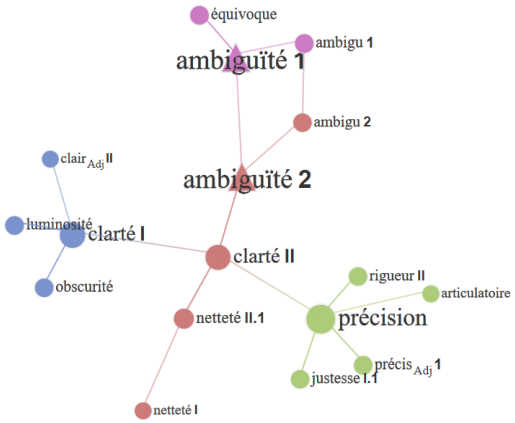
Travaux de désambiguïsation lexicale : rapport de stage

Clémentine Bleuze

L3 MIASHS Sciences Cognitives

23/05/2022 - 12/08/2022



Référent IDMC :
Geoffray Bonnin
bonnin@loria.fr

Référents ATILF :
Mathieu Constant
mathieu.constant@atilf.fr
Sandrine Ollinger
sandrine.ollinger@atilf.fr

Remerciements

Je tiens à remercier Mathieu Constant et Sandrine Ollinger pour leur accompagnement et leur disponibilité tout au long de mon stage. Leurs explications et leurs conseils m'ont à la fois permis d'acquérir de nouvelles connaissances et de découvrir le monde de la recherche, ce qui s'est révélé très intéressant.

Je remercie également l'ensemble des membres du laboratoire qui m'ont accueillie, aidée et ont répondu à mes questions ou échangé au sujet de leurs recherches. Un merci tout particulier à Delphine Barbier-Jacquemin, responsable Communication et Valorisation à l'ATILF, qui a pris le temps de m'éclairer sur l'organisation du laboratoire.

Table des matières

1 Introduction

1.1 La désambiguïsation lexicale

La *désambiguïsation lexicale* ou *désambiguïsation sémantique* (Word Sense Disambiguation) est une tâche centrale en Traitement Automatique des Langues (TAL) (Navigli, 2009). En effet, la plupart des mots que nous employons recouvrent plusieurs sens (ils sont *polysémiques*), et s'il est la plupart du temps aisé pour un locuteur de comprendre le sens d'un mot sans ambiguïté à l'aide de son contexte, cette tâche reste difficile à automatiser.

Dans une tâche de désambiguïsation lexicale, on cherche à attribuer aux occurrences de mots polysémiques d'un énoncé une classe correspondant à leur sens le plus probable : la désambiguïsation est une tâche de *classification*. On appelle *inventaire de sens* une proposition de découpage en classes (en sens) d'un ensemble de mots donné. On parle de *désambiguïsation ciblée* lorsqu'on chercher à lever l'ambiguïté sur un mot-cible d'un énoncé, par opposition à la *désambiguïsation globale* qui s'applique à tous les mots de celui-ci (All Words Disambiguation).

> **Remarque:** Si le terme "mot" facilite ici la compréhension de la tâche décrite, nous l'abandonnerons par la suite (justement en raison de son ambiguïté) pour des termes plus précis (3.1.1).

La désambiguïsation lexicale trouve de nombreuses applications dans le TAL : classification de documents (ex : analyse de sentiments), extraction d'informations, analyse et génération du discours, recherche au sujet du lexique...

1.2 Mission du stage

La mission que j'ai effectuée pendant mon stage consistait à tester des méthodes de désambiguïsation lexicale basiques sur des couples de sens issus du *Réseau Lexical du français (RL-fr)*, un système lexical créé et développé à l'ATILF (cf 3.1). Cette approche est originale en cela que l'inventaire des sens du RL-fr diffère des inventaires issus des dictionnaires traditionnels ou de ressources lexicales comme *WordNet*[1] : la richesse de la description des unités lexicales caractéristique au RL-fr pourrait en faire une ressource particulièrement pertinente pour la levée d'ambiguïté sémantique (Veronis, 2001).

Ce travail s'inscrit dans la continuité du projet *BEL-RL-fr*, lancé en 2018 sous la responsabilité de Sandrine Ollinger, ingénieure de recherche CNRS, dans le cadre des travaux sur les Systèmes Lexicaux menés à l'ATILF par Alain Polguère, professeur à l'Université de Lorraine et tous deux membres du laboratoire (cf 3.2).

1. `https://wordnet.princeton.edu/`

J'ai donc effectué des expériences de désambiguïsation ciblée, ramenées à des problèmes de classification binaire (deux sens seulement ayant été retenus par mot-cible, bien qu'ils en regroupent souvent davantage).

> **Exemple:** Ces énoncés illustrent les deux sens retenus pour le mot "long" :
>
> — *Elle porte une jupe d'été légère mais* ***longue*** *et un petit haut sans manches.*
> — *Cette nuit a été bien* ***longue****.*

Au fur et à mesure de mon stage, j'ai pu découvrir et mettre progressivement en place plusieurs méthodes de désambiguïsation lexicale basiques (au sens où elles ne font pas appel à des réseaux de neurones), faire des observations et des comparaisons entre les expériences, puis analyser qualitativement une partie des résultats de manière plus poussée.

Pour cela, j'ai également pu me documenter, tant sur les systèmes lexicaux et sur le RL-fr en particulier que sur les classifieurs existants pour la désambiguïsation ainsi que les librairies Python qui permettent de les implémenter.

Enfin, j'ai eu l'occasion de m'entretenir régulièrement avec mes maîtres de stage pour leur exposer mes avancées, poser mes questions et réfléchir avec eux aux ajustements nécessaires.

2 Présentation de l'ATILF

2.1 Le laboratoire

L'*ATILF*[2] (Analyse et Traitement Informatique de la Langue Française) est un laboratoire de recherche en sciences du langage issu de la fusion en 2001 de deux unités de recherche : *l'INaLF* (Institut National de la Langue Française - CNRS) et *LanDisCo* (Langues, Discours, Cognition - Université Nancy 2). En 2006, l'Atilf intègre également le *CRAPEL* (Centre de Recherches et d'Applications Pédagogiques en Langues - Université Nancy 2, désormais Université de Lorraine). L'ATILF est une unité mixte de recherche, sous la double tutelle du CNRS et de l'Université de Lorraine.

FIGURE 1 – Le site *Linguistique* de l'ATILF (CLSHS de Nancy)

Fort d'une double compétence en linguistique et en informatique, l'ATILF porte un projet scientifique orienté vers l'étude du lexique et des aspects dynamiques du langage, ainsi que le développement de corpus et de ressources linguistiques.

L'ATILF met à disposition du public et de la communauté scientifique de nombreuses ressources et outils de référence dans le domaine de la linguistique : le laboratoire est notamment porteur d'*ORTOLANG*[3] (Outils et Ressources pour un Traitement Optimisé de la LANGue), un Equipex[4] proposant un réservoir conséquent de données, corpus, lexiques, dictionnaires...utiles aux spécialistes comme au grand public. Depuis son lancement en 2015, *ORTOLANG* intègre les ressources issues de ses prédecesseurs : le *CNRTL*[5] (Centre National de Ressources Textuelles et Lexicales), créé dix ans plus tôt par le CNRS et adossé à l'Atilf, et le *SLDR* (Speech

2. `https://www.atilf.fr/`
3. `https://www.ortolang.fr/fr/accueil/`
4. Équipement d'excellence validé dans le cadre des investissements d'avenir PIA (`https://www.gouvernement.fr/le-programme-d-investissements-d-avenir`)
5. `https://www.cnrtl.fr/`

and Language Data Repository).

L'ATILF diffuse également le *TLFi*[6] (Trésor de la Langue Française informatisé), la version informatisée du *TLF*, un dictionnaire du français des XIXème et XXème siècles en 16 volumes, parus entre 1971 et 1994. Ce dictionnaire fait encore aujourd'hui figure de référence lexicographique et permet une consultation libre de définitions et d'exemples qui retracent l'histoire de plus de 100 000 mots de la langue française. Notons par ailleurs qu'une partie des données initialement rassemblées dans le but de fournir des exemples pour le TLF a donné naissance à la base de données *Frantext*, toujours développée et maintenue à l'ATILF. Désormais riche de plus de 5400 références issues de textes littéraires, philosophiques, scientifiques et techniques en langue française, Frantext permet de faire des recherches variées sur des formes, lemmes ou catégories grammaticales. Aujourd'hui, pas moins de 170 bibliothèques, universités et centres de recherches à travers le monde utilisent cette ressource.

Citons encore à titre d'exemple de projet mené à l'ATILF *EDOlang*[7], une plateforme pour l'apprentissage des langues bien connue des étudiants de l'Université de Lorraine.

2.2 Organisation

L'ATILF regroupe 135 membres organisés au sein de 5 équipes de recherche et 4 services de soutien à la recherche (voir organigramme page suivante).

Les équipes de recherche sont les suivantes :

- **Lexique**
- **Linguistique historique française et romane**
- **Discours**
- **Didactique des langues et sociolinguistique**
- **Ressources : normalisation, annotation et exploitation**

J'ai effectué mon stage sous la supervision de Mathieu Constant et de Sandrine Ollinger, respectivement membres des équipes Ressources et Lexique.

6. `http://atilf.atilf.fr/tlf.htm`
7. Environnement et Dispositifs Ouverts pour l'apprentissage des langues : `https://edolang-app.univ-lorraine.fr/accueilMenu/carnetBord`

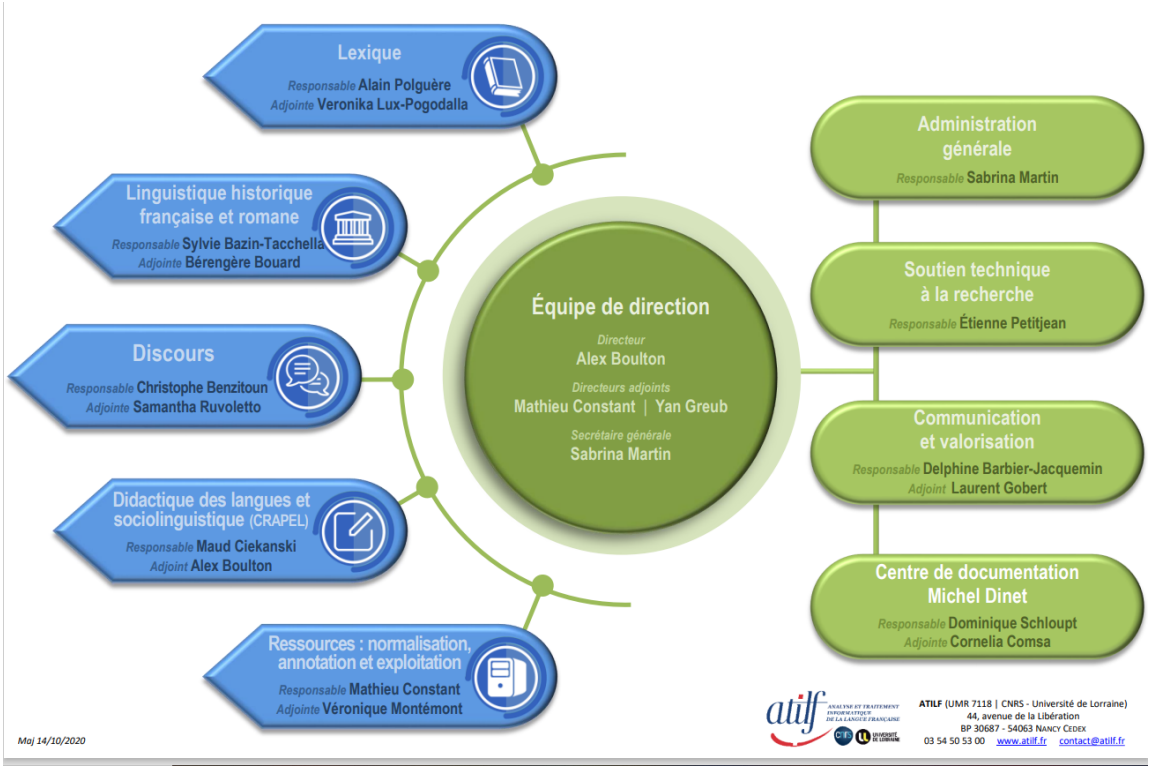


FIGURE 2 – Organigramme structurel de l'ATILF

2.2.1 L'équipe Ressources : normalisation, annotation et exploitation

L'équipe *Ressources : normalisation, annotation et exploitation* se donne pour mission de composer, d'annoter et d'outiller des corpus de texte destinés à l'exploitation scientifique. Elle s'articule autour des deux axes d'ingénierie et de recherche :

— **Côté ingénierie :** fabrication et traitement de corpus textuels, développement d'outils automatisés
— **Côté recherche :** analyse du contenu des corpus, TAL

L'équipe Ressources est notamment en charge de l'enrichissement de la base Frantext et de sa documentation (tandis que l'équipe de soutien technique à la recherche s'occupe de l'instrument d'interrogation de la base et de l'étiquetage grammatical des données). Par ailleurs, Mathieu Constant y a mené entre 2016 et 2020 le projet *PARSEME-FR* dédié au traitement automatique des expressions polylexicales du français (telles que "tirer son épingle du jeu" ou "tourner la page").

