candidat et pour valeur, le cosinus de l'angle entre les vecteurs du candidat et du mot cible. Pour finir, selon le pourcentage décidé par l'utilisateur, un vecteur de taille N, contenant un ensemble de substitutions sélectionnées pour chaque instance du corpus VICTOR, est obtenu.

Par conséquent, nous trouvons l'approche du chercheur assez simple et intéressante

à implémenter. En effet, nous avons déjà utilisé cette technologie pour la génération de candidats mais qu'en est-il pour la sélection ? Par ailleurs, nous avons également décidé de nous questionner sur un point essentiel qui sera abordé dans la sous-section suivante : le choix des modèles/bibliothèques.

9.2.2 Le choix du modèle

Il existe de nombreuses bibliothèques ou modèles de vectorisation. Par exemple, dans la section précédente, nous avons pris le parti de choisir BERT pour la génération de candidats. Mais, au vu des résultats peu satisfaisants à cause d'un corpus d'entrainement peu efficace, nous avons choisi de ne pas l'utiliser pour cette tâche.

Par conséquent, nous avons tranché pour FastText pour des raisons déjà énoncées dans la section précédente. Notre choix s'est porté vers cette librairie parce qu'elle permet de générer de meilleurs résultats pour les mots rares ou pour les mots hors vocabulaire étant donné que FastTextfonctionne par n-grammes. Nous pensons que les termes médicaux sont peu présents dans les corpus d'apprentissage non techniques voire absents étant donné que nous avons souvent affaire à des mots d'une grande technicité : prendre FastText peut se révêler être une bonne stratégie.

9.2.3 FastText

Nous proposons 2 façons différentes d'utiliser FastText pour la sélection des candidats :

1. La première se base uniquement sur la similarité entre les mots cibles et leurs candidats ;
2. La deuxième, à l'image de la classe WordVectorSelector, propose d'exploiter les mots voisins du mot complexe pour la vectorisation ;

Ces 2 modules prennent en entrée un fichier victor où nous nous retrouvons les candidats générés par les modules de génération et produisent en sortie un nouveau fichier avec les synonymes.

FastText selector word Dans un premier temps, il s'agit de récupérer les candidats qui ont été générés et imprimés au format VICTOR lors de l'étape précédente. Lorsque nous avons récupéré les candidats, nous utilisons la fonction comparaison vectors : nous comparons les vecteurs du candidat cible et du candidat à la substitution à l'aide de la fonction similarity du module KeyedVectors de Gensim et sélectionnons uniquement les paires cible - candidat dont le score de similarité est supérieur au seuil 0.5. Si le modèle ne trouve pas le mot cible ou le candidat, nous passons.

Pour déterminer le seuil, nous avons effectué plusieurs tests pour délmiter avec précision le moment où le modèle retient des mots entretenant une relation sémantique avec le candidat que nous estimons cohérents. En effet, si nous choisissons un seuil trop élevé, nous risquons de nous retrouver avec une fonction trop restrictive et peu de mots sélectionnés. Mais le contraire est également possible si nous choisissons un seuil trop bas. Pour éviter ce genre de situation, nous avons testé la fonction *comparaison vectors* avec les verbes *aimer* et *regarder* et leurs synonymes trouvés sur une ressource en ligne[31].

```
comparaison_vectors("regarder","voir")
comparaison_vectors("regarder","admirer")
comparaison_vectors("regarder","analyser")
comparaison_vectors("regarder","visionner")
```

```
0.70124084
0.5963082
0.5156275
0.69494045
```

FIGURE 18 – Sortie de la fonction comparaison vectors du module *FastText selector avant adaptation*

```
comparaison_vectors("aimer","adorer")
comparaison_vectors("aimer","apprécier")
comparaison_vectors("aimer","désirer")
comparaison_vectors("aimer","affectionner")
```

```
0.68052346
0.6011672
0.53008634
0.5740942
```

FIGURE 19 – Sortie de la fonction comparaison vectors du module *FastText selector* avant adaptation

Nous remarquons que tous les synonymes se situent au-dessus d'un seuil de 0.5. Par conséquent, nous avons choisi cette valeur. Par la suite, pour stocker ces résultats, nous avons créé un dictionnaire ayant pour clé, le candidat et pour valeur, la sortie de la fonction *comparaison vectors*, autrement dit, un mot partageant un score de similarité supérieur à 0.5 avec le mot cible. Pour finir, nous avons créé plusieurs fonctions annexes afin d'imprimer les candidats sélectionnés dans le format fichier victor adéquat. Conséquemment nous obtenons un fichier au format suivant où pour chaque ligne, nous avons : phrase mot indice candidat sélectionné candidat sélectionné candidat sélectionné

Paetzold Selector Le code de ce module est en entièrement basé sur celui de la classe WordVectorSelect de Paetzold et Specia (2015) : nous avons repris son code en retenant uniquement ce qui nous intéressait pour l'adapter à nos besoins : par exemple, nous avons utilisé un autre "tokenizeur" que celui fourni et nous avons nous-mêmes choisi le nombre de mots que nous voulions autour du mot complexe pour le calcul de son vecteur. Comme nous l'avons mentionné ci-dessus, Paetzold détermine le vecteur du mot cible en prenant en compte le contexte autour du mot cible. Nous voulons intégrer cette approche parce que nous sommes convaincus qu'elle peut permettre de gérer la polysémie de certains mots. Nous pensons également que la confrontation des résultats avec le module FastText selector word peut être intéressante. Dans un

31. http ://www.synonymo.fr/

premier temps, nous récupérons le fichier avec les candidats générés par les modules de génération. Nous procédons en trois temps :

1. Le calcul du vecteur du mot complexe à l'aide de la phrase entière et de la position du mot dans la phrase
2. Le calcul du vecteur des candidats
3. Le calcul de distance cosinus des vecteurs

Pour le calcul du vecteur du mot complexe, nous avons repris le code de la fonction get sent de la classe WordVectorSelector [32] : le vecteur est calculé à partir du contexte défini par l'utilisateur. C'est-à-dire que l'utilisateur choisit lui-même le nombre de mots autour du mot complexe qui interviendra dans le calcul du vecteur. Le procédé de Paetzold est le suivant :

> La phrase de départ est tokénizée à l'aide du tokénizeur de Nltk que nous avons choisi pour le français. Ensuite, les mots voisins sont retenus en fonction de la fenêtre déterminée par l'utilisateur : si ne ce sont pas des mots vides (vérification à partir de la liste de mot vide que nous avons initilisée avec Nltk), alors ils sont gardés dans une liste. Par la suite, pour chaque token de la liste, un vecteur est calculé à l'aide de Gensim. Pour finir, la somme des vecteurs des tokens est calculée et est stockée à l'aide de la bibliothèque NumPy.

Concernant le calcul du vecteur des candidats, la méthode est, on ne peut plus simple : pour chaque candidat d'un mot complexe, un vecteur est calculé à partir de Gensim. Pour finir, la distance cosinus entre les vecteurs du mot complexe et des candidats est obtenu grâce à SpaCy. Après obtention de cette valeur, au lieu de laisser le choix à l'utilisateur de choisir le pourcentage de synonymes restants comme le fait Paetzold, nous avons préféré, à l'image du module ci-dessus, fixer un seuil à 0.75 après plusieurs tests de la même nature que ceux effectués pour *FastText selector word*. De cette manière, nous pensons délimiter avec précision les synonymes des mots complexes. Enfin, nous avons à l'aide de plusieurs fonctions traité nos candidats afin qu'ils puissent se retrouver dans le bon format VICTOR.

9.2.4 Limitation

Nous ne pouvons pas utiliser cette technique de désambiguïsation sur les candidats générés par FastText. Etant donné que nous avons produit nos synonymes à l'aide de la fonction most similar de Gensim et que nous l'avons également utilisée pour sélectionner les mots les plus proches du mot complexe, le module ne sélectionnera aucun synonyme étant donné que, selon lui, ils sont déjà pertinents.

32. Nous avons seulement effacé certaines étapes qui nous semblaient pas pertinentes mais nous n'avons rien ajouté

9.3 Une ressource lexicale pour sélectionner des candidats

Dans le modèle précédent de sélection, nous avons utilisé la sémantique distributionnelle pour sélectionner les synonymes. Cette fois-ci, nous avons choisi comme critère de sélection, la similiarité sémantique.

Pour notre dernier module, nous nous inspirons de la classe WSDSelector exploitant la bibliothèque Pywsd [33] implémentant plusieurs technologies de désambiguïsation dont notamment l'algorithme de Lesk. Nous avons décidé de ne pas utiliser la bibliothèque puisqu'elle utilise Wordnet qui est une base de données en anglais où les mots sont regroupés en série de synset (ensemble de synonymes cognitifs), chacun exprimant un concept différent. Ceux-ci sont liés entre eux au moyen de relations conceptuelles, sémantiques et lexicales [34]. Par conséquent, nous avons recherché une base de données similaire à Wordnet pour implémenter l'algorithme Lesk Simple dans notre module.

9.3.1 Présentation de *Pywsd*

Plusieurs implémentations de technologies de désambiguïsation sont disponibles. Nous retranscrivons les informations les concernant à partir du github [35] :

1. **Les algorithmes de Lesk sous plusieurs formes** : original (Lesk, 1986), adapté/étendu (Banerjee et Pederson, 2003), simple (avec les définitions, exemples et hypero/hyponymes), cosinus (utiliser des cosinus pour calculer les chevauchements au lieu d'utiliser des comptes bruts)
2. **Maximiser la similarité(*Maximizing similarity*)** : similitude de chemins (Wu-Palmer, 1994 ; Leacock and Chodorow, 1998), contenu informatif (Resnik, 1995 ; Jiang and Corath, 1997 ; Lin, 1998)
3. **Baseline** : sens aléatoire, le premier sens NLTK ou le plus grand nombre de lemmes.

Nous avons choisi d'utiliser l'algorithme de Lesk parce qu'il nécessite seulement un dictionnaire et aucun apprentissage c'est-à-dire que nous n'avons nullement besoin de corpus annotés.

9.3.2 L'algorithme de Lesk

L'idée principale de l'algorithme, selon Vasilescu (2004), est basée sur la superposition (*overlap*), il s'agit de compter le nombre de mots communs entre les définitions du mot à désambiguïser et les définitions du contexte (mots se trouvant dans une fenêtre

33. https ://pypi.org/project/pywsd/
34. https ://wordnet.princeton.edu/
35. https ://github.com/alvations/pywsd

de x mots du mot à désambiguïser) . Dans la littérature, nous retrouvons très souvent l'exemple suivant pour expliquer l'algorithme (Vasilescu, 2004) :

> Par exemple, si on considère la cooccurrence des mots anglais *pine* et *cone* dans le même contexte, un programme de désambiguïsation automatique, basé sur cette idée simple, serait capable de choisir le sens arbre du mot pine en comptant les intersections entre les différentes définitions de sens des deux mots : "**pine** – 1. kind of **evergreen tree** with needle-shaped leaves . . . ; 2. waste away through sorrow or illness . . . ; **cone** – 1. solid body which narrows to a point . . . ; 2. something of this shape whether solid or hollow . . . ; 3. fruit of certain **evergreen tree**. . . ". Dans ce cas, le nombre maximal de mots communs (evergreen, tree) est donné par l'intersection entre les définitions 1, respectivement 3 de pine et cone, ce qui détermine le choix du sens correspondant pour le mot pine soumis à l'analyse.

Nous considérons qu'il serait intéressant d'implémenter l'algorithme de *Lesk* pour déterminer le sens du mot complexe. Ensuite, nous souhaitons comparer le sens du mot complexe choisi par l'algorithme avec les différents sens des candidats.

Lesk simple Nous avons choisi d'utiliser l'algorithme *Simple Lesk* pour désambiguïser le mot complexe. Pour l'implémentation, nous avons opté pour le pseudo-code suivant :

FIGURE 20 – Pseudo code de l'algorithme du Lesk Simple (Candito, 2017)

```
fonction simplifiedLesk(mot, phrase)
    meilleur_sens ← sens le plus fréquent de mot
                        (si l'info est dispo, sens random sinon)
    max_overlap ← 0
    contexte ← mots de phrase
    pour chaque sens existant pour mot
        signature ← signature(sens)
        overlap ← nb mots communs à signature / contexte
        si overlap > max_overlap alors
            max_overlap ← overlap
            meilleur_sens ← sens
    return meilleur_sens
```

Nous avons choisi l'agorithme Lesk Simple : au lieu de comparer la signature d'un mot à l'union des signatures (sens des mots du contexte de mot), nous comparons la signature du mot au contexte du mot. Le terme *signatures* est omniprésent dans cet algorithme et nous tenons à le définir pour une bonne compréhension de l'algorithme : celles-ci, toujours selon Candito (2017), désignent l'ensemble de mots pleins présents

dans les exemples/définitions de chaque sens. Par exemple, pour le terme anglais *bank*, il existe deux sens :

FIGURE 21 – Exemple des définitions du terme anglais *bank* (Candito, 2017)

$bank_1$	Def: Examples:	a financial institution that accepts deposits and channels the money into lending activities « he cashed a check at the bank » « that bank holds the mortgage on my home »
$bank_2$	Def: Examples:	sloping land (especially the slope beside a body of water) « they pulled the canoe up on the bank » « he sat on the bank of the river and watched the currents »

A partir de ces définitions, nous pouvons extraire les mots pleins suivants :

–signature de bank1 : financial, institution, accept, deposit, channel, money, lending, activity, cash, check, hold, mortgage, home

–signature de bank2 : sloping, land, body, water, pull, canoe, bank, sit, river

Pour obtenir ces signatures, nous avons recherché une ressource similaire à *WordNet* pour le français : WOLF [36]. Cette ressource a été construite à partir du Princeton Wordnet et de diverses ressources multilingues. Pour un traitement plus rapide et efficace, nous avons choisi d'exploiter une API python. Notre choix s'est dès lors tourné vers *FreNetic* [37].

9.3.3 FreNetic

FreNetic est une API python pour *WOLF*. Grâce à celle-ci, nous avons accès aux définitions, aux hyperonymes et hyponymes de manière aisée comme dans la figure 23 de la page suivante.

Le problème est que nous avons accès à ces informations uniquement à partir de l'identifiant du mot : or, il n'existe aucune fonction prévue à cet effet. Conséquemment, nous avons du nous arranger pour obtenir l'identifiant de chaque mot pour ensuite obtenir les définitions. Nous expliquons la création des fonctions et la résolution des problèmes dans la sous-section suivante.

36. http ://pauillac.inria.fr/ sagot/index.html#wolf
37. https ://github.com/hardik-vala/FreNetic

FIGURE 22 – Exemple d'utilisation de l'API FreNetic

```
>>> synset.sid() #Id
'eng-30-00001740-n'
>>> synset.literals() # Literals
[entité (96/4:fr.csen,fr.rocsen,fr.roen,enwikipedia;gwa2012(0.8601390479086624685 2);lrec12mllexwn(1.548);ManVal2012C
>>> synset.defn() # Definition
u"concept formulant la cat\xe9gorisation et l'identique des choses de notre environnement"
>>> synset.usages() # Usages
[]
>>> synset.bcs()
2
>>> synset.pos() # POS tag
'n'
>>> synset.hypernyms() # List of it's direct hypernyms (includes instance hypernyms)
[]
```

9.3.4 FreNetic selector

Dans un premier temps, nous avons créé la fonction *get list of definitions* prenant en argument un mot. Cette fonction a pour but de retourner une liste de définitions pour chaque sens/hyperonyme/hyponyme du mot. Nous avons remarqué que la base de données avait du mal avec les mots qui ne sont pas dans leurs formes canoniques : par conséquent, nous avons ajouté une fonction de lemmatisation à partir de *Spacy*. Ensuite, nous utilisons la méthode *synsets* de la bibliothèque *FreNetic* pour obtenir une liste contenant toutes les informations sur notre mot afin de récupérer l'identifiant de chaque sens/hyperonyme/hyponyme :

FIGURE 23 – Sortie après utilisation de la méthode "synsets" sur le mot *transgresser*

```
[Synset(Id: eng-30-02566528-v, Literals: [briser (lrec12mllexwn(2.336)), enfreindre (lrec12clwolf(1.4302)), rompre (lrec12m
llexwn(2.197)), transgresser (lrec12mllexwn(2.344)), violer (gwa2012(0.2470861851623971 3725);lrec12clwolf(1.4302))], Def.:
act in disregard of laws, rules, contracts, or promises, Usages: ['offend all laws of humanity', 'violate the basic laws or
human civilization', 'break a law', 'break a promise'], POS: v, Hypernyms: ['eng-30-02457825-v'])]
```

Nous avons utilisé une expression régulière pour récupérer l'identifiant de chaque sens différent/hyperonyme/hyponyme associé au mot complexe. Par exemple dans la figure ci-dessus, nous avons seulement un sens pour le mot *transgresser* identifiant **eng-30-025665-v** mais nous avons également un hyperonyme correspondant au mot **eng-30-02457825-v** dont nous allons aussi extraire la définition. Nous pensons qu'exploiter les hyperonymes et hyponymes peut permettre un plus large recouvrement entre le contexte et les définitions du mot complexe.

A partir de cet identifiant, nous avons pu enfin utiliser la fonction *defn()* de *FreNetic* permettant de donner les définitions de chaque sens/hyperonymme/hyponyme d'un mot. Nous avons ajouté chaque définition dans une liste. Si nous reprenons notre exemple, notre fonction retourne la liste suivante pour le mot *transgresser* : ['act in disregard of laws, rules, contracts, or promises', 'show a lack of respect for'].

Ensuite, nous avons créé la fonction *LeskSimplified* qui a été implémentée selon le pseudo-code que nous avons montré dans la sous-section suivante. Pour fonctionner cette fonction demande la liste de sens du mot complexe que nous obtenons grâce à la fonction présentée précédement. Mais nous avons également du créer trois autres fonctions annexes :

- *contexte(mot,phrase)* : cette fonction permet de retourner le contexte du mot complexe. Par exemple, dans la phrase "*Elle aime transgresser les règles"*, nous voulons "*aime,règles.*" étant donné que *elle* et *les* sont peu informatifs. De cette manière, nous pouvons ensuite regarder si ces mots se retrouvent dans les signatures de chaque sens/hyperonyme/hyponyme du verbe *transgresser*. Mais nous avons du faire face à un problème de taille : les usages et définitions sont en anglais. Conséquemment, il est impossible de vérifier si les mots voisins se retrouvent dans les signatures. Nous avons alors utilisé la librairie *googletrans*[38] pour contourner ce problème. De cette manière, nous prenons en argument le mot complexe et la phrase dans laquelle se trouve le mot complexe. Nous traduisons la phrase à l'aide de notre module. Nous enlevons le mot complexe de la phrase pour obtenir le contexte. A partir du contexte obtenu en anglais, nous tokenizons, enlevons les mots vides et nous lemmatisons à l'aide du module *Spacy*. Nous obtenons, toujours pour le même exemple, la sortie suivante : [like,rule]
- *signature(definition)* : Pour chaque sens/hyperonyme/hyponyme du mpt (ici, *transgresser*), nous voulons déterminer sa signature afin de pouvoir déterminer le nombre de mots communs entre la signature de chaque sens du mot et le contexte du mot complexe. Cette fonction prend en argument la définition d'un des sens/hyperonyme/hyponyme du mot complexe et donne en sortie les mots pleins de cette définition grâce à une liste de mots vide du module *Nltk*. Pour transgresser, nous obtenons les signatures suivantes : ['act', 'disregard', 'law', 'rule', , 'contractpromise'] ['show', 'lack', 'respect']
- *overlap(contexte,signature)* : nous regardons si des mots du contexte se trouvent dans la signature, si c'est le cas, nous incrémentons le compteur. Dans ce cas nous obtenons le meilleur sens : *act in disregard of laws, rules, contracts, or promises* puisque nous retrouvons le mot voisin *rules* dans la signature ['act', 'disregard', 'law', 'rule', 'contractpromise'].

En sortie de cette fonction, nous obtenons donc le meilleur sens du mot complexe en fonction de son contexte. Par la suite, nous regardons si la définition du sens du mot complexe se retrouve dans chaque définition de chaque sens de chaque candidat : nous avons préprocessé les définitions (tokenization, suppression des mots vides et de la ponctuation) et incrémenté à chaque fois qu'un token est à la fois présent dans la définition du sens sélectionné et dans la définition d'un des sens du candidat. Par exemple, pour notre phrase "*Elle aime transgresser les règles*", nous avons les candidats *rompre, briser et couper*. Seuls les deux premiers ont été en commun avec *transgresser* . Enfin, pour stocker toutes ces informations, nous avons créé un dictionnaire avec comme

38. https ://pypi.org/project/googletrans/

clé le mot complexe et comme valeur, le candidat à la substitution si et seulement si celui-ci obtient un score égal ou supérieur à 4. Nous avons effectué plusieurs tests et nous avons déterminé que 4 était un seuil convenable : en dessous de 4, nous risquons de sélectionner des candidats qui ont des mots communs "par hasard" et qui donc ne partagent pas forcément de similarité sémantique. Pour finir, à l'image des autres modules, nous créé plusieurs fonctions afin d'imprimer l'ensemble des candidats sélectionnés dans le format adéquat.

9.3.5 Limites

Cette méthode comporte plusieurs limites que nous listons et expliquons dans le paragraphe suivant.

Limite de l'algorithme Nous avons choisi l'algorithme simple de Lesk pour sa facilité d'implémentation mais également parce que nous pouvons désambiguïser de manière non supervisée, ce qui est un avantage considérable étant donné qu'annoter un corpus demande beaucoup de temps. Mais, évidemment, comme tout algorithme, celui-ci comporte des limitations. Le problème principal est que le sens est déterminé à partir du chevauchement entre les définitions du mot complexe et le contexte, or, ces définitions sont souvent limitées et non exhaustives. Il est donc possible qu'aucun mot du contexte ne se retrouve dans les signatures de chaque sens du mot complexe. Nos solutions ont été les suivantes :

1. Nous utilisons également les définitions des hyperonymes et hyponymes afin d'élargir les possibilités ;
2. Nous avons lemmatisé les mots du contexte et les signatures des sens/hyponymes/hyperonymes ;

Problème d'outils/ressources Comme nous l'avons mentionné plus haut, la qualité de notre dictionnaire est discutable étant donné que nous obtenons des définitions en anglais pour des mots en français. Nous avons cherché, en vain, une autre ressource similaire à FreNetic pour pallier ce problème. Nous avions pensé à WoNeF qui se révèle être une traduction de WordNet 3.0 mais les définitions sont également en anglais. Partant de ce constat, nous avions proposé d'utiliser un traducteur pour contourner le problème de langue. Nous tenons également à souligner le problème que nous avons rencontré avec les outils utilisés pour la lemmatisation : il arrive que le lemmatiseur ne trouve pas la forme canonique d'un mot et par conséquent, il n'est pas possible de trouver ce mot dans la base données. La meilleure solution serait de proposer aux utilisateurs d'entrer la forme canonique directement dans le fichier VICTOR mais cela nous parait fort contraignant pour l'utilisateur. De plus, nous nous situons dans le

domaine du traitement automatique du langage donc, il est important d'automatiser le plus de processus possibles.

Problème de traduction Enfin, si utiliser un traducteur nous permet de vérifier si les mots voisins, qui se trouvent initialement en français, se retrouvent dans les signatures, qui sont elles en anglais, cela implique également des erreurs supplémentaires. Effectivement, l'utilisation d'un traducteur signifie de facto des erreurs de traduction et notamment -de façon ironique -des erreurs de désambiguïsation.

10 Module de classement des candidats

Pour finir le processus de simplification, Paetzold et Specia proposent 8 classes traitant le problème de classement sous différents angles soit en utilisant des prescripteurs linguistiques (MetricRanker, GlavasRanker), une machine à vecteurs de support (SVMRanker, SVMBoundaryRanker), un classificateur (Boundary Ranker), un modèle de langue (BiranRanker) ou en calculant des fréquences dans des corpus simples (BottRanker, YamamotoRanker).

Nous avons choisi de créer deux modules de classement : l'un à partir de prescripteurs linguistiques à la manière des classesMetricRank, GlavasRanker et l'autre à partir d'une machine à vecteurs de support comme pour les classes SVMRanker et SVMBoundaryRanker. Nous avons laissé les autres techniques de côté parce que nous intégrons déjà la fréquence dans nos prescripteurs linguistiques, dès lors, nous pensons qu'il serait redondant et inutile de créer un module semblable à BottRanker et YamamotoRanker. Nous raisonnons de la même manière pour le classificateur Boundary Ranker, nous utilisons déjà une machine à vecteurs de support qui permet des résoudre des problèmes de classification.

Par conséquent, cette section se divise en deux sections, chacune étant réservée pour un type de module. Pour chacun des deux, nous motivons nos choix et expliquons nos implémentations.