2.2.2 L'équipe Lexique

L'équipe *Lexique* se structure autour de 5 axes de recherche, dans une perspective essentiellement synchronique :

— Morphologie constructionnelle
— Lexicologie théorique, descriptive et appliquée
— Lexicographie franco-allemande
— Du lexique à la phrase
— Lexique et corpus

C'est dans l'axe *Lexicologie théorique, descriptive et appliquée* que s'inscrivent les travaux sur les systèmes lexicaux dont dérive le projet BEL-RL-fr auquel contribue mon stage.

3 Contexte du stage

Pour mon stage, j'ai mené des expériences de désambiguïsation lexicale basées sur un inventaire de sens ainsi que des données issues d'une ressource créée à l'ATILF : le Réseau Lexical du français (RL-fr). Ce travail s'inscrit dans le cadre du projet BEL-RL-fr, visant à valoriser la base d'exemples lexicographiques du RL-fr, pour mener une réflexion sur les notions d'ambiguïté et de polysémie. Après avoir présenté ces deux éléments, je présenterai les données de travail de mon stage et les expériences menées.

3.1 Le Réseau Lexical du français (RL-fr)

Le Réseau Lexical du français (que nous noterons désormais exclusivement RL-fr) est un *système lexical* du français contemporain, créé et développé à l'ATILF depuis 2011. Pour illustrer cette présentation, nous utiliserons des captures obtenues à l'aide de *Spiderlex*, le navigateur d'exploration du RL-fr accessible en ligne [8].

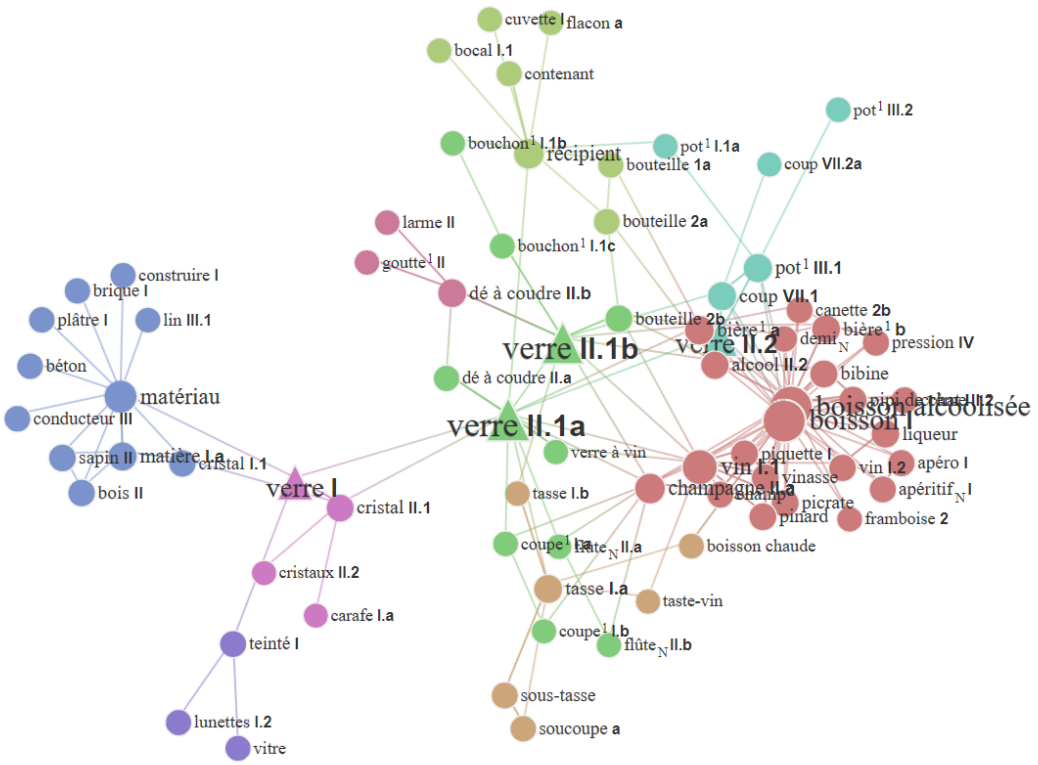


FIGURE 3 – Espace lexical du lemme VERRE

8. https://spiderlex.atilf.fr/fr

3.1.1 Le projet RELIEF

L'histoire du RL-fr remonte à 2011, avec le lancement du projet *RELIEF* (REssource Lexicale Informatisée d'Envergure sur le Français) sous la responsabilité d'Alain Polguère et de Jean-Marie Pierrel (chef de file), avec le soutien de l'Agence de Mobilisation Economique de la Region Lorraine et du FEDER Lorrain, et la collaboration de la société MVS de Saint-Dié [9]. Achevé en 2014, ce projet se donnait pour double objectif de développer une ressource lexicale de nouvelle génération et de valoriser cette dernière dans des activités liées au TAL et à la lexicographie synchronique du français à l'ATILF.

C'est sur la base des recherches en *Lexicologie Explicative et Combinatoire* menées à l'*OLST* (Observatoire de linguistique Sens-Texte - Université de Montréal) par Igor Mel'čuk et ses collaborateurs (dont Alain Polguère) que s'inscrivent les travaux sur les Systèmes Lexicaux actuellement en cours à l'ATILF. Le RL-fr, produit du projet RELIEF, constitue donc formellement un système lexical : c'est un modèle lexical de type *réseau*, dont les noeuds sont principalement des *unités lexicales*, et les arcs des *fonctions lexicales* orientées [10]. La particularité du RL-fr (par rapport aux dictionnaires classiques) repose justement sur cette ossature relationnelle qui structure l'information lexicale. Nous précisons la notion de fonction lexicale en section 3.1.3.

> **Remarque:** Il est utile d'introduire certains termes de vocabulaire auxquels nous aurons désormais fréquemment recours (nous abandonnons ici le terme "mot") :
> - un **lemme** est la forme canonique d'un mot variable. Il regroupe plusieurs formes fléchies appelées **mots-formes**. Ex : "dormirai", "dormant", "dors" sont des mots-formes du lemme DORMIR.
> - une **lexie** ou **unité lexicale** correspond à un **sens** donné d'un lemme. Un lemme possédant plusieurs sens est dit **polysémique**. Ex : Le lemme VERRE est polysémique. Il peut désigner un *matériau*, comme un *objet* fait de ce matériau.

3.1.2 Noeuds du RL-fr (unités lexicales)

Chaque unité lexicale recensée dans le RL-fr est associée à un code (constitué de chiffres romains, de chiffres arabes et de lettres) qui permet de la distinguer des autres sens du lemme auquel elle est associée (voir fig.3 : plusieurs lexies associées au

9. `https://www.mvs.fr/index.php?page=presentation`
10. Quelques rares noeuds correspondent également à des *clichés linguistiques* (ex : "Je vous en prie") ou à des *constructions* (ex : "de_N en_N" où les N sont des noms) . En plus des fonctions lexicales, le RL-fr encode des relation de copolysémie , d'inclusion formelle et d'inclusion sémantique définitionnelle (Polguère, 2014).

lemme VERRE, et leurs voisins lexicaux). Un noeud correspond à une unité lexicale, et encapsule plusieurs informations (fig.4) :

— des **caractéristiques grammaticales :** classe grammaticale, traits morpho-syntaxiques
— une **définition** rendant compte du schéma actanciel de la lexie
— des **exemples lexicographiques** illustrant sans ambiguïté l'emploi de la lexie
— un ensemble d'**arcs entrants et sortants** des autres noeuds du graphe, étiquettés par une fonction lexicale.
— des **expressions phraséologiques** incluant cette lexie ...

FIGURE 4 – Quelques informations contenues dans le noeud $verre_{II.1a}$

Toujours en construction, le RL-fr recense 29 220 unités lexicales dans sa version 2.1.

3.1.3 Arcs du RL-fr (fonctions lexicales)

Nous avons dit que l'information du RL-fr reposait sur son ossature relationnelle, et plus particulièrement sur des *fonctions lexicales* (FL). Développées dans le cadre de la théorie *Sens-Texte*, ce sont des fonctions (au sens mathématique du terme)

dont les arguments sont des unités lexicales et les valeurs des ensembles d'unités lexicales (Mel'čuk et Polguère, 2021). Elles servent à décrire la combinatoire restreinte des unités lexicales du langage, et se divisent en deux catégories [11] :

— les **FL paradigmatiques** : leurs arguments et valeurs sont liés par une relation paradigmatique, c'est-à-dire de *substitution* : arguments et valeurs sont substituables
— les **FL syntagmatiques** : leurs arguments et valeurs sont liés par une relation syntagmatique, c'est-à-dire de *collocation* : arguments et valeurs apparaissent ensemble en contexte

Exemple: Syn est une FL paradigmatique, tandis que Magn est une FL syntagmatique.
— **Synonymie** : Syn(*voiture*) = *automobile*, *auto*, *bagnole* (fam.)
— **Intensificateur** : Magn(*peur*) = *bleue* (postposé)

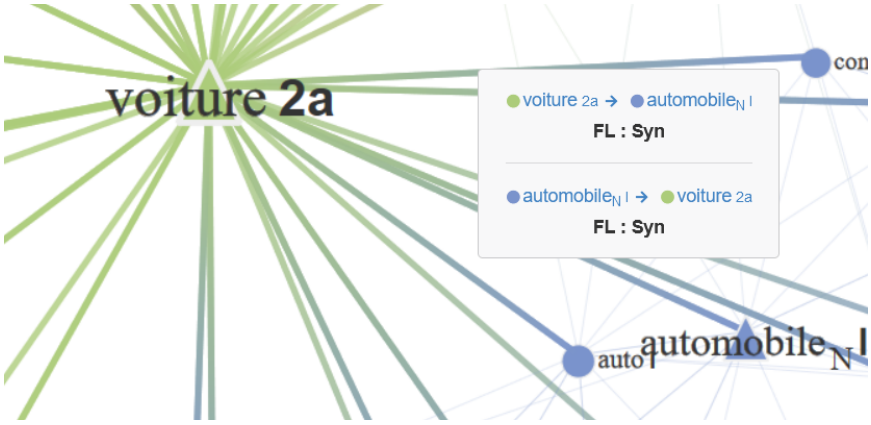


FIGURE 5 – Visualisation de la relation entre $voiture_{2a}$ et $automobile_{N.I}$

Si les fonctions lexicales standard sont au nombre d'une soixantaine, le RL-fr compte en tout plus de 600 fonctions lexicales distinctes instanciées dans 62 641 arcs (version 2.1).

3.1.4 Comparaison avec l'existant

Le RL-fr constitue une ressource lexicale d'un tout nouveau genre. En effet, les ressources les plus couramment utilisées en TAL appartiennent à la famille des réseaux lexicaux *ontologiques* comme *WordNet* [12] : dans ces réseaux, l'information est

11. On acceptera pour cette présentation l'approximation suivante. Au sujet de l'interconnection entre FL paradigmatiques et syntagmatiques, voir (Mel'čuk et Polguère, 2021).
12. `https://wordnet.princeton.edu/`

principalement structurée de manière *hiérarchique*, avec des relations caractérisant l'héritage, la composition... Cette organisation permet de structurer l'information en classes et en sous-classes. Au contraire, le RL-fr constitue un réseau *non-ontologique* structurant l'information de manière fondamentalement relationnelle, avec la grande variété de relations que nous avons évoquée précédemment (Polguère, 2014). De plus, contrairement aux relations représentées dans WordNet, celles du RL-fr lient bien souvent des unités lexicales de catégories grammaticales différentes.