10.1 Les prescripteurs linguistiques pour le classement

Pour ce premier module, nous avons choisi de nous inspirer de la première classe du module generators de *LEXenstein* (Paetzold et Specia, 2015), MetricRanker dont la méthode de classement est basée sur une série de mesures choisies par l'utilisateur. Cette classe fonctionne avec le modulefeatures qui propose 39 mesures regroupées en 8 catégories différentes : axé sur le lexique, morphologique (longueur, nombre de syllabes,etc), "collocatif" (fréquences "brutes" et probabilités à partir d'un modèle de langage), "pop collocatif"(fréquences et prbabilités de modèle de langage de n-grammes "pop"), "collocatif tagué"(comptage brut de fréquences de ngrammes tagués), axé sur le sens (le nombre de synonymes,de sens, d'hyperonymes, d'hyponymes, etc), syntaxique (mesure la probabilité qu'un candidat à la substitution assume le rôle syntaxique du mot cible), et sémantique (similitude sémantique par exemple).

Notre motivation première pour cette technique tient au fait que celle-ci est facile, non supervisée et ne demande aucun corpus annoté. C'est un avantage indéniable étant donné que, comme nous l'avons déjà répété plusieurs fois, l'annotation de corpus demande un temps considérable. Il est important de noter que le classement des candidats à partir de prescripteurs linguistiques est compliqué parce que la complexité/facilité d'un mot varie indéniablement d'une langue à l'autre [39] : par exemple, l'acronyme médical *samu* a très peu de lettres (4) et est considéré comme compliqué tandis que l'adjectif *beau* de la langue française est également court mais pourtant moins complexe.

10.1.1 Les prescripteurs sélectionnés

A l'instar de *LEXenstein* proposant toute une série de prescipteurs à son public, nous choisissons d'intégrer les prescripteurs qui ont été abordés tout au long de notre parcours théorique. Nous motivons nos choix étape par étape.

La longueur Ce prescripteur est discutable : s'il est utilisé pour la langue française générale, il est boudé pour la langue médicale. Pour rappel, dans la section théorique où nous abordons les textes médicaux, nous avons remarqué que la concision était une caractéristique des termes médicaux comme en témoignent les nombreux acronymes, abréviations, sigles, etc. Il existe pourtant pas mal de termes médicaux très longs tels que éponymes, les technicismes spécifiques, etc. Conséquemment, nous pensons qu'il est aussi intéressant de tester ce prescripteur sur le langage médical.

39. Dans ce cas-ci, nous parlons de la langue médicale et de la langue courante.

Le nombre de sens De manière intuitive, nous serions tentés de croire que plus un mot a de sens, plus celui-ci a de fortes chances d'être complexe. Néanmoins, Elhadad (2006) a souligné une caractéristique importante d'un terme : la monosémie. Partant de ce constat, nou envisageons de considérer comme plus complexes, les mots ayant un seul sens.

La fréquence La fréquence est, selon nous, un indicateur cohérent de complexité pour la langue française mais également pour la langue médicale. A la manière d'Elhadad (2006), nous postulons que plus un mot est fréquent, plus celui-ci a des chances d'être compris par le lecteur. Pour ce faire, nous avons téléchargé *Lexique*[40], qui est une base de données fournissant pour 140.000 mots de la langue française des informations dont notamment les fréquences d'occurences dans différents corpus (sous-ensemble de textes littéraires récents tirés du corpus Frantext et un corpus de sous-titres de films). Ci-dessous, nous avons inclus un exemple pour l'adjectif *aberrant*, nous retrouvons ses versions aux masculin pluriel, féminin singulier et féminin pluriel.

FIGURE 24 – Exemple de l'entrée "aberrant" dans Lexique3

aberrant	abER@	aberrant	ADJ	m	s	0,84	2,5	0,58	1,08
aberrant	abER@	aberrer	VER			0	0,2	0	0,2
aberrante	abER@t	aberrant	ADJ	f	s	0,84	2,5	0,04	0,61
aberrantes	abER@t	aberrant	ADJ	f	p	0,84	2,5	0,2	0,34
aberrants	abER@	aberrant	ADJ	m	p	0,84	2,5	0,02	0,47

Evidemment, pour de meilleurs résultats, il aurait été intéressant de réaliser un module unique pour les termes médicaux en recherchant une ressource similaire à celle de *Lexique* mais pour la langue médicale. Malheureusement, après avoir épluché internet, nous nous sommes rendus à l'évidence : il n'existe pas une telle ressource pour les textes médicaux français. Nous aurions pu, comme Vinod et al. (2014), déterminer la fréquence de ces termes dans le Wikipédia Simple : si le candidat est fréquent dans un corpus grand public, alors ce candidat est général. Par ailleurs, puisque le but de notre mémoire n'est pas de créer des ressources et surtout parce que cela n'est pas possible dans les limites du temps imparti, nous nous sommes limités aux données fournies par *Lexique*.

10.1.2 Features ranking

L'idée de ce module est la même que celle de Paetzold et sa classe MetricRanker : sélectionner à partir de prescripteurs linguistiques (longueur, sens, fréquence). L'utilisateur choisit parmi les 3 mesures lesquelles il souhaite utiliser pour classer les candidats sélectionnés. Nous obtenons pour chacune des 3 mesures un fichier au format VICTOR

40. http ://www.lexique.org/

où nous pouvons lire pour chaque ligne : la phrase le mot complexe l'indice le candidat n1 le candidat n2 etc.

La longueur comme prescripteur Nous ne commentons pas l'implémentation de cette fonction puisque celle-ci est très basique. La fonction retourne simplement le nombre de lettres du mot entré en paramètre.

Le sens comme prescripteur Nous avons réutilisé l'API python FreNetic nous donnant accès aux définitions, sens de chaque candidat. Nous utilisons la même technique que pour FreNetic selector mais récupérons uniquement les différents sens du candidat en utilisant une expression régulière pour obtenir l'identifiant du candidat. Ensuite, nous utilisons un compteur que nous incrémentons à chaque fois que nous rencontrons une sens différent pour le candidat donné en argument. La fonction retourne le nombre de sens du candidat. Le problème principal de cette fonction est la qualité du lemmatiseur : les erreurs de celui-ci influencent directement les résultats puisque si la lemmatisation n'est pas bien réalisée, aucun sens n'est trouvé.

La fréquence comme prescripteur Nous recherchons dans la base de données le candidat et retournons sa fréquence. Si le mot n'est pas présent, alors nous retournons la valeur 0. Nous avons privilégié la fréquence provenant d'un corpus de sous-titres de films étant donné que ceux-ci contiennent du vocabulaire plus actuel. Contrairement à la fonction précedénte, nous n'avons pas dû utiliser de lemmatiseur étant donné que Lexique contient des lemmes assortis de leurs versions fléchies.

Pour finir, pour chaque fonction, nous avons créé une fonction principale recevant en argument le fichier victor avec les candidats sélectionnés à l'étape précédente. De cette façon, nous pouvons aisément calculer pour chaque candidat soit le nombre de sens, de lettres, sa fréquence ou vérifier son annotation dans un lexique spécialisé.

Nous avons créé respectivement les fonctions *main sens, main longueur, main frequence* qui retournent un dictionnaire où les clés sont les candidats et les valeurs soit des fréquences, un nombre de sens, un nombre de lettres. Dans chacune d'entre elle, nous avons dû faire appel à une fonction annexe classant soit les valeurs des plus grandes aux plus petites (*result ranking sens freq*) comme c'est le cas pour les fonctions retournant le nombre de sens et de fréquences, soit des plus petites aux plus grandes (*result ranking*) comme pour les fonctions retournant un nombre de lettres.

10.2 Une machine à vecteurs de supports pour classer des candidats

Pour finir, nous choisissons à l'image de la classe *SVMRanker* de Paetzold (2015), d'utiliser une machine à vecteurs de supports. Notre première idée a été de réutiliser entièrement le code de Paetzold. Cependant, nous avons eu du mal à comprendre le code de celui-ci et sans compréhension du code, il est très difficile de l'adapter à nos données. Pour pallier ce problème et parce que nous voulons tester une méthode supervisée, nous avons décidé de créer le module de A à Z.

Par ailleurs, nous pensons qu'implémenter nous-mêmes une machine à vecteurs de support avec la librarie *Scikit-learn* nous permettra de découvrir le monde merveilleux du machine learning. A la différence de Paetzold et François et al (2016), nous envisageons de traiter notre problème de classement comme un problème de classification, non pas sous la forme d'une classification par paires. Dès lors, à la fin du processus, nous nous retrouverons avec un résultat binaire (complexe ou non complexe) au lieu d'un classement à proprement parler mais nous pensons qu'une telle information peut être intéressante à la fin du processus de simplification. Afin d' obtenir un classement à la fin du processus, nous avons décider de coupler le prescripteur de fréquence, lequel donne de meilleurs résultats (cf. Résultats), à notre machine à vecteurs de support. De cette manière, nous supprimons les mots les plus complexes des candidats et effectuons un classement sur les mots considérés comme les plus simples.

10.2.1 Constitution d'un corpus d'apprentissage

Dans un premier temps, nous avions décidé de créer nous-mêmes un corpus d'apprentissage à partir d'un fichier fourni par Thomas François où les termes ont été annotés en fonction de leur complexité. Cependant, étant donné que nous utilisons également ces données pour l'établissement de notre corpus test, nous risquons le sur-apprentissage : le modèle prédictif (trop spécialisé) s'ajuste trop bien aux données d'entrainement. Ce sur-apprentissage a pour conséquence un modèle faisant de bonnes prédictions sur les données déjà vues mais celui-ci prédira mal de nouvelles données.

Dès lors, nous avons été forcés de trouver une autre méthode pour constituer notre corpus. Nous avons décidé d'utiliser un générateur de mot random [41] et d'utiliser notre module *features ranking* pour leur attribuer plusieurs caractéristiques ("*features*") comme la longueur, la fréquence et le nombre de sens. Nous avons utilisé le lexique annoté de Grabar et Hamon (2016) qui contient des termes techniques assortis des signes "/", "-" ,"+" [42] correspondant à des différents niveaux de compréhension ("je ne comprends

41. https ://www.palabrasaleatorias.com/mots-aleatoires.php
42. Nous n'avons pas pris l'ensemble des mots techniques présents dans le fichier parce que ceux-ci auraient été plus nombreux que les mots de la langue générale, nous pensons que cela aurait pu biaiser

pas","je ne suis pas sur de comprendre", "je comprends" . Ensuite, un membre de notre famille et nous-mêmes avons débattu sur plus ou moins 80 mots : un terme est considéré comme complexe lorsque nous jugeons qu'il est compliqué ou que nous avons des difficultés à l'expliquer alors qu'un terme est considéré comme simple lorsque celui-ci nous parait évident. Par la suite, nous avons réalisé un fichier VICTOR avec nos mots complexes et simples pour pouvoir utiliser le module *features ranking* puisque nous voulons prédire la complexité d'un mot à partir de son nombre de sens, de sa fréquence et de sa longueur. Enfin, nous avons créé un fichier csv reprenant toutes les informations utiles à l'entrainement de notre modèle.

10.2.2 SVM selector

Pour rappel, les machines à vecteurs de support (SVM) sont des algorithmes d'apprentissage automatique se basant sur un alogrithme linéaire imaginé par Vladimir Vapnik et Aleksandr Lerner en 1963 (Azencott, 2018).

Pour réaliser ce code, nous nous sommes inspirés du github suivant [43] et nous avons utilisé la bilbiothèque *Scikit-Learn* [44] destinée à l'apprentissage automatique. Le classificateur prend en entrée deux tableaux : le premier tableau *simplification* contient les données d'apprentissage (longueur, fréquence, sens) et le deuxième *type label*, les étiquettes de classes (0 ou 1). Après avoir adapté le modèle à l'aide de la fonction *fit*, nous sommes en mesure de prédire de nouvelles valeurs. Ensuite, nous avons créé la fonction *complexe or simple* pour prédire si un candidat est complexe ou simple à partir de sa longueur, le nombre de sens et sa fréquence à l'aide du module *features.py* et de ses fonctions *prescripteur_longueur, prescripteur_sens, prescripteur_frequence*. Puisque les fréquences étaient faibles pour la plupart des mots que nous avions testés, nous avions enlevé ce prescripteur. Cependant, nous avons relevé de meilleurs résultats avec le prescripteur *fréquence*. Effectivement, après avoir réalisé notre fonction, nous l'avons testée avec 10 mots et nous avons comparé nos résultats avec ceux de *ReSyf* :

nos résultats.