3.2 Le projet BEL-RL-fr

Lancé en 2018 sous la responsabilité de Sandrine Ollinger, le projet BEL-RL-fr s'inscrit dans le cadre des travaux sur les Systèmes Lexicaux menés par Alain Polguère. Il a bénéficié de financements de la part du consortium CORLI [13], du CPER LCHN [14] et du projet IMPACT OLKi-LUE [15].

3.2.1 Objectifs, enjeux et applications

Le projet BEL-RL-fr (Base d'Exemples Lexicographiques du Réseau Lexical du français) a pour but d'exploiter et de valoriser la base de citations lexicographiques du RL-fr. Cette base contient l'ensemble des exemples lexicographiques associés à des unités du RL-fr (cf fig.4), principalement issus de trois sources :

— la base *Frantext* (cf 2.1)
— un corpus journalistique issu de l'*Est Républicain* [16]
— le corpus web *frWac* [17]

Le BEL-RL-fr est distribué en tant que corpus indépendant depuis 2019 sur la plateforme ORTOLANG [18]. Notons qu'il s'agit d'un corpus représentant un genre textuel particulier : les exemples ont en effet été soigneusement choisis par des lexicographes pour illustrer précisément les lexies du RL-fr et rendre compte de leur combinatoire. Ce sont, de plus, des données textuelles particulièrement propres, car l'orthographe et la typographie ont été systématiquement vérifiés. Enfin, chaque exemple du BEL-RL-fr illustre jusqu'à 12 unités lexicales différentes (voir fig.6) : le BEL-RL-fr peut donc être vu comme un *corpus de citations partiellement désambiguïsé* (selon l'inventaire de sens du RL-fr).

13. https://corli.huma-num.fr/
14. Contrat de plan Etat / Region Lorraine ; Langues, Connaissances et Humanités Numériques : http://lchn.fr/
15. Open Language and Knowledge for citizens (projet de l'initiative Lorraine Université d'Excellence : https://www.univ-lorraine.fr/lue/les-projets-impact/open-language-and-knowledge-for-citizens-olki/
16. https://www.ortolang.fr/market/corpora/est_republicain/v4
17. https://wacky.sslmit.unibo.it/doku.php?id=corpora
18. https://www.ortolang.fr/market/corpora/examples-ls-fr/V1

Dans sa distribution V2, le BEL-RL-fr compte 31 131 citations et 51 347 associations citation-lexie, avec 27 343 lexies représentées.

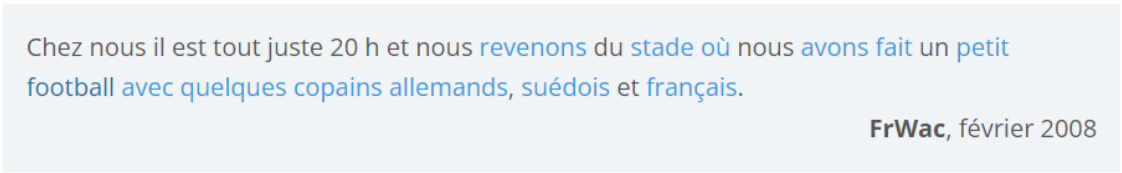


FIGURE 6 – Un exemple de citation du BEL-RL-fr associé à 12 unités lexicales

Le projet BEL-RL-fr englobe donc l'ensemble des travaux en lien avec le corpus du même nom, essentiellement centrés sur les perspectives suivantes :

— **son enrichissement** : automatiser l'identification de citations candidates parmi les sources, intégrer de nouveaux exemples
— **son annotation** : automatiser la désambiguïsation de segments textuels pas encore annotés parmi les citations
— **son exploration** : utilisation comme corpus d'apprentissage pour des tâches de désambiguïsation, étude de l'ambiguïté lexicale, de la polysémie, applications pédagogiques, amélioration du RL-fr...

3.2.2 Chronologie

- **2018**

— lancement du projet
— annotation manuelle de segments de citations en unités lexicales (AUBE et SOLEIL)

- **2019**

— distribution v1 du BEL-RL-fr sur ORTOLANG

- **2020**

— export TXM
— révision et validation d'exemples

- **2021**

— distribution v2 du BEL-RL-fr sur ORTOLANG
— expériences de désambiguïsation automatique à base de réseaux de neurones (Mathieu Constant, Sandrine Ollinger, Aman Sinha, Alain Polguère)

3.3 Mission du stage

La mission de mon stage a consisté à mettre en place plusieurs méthodes de désambiguïsation lexicale basiques (n'utilisant pas de réseaux de neurones) sur un corpus mixte (une partie issue du BEL-RL-fr, l'autre d'un corpus web) ciblant 25 couples d'unités lexicales issus du RL-fr. En d'autres termes, le problème est ramené à 25 tâches de classification binaire.

Cette mission est à mettre en regard avec les expériences de désambiguïsation automatique à base de réseaux de neurones récemment conduites à l'ATILF (Sandrine Ollinger, Mathieu Constant, Aman Sinha, Alain Polguère) sur un corpus de plus de 30 000 énoncés du BEL-RL-fr, ciblant les noms ainsi que les verbes. Dans les deux cas, il s'agit de se questionner sur l'impact de l'utilisation de ressources relevant d'une description fine de la langue (comme le RL-fr) plutôt que d'ontologies (comme WordNet) dans les tâches de désambiguïsation automatique ; une hypothèse étant que ce premier type de ressource devrait améliorer les résultats (Veronis, 2001).

L'approche adoptée dans mon stage peut être vue comme une simplification de l'approche par réseaux de neurones : si mes résultats ne peuvent prétendre être aussi représentatifs et complets, ils permettent tout d'abord une comparaison des performances des réseaux de neurones à celles de méthodes plus simples sur nos données. Par ailleurs, la petite taille de mon corpus permet bien davantage de consulter et d'interpréter les prédictions dans l'optique d'une analyse qualitative plus fine.

3.3.1 Description des données

Le corpus de départ a été constitué par mes maîtres de stage. Nous nous sommes intéressés à 25 lemmes, de catégories grammaticales différentes, et pour chacun desquels deux sens distincts ont été sélectionnés dans le RL-fr [19] :

- **adjectifs :** LONG
- **prépositions :** AVEC
- **adverbes :** AILLEURS, PRÈS
- **noms :** ANTENNE, BOUCHER, BOUCHON, ÉCLAIR, FAUTEUIL, LESSIVE, LIT, MARCHE, ROUGE, SOLEIL, VESTE
- **verbes :** ALLER, ALLONGER, CHANTER, COURIR, DORMIR, MANGER, MARCHER, NETTOYER, SORTIR, VOLER

On dispose donc de 25 lemmes et de 50 sens répartis dans ces 5 catégories. Ajoutons que nos 25 couples permettent d'illustrer plusieurs types de relations entre sens d'un même lemme (on parle de *relations de copolysémie*), comme par exemple :

- **métaphore** : relation analogique entre les deux lexies (ex : $rouge_{IV}$ - synonyme : *vin rouge* et $rouge_{I.1b}$ - générique : *couleur*)
- **métonymie** : contiguïté entre les deux sens (ex : $soleil_{I.a}$ - générique : *astre* et $soleil_{II.1}$ - synonyme : *beau temps*)
- **homonymie** : aucun lien entre les deux sens (ex : $éclair^1_{I.1}$ - synonyme : *foudre* et $éclair^2$ - générique : *pâtisserie*)

19. voir annexe en section 5 pour un inventaire complet des sens sélectionnés

Finalement, les données sur lesquelles j'ai travaillé se composent de :

— **3016 énoncés** (chacun associé à un unique sens de notre inventaire) dont :
  — 525 proviennent du BEL-RL-fr (9 à 25 énoncés par sens)
  — 2491 proviennent du frTenTen17 [20] (50 énoncés par sens, sauf $manger_{V.I.1a}$ qui en a 41)

  Tous les énoncés pour une lexie sont rassemblés dans un fichier texte, stocké dans un dossier correspondant à sa source (BEL-RL-fr ou frTenTen17).

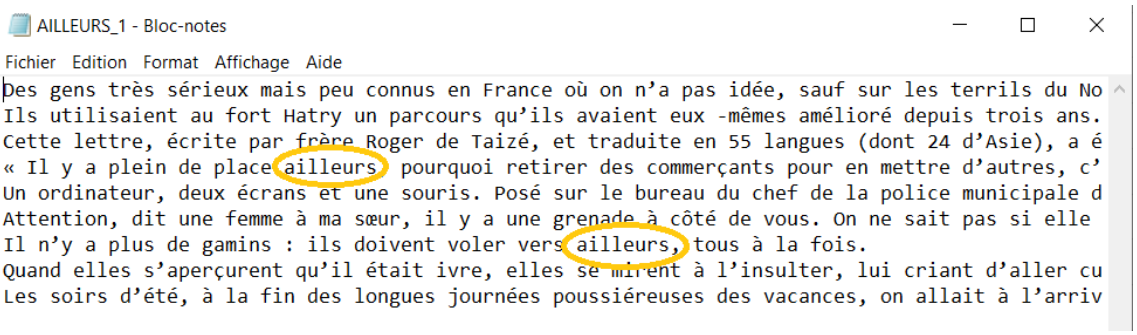


FIGURE 7 – Une partie du fichier contenant des énoncés pour la lexie $ailleurs_1$

— la liste de tous les **voisins lexicaux** de niveau 1 de nos 50 lexies, c'est-à-dire la liste des lexies du RL-fr directement liées à chacun de nos sens par une fonction lexicale (voir fig.8)

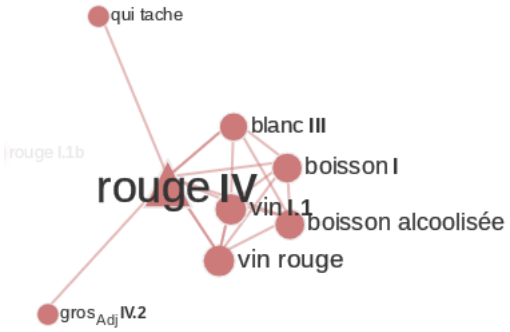


FIGURE 8 – Les voisins lexicaux de niveau 1 de $rouge_{IV}$

20. Le French Web Corpus (frTenTen) est un corpus de textes provenant du web et illustrant plusieurs variétés du français (France, Canada, Afrique). Au contraire des exemples du BEL-RL-fr, les exemples issus de ce corpus peuvent présenter de nombreuses imperfections : fautes d'orthographe, de grammaire, mauvaise typographie...

4 Travail réalisé pendant le stage

Dans un premier temps, je présente la manière dont je m'y suis prise pour extraire les données de mon corpus de travail et les intégrer à une structure de code me permettant de les manipuler et de les soumettre à des premiers traitements. Puis, je présente les méthodes de désambiguïsation abordées et la réflexion autour de leur implémentation. Enfin, je présente les résultats obtenus ainsi qu'une partie de l'analyse effectuée sur ces-derniers.

4.1 Prise en main des données

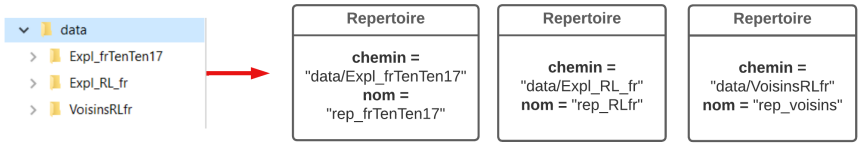
Matériel et logiciels utilisés : Tout au long de mon stage, j'ai travaillé avec un ordinateur portable sous Windows, fourni par l'ATILF. L'ensemble du code a été écrit en **Python 3.8**, dans **Visual Studio Code**. J'ai également rédigé plusieurs **Jupyter Notebook** pour documenter mes avancées et mes résultats. Enfin, mes maîtres de stage et moi avons utilisé **GitLab** pour stocker les données de travail, déposer progressivement les fichiers du projet, et discuter autour des questionnements rencontrés.

4.1.1 Structure du code

Les données décrites en 3.3.1 étant fournies sous forme de fichiers .txt déposés dans notre répertoire GitLab, j'ai dû commencer par définir une structure de données permettant d'extraire les données de ces fichiers, pour les stocker puis les manipuler. Alors que, dans une première approche, j'utilisais pour cela un Dictionnaire Python, j'ai vite rencontré les limites d'une telle manière de procéder : les fonctions commençaient à s'accumuler, il devenait difficile de se retrouver dans mon code et d'effectuer des modifications sans rendre l'ensemble du code déficient.

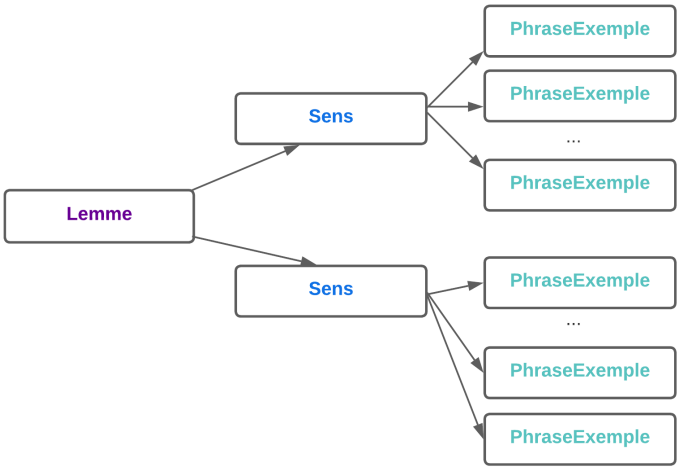
J'ai donc progressivement segmenté mon code en des fichiers distincts dédiés à des tâches précises (importer les données, leur associer des prédictions, afficher des résultats), puis j'ai codé des classes Python pour représenter les différents objets que je souhaitais manipuler. Je me suis pour cela documentée sur la Programmation Orientée Objet (si j'ai déjà eu l'occasion d'en faire en Java, c'était la première fois en Python). Un diagramme présentant l'ensemble des classes codées est disponible en annexe 5, nous évoquerons simplement ici les principales :

— un **Repertoire** correspond à un dossier de fichiers .txt pour un de nos sous-corpus de données (BEL-RL-fr, frTenTen17 ou les voisins du RL-fr).

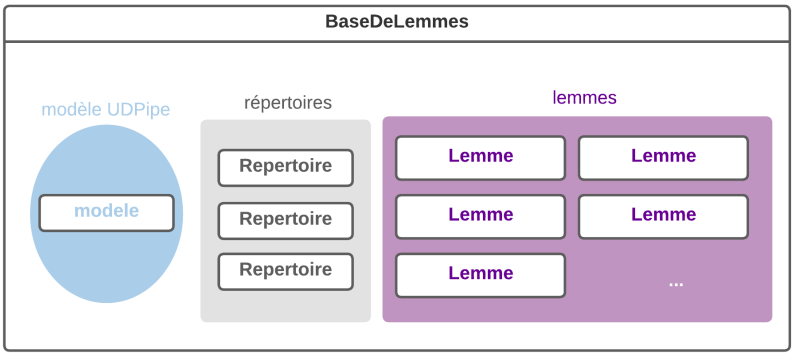


Cet objet permet d'aller explorer le dossier correspondant pour en extraire les données, et constituer au fur et à mesure des objets **Lemmes**, **Sens** et **PhraseExemple**.

— un **Lemme** est un objet qui a un *nom*, qui est associé à deux **Sens** dont chacun stocke des *données*, sous forme de liste de **PhraseExemple**.



— enfin, l'ensemble de ces données est stocké dans un unique objet **BaseDeLemmes (bdl)** sur lequel on va travailler.



Voici pour les classes principales. Il y a encore d'autres classes "de service" qui ne modélisent pas directement les données (Parametres, Predictions, Evaluation).

En tout, les fichiers de classes Python représentent environ **650 lignes de code**, auxquelles il faut encore ajouter **800 lignes** de code réparties dans des fichiers Python de fonctions dédiées à des tâches particulières :

- initialisation et pré-traitements sur les données
- méthode naive bayes
- méthode random forest
- méthode de régression logistique
- recherche d'entités nommées
- séparation des données en sous-corpus (apprentissage, dev, test)
- test des méthodes et affichage de graphiques

Mon code a été livré sur le GitLab du projet sous la forme d'un dossier **src** conrtenant l'ensemble des fichiers décrits ci-dessus. Un Jupyter Notebook intitulé **main** en racine de projet permet à l'utilisateur de tester le code et de lancer ses propres expériences de désambiguïsation ; le format de Notebook permet de fournir des explications précises pour la prise en main. D'autres Notebooks de documentation sont fournis : pour chaque méthode, ils résument son fonctionnement, son implémentation, ainsi que les résultats que j'ai pu obtenir sous forme de chiffres et de graphiques.

4.1.2 Pré-traitements sur les données

Après une première étape de chargement des énoncés dans la BaseDeLemmes **bdl**, l'initialisation n'est pas tout à fait terminée : je précise ici l'ensemble des pré-traitements appliqués aux données avant qu'elles soient utilisables pour nos expériences de désambiguïsation.

Chargement des voisins du RL-fr : Il reste un Repertoire dont il faut également charger les données : le répertoire contenant la liste des voisins du RL-fr de niveau 1 de nos lexies. Nous avons en effet vu en 3.3.1 que le RL-fr permettait d'accéder aux unités entretenant des relations privilégiées avec chacun des sens qui nous intéressent. Ces informations sont exploitables dès le chargement du Repertoire qui les contient. A l'issue de cette étape, chaque objet Sens de **bdl** se voit complété par un attribut "voisins".

```
from src import initialisation, separer_donnees, tester_methodes, Parametres, Evaluation
bdl = initialisation.initialiser_base()
rouge = bdl.get_lemme("rouge")
for sens in rouge.sens:
    print(f"Voisins du sens '{sens.nom}': {sens.voisins}\n")
```

```
Voisins du sens 'I.1b': ['couleur', 'sombre', 'foncé', 'vif', 'brillant', 'ardent', 'pâle', 'clair', 'vermillon', 'rubis', 'pou
rpre', 'nacarat', 'lie-de-vin', 'incarnat', 'grenat', 'écarlate', 'carmin', 'bordeaux', 'virer', 'passer']

Voisins du sens 'IV': ['qui tache', 'gros', 'vin rouge', 'vin', 'blanc', 'boisson', 'boisson alcoolisée']
```

FIGURE 9 – Consultation des voisins associés à ROUGE après initialisation

Analyse syntaxique (UDPipe) : L'analyse syntaxique des énoncés a été effectuée avec l'outil *UDPipe*, disponible comme librairie Python (**ufal.udpipe**), mais également utilisable en ligne[21]. Le choix d'UDPipe plutôt que d'autres modèles comme ceux que propose **spacy** pour le français repose notamment sur la manière dont on souhaite traiter les noms dénotant des êtres animés : dans le RL-fr, ces noms constituent des entrées distinctes (lorsque 2 formes existent) au masculin et au féminin. BOULANGER et BOULANGÈRE sont donc deux lemmes distincts du RL-fr, regroupant éventuellement chacun plusieurs lexies. Au contraire des modèles spacy qui regroupent les deux formes sous le lemme BOULANGER, UDPipe fait ce même choix.

UDPipe permet d'effectuer la segmentation en phrases et la tokénisation des énoncés (le découpage en *tokens*, c'est-à-dire en unités lexicales). Un arbre de dépendances syntaxiques est calculé pour chaque phrase. Enfin, chaque token est associé à un lemme, une catégorie grammaticale ainsi que des traits morpho-syntaxiques. Toutes ces données sont alors conservées comme attributs des PhraseExemple de **bdl**.

Id	Form	Lemma	UPosTag	XPosTag	Feats	Head	DepRel	Deps	Misc
# newdoc									
# newpar									
# sent_id = 1									
# text = La couleur dominante pour 1999 est le rouge assortie au rose avec quelques touches de jaune.									
1	La	le	DET	_	Definite=Def|Gender=Fem|Number=Sing|PronType=Art	2	det	_	TokenRange=0:2
2	couleur	couleur	NOUN	_	Gender=Fem|Number=Sing	8	nsubj	_	TokenRange=3:10
3	dominante	dominant	ADJ	_	Gender=Fem|Number=Sing	2	amod	_	TokenRange=11:20
4	pour	pour	ADP	_	_	5	case	_	TokenRange=21:25
5	1999	1999	NUM	_	_	2	nmod	_	TokenRange=26:30
6	est	être	AUX	_	Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin	8	cop	_	TokenRange=31:34
7	le	le	DET	_	Definite=Def|Gender=Masc|Number=Sing|PronType=Art	8	det	_	TokenRange=35:37
8	rouge	rouge	NOUN	_	Gender=Masc|Number=Sing	0	root	_	TokenRange=38:43
9	assortie	assortier	VERB	_	Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin	8	acl	_	TokenRange=44:52

FIGURE 10 – Analyse UDPipe d'un énoncé (depuis le service en ligne)

21. `http://lindat.mff.cuni.cz/services/udpipe/`

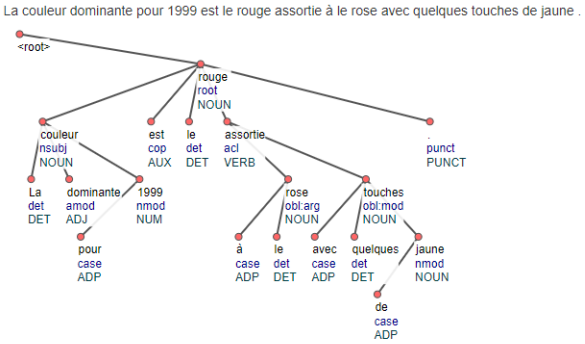


FIGURE 11 – Arbre en dépendances syntaxiques d'un énoncé

Recherche des occurrences-cibles : L'analyse UDPipe permet de rechercher, dans chaque énoncé, le token sur lequel va porter la tâche de désambiguïsation. Si UDPipe n'a commis aucune erreur, il s'agira donc du token lemmatisé comme le lemme-cible associé à notre énoncé (voir colonne "lemma" dans la fig.10 : on cherche le token lemmatisé en "rouge", ici c'est celui d'indice 8).

Cependant... UDPipe commet des erreurs. Il peut arriver que certains tokens soient mal lemmatisés : on ne trouve alors aucune occurrence-cible. J'ai donc écrit une fonction alternative, appelée uniquement dans ce cas précis : on choisit comme occurrence-cible le token dont la forme présente le plus long préfixe commun avec le lemme-cible.

> **Exemple:** Pour l'énoncé *"Un peu plus et l'héroïne aux **longues** jambes aurait sincèrement apprécié un match de football."*, UDPipe lemmatise "longues" en LONGUE (au lieu de la cible LONG). Cependant la stratégie alternative sélectionne bien "longues" comme ayant le plus long préfixe commun avec "long".

Recherche d'entités nommées : C'est en travaillant sur une représentation de données pour la méthode de **régression logistique** (cf 4.3.4) que j'ai eu l'idée de m'intéresser aux entités nommées [22]. Il peut en effet être intéressant de savoir si un énoncé fait référence à des lieux géographiques ou à des personnes pour désambiguïser le contexte autour de la cible. Cette recherche a donc été ajoutée aux pré-traitements. J'ai choisi d'utiliser **spacy**, avec le modèle *fr_core_news_lg* (modèle le plus performant pour cette tâche parmi ceux proposés par spacy).

On peut donc, pour chaque énoncé, rechercher les entités nommées de type ORG (organisations), PER (personnes), LOC (lieux) et MISC (autres entités nommées).

22. Une entité nommée est une expression linguistique référentielle, souvent associée aux noms propres et aux descriptions définies (Source : Wikipédia).