43. https ://github.com/adashofdata/muffin-cupcake/blob/master/muffin$_v s_c upcake_d emo.ipynb$
44. https ://scikit-learn.org/stable/modules/svm.htmlscores-probabilitie

Mot	SVM.py	ReSyf
inflammation	complexe	classé 4/5
myoplastie	complexe	/
oedème	complexe	classé 5/15
joli	simple	classé 8/66
superbe	simple	classé 15/66
ravissant	simple	classé 11/35
bellissime	complexe	classé 5/5
supercoquentieux	complexe	classé 4/4
charmant	simple	classé 8/58
sympathique	complexe	classé 4/9 et 1/2

Nous sommes en désaccord avec ReSyf sur les mots suivants : oedème et sympathique. Evidemment, à cette étape, nous avons uniquement testé avec très peu de mots mais ces résultats semblent plus en adéquation avec ceux de ReSyf. Pour cette raison, nous avons réintégré le prescripteur fréquence.

Enfin, nous avons inclus la fonction *complexe or simple* dans la fonction principale *main VM* où nous lisons le fichier VICTOR_CORPUS, extrayons les candidats, calculons les prescripteurs de chaque candidat à l'aide de notre module précédent, décidons si un mot est complexe ou pas. Si le mot est simple nous le rajoutons dans un dictionnaire dont la structure est la suivante : chaque clé contient le mot à substituer et chaque valeur, un tuple composé du candidat simple et de sa fréquence. Enfin, à l'aide de fonctions annexes, nous classons en ordre décroissant chaque candidat de chaque mot complexe et les imprimons dans un format victor adéquat. A la fin nous obtenons un fichier au format suivant : phrase mot candidat n1 candidat n2 etc.

Enfin, nous proposons un module final qui produit en sortie les phrases simplifiées. Nous avons utilisé, pour chaque étape, les meilleurs modules dans le but d'obtenir les meilleurs résultats que l'on puisse espérer. Ces modules ont été sélectionnés à partir des résultats que nous obtenus et commentés dans la section suivante. Dès lors, nous avons généré des synonymes à partir de la ressource ReSyf, sélectionné les meilleurs synonymes à partir du module FastText selector word et rangé les candidats à partir de notre machine à vecteur de support couplée au prescripteur de la fréquence. Etant donné que chaque candidat a été classé en fonction de sa complexité, nous avons sélectionné à chaque fois le premier candidat proposé parmi l'ensemble des candidats sélectionnés.

11 Conclusion

FreMLy propose 4 générateurs de candidats : deux exploitent la proximité sémantique dans l'espace vectoriel, l'un produit des substituts à l'aide d'une ressource synonymique et l'autre exploite le sens des mots complexes pour la création de paraphrases. Nous avons mis en exergue pour chacune des techniques les limites qui sont très variées puisque nous avons, par exemple, remarqué que le modèle préentrainé multilingue de BERT possédait un vocabulaire pauvre responsable de mauvais résultats. Nous avons également remarqué que ReSyf n'était pas une ressource efficiente pour les termes médicaux. Enfin, nous avons pointé du doigt le temps de chargement du modèle préentrainé de FastText.

Ensuite, l'outil propose 3 sélecteurs de mots en contexte : deux se basent encore sur la proximité sémantique partagée par deux mots dans l'espace vectoriel tandis que l'autre est basé sur l'aglortihme de Lesk Simple. Nous émettons des réserves quant au dernier module étant donné que la sélection des mots est basée sur l'enchassement entre les tokens de la définition du mot (signature) à désambiguïser et de son contexte. Or, les définitions des mots ne sont pas exhaustives.

Pour finir le processus, *FreMLy* propose de classer les candidats soit à l'aide de la fréquence de ceux-ci dans le corpus Lexique, soit à l'aide de leur longueur ou encore à l'aide du nombre sens de ceux-ci via l'api FreNeTic. Nous avions également opté pour une approche supervisée en couplant une machine à vecteurs de support au prescripteur de fréquence.

Troisième partie

Résultats

Cet utilme chapitre a pour but de présenter les résultats obtenus après avoir utilisé les différents modules créés pour la simplification automatique de textes médicaux et de la langue française générale. Notre objectif est d'évaluer chaque étape du processus de simplification en confrontant chaque technique utilisée à l'aide de deux corpus test (langue française et langue médicale). Dès lors, nous diviserons ce chapitre en 3 parties afin de discuter des résultats obtenus pour la génération de syonymes, la sélection des candidats et le classement de ceux-ci. En dernier lieu, nous discuterons des phrases simplifiées dans leur globabilité. Pour ce faire, nous employons diverses mesures mathématiques (rappel et précision) et linguistiques (préservation du sens, de la simplification et de la grammaticailité) qui différent en fonction du module évalué.

12 Evaluation des modules de génération de candidats

Pour l'évaluation de génération des candidats, nous avons choisi d'utiliser deux "mesures" sous les conseils de Thomas François : le nombre de candidats générés et la précision en calculant le nombre de bons candidats, c'est-à-dire ceux qui sont considérés comme cohérents, sur le nombre de candidats générés. Nous avons choisi d'annoter les bons candidats nous-mêmes sur base de notre intuition linguistique. Nous sommes conscients que pour plus de précision et pour un souci d'objectivité, nous aurions dû solliciter l'aide de tierces personnes pour réaliser l'exercice d'annotation. Cependant, étant donné les limites du temps imparti, nous n'avons pas été en mesure de réaliser un accord inter-juge.

12.1 Discussion des résultats

Pour obtenir les résultats ci-dessous, nous avons testé nos différents modules de génération de synonymes sur les deux corpus test (français et médical) que nous avions pris soin de constituer.

Nom du module	Candidats corpus français	Candidats corpus médical
Resyf Generators	2660 candidats générés	282 candidats générés
FastText Generators	1424 candidats générés	1304 générés

Avant d'entammer une discussion des résultats, nous souhaitons aborder le cas du module *BERT Generator* dont les résultats n'ont pas été repris dans cette section. Au cours de l'implémentation, nous nous sommes aidés d'une dizaine de phrases issues de notre corpus test. Nous avons constaté que sur la dizaine de phrases, seulement 5 ont reçu un résultat. Sur les 5 résultats, aucun n'a été jugé pertinent. Nous avons repris ci-dessous les 10 phrases et nous avons mis en évidence les termes à remplacer en gras. Les synonymes proposés se trouvent à côté de la phrase.

- Un matin de **vacances**, Delphine et Marinette s'insallèrent dans le pré avec leurs boites de peinture. Noël
- Marinette fit poser l'âne de **profil** et se mit à peindre. pierre
- Comme elles emplissaient les **godets** d'eau claire, l'âne vint à elles du fond du pré. None
- Les hommes doivent revêtir de lourds scaphandres qui assurent leur **survie** dans l'espace. None
- La vie dans l'espace, en dehors de l'atmosphère, présente des conditions très **particulières**. None
- Le castor est un excellent **nageur**. None
- Le castor peut non seulement abattre adroitement des arbres, mais il est aussi un **expert** pour la construction de barrages. moyen
- Il amusait tout le monde par ses pitreries, ses **propos** vifs et son accent si particulier. yeux
- Il amusait tout le monde par ses pitreries, ses propos vifs et son accent si **particulier**. bien
- Il était joufflu et **robuste**, souriant aux badauds accourus des ruelles environnantes. None

Conséquemment, nous avons considéré qu'il n'était pas intéressant de générer d'autres résultats, particulièrement pour les textes médicaux qui contiennent énormément de mots techniques. Nous pensons que ces mauvais résultats sont à imputer, comme nous l'avons déjà explicité dans la section précédente, à un modèle multilingue qui ne possède pas un vocabulaire assez étendu. Cette hypothèse s'est d'ailleurs confirmée lorsque nous avons testé notre module avec le modèle préentrainé anglais sur le fichier test de LEXenstein (lexmturk.txt) composé de 3 phrases. Sur l'ensemble des phrases, les 2 premières ont reçu des candidats qui se sont retrouvés dans les candidats proposés par Paetzold. Nous rapportons les phrases ci-dessous :

- Their escapades earned them a cover shot and a nude **pictorial** in a playboy spread. photo
- In 1971 , she **witnessed** the impoverished conditions of the trash collectors in cairo , egypt , and decided to live among them. noticed
- Nearby there is the goobang national park, and peak hill which **features** an open cut mine that can be toured during holidays. is

Dès lors, nous nous sommes concentrés sur les résultats de notre ressource lexicale ReSyf et des plongements lexicaux de FastText en mettant de côté le modèle de langage développé par Google. Lors de l'établissement de notre méthodologie, nous avions

rencontré des difficultés à trouver une ressource de qualité pour le domaine médical recoouvrant un large éventail de termes médicaux mais qui possèdent également des équivalents vulgarisés. Si ReSyf nous semblait être la ressource idéale pour la langue courante, elle paraissait moins intéressante pour la langue médicale. Dès lors, nous avions postulé que les plongements lexicaux étaient une réponse à notre problème étant donné que le modèle pré-entrainé pour les longements lexicaux sont souvent entrainé sur des énormes corpus dont notamment Wikipédia. Cette dernière est non seulement volumineuse mais contient égakement un grand nombre d'articles relatifs au domaine médical.

Si nous regardons les résultats ci-dessus, nous remarquons que notre postulat est confirmé : FastText a produit plus de candidats que ReSyf pour le domaine médical. Cependant, pour les phrases de la langue générale, nous remarquons une tendance inverse : ReSyf génère plus de candidats. Nous pensons que ce résultat est dû au fait que nous avons limité la génération de 10 synonymes par mot pour le module FastText Genrator : en effet, au-delà, le modèle propose des mots incohérents, tandis que ReSyfproduit parfois jusqu'à 50 synonymes par mot (tous sens confondus).

Cependant, générer un grand nombre de candidats n'implique pas nécessairement que le système ait généré des candidats cohérents. C'est ce que nous avons vérifié manuellement : pour chaque candidat proposé par mot complexe, nous avons sélectionné ceux qui nous semblaient les plus cohérents. Dès lors, nous avons pu calculer les valeurs suivantes :

Nom du module	Précision corpus français	Précision corpus médical
Resyf Generators	14% (382 bons candidats/2660 propositions)	23% (66/282)
FastText Generators	11% (158 bons candidats/1424 propositions)	3%(44/1304)

FastText Nous observons que malgré le fait que FastText ait proposé de nombreux candidats, très peu sont cohérents surtout dans le domaine médical. Nous notons les différentes observations réalisées lors de l'analyse des résultats. Parmi les "mauvais" résultats, nous avons remaqué que FastText propose :

— Des mots partageant un même espace sémantique mais qui ne sont pas synonymes : par exemple, le modèle propose *césarienne* pour *épisiotomie* parce que ce sont deux opérations chirurgicales réalisées dans le cadre d'une grossesse difficile et que ces deux mots sont souvent l'un a côté de l'autre. Cependant, ce sont deux opérations distinctes. C'est également la même chose pour *paludisme* et *onchocercose* qui sont toutes les deux des maladies dont la cause est parasitaire.

— Des mots ayant le même préfixe : nous remarquons que FastText propose pas mal de mots partageant le même préfixe que le mot cible comme c'est le cas pour le mot *fibrinogène* (protéine présente dans le plasma sanguin) [45] et les propo-

45. https ://www.doctissimo.fr/html/sante/analyses/ana$_p$*roteines*08.*htm*

sitions *fibrinolytique (agents antithrombotique*[46]*,fibrine (fibrinogène transformé lors de la coagulation*[47]. Même si ces mots sont parents, ils ne signifient pas la même chose et ne sont pas substituables.

— Des mots partageant les mêmes suffixes : pour *gangliosides* nous retrouvons *osides, polyosides, oglioside, glucioside.*

— Des n-grams : Nous avions choisi FastText parce que celui-ci, à la différence de Word2vec, fonctionne par n-grammes et cette différence semblait être un avantage surtout pour les termes médicaux. En effet, fonctionner par n-grammes permet de générer de meilleurs résultats pour les mots raires ou pour les mots hors vocabulaire. Cependant, surtout lors de l'analyse des résultats pour le corpus de la langue générale, nous avons remarqué que certaines propositions possédaient des "traces" de ces n-grammes, comme c'est le cas pour le mot *vacances* : vacances. vacances.c' vacances.pour.

— Des nombreuses variantes du terme à substituer : par exemple pour le mot *escalader*, nous obtenons les propositions suivantes *escaladera, escladait escalada, escaladés.*

Concernant les bons résultats, nous avons remarqué que le modèle propose :

— Des hyperonymes : par exemple, pour *onychomycose*, nous obtenons l'hyperonyme *mycose.* C'est le cas également pour *carotidien* qui obtient comme hyperonyme *artère* étant donné que les carotides sont des artères ou le terme *arthrose* qui obtient *rhumatisme*, un terme générique pour tous les problèmes articulaires.