J'ai choisi de ne pas conserver cette dernière catégorie, peu informative pour notre tâche.

```
pres = bdl.get_lemme("près")
p = pres.sens[0].donnees[0]
print(f"{p.phrase}\n-> entités nommées: {p.en}")
```

```
Un train de fret composé de 24 wagons a déraillé hier matin vers 10 h lors d'une manœuvre à proximité de Berthelming près de Sa
rrebourg en Moselle. « Ce train transportait du gaz liquéfié du type propane, seuls deux wagons sont sortis des rails en positi
on debout », a indiqué hier la SNCF dans un communiqué.
-> entités nommées: {'LOC': ['Berthelming', 'Sarrebourg', 'Moselle'], 'ORG': ['SNCF']}
```

FIGURE 12 – Un exemple d'énoncé comportant des entités nommées

4.2 Méthodes de désambiguïsation

Durant mon stage, j'ai pu comparer en tout 6 méthodes de désambiguïsation : méthode **aléatoire**, méthode **des voisins**, méthode **des cooccurrents**, méthode **Naive Bayes**, méthode de **Régression Logistique** et méthode **Random Forest**. Dans cette partie, je présente chacune de ces méthodes ainsi que quelques éléments clés de leur implémentation. Les méthodes les plus basiques ont été conçues et implémentées "à la main", tandis que pour les deux dernières j'ai utilisé des librairies Python dédiées. En ce qui concerne les méthodes Naive Bayes, de Régression Logistique, et Random Forest, je me suis documentée à leur sujet, car ce sont des classifieurs de référence pour le TAL (Jurafsky et Martin, 2022).

4.2.1 Méthode aléatoire

Cette méthode (*baseline*) consiste simplement à tirer aléatoirement un sens parmi les deux possibles pour un énoncé donné. Le code repose sur une simple fonction de tirage faisant appel à la librairie *random*.

4.2.2 Méthodes des voisins et des cooccurrents

Dans la méthode des voisins, on consulte les voisins RL-fr de chacun des sens possibles pour l'énoncé considéré. Plus précisément, on compte le nombre de voisins du premier sens présents dans l'énoncé, puis du second, et on prédit le sens ayant produit le meilleur recouvrement. Dans la méthode des cooccurrents, on compare cette fois-ci le recouvrement de l'énoncé par deux ensembles de *cooccurrents* associés à chaque sens : ce sont les lemmes du corpus d'apprentissage qui apparaissent fréquemment dans le contexte de ce sens [23]. Le processus de prédiction des deux méthodes est illustré fig.13 :

23. Cette méthode s'appuie sur l'*hypothèse distributionnelle* selon laquelle des mots qui se trouvent dans des contextes d'apparition similaires tendent à avoir des sens similaires (Firth, 1957)

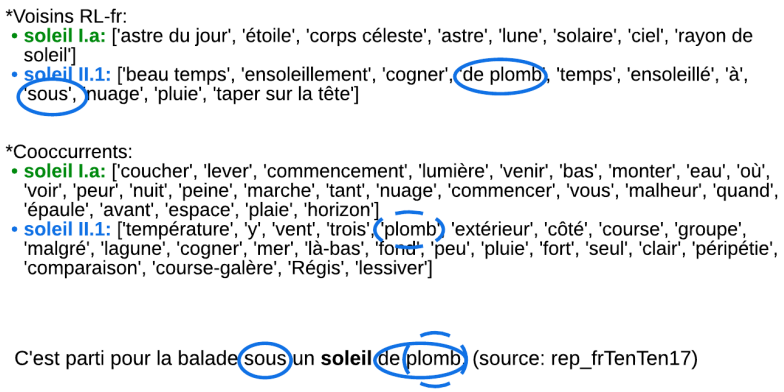


FIGURE 13 – Exemple de décompte des voisins RL-fr et des cooccurrents pour SOLEIL : les deux méthodes prédisent le sens $soleil_{I.1a}$ (2 voisins RL-fr et 1 cooccurrent)

En raison de leur proximité, j'ai écrit une fonction commune aux deux méthodes : cette fonction calcule le nombre d'apparitions des lemmes d'un ensemble donné dans une PhraseExemple (un énoncé). Cet ensemble sera donc respectivement un ensemble de voisins RL-fr ou un ensemble de cooccurrents.

> **Remarque:** Une égalité du nombre de voisins ou de cooccurrents des deux sens est tout à fait possible. Dans ce cas, un des deux sens est déterminé aléatoirement (retour à la méthode *baseline*).

L'extraction des cooccurrents se base quant à elle sur un calcul de *tf-idf* appliqué à l'ensemble des lemmes contenus dans le corpus d'apprentissage. Ce calcul prend à la fois en compte la fréquence d'apparition d'un lemme pour un sens, et le nombre de sens dans le contexte desquels il apparait (ce qui permet d'éliminer les termes très fréquents ou *stopwords* comme *et*, *le*, *à*...). Les cooccurrents d'un sens sont donc les termes qui apparaissent fréquemment dans le contexte de ce sens, tout en n'apparaissant pas ou peu dans les autres. On sélectionne ensuite pour chaque sens les n lemmes ayant le tf-idf le plus élevé pour ce sens (avec n le nombre de cooccurrents désirés).

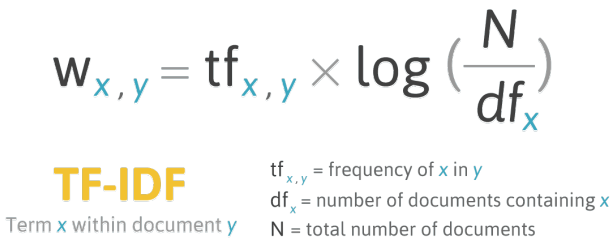


FIGURE 14 – tf-idf :mesure de la pertinence lexicale d'un terme dans un document

4.2.3 Naive Bayes

Il s'agit d'un algorithme de classification linéaire reposant sur le calcul de probabilités conditionnelles du type $P(élément|classe)$ (où un élément est un énoncé, et une classe un sens), à l'aide d'estimations réalisées sur un corpus d'apprentissage. C'est un *modèle génératif* ; il répond à la question "Connaissant les caractéristiques des énoncés produits par les deux sens du lemme L, lequel de ces sens est le plus susceptible d'avoir produit l'énoncé e ?"

Après m'être documentée sur cette méthode dans (Jurafsky et Martin, 2022), j'ai ré-implémenté l'algorithme moi-même en Python. Cet algorithme utilise une représentation des données basées sur le vocabulaire. En d'autres termes, un énoncé est représenté par l'ensemble des lemmes qu'il contient, indépendamment de leur nombre d'apparitions ou de leur position dans la phrase [24].

Il existe quelques variantes pour cet algorithme : on peut par exemple choisir de ne représenter un énoncé que par les lemmes de *classes ouvertes* qu'il contient : ce sont les lemmes des catégories grammaticales ADJ, ADV, INTJ, NOUN, PROPN et VERB [25]. Après avoir essayé cette méthode, j'ai finalement conservé une représentation avec l'ensemble des classes, qui me donnait de meilleurs résultats, bien qu'elle allonge le temps d'exécution des prédictions.

4.2.4 Régression Logistique

Au contraire de la méthode précédente, la régression logistique constitue un algorithme de classification *discriminant*. Etant donné une *représentation* des données du corpus d'apprentissage, l'algorithme cherche à déterminer les caractéristiques de cette représentation qui sont le plus discriminantes entre les différentes classes. Le modèle calcule ensuite les poids optimaux à accorder à chaque caractéristique de la représentation pour effectuer des prédictions.

Le choix d'une représentation des données pour la régression logistique a fait l'objet d'un travail important dans mon stage : je développe ce point dans la section 4.3.4, qui s'applique aussi à la représentation utilisée pour la méthode Random Forest. Une fois la représentation des données définies, j'ai utilisé la librairie Python *sklearn* [26] et plus particulièrement le module *LogisticRegression* pour entraîner et appliquer le modèle.

24. C'est cette forte hypothèse d'indépendance entre les caractéristiques de la représentation qui vaut à cette méthode le qualificatif de *naive*

25. Les classes ouvertes sont celles qui sont susceptibles d'évoluer dans le temps, par ajout de nouveaux mots (ex : *covidé*, *googliser*...). Les classes fermées comportent les mots grammaticaux (ex : les prépositions)et sont donc bien plus stables dans le temps.

26. Documentation disponible sur : `https://scikit-learn.org/stable/`

4.2.5 Random Forest

Le classifieur Random Forest utilise un ensemble d'*arbres de décision*[27] (d'où *forest*) dont il extrait une unique prédiction comme moyenne des prédictions individuelles de chaque arbre. Précisons encore que les arbres qui constituent la forêt sont des "variations aléatoires" les uns des autres (d'où *random*)[28].

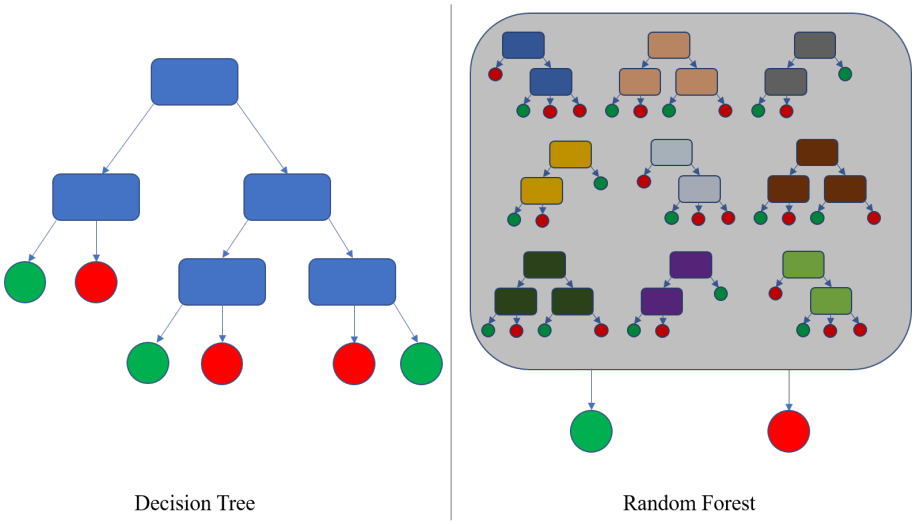


FIGURE 15 – Représentation schématique d'une Random Forest

Là encore, j'ai fait appel à la librairie *sklearn* (module *RandomForestClassifier*) pour instancier, entraîner et utiliser le modèle.

4.3 Évaluation des prédictions

Par souci de clarté, il m'a semblé préférable de présenter l'ensemble des méthodes de désambiguïsation abordées avant d'évoquer les résultats obtenus. Cependant, au cours de mon stage, les phases de documentation et d'implémentation des méthodes se sont effectuées en parallèle de l'extraction de premiers résultats. Ce sont justement les observations et analyses de ces résultats temporaires (sur les données de développement) qui m'ont permis, au fur et à mesure, d'affiner les méthodes et de chercher à exploiter de nouvelles caractéristiques des données.

Je présente premièrement le procédé de **cross-validation** employé pour évaluer mes méthodes, ainsi qu'une partie des résultats intermédiaires utiles dans le

27. Les arbres de décision sont des outils qui servent à répartir des données en groupes homogènes en un certain nombre de séparations successives, suivant des variables discriminantes.

28. Soit C le corpus d'apprentissage initial, et $F = \{f_1, \ldots, f_n\}$ un ensemble de features pour la représentation des données. Chaque arbre est associé à un sous-corpus aléatoire $c*$ de C, et à sous-set aléatoire de features $f*$ de F.

cheminement de mon travail. Enfin, je présente les résultats finaux obtenus pour l'ensemble des méthodes.

4.3.1 Division des données et cross-validation

A l'exception des méthodes aléatoires et des voisins, toutes ces méthodes requièrent une première phase d'apprentissage sur une partie des données, avant de pouvoir être appliquées sur les données restantes. On commence donc par isoler une première partie des données qui ne serviront qu'au moment de l'évaluation finale (sous-corpus de **test** ou *test set*), et qui ne seront jamais consultées pendant la phase de développement. Les données restantes sont à leur tour réparties en données d'**apprentissage** (*training set*) et de **développement** (abrégé en **dev**, *dev set* ou *validation set*). Ce dernier sous-corpus sert à ajuster les paramètres de notre modèle : au contraire des données de test qui sont "aveugles", on peut consulter les résultats et les prédictions obtenus sur les données de dev. Leur analyse permet de tester différents paramétrages des méthodes pour améliorer au mieux leurs performances.

Ainsi, la *cross-validation* consiste à redistribuer aléatoirement les données d'apprentissage et de dev sur plusieurs exécutions (le test étant toujours isolé) pour attribuer un score à un modèle testé : il s'agit de la moyenne des scores obtenus à chaque exécution. Ce procédé permet de faire plusieurs évaluations d'un modèle sur un seul ensemble de données, et aboutit à une estimation plus robuste qu'un calcul sur une seule exécution. Une fois le paramétrage du modèle terminé, on ajoute les dernières données de dev au sous-corpus d'apprentissage, et on lance enfin le modèle sur les données de test.