Par conséquent, FastText propose parfois des mots cohérents (qui se retrouvent également dans ceux générés par ReSyf mais la plupart du temps, le modèle propose des mots que nous ne pouvons pas considérer comme étant des synonymes. Cette conclusion rejoint celle de Hmida et al (2019) qui a fait l'observation suivante : les plongements lexicaux pouvaient produire des mots qui ne sont pas des synonymes provenet de relations non appropriées comme l'antonymie.

ReSyf Nous trouvons que ReSyf est très exhaustif dans ses propositions de synonymes : en effet, plusieurs synonymes sont proposés en fonction du sens du mot. De cette manière, comme nous l'avons dit plus haut, la ressource propose parfois une cinquantaine de mots alors que seulement 5 sont cohérents en contexte. C'est pour cette raison que ReSyf obtient beaucoup de propositions pour peu de "bons" candidats. Cependant, à la différence de FastText, nous trouvons la qualité des synonymes bonnes étant donné que nous ne trouvons pas de "déchets", c'est-à-dire des mots qui n'ont strictement avoir avec le terme à substituer. Evidemment, cette qualité est due au fait que la ressource a été réalisée expréssement dans l'optique de proposer des synonymes. Pour finir, il est amusant de remarquer que FastText obtient une meilleure précision pour les termes médicaux, cela est du au fait que celle-ci propose beaucoup moins de

46. https ://www.em-consulte.com/article/15728/traitements-fibrinolytiques
47. https ://fr.wikipedia.org/wiki/Fibrinog%C3%A8ne

synonymes, et ce qui implique un écart moins important entre les bons candidats et le nombre de candidats générés par mot, mais également parce que certains termes de la langue courante ont été empruntés par la langue médicale. Conséquemment, ReSyf génère quant même des candidats pour certains termes mais quand ceux-ci sont trop techniques, la ressource ne propose aucune alternative.

Dès lors, lorsque les candidats ont été générés, il s'agit de sélectionner ceux qui sont cohérents en contexte. Grâce aux annotations que nous avons réalisées précédemment, nous allons pouvoir comparer si les mots que nous considérions comme cohérents ont également été sélectionnés par nos modules de sélection qui sont au nombre de 3. Comme mesures, nous avons choisi de calculer "rappel" et précision dans la section suivante.

13 Evaluation des modules de sélection de candidats

Comme nous l'avons annoncé dans la fin de la section précédente, nous avons choisi de calculer les rappel et précision de chaque module de sélection. Pour rappel, nous avons réalisé un module desambiguïsation basé sur l'algorithme de Lesk principalement parce que celui-ci ne nécessite pas de corpus annotés et que ceux-ci sont rares pour le domaine médical. Ensuite, nous avons également utilisé les plongements lexicaux pour la phase de sélection puisqu'une fois de plus nous n'avons pas besoin de corpus annotés, seulement d'un modèle préentrainé. Nous avons procédé de deux manières différentes : le premier module propose de comparer le vecteur du mot complexe avec celui de chaque candidat tandis que le deuxième, réalisé par Paetzold (2016), compare le vecteur de la phrase avec le mot complexe à chaque vecteur de chaque candidat.

Pour calculer la précision, nous proposons de regarder le nombre de bons candidats sur le nombre de mots sélectionnés par le module. Tandis que pour le rappel, nous regardons le nombre de bons candidats sur le nombre total de bons mots à récupérer : de cette manière, le fait qu'une méthode rate des candidats sera perceptible. Dès lors, nous avons testé notre module basé sur l'algorithme de Lesk Simple (Frenetic Selector) sur les ressources de langues médicale et courante produites par ReSyf et FastText. Ensuite, nous avons testé les deux modules (FastText selector word et Paetzold vector selector) de plongements lexicaux sur uniquement les ressources de langues médicale et courante produites par ReSyf. Nous avons remarqué que lorsque nous utilisons les plongements sur les résultats produits par FastText, tous les candidats sont sélectionnés. Cette conséquence est plutôt logique étant donné que nous avons utilisé FastText à la fois pour la génération et pour la sélection : le modèle a généré des synonymes qui lui semblaient cohérents en contexte.

13.1 Discussion des résultats

13.1.1 Sélection des candidats générés à partir de la ressource lexicale *ReSyf*

Ci-dessous, nous pouvons observer les différentes mesures de rappel et précision obtenues pour les différents modules. Pour rappel nous déterminons la mesure selon la formule suivante : nombre de bons candidats/nombre de candidats à sélectionner. D'abord, nous rapportons les résultats obtenus à partir des candidats générés par *ReSyf* :

Nom du module	Rappel corpus français	Rappel corpus médical
Frenetic selector	13% (52/382)	15 % (10/66)
Paetzold vector selector	34% (112/382)	51% (34/66)
FastText selector word	35% (138/382)	30% (26/66)

Pour la précision, nous calculons le nombre de bons candidats sur l'ensemble des candidats sélectionnés par le module :

Nom du module	Précision corpus français	Précision corpus médical
Frenetic selector	22% (52/232)	23 % (10/42)
Paetzold vector selector	8% (112/1264)	19% (34/175)
FastText selector word	25% (138/540)	49% (26/53)

Fenetic Selector Nous remarquons que Frenetic selector obtient les plus mauvais résultats en terme de rappel par rapport aux deux autres modules, tandis que si nous regardons la précision, le module surpasse celui de Paetzold. Ces résultats sont directement en lien avec les limites de l'algorithme que nous avions déjà énoncées auparavant. Le problème avec Lesk Simple, c'est que celui-ci est basé sur la disponibilité de définitions et le chevauchement entre les définitions trouvées pour chaque candidat et celle du mot complexe. Dès lors, lorsque les candidats sont très proches du mot complexe et qu'ils ne sont pas trop spécifiques, le modèle sélectionne plutôt bien comme en témoigne les exemples ci-dessous :

— Regarde, Jean, dit-elle, on dirait quelque enfant perdu, abandonné par ses **compagnons**. complice compagnie ami familier
— Tout à coup il vit une **auto** noire qui s'arrêtait tout près de lui. automobile
— Promets-moi de me suivre dans sept ans et je veillerai à ce que vous viviez jusque-là dans l'**opulence**. richesse fortune abondance

Mais lorsque celui-ci doit faire face à des mots trop complexes, l'algorithme ne trouve pas de définitions soit pour le mot compliqué soit pour les candidats soit pour les deux en même temps et ne sélectionne aucun mot. Dès lors, puisque très peu de candidats

sont sélectionnés par le module, nous avons moins de chance que les mots sélectionnés se retrouvent dans notre liste de référence.

Cependant, nous remarquons que le module rapporte de meilleurs résulats pour la langue médicale alors que celle-ci comporte des termes spécifiques : selon notre raisonnement, nous devrions avoir moins de chances de trouver des définitions pour ces termes. Par exemple, le module a sélectionné des substituts pour les termes suivants : antipsychotique, tuberculose, hémorragie, aïgue, carcinome, mastectomie, etc. Nous pensons que ces mots bien que faisant partie du langage médical sont des mots qui sont rentrés dans la langue courante, dans notre usage : dans ce cas, ceux-ci ont leurs entrées dans les dictionnaires et donc des définitions. Par contre, les termes tels *asthénie, dyskinésie, ménorragie* qui pourtant avaient de bons candidats au départ n'ont pas été sélectionné faute d'une trop grande technicité.

Paetzold vector selector Nous obtenons de très bons résultats en comparaison au module précédent, surtout pour les textes médicaux : le module sélectionne 51 % des synonymes médicaux de notre liste de référence. Cependant, lorsque nous regardons la précision, nous obtenons des mauvais scores. En effet, si le rappel est très bon c'est parce que le module *Paetzold Generator* a sélectionné beaucoup de candidats. Dès lors, nous avons beaucoup plus de chance de trouver de bons synonymes parmi ceux sélectionnés.

Lors de la sélection des termes dans le corpus de langue générale, nous remarquons que le module présente des difficultés à faire une sélection lorsque les mots possèdent beaucoup de candidats. Or, contrairement au corpus médical, certains termes possèdent une cinquantaine de synonymes. Par exemple, pour le terme *abracadrantes*, le module sélectionne 17 candidats, parmi les 57 originaux. Lors de la sélection, nous avons nous-mêmes éprouvé des difficultés lorsque de nombreux synonymes étaient proposés étant donné que beaucoup se ressemblent à quelques nuances près. Dès lors, nous pensons que nous pouvons attribuer ces erreurs au seuil décidé lors de notre méthodologie. Malgré plusieurs tests, il semblerait que nous ayons restreint le seuil de manière trop laxiste et par conséquent sélectionne trop de candidats.

Pour finir nous avions choisi ce module parce que nous pensons qu'il serait plus performant lorsque les candidats étaient polysémiques et portaient à confusion. Nous avions relevé les phrases suivantes :

— Mais la plus spectaculaire adaptation des dromadaires au désert, c'est leur **faculté** de survivre pendant des mois sans boire (...)
— L'accident survient la nuit, par des défaillances du circuit secondaire, non-nucléaire, suivies par la perde d'éanchéité de l'**enceinte** du circuit d'eau primaire.
— La dissection de l'**artère** carotide interne extracrânienne est une cause majeure d'AVC chez les patients plus jeunes.

Le module a mal interprété le terme *faculté* puisqu'il propose des termes qui n'ont pas le sens de faculté en tant que *aptitude*. Il en va de même pour le terme *enceinte* puisqu'il a parmi des bons candidats, sélectionné *grossesse*. Par contre, pour artère, le module propose *vaisseau sanguin* dans ses sélections contrairement à *Frenetic selector* qui ne propose aucun synonyme tandis que *Word Vector selector* propose le synonyme avenue. Les résultats ne correspondent pas à nos attentes et infirme le postulat de départ selon lequel nous aurions de meilleurs résultats étant donné une prise en compte du contexte du mot complexe.

FastText selector word Ce dernier module obtient des résultats similaires au module précédent pour le rappel que nous avons mesuré à partir du corpus français. Cependant, nous remarquons un rappel nettement différent pour les textes médicaux : Paetzold Generator obtient 51% contre 30% pour celui que nous avons réalisé. Cette différence peut s'expliquer en observant les précisions des deux modules. Pour rappel, Paetzold generator décroche un très bon score de rappel mais un mauvais score précision étant donné que parmi sa sélection, seulement 8% et 10% sont de bons candidats. Tandis que le module FastText selector word reçoit de meilleures scores de précision puisque 25% des mots sélectionnés, dans le corpus français et 49% des mots sélectionnés dans le corpus médical sont considérés comme justes et cohérents. Donc, le premier retient pas mal de synonymes qui figurent das notre liste de référence en raison d'une très grande sélection tandis que le deuxième retient moins de synonymes mais ceux-ci sont plus souvent considérés comme bons. Notre solution est également la même, nous pensons que pour de meilleurs résultats, il serait intéressant restreindre le seuil que nous avions délimité.

Si nous continuons à le comparer avec le module précédent, le module de vecteurs que nous avons créé sélectionne pour la langue française la moitié des candidats sélectionnés par le sélecteur créé par Paetzold. Cependant, nous pensons qu'il sélectionne également beaucoup de synonymes pour le corpus de langue général : nous avions retenu 382 mots cohérents et le module en sélectionne 540. C'est pourquoi nous pouvons lui adresser la même critique qu'à son concurrent : le module est indécis lors de la sélection de candidats parmi un grand nombre de propositions.

13.2 Sélection des candidats générés à partir des plongements lexicaux (FastText)

De la même manière que pour la sous-section précédente, nous discuterons des rappels et précisions obtenus suite à la sélection réalisée par Frenetic selector :

Nom du module	Rappel corpus français	Rappel corpus médical
Frenetic selector	21% (33/158)	9 % (4/44)

Nom du module	Précision corpus français	Précision corpus médical
Frenetic selector	13% (33/239)	4 % (4/103)

Nous obtenons des moins bons résultats avec ce module qu'avec la ressource ReSyf. Comme nous l'avons déjà mentionné auparavant, FastText produit pas mal de mauvais substituts dont nous avons établi une typologie dans la section précédente. Conséquemment, très peu de synonymes sont sélectionnés et quand ceux-ci le sont, ils se révèlent souvent être des variantes du mot complexe. Nous pensons, par exemple, au verbe *acomplit* dont les propsistions suivantes ont été sélectionnées *accomplissant, accomplira, accomplissent, accomplissait, accompli, acomplisse, effectue*. Par conséquent, si nous couplons l'algorithme de Lesk dont nous avons déjà expliqué les limites à une mauvaise ressource, nous obtenons des résultats peu convaincants.

En comparant et en discutant des résultats, il ressort que le sélecteur FastText Selector Word obtient les meilleurs résultats. Dès lors, pour l'analyse des résultats de classement, nous avons réalisé nos test sur des ressources produites par Resyf Generator pour la génération de synonymes et par Word Vector Selector pour la sélection.

14 Evaluation des modules de classement des candidats

Pour finir le processus de simplification, nous devons classer les candidats qui ont été sélectionnés en fonction de leur facilité sur base de plusieurs critères. Pour ce faire, nous avons réalisé deux modules : l'un se basant sur des prescripteurs linguistiques (sens, fréquence, longueur), l'autre utilisant principalement une machine à vecteurs de support. Pour l'évaluation, nous avons établi, sous les conseils de Rémi Cardon, que calculer des mesures de rappel et de précision n'était pas approprié. C'est pourquoi nous expliquons notre méthode d'évaluation dans le paragraphe suivant.

Nous pensons qu'évaluer le classement des mots selon notre intuition serait trop subjectif et beaucoup trop ardu. Par conséquent, nous avons utilisé la ressource lexicale ReSyf puisque celle-ci contient des synonymes gradués en fonction de leur complexité. Lorsque le candidat a plus d'un synonyme, nous vérifions dans la ressource si les synonymes suivent le même ordre que dans la ressource. Si c'est le cas, nous attribuons un score de 1. Dans le cas contraire, un score de 0. Lorsque nous avons plus de 3 synonymes, nous avons décidé de vérifier à chaque fois l'ordre des 3 premiers.

14.1 Discussion des résultats

Pour comparer nos résultats, nous utilisons la formule suivante : le nombre de bons classement/le nombre de mots possédant plus d'un synonyme.

Nom du module	Corpus test français	Corpus test médical
Longueur	36/82 (44%)	5/15 (33%)
Sens	28/82 (32%)	7/15 (47%)
Fréquence	37/82 (45%)	11/15 (73%)
SVM + Fréquence	43/63 (68%)	3/3 (100%)

14.1.1 Features

Ce module propose de classer les candidats soit à partir de leur longueur, de leur fréquence ou de leur nombre de sens. Nos différents postulats ont déjà été énoncés dans la section précédente.

La longueur S'il a déjà été démontré que les termes les plus faciles sont souvent les plus courts, nous remarquons que nous ne pouvons pas faire de cette observation, une règle générale. Cependant, nous observons quand même un résultat sensiblement meilleur du côté du corpus de langue générale. Effectivement, comme nous l'avions postulé auparavant, utiliser la longueur comme prescripteur de difficulté pour des termes médicaux n'est pas cohérent étant donné la concision de certains mots (acronymes, abréviations, sigles, etc).

Le sens Pour aller plus loin qu'un simple precripteur de surface, nous avions décidé d'exploiter le nombre de sens. Nous avions repris le postulat d'Elhadad (2006) pour construire notre module : plus un mot de sens, plus celui-ci a des chances d'être simple. Finalement, les résultats sont, à quelques mots près, les mêmes que pour la longueur. Le problème qui, avait déjà été souligné par Elhadad (2006), est que certains mots faciles possèdent seulement 1 voire 2 définitions . Il arrive aussi que nous ne trouvons pas le terme dans WoNeF, dès lors, celui-ci est directement considéré comme complexe. Par exemple, si nous prenons le terme complexe *compassion* et les trois premiers synonynes *sensibilité, humanité, pitié*. Ceux-ci obtiennent respectivement des scores de 12, 9 et 6. Cependant, *pitié* est indiscutablement plus facile que les termes le précèdant. Dès lors, nous remarquons qu'à l'image du prescripteur de la longueur, utiliser le nombre de sens n'est pas très précis.

La fréquence Pour finir, nous avons choisi d'utiliser la fréquence comme indicateur de complexité. Nous remarquons que la fréquence est le meilleur prescripteur pour la fréquence puisque nous avons réalisé 11 classements corrects sur base de ReSyf. Pour la langue médicale, les résultats sont beaucoup plus modérés puisque nous obtenons le même résultat que lorsque nous utilisons le prescripteur de longueur.

Conséquemment, nous pensons qu'utiliser uniquement un seul prescripteur pour classer les candidats est une technique qui n'apporte pas des résultats satisfaisants. C'est pour cette raison que nous avons choisi d'utiliser une machine à vecteurs de supports couplée au meilleur prescripteur, celui de la fréquence.

SVM Enfin, les résultats obtenus suite au module SVM sont nettement meilleurs que les précédents. L'avantage avec cette méthode est que nous avons procédé à une nouvelle étape de sélection puisque nous avons à l'aide de notre machine défini et retenu les mots simples parmi les sélectionnés et supprimé les complexes. De cette manière, nous obtenons non seulement des candidats qui sont bien classés mais qui sont également plus faciles.

15 Evaluation globale des phrases simplifiées

Pour finir, nous choisissons d'évaluer les phrases simplifiées que nous obtenons grâce aux différentes combinaisons de modules. Nous choisissons d'utiliser toutes les mesures utilisées auparavant (rappel et précision) ainsi que les mesures linguistiques énoncées dans l'introduction des résultats.

15.1 Phrases simplifiées pour la langue générale

Comme nous l'avons explicité dans la section précédente, pour arriver aux résultats que nous allons analyser, nous avons réalisé chaque étape avec le module obtenant les meilleurs scores. Nous avons choisi d'analyser nos résultats à l'aide des mesures de précision et de rappel. Pour déterminer si le terme remplaçant est plus facile que le mot à substituer, nous nous sommes référés à la ressource ReSyf de façon à réaliser une évaluation objective. A la manière de Rémi Cardon (2018), nous avons également choisi d'évaluer manuellement la grammaticalité et la préservation du sens des phrases qui ont été simplifiées.

15.1.1 La précision

Sur l'ensemble de nos 200 phrases tests, 102 ont été simplifiées. Nous obtenons donc donc une précision de 51%. Parmi ces phrases non simplifiées, certaines auraient dû être simplifiées comme c'est le cas pour les phrases :

- Comme elles emplissaient les **godets** d'eau claire, l'âne vint à elles du fond du pré.
- Mariette alla chercher une **brassée** de branches qu' elle jeta sur les braises.
- Mme Morot **interrompt** son récit.

Tandis que d'autres n'attendaient pas nécessairement des simplifications étant donné qu'elles comportaient déjà des mots simples : lorsque notre système ne proposait pas de simplification, nous nous sommes référés à ReSyf pour voir s'il existait des synonymes plus faciles que le mot. Ce cas de figure démontrent que notre outil fait des choix "intelligents". Par exemple, il ne simplifie pas les phrases suivantes :

- Un matin de **vacances**, Delphine et Marinette s'installèrent dans le pré avec leurs boites de peinture.
- Tiens, à **présent** tout son manteau blanc a fondu
- Il assena au malheureux un second coup, à la suite duquel la **moelle** de la jambe s'écoula, et le chevalier expira sur l'heure.
- Le vent c'est l'air mis en mouvement par différentes **causes**.

15.1.2 Le rappel

Ensuite, sur l'ensemble des phrases simplifiées, nous avons évalué celles qui étaient réellement simplifiées : sur les 102 simplifiées, 73 ont été correctement simplifiées c'est-à-dire qu'elles proposent un synonyme plus simple que le mot de départ :

- Je n'aurais jamais cru que j'avais une tête de **bouledogue**. -> Je n'aurais jamais cru que j'avais une tête de **chien**.
- Les hommes doivent revêtir de lourds scaphandres qui assurent leur **survie** dans l'espace. -> Les hommes doivent revêtir de lourds scaphandres qui assurent leur **vie** dans l'espace.
- Le cheval **paisible** et fort, domestique (on voit le dessous de son fer près de la tête du soldat mort) représente l'humanité souffrante. -> Le cheval **tranquille** et fort, domestique (on voit le dessous de son fer près de la tête du soldat mort) représente l'humanité souffrante.

Il arrive que notre système propose des synonymes plus complexes comme dans les exemples suivants :

- Les feuilles mortes s'envolent alors dans des **tourbillons**. -> Les feuilles mortes s'envolent alors dans des **vortex**.
- Pendant des milliers d'années, cette énergie a permis le transport des hommes et des **marchandises** et la découverte de terres inconnues. -> Pendant des milliers d'années, cette énergie a permis le transport des hommes et des **cargaisons** et

la découverte de terres inconnues.
— Le castor peut non seulement abattre adroitement des arbres, mais il est aussi un **expert** pour la construction de barrages. -> Le castor peut non seulement abattre adroitement des arbres, mais il est aussi un **spécialiste** pour la construction de barrages.

15.1.3 La préservation du sens

Dans cette sous-section, nous jugeons, parmi l'ensemble des phrases simplifiées dont nous avons considéré que le mot proposé était plus simple que le mot cible, la préservation du sens. En effet, il arrive que le candidat soit plus simple mais qu'il ne soit pas sémantiquement correct :
— Pour ne pas se mettre hors de **combat** elle-même, l' araignée se laisse des passages dans sa toile. -> Pour ne pas se mettre hors de **guerre** elle-même, l' araignée se laisse des passages dans sa toile.
— Tous les animaux dorment, c'est obligatoire pour vivre mais tous ne le font pas au même **moment**. -> Tous les animaux dorment, c'est obligatoire pour vivre mais tous ne le font pas au même **temps**.
— Le castor peut non seulement **abattre** adroitement des arbres , mais il est aussi un expert pour la construction de barrages. -> Le castor peut non seulement **tuer** des arbres , mais il est aussi un expert pour la construction de barrages.
Ce sont les seuls cas que nous avons relévé parmi les 73 phrases simplifiées correctement.

15.1.4 La grammaticalité

Nous n'avons pas utilisé de module permettant de corriger la grammaticalité de nos énoncés. Par conséquent, nous avons retrouvé des termes qui étaient plus simples, qui étaient cohérents mais dont la grammaticalité n'a pas été respectée. Par exemple, dans certaines phrases, nous rencontrons des problèmes d'accords :
— Il suivit l'**allée** des arbres. -> Il suivit l'**rue** des arbres.
— Par une nuit d'hiver, ils virent quelque chose d'**étrange** dans la cour. ->Par une nuit d'hiver, ils virent quelque chose d'**drôle** dans la cour
— Mariette alla chercher une brassée de branches qu' elle jeta sur les **braises**. -> Mariette alla chercher une brassée de branches qu' elle jeta sur les **feu**.
Nous obtenons un score de 9% puisque seulement 7 énoncés ne sont pas grammaticalement corrects parmi les phrases correctement simplifiées. Cependant, nous avons remarqué que nous avons eu, par exemple, peu de verbes conjugués simplifiés. Dès lors, il est normal que notre système obtienne un score honorable pour la préservation de la grammaire. Nous pensons toujours qu'un module du style de *MorphAdorner* serait intéressant.

15.2 Phrases simplifiées pour la langue médicale

Comme nous l'attendions, les résultats pour la langue médicale sont extrêment pauvres. Ceux-ci démontrent que nous avons utilisé des ressources qui n'étaient pas adaptées aux textes médicaux. Nous nous attendions à des résultats décevants étant donné que nous avions testé la ressource *ReSyf* avec des termes médicaux mais nous étions curieux de savoir à quel point celle-ci n'était pas adaptée. En réaction, nous avons testé une autre manière de simplifier les textes médicaux avec une ressource plus spécialisée. Avant d'en commenter les résultats, nous décortiquons d'abord les résultats ci-dessous.

15.2.1 Précision, rappel, sens et grammaticalité

Nous obtenons seulement 9 phrases simplifiées sur un total de 200 phrases et donc une précision de 4%. Sur les 9 simplifiées, 5 donnent des synonymes plus simples comme en témoignent les phrases suivantes :

- Il se présente aux urgences pédiatriques d'un centre hospitalier général pour fièvre depuis la veille, vomissements, refus alimentaire, **asthénie**. -> Il se présente aux urgences pédiatriques d'un centre hospitalier général pour fièvre depuis la veille, vomissements, refus alimentaire, **fatigue**.
- L'homme a des antécédents de douleur chronique, de dépression, de reflux, d'onychomycose, de rétention urininaire et du VIH. -> L'homme a des antécédents de douleur chronique, de dépression, de reflux, d'**mycose**, de rétention urininaire et du VIH.