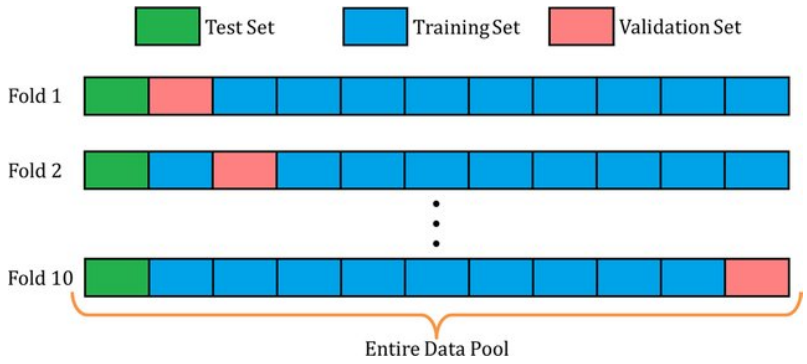


FIGURE 16 – Principe de la méthode de cross-validation : illustration sur 10 exécutions

J'ai donc isolé **10% de mes données** (soit 302 énoncés) pour l'évaluation finale de l'ensemble des méthodes : ces données n'ont pas été consultées pendant les phases

de développement. A chaque exécution (100 ou 1000 en tout selon les expériences), 10% ont servi de corpus de **dev**, et les 80% restants à l'**apprentissage**.

4.3.2 Consultation des prédictions

La prédiction de sens sur la base **bdl** se fait en trois temps :
— répartition des données entre les différents corpus app/dev/test
— définition d'un objet ParametresTest spécifiant la méthode à appeler ainsi que ses paramètres spécifiques (par exemple, pour la méthode des cooccurrents, le nombre de cooccurrents que l'on souhaite calculer)
— appel à la méthode *BaseDeLemmes.predire_sens*, avec en arguments les paramètres de test ainsi qu'un booléen *dev* indiquant si on souhaite prédire les données du dev/du test

La méthode *BaseDeLemmes.predire_sens* associe alors à chaque PhraseExemple concernée par l'appel de la méthode une Prediction, directement consultable par l'utilisateur.

```
# Exemple sur le lemme ROUGE
for p in rouge.fold_dev:
    print(p.phrase)
    print(f"-> prédiction: {p.sens_predit.prediction} (correct: {p.sens.nom})\n")
```

```
Le mélange du rouge et du blanc saura équilibrer votre bouquet de fleurs par un profond message.
-> prédiction: IV (correct: I.1b)

La pierre fine révèle plusieurs variétés de couleurs allant du rouge au rose.
-> prédiction: I.1b (correct: I.1b)

Sur le plan plastique, le rouge est très difficile à mélanger avec les autres couleurs .
-> prédiction: I.1b (correct: I.1b)
```

FIGURE 17 – Ex : quelques prédictions sur ROUGE (Naive Bayes)

4.3.3 Evaluation méthode par méthode

Scores - J'ai mesuré la performance des méthodes selon plusieurs scores : la **précision (p)**, le **rappel (r)** et le **f-score (f)** :

$$p = 100 * nb\ de\ prédictions\ correctes\ /\ nb\ de\ prédictions\ effectuées$$

$$r = 100 * nb\ de\ prédictions\ correctes\ /\ nb\ de\ prédictions\ attendues$$

$$f = (2 * p * r)/(p + r)$$

Dans le cas général, le nombre de prédictions effectuées est égal au nombre total de prédictions attendues ; ces trois scores sont donc équivalents, et on parlera simplement de la précision. Nous verrons toutefois dans le paragraphe suivant des cas

où le rappel et le fscore entrent en jeu. Ces scores peuvent être consultés pour un lemme en particulier, ou bien sur l'ensemble de la base.

```
from src.Evaluation import *
# Evaluation du lemme ROUGE -> Il faut passer en paramètres le dev que l'on veut évaluer
print(Evaluation.evaluer_resultats(rouge.fold_dev))
```

```
total à prédire: 12
--prédit: 12
----correct: 9
* précision: 75.0 %
* rappel: 75.0 %
* f-score: 75.0 %
```

FIGURE 18 – Ex : évaluation des prédictions sur ROUGE (Naive Bayes)

Evaluation de la méthode des voisins - Pour chaque méthode implémentée, j'ai évalué les scores présentés ci-dessus pour un nombre important d'exécutions (sur un corpus de dev, suivant la méthode de cross-validation). Pour pouvoir mieux comprendre la manière dont se comportent les méthodes, j'ai écrit une fonction qui me permet de visualiser les performances **lemme par lemme** et score par score. Il serait long et redondant de décrire l'évaluation de chacune des méthodes (d'autant que les résultats finaux sont présentés en section 4.3.5) ; je reviens ici uniquement sur la réflexion menée autour de l'évaluation de la méthode des voisins, ce qui permet d'illustrer la manière dont j'ai procédé méthode par méthode.

Mes premiers résultats concernant les voisins ont montré une précision moyenne proche de **63%** sur l'ensemble de la base, avec de fortes disparités entre les lemmes et même entre les énoncés du BEL-RL-fr et du frTenTen :

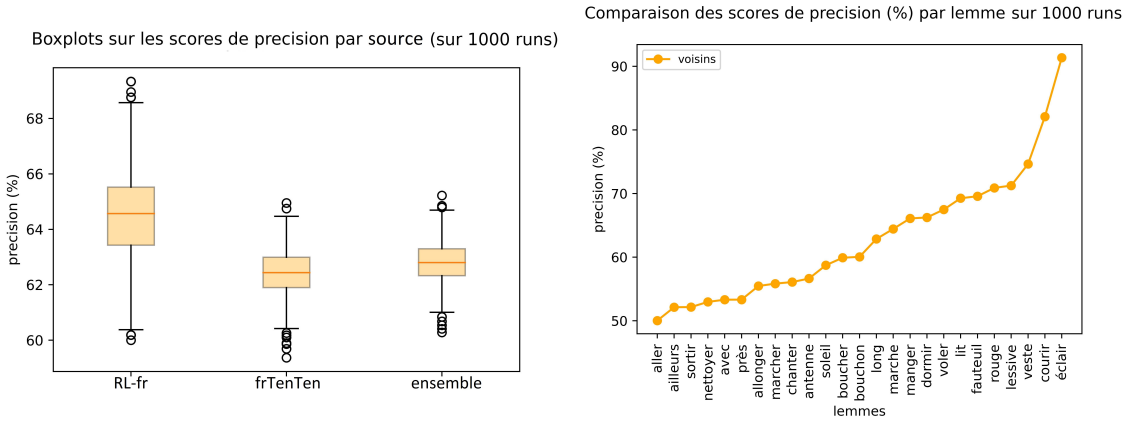


FIGURE 19 – Méthode des voisins : précision par source (à gauche) et par lemme (à droite)

Ces premiers résultats permettent déjà de soulever des pistes de réflexion :

— Il pourrait exister un biais de sur-représentation des voisins lexicaux issus du RL-fr dans les énoncés eux-mêmes issus du RL-fr, ce qui augmente la précision pour cette source
— "aller" n'est pas mieux prédit qu'en baseline aléatoire, tandis qu'"éclair" est proche de 100% de précision : il serait intéressant de comparer leurs voisins

Un examen rapide des voisins de chaque lexie révèle :
— des écarts considérables du nombre de voisins entre certains sens d'un même lemme (ex : $manger_{I.1.a}$ a 98 voisins, contre 3 pour $manger_{VI.1}$) : rappelons que le RL-fr est une ressource encore en construction, dans laquelle chaque lexie n'est pas également décrite. Cependant cela peut largement orienter le décompte des voisins vers le sens en possédant le plus ;
— que certains voisins ont une forme qui ne leur permet pas toujours d'être identifiés par les fonctions écrites (combinaison d'identification de lemme / recherche d'expression régulières avec la librairie *re*) : "mettre_N derrière la cravate" (où N est remplacé par un nom) est une construction qui ne sera jamais repérée ;
— que certains voisins risquent d'être assez peu fréquents en contexte : "bergeronnette" pour $chanter_{I.2}$, "se faire péter la sous-ventrière" pour $manger_{I.1.a}$...

Il est intéressant de noter que dans ce dernier cas (celui où les voisins n'apparaissent pas en contexte), on risque de se retrouver avec une égalité de 0 voisins pour chaque sens dans nos énoncés : on recourt alors à l'aléatoire pour effectuer une prédiction. Mais dans combien de cas est-ce que cette situation se produit ? Pour quantifier la part de l'aléatoire dans les résultats (qui sont d'ailleurs la seule source de variation des prédictions possible, la méthode étant sans cela déterministe), j'ai écrit une variante de la méthode en mode **non aléatoire**. Avec cette variante, en cas d'égalité du nombre de voisins des deux sens dans un énoncé, on ne prédit rien. De cette manière, on distingue dans l'évaluation les énoncés mal prédits des énoncés non-prédits : le rappel et le f-score ne sont plus équivalents à la précision.

En version non aléatoire, j'obtiens cette fois-ci sur l'ensemble de la base une précision de 81.54%, un rappel de 33.26% et un **f-score** de **47.25%**.
Cette variante révèle que seuls 41% des énoncés se voient assigner une prédiction en version sans aléatoire : c'est pour cette raison que le rappel est si faible, et que le f-score descend en dessous de 50% [29]. Une part d'aléatoire de 59% semble très élevée : il faudrait réfléchir à une manière de la faire baisser, d'autant que la précision de la méthode semble en réalité assez bonne (81.54%).

29. La moyenne harmonique, au contraire de la moyenne arithmétique, valorise les faibles valeurs.

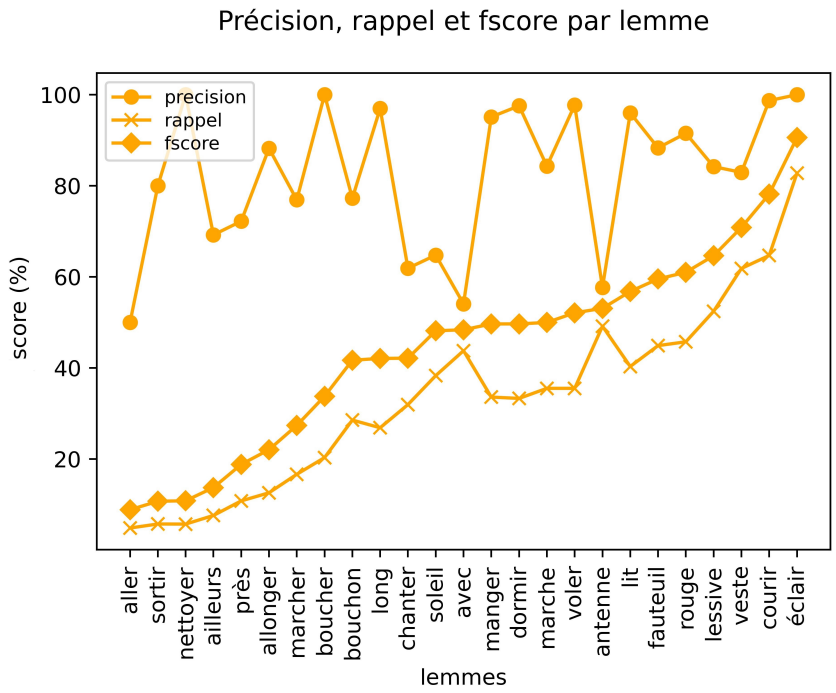


FIGURE 20 – Méthodes des voisins : scores par lemme (sans aléatoire)

En regardant de plus près les énoncés non-prédits, j'ai constaté que dans **93%** des cas, il y avait eu égalité du nombre de voisins à **0 pour chaque sens** (égalité à 1 voisins par sens dans 6% des cas, les échecs restants ne dépassent pas une égalité à 2) : c'est donc bien que les voisins n'apparaissent *pas assez*.
On retient donc :

- l'utilisation de voisins lexicaux issus d'une description fine de la langue permet des prédictions relativement précises
- cependant, en l'état, la méthode des voisins n'est pas adaptée à des prédictions massives : la majorité des lemmes obtiennent un score de rappel inférieur à 50% (cf fig.20)
- on aurait intérêt à dégager des lemmes supplémentaires apparaissant plus fréquemment dans le contexte de nos sens

C'est ainsi que j'ai pu, par la suite, implémenter la méthode des cooccurrents qui permet de tester ce dernier point ; pour cette méthode encore, j'ai implémenté une variante sans aléatoire pour comprendre les résultats de la méthode "brute". Cette fois-ci, en moyenne [30] : 66% des énoncés sont prédits, avec une précision de 90.47%, un rappel de 59.50% et un **f-score** de **71.76%**.