Cependant, les 5 synonymes sélectionnés ne sont pas tous grammaticaux comme le démontre l'exemple précédent puisque nous pouvons observer un problème de déterminant. Nous observons également un problème grammatical avec la phrase suivante où nous rencontrons un souci de déterminant et d'accord :

- La rétinopathie de la prématurité (RP) est une maladie complexe des vaisseaux sanguins rétiniens en développement et est l'une des principales causes de **cécité** infantile évitable. -> La rétinopathie de la prématurité (RP) est une maladie complexe des vaisseaux sanguins rétiniens en développement et est l'une des principales causes de *aveugle* infantile évitable.

L'exemple précédent démontre également un problème au niveau sémantique étant donné que nous attendions un nom et que notre outil a proposé un adjectif. Conséquemment nous obtenons des scores de 60 % (3/5) pour la préservation du sens et de la grammaticalité.

15.3 Phrases simplifiées pour la langue médicale bis

Contrairement au module précédent, nous avons choisi non pas de simplifier en utilisant des synonymes mais en paraphrasant c'est-à-dire en expliquant les mots complexes. Nous pensons que ce module va obtenir une meilleure précision que le précédent étant donné que le dictionnaire électronique des mots est 2 fois plus volumineux que la ressource lexicale *ReSyf*.

Conséquemment, nous n'utilisons pas les mêmes mesures d'évaluation que pour les autres résultats de cette sous-section. En effet, ce module génère des paraphrases, les mesures telles que la préservation du sens et de la grammaticalité sont donc exclues. Nous évaluons uniquement le rappel, la précision et la simplification (est-ce que la paraphrase générée aide le lecteur ?).

15.3.1 La précision

Nous estimons qu'utiliser la précision est une mesure intéressante dans la simplification lexicale médicale : cela nous permet de mesurer la couverture des termes spécifiques. Plus la couverture est importante, plus nous aurons de chance d'obtenir des synonymes (des paraphrases, dans ce cas). Nous obtenons une précision de 56 % étant donné que 112 mots trouvent une paraphrase explicative sur les 200 identifiés et sont donc simplifiés. Parmi les mots non-identifiés, nous avons pu dégager certaines tendances :

- **Une mauvaise lemmatisation** : certains mots ne sont pas bien lemmatisés comme *nephrologue* lemmatisé *nephrologu*, *anticholériques* qui reste au pluriel ou bien *folinique* lemmatisé *foliniq*. De cette manière, ces mots mal lemmatisés ne peuvent se retrouver dans le dictionnaire électronique.
- **Des mots issus de la langue générale** : pour limiter les erreurs dûes à la polysémie de certains termes, nous avions décidé d'exploiter uniquement les sens des domaines suivants : "anatomie", "pathologie', "biologie", "médecine", etc. Cependant, la langue médicale emprunte de nombreux termes à la langue générale ou à d'autres domaines tels que *fongique* appartenant à la botanique, *systémique* aux domaines de l'économie ou de l'éducation ou encore *site*, *aigu*, agent, etc.
- **Des mots trop techniques** : plusieurs ne se retrouvent pas dans le dictionnaire à cause d'une trop grande technicité (*colloïdale, intracapsulaire, laparoscopique*) ou parce qu'ils forment une expression (*carcinome hépatocellulaire, drainage postural*. Nous retrouvons également les acronymes ou sigles (*CEC, BPO, NSC*) qui sont assez fréquents dans le jargon médical.

15.3.2 Le rappel

Pour le rappel, nous comptons le nombre de paraphrases qui aident le lecteur et rendent la phrase plus simple. Pour réaliser cette analyse, nous avons décidé d'évaluer nous-mêmes manuellement ce critère avec l'aide d'un membre de nos familles. Nous avons calculé que sur les 112 paraphrases générées, 45% apportent des informations supplémentaires au lecteur et facilitent sa lecture. Nous avons réalisé une typologie parmi les paraphrases :

- **Celles qui simplifient** :
  - L'homme a des antécédents de douleur chronique, de dépression, de reflux, d'onychomycose (mycose des ongles), de rétention urininaire et du VIH.
  - Nouvelle visite à domicile par la remplaçante qui pose la diagnostic d'une artérite(inflammation artères) sévère.
  - En cas de sténose(étroitesse conduit) de l'artère vertébrale, une intervention chirurgicale est techniquement difficile, potentiellement dangereuse et n'est pas envisagée de la plupart des centres.
  - La réparation de la hernie inguinale(jonction cuisse-tronc) est l'opération la plus fréquente en chirurgie générale.
  - Le patient est en arrêt cardiaque avec un glasgow, une mydriase(dilatation pupille) bilatérale et malgré la réanimation, il n'y a aucune reprise de l'activité cardiaque.
- **Celles qui utilisent des mots complexes** : certaines font appel à d'autres termes complexes comme en témoignent les phrases ci-dessous.
  - Les gangliosides(glycolipide) pourraient avoir un effet protecteur sur les systèmes nerveux central et périphérique.
  - Les personnes infectées par le virus de l'immunodéficience humaine présentent un risque augmenté de développer une tuberculose(dû à bacille Koch) (TB) active.
  - Il existe de plus en plus de preuves indiquant que les hormones sexuelles stéroïdes(hormone endocrine) ont un effet bénéfique sur un certain nombre de facteurs de risque des maladies artérielles périphérique.
  - La nicotine est un anticholinergique qui possède également un effet présynaptique en libérant de l'acétylcholine(médiateur parasympathique).
- **Celles qui sont expliquées par des abréviations** Les sens dans le dictionnaire électronique sont parfois raccourcis à l'aide d'abréviations. Parfois, celles-ci sont naturellement compréhensibles :
  - Il est possible qu'un déficit d'oestrogène soit un facteur étiologique(causes d maladies) dans le développement de l'incontinence urinaire chez la femme.

— La clonidine(ct hypertension) était à l'origine utilisée pour abaisser la tension artérielle.
— La thrombolyse artérielle périphérique est utilisée dans la prise en charge de l'ischémie(rupt circul sang) artérielle périphérique.
— "La caféine a des actions pharmacologiques ; c'est un faible bronchodilatateur(q dilate bronches) et elle réduit également la fatigue des muscles respiratoires"

D'autres fois, elles sont totalement incompréhensibles et embrouillent le lecteur :

— Il constatait que l'enfant était cyanosée(qn r/d cyanosé), sans pouls fémoral.
— Évaluer la sécurité et l'efficacité de différents matériaux pour l'angioplastie(pr réfect vaisseau) carotidienne

15.4 Résumé

Pour clôturer nos recherches nous proposons une discussion sur la recherche que nous avons menée en faisant une synthèse du dernier chapitre mais également en mettant en lumière les différents aspects qui peuvent être améliorés.

Lors de l'évaluation du premier module, ReSyf s'est démarqué du modèle de plongements lexicaux pour la langue générale en générant 2680 candidats contre 1424 pour FastText. La ressource lexicale, en plus de produire une quantité importante de synonymes, a généré 14% de candidats cohérents pour notre corpus de 200 phrases issus de la langue générale contre 11% pour FastText. Par contre, concernant la langue médicale, nous remarquons que FastText propose beaucoup plus de candidats que sa concurrente, mais seulement un faible pourcentage (3%) de candidats s'est retrouvé dans notre liste de référence tandis que ReSyf obtient un score de 23%. Quant au réseau neuronal BERT, celui-ci a été mis hors compétition suite à des résultats trop incohérents

Ensuite, après avoir analysé nos 3 modules de sélection sur les résultats générés par ReSyf Generator et FastText Generator : nous avons obtenu les moins bons résultats pour le module basé sur l'algorithme de LeskSimple. Les meilleurs résultats sont attribués aux modules de sélection par vecteurs. Cependant, si Paetzold vector selector bat son concurrent au niveau de la précision, FastText selector word le bat également au niveau du rappel. En effet, le Paztzold vector selector a tendance a sélectionné beaucoup de candidats. Dès lors, il y a plus de chances pour que les candidats que nous avons jugé cohérents se retrouvent dans la sélection du module.

Enfin, les derniers modules de classement ont été testés sur les résultats générés par la ressource lexicale ReSyf et sélectionnés à partir du module FastText selector word puisque nous avons établi que ceux-ci étaient les meilleurs lors de la comparaison

des résultats. Sans surprise, nous avons observé que les prescripteurs employés seuls donnaient des résultats peu convaincants. Cependant, nous avons noté des résultats sensiblement différents pour le prescripteur de la fréquence. Ce prescripteur a donné d'avantages de bons résultats lorsque nous l'avons couplé à une machine à vecteurs de supports que nous avons entrainée sur un corpus d'apprentissage composé par nos soins.

Pour finir, nous avons confronté nos résultats finaux. Dans cette section, ce qui nous intéressait, c'était la comparaison des résultats entre la simplification de textes médicaux réalisée à partir de ReSyf et du dictionnaire électronique DEM. Comme nous l'avons postulé dès le départ, les résultats du premier sont très pauvres en raison de l'utilisation d'une ressource inadaptée puisque sur le total de 200 phrases, seulement 9 ont été simplifiées. Les résultats du deuxième sont nettement meilleurs puisque nous obtenons une précision de 56 %. Evidemment, la couverture des termes techniques n'est pas encore totale puisque plusieurs ne se retrouvent pas dans le dictionnaire électronique. Mais nous jugeons cette méthode efficiente puisque nous obtenons un rappel de 45 % signifiant que parmi les phrases simplifiées, 45% d'entre elles apportent des informations supplémentaires et facilitent la lecture. Si nous nous risquons à une comparaison avec le travail de Cardon (2018), nous obtenons une meilleure couverture de nos termes techniques puisque celui-ci obtient sur un total de 7 893 phrases, 86 phrases simplifiées. Parmi ces 86 phrases simplifiées, il juge que 14.86 % sont plus faciles après simplification. Evidemment, pour que notre comparaison soit plus cohérente, nous devrions tester nos modules sur un corpus test beaucoup plus conséquent.

Concernant les résultats de nos simplifications sur les textes de langue générale, nous aurions voulu les comparer aux résultats effectués par Paetzold et Specia (2015). Cependant, celui-ci n'a pas utilisé les mêmes mesures pour l'obtention de ses résultats. Il nous semble donc inutile de comparer des résultats qui sont incomparables. Dès lors, nous avons préféré les comparer de la même manière que Cardon (2018) aux résultats de *Devlin* et *Biran* qui utilisent comme critères la simplicité, le sens et la grammaticalité :

Critères	Notre module	Devlin	Biran
Simplicité	71.5%	46.43%	75.58%
Grammaticalité	95.8%	46.43 %	75.58%
Sens	41.09%	55.95%	46.43%

Comme nous l'avions fait remarqué précédement, il aurait été plus cohérent de tester nos modules sur un ensemble plus large de phrases test afin d'avoir un échantillon plus varié.

16 Conclusion

L'objectif de notre mémoire était de créer un outil de simplification pour les textes médicaux et de langue générale. En effet, grâce à notre implication autour du projet CLEAR, nous avons été séduits par le fait d'adapter des méthodes de TAL au domaine médical. Dès lors, nous avons souhaité reprendre notre sujet de stage qui était l'adaptation de la bibliothèque anglophone *LEXenstein* (Paetzold et Specia, 2015) aux textes médicaux et la langue française.

Pour réaliser notre outil, nous avons d'abord procédé à un état de l'art sur les domaines de la simplification automatique, de la simplification lexicale pour la langue générale et pour les textes médicaux. Nous avons également ajouté une section orientée linguistique qui avait pour but de mettre en lumière les différentes composantes du langage médical. Grâce à cette partie théorique, nous avons établi que la simplification des textes médicaux et de langue courante utilisent les mêmes techniques à quelques nuances près. Pour l'étape de l'identification des mots complexes et le classement, Elhadad (2006), par exemple, a préféré adapter les prescripteurs classiques au domaine médical démontrant ainsi que la langue médicale comporte plusieurs spécifités par rapport au français. Par exemple, celle-ci comporte une grande proportion de termes dont le sens est défini par des spécialités et caractérisés par leur monosémie. Dans ce cas, Elhadad (2006) décide d'exploiter les sens des candidats pour identifier les termes complexes. L'étape de génération de candidats ne diffère pas non plus puisque pour les deux langues, nous pouvons utiliser des plongements lexicaux, des ressources lexicales ou corpus parallèles. La plus grande difficulté est de trouver des modèles d'entrainement ou corpus pour la langue médicale étant donnné que ceux-ci doivent être à la fois spécifiques (pour assurer une large couverture des termes médicaux) et simples (pour produire des synonymes plus simples). Grabar et Hamon (2016), eux, ont choisi d'exploiter une des spécificité de la langue médicale : sa morphologie. De cette manière, il peuvent créer des paraphrases expliquant les mots complexes.