30. Les scores diffèrent toujours d'un run à l'autre en raison du calcul aléatoire du corpus d'apprentissage, dont dépendent directement les cooccurrents

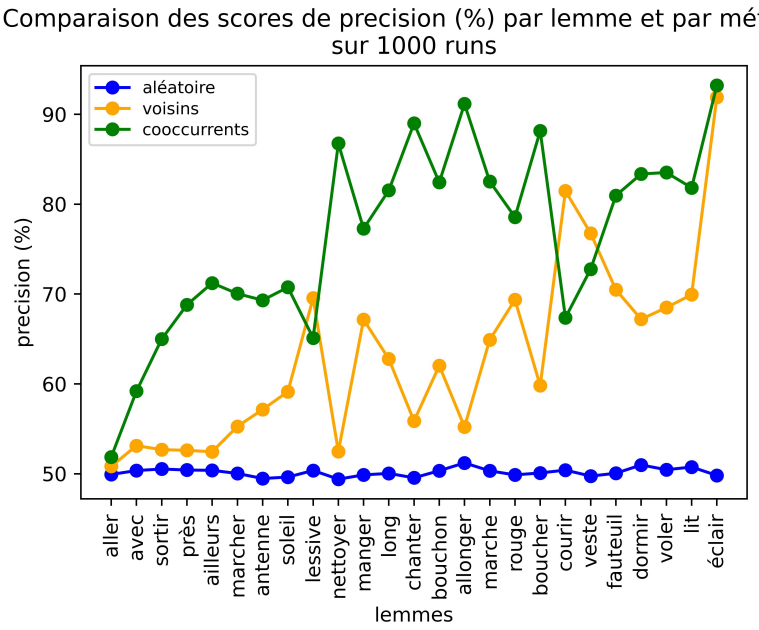


FIGURE 21 – Complémentarité des méthodes des voisins et des cooccurrents

Une comparaison de ces deux méthodes (ainsi que de la baseline aléatoire) permet de voir qu'il serait intéressant de les combiner, ce qui sera fait dans la méthode de la régression logistique.

4.3.4 Choix d'une représentation des données

Pour les méthodes suivantes, j'ai eu besoin de m'intéresser à la notion de représentation des données. Nous avons dit que les classifieurs automatiques utilisaient un corpus d'apprentissage pour apprendre un modèle, avant de pouvoir l'appliquer à des données de test. En réalité, le modèle ne s'entraîne pas sur les données d'apprentissage brutes, mais sur une *représentation* de ces données. Un énoncé est vu comme un *vecteur* à n dimensions, dont la i-ème coordonnée correspond à une certaine caractéristique (ou *feature*) de la représentation choisie.

Exemple: La méthode Naive Bayes représente un énoncé par l'ensemble des lemmes qu'il contient. Si on note V le vocabulaire appris par le modèle lors de l'apprentissage, alors tout énoncé peut être vu comme un vecteur de taille $|V|$, dont la i-ème coordonnée vaut 1 si il contient le i-ème mot de V, et 0 sinon.

V = vocabulaire appris
F = features (caractéristiques de la représentation)

v_i	arbre	avoir	bien	ce	...	nuit	pouvoir	sable	zèbre
f_i	0	1	1	1	...	1	0	0	0

Cette nuit a été bien longue. → **(0 1 1 1 … 1 0 0 0)**

FIGURE 22 – Représentation d'un énoncé par un vecteur (Naive Bayes)

J'ai donc été amenée à réfléchir à une manière de représenter les données pour la méthode de la régression logistique (et par la suite, random forest). L'objectif, à ce point du stage, était de pouvoir intégrer dans la régression une information la plus complète possible. En effet, les méthodes précédentes ont toutes utilisé des caractéristiques spécifiques des lexies à désambiguïser : leurs **voisins lexicaux** de niveau 1 (méthode des voisins), leurs **cooccurrents** (méthode des cooccurrents), et le **vocabulaire** de leur contexte d'apparition (Naive Bayes). Notons que si on conserve aussi la fréquence d'apparition des mots du vocabulaire (au contraire de ce que fait Naive Bayes), la méthode de la régression peut reconstituer les cooccurrents. En notant F l'ensemble de features à représenter, on peut donc esquisser informellement une première approche :

$$F = \{voisins\ RLfr, vocabulaire\}$$

Nous avons ensuite intégré à F de nouvelles caractéristiques encore non exploitées comme le **voisinage syntaxique**[31] des lexies, la présence d'**entités nommées** (cf 4.1.2) ou encore les **catégories grammaticales** des mots proches de la cible. En effectuant un classement des lemmes par précision moyenne des prédictions, j'ai en effet été surprise de constater que le lemme PRÈS arrivait avant-dernier, alors que ses sens me paraissaient instinctivement très distincts : $près_{I.1}$ s'accompagne d'une localisation (*ex : près de Paris*), tandis que $près_{II.1}$ signifie "environ" (*ex : près de 100 000 habitants*). Il m'a donc semblé intéressant de pouvoir quantifier le nombre

31. L'ensemble des lemmes situés juste au-dessus et juste au-dessous de la cible dans l'arbre de dépendances syntaxiques calculé par UDPipe (voir un exemple d'arbre en fig.11).

d'entités nommées de type LOC (localisation) dans nos énoncés. Les nombres tels que "100 000" ne peuvent pas être identifiés par des entités nommées, en revanche ils correspondent à la catégorie grammaticale NUM. On a donc :

$$F = \{voisins\ RLfr, vocabulaire, voisins\ syntaxiques, entités\ nommées,$$
$$catégories\ grammaticales\}$$

Cependant, il est clair que dénombrer les catégories grammaticales de tout un énoncé est peu informatif : c'est ici le contexte local qui nous intéresse, c'est-à-dire les mots proches de l'occurrence-cible [32]. Encore reste-t-il à définir ce qu'on entend par "proche" : uniquement le mot suivant ? Une fenêtre centrée sur l'occurrence-cible ? Faut-il distinguer une apparition juste à gauche de la cible d'une apparition juste à droite ? Par ailleurs, le vocabulaire peut toujours être extrait de différentes manières (uniquement les classes ouvertes, ou aussi les classes fermées), tout comme le voisinage syntaxique...

Alors que mes différents essais commençaient à devenir très confus, mes maîtres de stage m'ont aidée à établir un protocole rigoureux et progressif pour constituer F. Il est en effet important de ne changer qu'un critère d'une expérience à l'autre, pour pouvoir s'assurer de la meilleure configuration [33]. J'ai alors mené plusieurs expériences testant un à un les différents critères de F, ainsi que différents contextes pour ces critères (contexte global : toute la phrase, contexte local : proche de l'occurrence) et différentes manières de les calculer. Au fur et à mesure de ces expériences, nous avons pu vérifier l'apport de chaque critère et combiner le contexte global au contexte local ayant fourni les meilleurs résultats.

32. Par exemple, *éclair*[2] est souvent suivi d'une préposition puis d'un déterminant, comme dans *éclair au chocolat* = ÉCLAIR+À+LE +CHOCOLAT. Le décompte local des catégories grammaticales est donc pertinent. En revanche, le nombre total de prépositions et de déterminants de l'énoncé contenant *éclair au chocolat* dépend de la taille de cet énoncé, et ne nous dit rien sur l'emploi de *éclair*.

33. Sur l'étude systématique des critères utilisés pour la désambiguïsation, voir (Audibert, 2004) et (Audibert, 2007)

	nb de run	précision moyenne	variance	durée (min)
Expé 1	1000	80.37%	5.04	~30
Expé 2	1000	81.01%	4.25	~50
Expé 3	1000	84.03%	4.05	~30
Expé 2a	100	81.73%	5.24	~5
Expé 2b	100	79.87%	5.24	~5
Expé 2c	100	78.70%	3.15	~5
Expé 2d	100	81.04%	4.12	~5

FIGURE 23 – Extrait du tableau suivant la progression des expériences

Finalement, à l'issue de ces expériences :
- l'ensemble des critères retenus dans F sont pertinents
- on définit un **contexte global** C_t qui correspond à l'ensemble de l'énoncé
- on définit deux **contextes locaux** : un **contexte gauche** C_g (la cible + les deux lemmes à sa gauche) et un **contexte droit** C_d (la cible + les deux lemmes à sa droite)

critère	procédé d'évaluation du critère	C_t	C_g	C_d
voisins RL-fr	décompte des voisins de chaque sens dans C			
mots du vocabulaire	fréquence d'apparition dans C			
voisins syntaxiques	décompte des voisins de chaque sens dans C			
entités nommées	décompte des EN de chaque catégorie dans C			
catégories grammaticales	décompte des CG par catégorie dans C			

FIGURE 24 – Critères retenus pour la représentation finale

Par exemple, ce tableau nous dit qu'on compte le nombre de lemmes par catégories grammaticales dans C_g (ex : 1 NOUN, 1 ADP, 1 DET) et dans C_d (ex : 2 NOUN, 1 DET), mais pas dans C_t. Pour les trois premiers critères du tableau, la combinaison des contextes locaux *et* du contexte global permet de capter une information large tout en accentuant l'importance du contexte local.

> **Exemple:** Supposons que $V(rouge)$ contienne les mots *vin* et *boire*. Dans l'énoncé *"J'ai bu un excellent vin* ***rouge****."*, *boire* est repéré dans C_t, tandis que *vin* est repéré dans C_t *et* dans C_g, ce qui accentue son importance.

Une fois la représentation des données définie, j'ai pu écrire la fonction permettant de transformer nos énoncés en vecteurs : cette fonction génère des fichiers .csv qui sont lus par le modèle comme données d'entrée (avec la librairie *pandas*).

	nb_voisins_0_g	nb_voisins_0_d	nb_voisins_0_t	nb_voisins_1_g	nb_voisins_1_d	nb_voisins_1_t	nb_voisins_synt_0_g	nb_voisins_synt_0_d
0	0.0	0.333333	0.250000	0.0	0.0	0.000000	0.333333	0.333333
1	0.0	0.000000	0.055556	0.0	0.0	0.055556	0.666667	0.666667
2	0.0	0.000000	0.000000	0.0	0.0	0.047619	0.333333	0.000000
3	0.0	0.000000	0.111111	0.0	0.0	0.000000	0.500000	0.666667
4	0.0	0.000000	0.062500	0.0	0.0	0.000000	0.666667	0.000000

nb_voisins_synt_0_t	nb_voisins_synt_1_g	...	acceptable_g	acceptable_d	acceptable_t	2008_g	2008_d	2008_t	plaisir_g	plaisir_d	plaisir_t	id_sens
0.500000	0.333333	...	0	0	0	0	0	0	0	0	0	0
0.555556	0.666667	...	0	0	0	0	0	0	0	0	0	0
0.142857	0.666667	...	0	0	0	0	0	0	0	0	0	1
0.555556	0.500000	...	0	0	0	0	0	0	0	0	0	0
0.562500	0.333333	...	0	0	0	0	0	0	0	0	0	0

FIGURE 25 – Aperçu de quelques énoncés représentés comme vecteurs (chaque ligne correspond à un énoncé)

C'est donc avec des données de cette forme que les méthodes de Régression Logistique et Random Forest ont entraîné leurs modèles.

4.3.5 Résultats

Résultats obtenus pendant le développement - Au cours de la phase de développement, chaque méthode a montré une performance meilleure que la précédente, à l'exception de la méthode Random Forest, légèrement moins précise en moyenne que la Régression Logistique. La figure 26 illustre cette progression sur une expérience menée en phase de développement : on affiche les boxplots de la répartition des scores de précision par méthode, sur 1000 exécutions. On s'attend donc à retrouver cette progression dans les résultats sur les données de test.