Après avoir établi les fondements théoriques de nos recherches, nous pu commencer l'implémentation de *FreMLy*. Nous avons choisi pour la génération de candidats d'exploiter les plongements lexicaux BERT et FastText et leurs capacités à trouver dans un espace de représentation vectoriel des mots ayant des sens proches. Nous avons également exploité deux ressources lexiques : ReSyf et DEM. Nous avons traité la deuxième ressource d'une manière différente et originale puisque nous avons extrait le sens des définitions des candidats à la simplification de sorte à obtenir des paraphrases comme Grabar et Hamon (2016). Enfin, nous avons proposé l'implémentation l'algorithme de Lesk Simple et l'utilisation de vecteurs pour l'étape de désambiguïsation. Pour finir, nous avons choisi de classer les candidats à l'aide de prescripteurs linguistiques tels que le nombre de sens, la fréquence et la longueur. Nous avons calculé le nombre de sens par le biais de l'API FreNeTic similaire à Wordnet mais en français et la fréquence à partir de Lexique. Nous avons souhaité, pour clôturer la partie méthodologie, op-

ter pour une méthode d'apprentissage supervisée telle que les machines à vecteurs de supports et créé un corpus dans ce but.

Enfin, nous avons analysé nos résultats. Au regard des résultats obtenus par Cardon (2018), Devlin et Biran, les nôtres semblent honorables. Cependant, nous considérons pour notre module de langue médicale bis qu'une ressource plus spécialisée serait intéressante puisque nous avons remarqué que plusieurs termes techniques n'ont pas été couverts dont des abréviations. Or, ce type de terme est très présent dans le langage médical et nous pensons qu'il serait judicieux de s' y intéresser de plus près.

17 Discussion

Nous souhaitons discuter des limites de notre outil de simplification. Outre notre souhait de trouver une ressource encore plus spécialisée pour une meilleure couverture des termes techniques, nous avons identifié 3 autres limites à notre travail. La première limite a mettre en lumière est le format utilisé. En effet, celui-ci peut apparaitre comme contraignant puisque l'utilisateur doit lui-même rentrer ses données. Dès lors, il aurait été intéressant d'intégrer un module qui puisse convertir les données dans le format adéquat.

Une des améliorations qu'il est également souhaitable d'apporter est l'utilisation d'un module de flexion lors du processus de simplification. Puisque nous ne touchons pas aux candidats, ni aux mots de la phrase lors du processus, nous risquons des énoncés agrammatiaux. Bien sûr, puisque nous avons testé nos modules sur un petit échantillon et qu'aucune phrase avec un verbe n'a été simplifiée, nos chances d'obtenir des énoncés agrammaticaux dans ces recherches a été réduite. Mais nous observons des problèmes au niveau des déterminants et de certains accords.

Enfin, au niveau méthodologique, nous pensons que solliciter l'aide d'annotateurs aurait été une méthode plus rigoureuse. En effet, même si nous avons essayé pour la constitution de nos corpus, de demander de l'aide à un membre de la famille, il aurait été préférable de réunir plus de personnes. C'est également le cas pour l'analyse de nos résulats, même si nous nous sommes aidés de *ReSyf* pour respecter un maximum d'objectivité, l'intervention de personnes tierces aurait été souhaitable.

Références

[1] Bert : Le "transformer model" qui s'entraîne et qui représente. https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente. Accès : 11-10-19.

[2] Deep transfer learning – le traitement du langage À l'aube d'une rÉvolution ? https://weave.eu/deep-transfer-learning-nlp-revolution/. Accès : 11-10-19.

[3] Using bert with pytorch. https://medium.com/@noa.kel/using-bert-with-pytorch-b9624edcda4e. Accès : 11-10-19.

[4] Sandra Aluisio and Caroline Gasperin. *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, chapter Fostering Digital Inclusion and Accessibility : The PorSimples project for Simplification of Portuguese Texts, pages 46–53. Association for Computational Linguistics, 2010.

[5] Drilon Avdiu, Vanessa Bui, and Klára Ptačinová Klimčíková. Predicting learner knowledge of individual words using machine learning. In *Proceedings of the 8th Workshop on NLP for Computer Assisted Language Learning*, pages 1–9, Turku, Finland, 30 September 2019. LiU Electronic Press.

[6] Chloé-Agathe Azencott. Introduction au machine learning. Dunod, 2018.

[7] Mokhtar Boumedyen Billami and Núria Gala. Creating and validating semantic signatures : application for measuring semantic similarity and lexical substitution. In *Traitement Automatique des Langues Naturelles TALN 2017*, Orléans, France, 2017.

[8] Or Biran, Samuel Brody, and Noémie Elhadad. Putting it simply : a context-aware approach to lexical simplification. In *Proceedings of the 49th ACL*, pages 496–501, 2011.

[9] Stefan Bott, Luz Rello, Biljana Drndarevic, and Horacio Saggion. Can spanish be simpler ? lexsis : Lexical simplification for spanish. In *Proceedings of CoLing*, pages 357–374, 2012.

[10] Marie Candito. Analyse sémantique automatique. 2017.

[11] Rémi Cardon. Approche lexicale de la simplification automatique de textes médicaux. In *COnférence en Recherche d'Informations et Applications - CORIA 2018, 15th French Information Retrieval Conference, Rennes, France, May 16-18, 2018. Proceedings*, 2018.

[12] John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10, 1998.

[13] John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, June 1999. Association for Computational Linguistics.

[14] R. Chandrasekar, Christine Doran, and B. Srinivas. Motivations and methods for text simplification. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 1041–1044, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[15] R. Chandrasekar and B. Srinivas. Automatic induction of rules for text simplification, 1997.

[16] Jinying Chen, Abhyuday Jagannatha, Samah Fodeh, and Hong Yu. Ranking medical terms to support expansion of lay language resources for patient comprehension of electronic health record notes : Adapted distant supervision approach. *JMIR Medical Informatics*, 5 :e42, 10 2017.

[17] Jinying Chen, Jiaping Zheng, and Hong Yu. Finding important terms for patients in their electronic health records : A learning-to-rank approach using expert annotations. *JMIR Medical Informatics*, 4 :e40, 11 2016.

[18] Jan De Belder and Marie-Francine Moens. A dataset for the evaluation of lexical simplification. In *Proceedings of the 13th CICLing*, 2012.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[20] Mark Dras. Tree adjoining grammar and the reluctant paraphrasing of text. Technical report, 1999.

[21] PHILIP EDMONDS and Adam Kilgarriff. Introduction to the special issue on evaluating word sense disambiguation systems. *Natural Language Engineering*, 8 :279 – 291, 12 2002.

[22] Noemie Elhadad. Comprehending technical texts : Predicting and defining unfamiliar terms. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, 2006 :239–43, 02 2006.

[23] Christiane Fellbaum. *WordNet : An Electronic Lexical Database*. Bradford Books, 1998.

[24] Thomas François, Mokhtar Boumedienne Billami, Núria Gala, and Delphine Bernhard. Automatic ranking of synonyms according to their reading and comprehension difficulty. In *JEP-TALN-RECITAL 2016*, volume 2 of *TALN*, pages 15–28, Paris, France, 2016.

[25] Nuria Gala, Thomas François, Delphone Bernhard, and Cédrick Fairon. Un modèle pour prédire la complexité lexicale et graduer les mots. *TALN*, 2014.

[26] Goran Glavaš and Sanja Stajner. Simplifying lexical simplification : Do we need simplified corpora ? 07 2015.

[27] Natalia Grabar, Vincent Claveau, and Clément Dalloux. CAS : French Corpus with Clinical Cases. In *LOUHI 2018 - The Ninth International Workshop on Health Text Mining and Information Analysis*, Ninth International Workshop on Health Text Mining and Information Analysis (LOUHI) Proceedings of the Workshop, pages 1–7, Bruxelles, France, October 2018.

[28] Natalia Grabar and Thierry Hamon. Exploitation de la morphologie pour l'extraction automatique de paraphrases grand public des termes médicaux. *Traitement Automatique des Langues*, 57(1) :85–109, October 2016.

[29] Serge Guérin. Emploi de termes hybrides gréco-latins dans le langage médical. *Meta : Journal des traducteurs*, 46 :7, 01 2001.

[30] Firas Hmida, Mokhtar Boumedyen Billami, Thomas François, and Nuria Gala. Assisted Lexical Simplification for French Native Children with Reading Difficulties. In *The Workshop of Automatic Text Adaptation, 11th International Conference on Natural Language Generation*, Tilbourg, Netherlands, 2018.

[31] Colby Horn, Cathryn Manduca, and David Kauchak. Learning a lexical simplifier using wikipedia. volume 2, pages 458–463, 06 2014.

[32] Arnaldo Jr, Erick Maziero, Caroline Gasperin, Thiago Pardo, Lucia Specia, and Sandra Aluisio. Supporting the adaptation of texts for poor literacy readers : a text simplification editor for brazilian portuguese. *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, 06 2009.

[33] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*, volume 4. 04 2011.

[34] Anne-Laure Ligozat, Cyril Grouin, Anne Garcia-Fernandez, and Delphine Bernhard. Approches à base de fréquences pour la simplification lexicale. In *TALN-RECITAL 2013*, volume 1, pages 493–506, Les Sables d'Olonne, France, 2013.

[35] Sylvie Monin. La siglaison en langue médicale et problèmes de traduction. 1993.

[36] Sylvie Monon. Termes éponymes en médecine et application pédagogique. *ASp*, pages 217–237, 1996.

[37] Marie-Françoise Mortureux. Les vocabulaires scientifiques et techniques. 12 2019.

[38] Gustavo Paetzold and Lucia Specia. LEXenstein : A framework for lexical simplification. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 85–90, Beijing, China, July 2015. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

[39] Gustavo Paetzold and Lucia Specia. SemEval 2016 task 11 : Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California, June 2016. Association for Computational Linguistics.

[40] Gustavo Paetzold and Lucia Specia. Understanding the lexical simplification needs of non-native speakers of English. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, pages 717–727, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[41] Bénédicte PierreJean. Comprendre et utiliser les words embddings. 2019.

[42] Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. A simple bert-based approach for lexical simplification. *CoRR*, abs/1907.06226, 2019.

[43] Luz Rello and Ricardo Baeza-Yates. Good fonts for dyslexia. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 14 :1–14 :8, New York, NY, USA, 2013. ACM.

[44] Maurice Rouleau. Complexité de la phrase en langue de spécialité : mythe ou réalité ? le cas de la langue médicale. 2006.

[45] Horacio Saggion and Graeme Hirst. *Automatic Text Simplification*. Morgan & Claypool Publishers, 2017.

[46] Isabel Segura-Bedmar and Paloma Martinez. Simplifying drug package leaflets written in spanish by using word embedding. *Journal of Biomedical Semantics*, 8, 12 2017.

[47] Advaith Siddharthan. A survey of research on text simplification. *ITL - International Journal of Applied Linguistics*, 165 :259–298, 01 2014.

[48] Lucia Specia and Gustavo Paetzold. Lexical simplification with neural ranking. In *EACL*, 2017.

[49] Anaïs Tack, Thomas François, Anne-Laure Ligozat, and Cédrick Fairon. Modèles adaptatifs pour prédire automatiquement la compétence lexicale d'un apprenant de français langue étrangère. *JEP-TALNRECITAL*, 01 2016.

[50] Phillipe Thoiron and Henri Béjoint. La terminologie, une question de termes ? *Meta, 55(1)*, pages 105–118, 2010.

[51] Laurens van den Bercken, Robert-Jan Sips, and Christoph Lofi. Evaluating neural text simplification in the medical domain. In *The World Wide Web Conference*, WWW '19, pages 3286–3292, New York, NY, USA, 2019. ACM.

[52] Florentina Vasilescu, Philippe Langlais, and Guy Lapalme. Evaluating variants of the lesk approach for disambiguating words. *Proceedings of the Conference of Language Resources and Evaluations (LREC 2004)*, 01 2004.

[53] Sara Vecchiato and Sonia Gerolimich. La langue médicale est-elle « trop complexe » ? *Nouvelles perspectives en sciences sociales*, 9 :81, 01 2013.

[54] VG Vinod, Q Mei, and K Zheng. Amia annu symp proc. *Journal of Biomedical Semantics*, 11 2014.

[55] Qing Zeng-Treitler, Eunjung Kim, Jon Crowell, and Tony Tse. A text corpora-based estimation of the familiarity of health terminology. volume 3745, pages 184–192, 10 2005.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté de philosophie, arts et lettres
Place Blaise Pascal, 1 bte L3.03.11, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/fial**