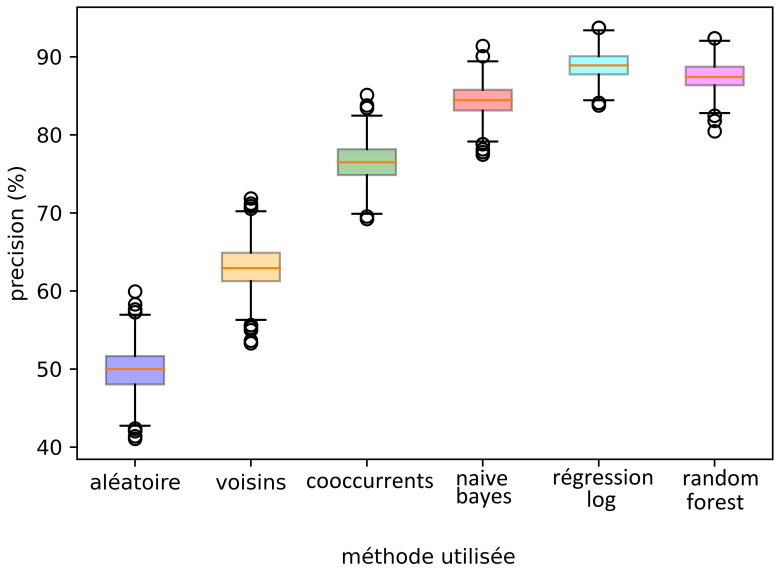


FIGURE 26 – Comparaison des méthodes sur une expérience de 1000 exécutions (boxplots de la répartition des scores de précision)

En ce qui concerne les résultats lemme par lemme (cf fig.27), on constate des disparités importantes dans la précision des prédictions. Le lemme le plus difficile à prédire en phase de développement est ALLER, tandis qu'ÉCLAIR présente les meilleures performances.

De manière générale, toujours pendant le développement :

— les couples de sens en relation d'**homonymie** (pas de lien de sens) obtiennent des résultats parmi les meilleurs : ALLONGER est $7^{ème}$, VESTE $4^{ème}$, VOLER $3^{ème}$ et ÉCLAIR 1^{er}. Du fait qu'ils présentent des contextes d'apparition bien distincts, et qu'ils ne partagent pas de composantes sémantiques, ces types de couples sont en effet plus simples à désambiguïser.

— les couples de sens en relation de **métonymie** (SOLEIL, LESSIVE, ANTENNE) présentent des résultats très proches, plutôt en fin de classement ($17^{ème}$, $19^{ème}$ et $20^{ème}$. Les couples de sens liés par une relation **métaphorique** (les plus nombreux dans nos données) occupent globalement le milieu du classement (PRÈS, NETTOYER, ROUGE, FAUTEUIL, DORMIR...).

— Il n'est pas spécialement étonnant de retrouver ALLER en dernière position : des imperfections ont été relevées quant à sa description dans le RL-fr, ce qui explique qu'il ait été inclus par mes maîtres de stage dans mes données de travail. Par ailleurs, il reste assez difficile à désambiguïser, même pour l'humain.

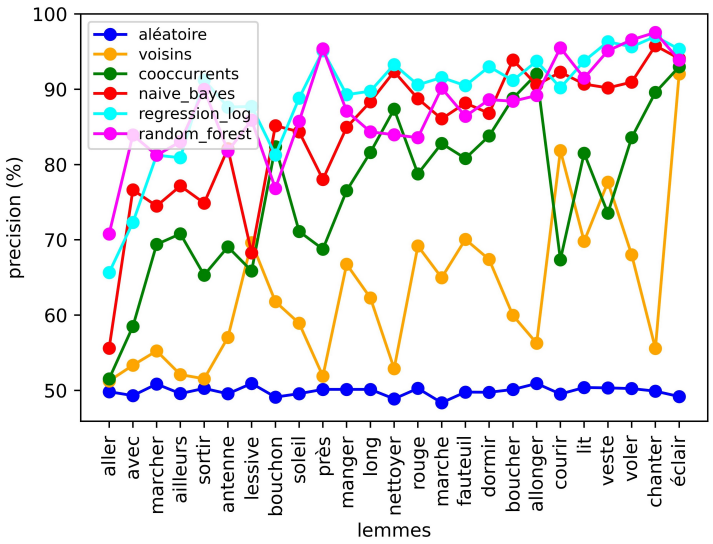


FIGURE 27 – Classement des lemmes, du moins bien au meilleur prédit (en moyenne) sur une expérience de 1000 exécutions

Résultats finaux - Finalement, à l'issue de la phase de développement, on lance les méthodes sur les données de test précédemment isolées. Le corpus de test final comprend **302 énoncés** (49 issus du BEL-RL-fr et 253 du frTenTen17) ; le reste des données est utilisé pour l'apprentissage, soit un corpus de **2714 énoncés** (476 issus du BEL-RL-fr et 2238 du frTenTen17). La figure 28 reprend les f-scores par méthode sur l'ensemble des catégories grammaticales représentées dans le corpus, donc sur les 25 lemmes (scores moyens sur 100 exécutions), tandis que la figure 29 présente le détail de ces résultats par lemme [34].

méthode	paramètres	toutes catégories (25)
aléatoire	/	**49,74**
voisins	avec aléatoire	61,25
voisins	sans aléatoire	44,71
cooccurrents	avec aléatoire, nb_cooc = 25	76,56
cooccurrents	sans aléatoire, nb_cooc = 25	72,08
naive_bayes	/	83,44
regression log	Set 230	88,74
random forest	Set 230	87,09

FIGURE 28 – F-scores moyens (%) par méthode sur 100 exécutions

34. Sur la figure 29 on ne représente pas les variantes sans aléatoire des méthodes des voisins et des cooccurrents : la précision et le f-score sont équivalents.

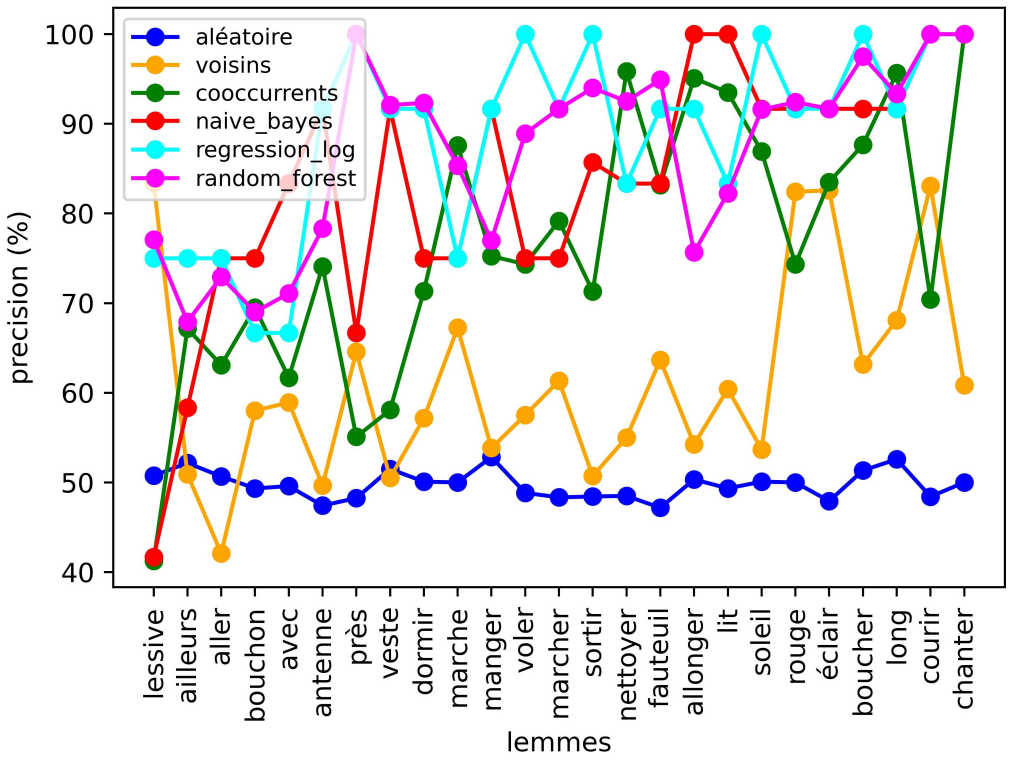


FIGURE 29 – Classement final des lemmes par précision moyenne croissante sur l'ensemble des méthodes (100 exécutions)

On constate que la position relative des méthodes les unes par rapport aux autres concernant la précision et le f-score est conservée. Le classement des lemmes par précision moyenne croissante est quant à lui modifié : ALLER n'est plus dernier mais $23^{ème}$, et ÉCLAIR n'est plus 1^{er} mais $5^{ème}$. Les couples d'homonymes ont été dispersés dans le classement (aux positions 5, 9, 14 et 18), tout comme les couples en relation métonymique (aux positions 7, 20 et 25). Cela laisse penser que notre corpus de test compte trop peu de données pour tirer des conclusions robustes quant à l'influence de la nature de la relation de copolysémie des couples sur la tâche de désambiguïsation.

Il apparaît aussi que les méthodes employées se compensent alternativement : aucune ne surpasse toutes les autres sur l'ensemble des lemmes ; il serait donc intéressant d'envisager par la suite une approche combinée.

Comparaisons - Il est intéressant d'effectuer une première comparaison de ces résultats avec ceux obtenus par mes maîtres de stage et Aman Sinha dans le cadre des expériences de désambiguïsation évoquées en section 3.3. Pour simplifier la lecture, nous désignerons dans ce paragraphe par ERN cet ensemble d'Expériences de désambiguïsation automatique à base de Réseaux de Neurones, et par S les expé-

riences menées pendant mon Stage. Rappelons que mon travail peut être vu comme une simplification de cette précédente expérience, dans la mesure où ERN :

- utilise des réseaux de neurones (S : des méthodes basiques)
- porte sur un corpus de plus de **8000 énoncés pour les verbes** et **19 000 énoncés pour les noms** (S : 1221 et 1317 énoncés, respectivement pour les verbes et les noms)
- utilise l'intégralité des données du BEL-RL-fr (S : seulement 525 énoncés)
- intègre des informations quant au poids sémantique associé [35] aux liens entre lexies du RL-fr (non pris en compte dans S)

Surtout, dans ERN, la tâche de désambiguïsation porte aussi bien sur des lemmes n'ayant que 2 sens que sur des lemmes en comptant des dizaines, là où j'ai désambiguïsé dans S exclusivement des couples de sens. Une comparaison directe est donc impossible : je me limite à comparer les résultats ERN sur les lemmes portant 2 sens aux résultats de S, calculés plus particulièrement sur les verbes et sur les noms (détail fig.30).

méthode	paramètres	toutes catégories (25)	noms (11)	Verbes (10)
aléatoire	/	**49,74**	49,53	49,64
voisins	avec aléatoire	61,25	**64,98**	57,58
voisins	sans aléatoire	44,71	**54,5**	28,47
cooccurrents	avec aléatoire, nb_cooc = 25	76,56	76,32	**79,58**
cooccurrents	sans aléatoire, nb_cooc = 25	72,08	72,02	**73,31**
naive_bayes	/	83,44	84,09	**86,07**
regression log	Set 230	88,74	87,12	**92,5**
random forest	Set 230	87,09	86,52	**88,5**

FIGURE 30 – F-scores moyens (%) par méthode et par POS sur 100 exécutions

Dans S, on observe par rapport aux autres catégories grammaticales un meilleur f-score sur les noms pour la méthode des voisins, et sur les verbes pour l'ensemble des méthodes suivantes. Au contraire, ce sont les noms qui sont les mieux prédits par ERN : pour les lemmes comptant 2 sens (3546 noms et 920 verbes), ERN obtient une précision maximale de **53.97% pour les noms**, et **22.58% pour les verbes** [36]. Ces faibles précisions peuvent s'expliquer par le fait qu'au contraire des couples de lexies S, ceux d'ERN peuvent englober des couples encore peu décrits dans le RL-fr et associés à peu voire pas d'exemples lexicographiques (puisqu'ERN utilise toutes les données du BEL-RL-fr).

35. Ce poids sémantique peut valoir 0,1 ou 2. Il représente l'importance du contenu sémantique partagé par les 2 lexies (0 étant le minimum et 2 le maximum).
36. Résultats communiqués par mes maîtres de stage, non publiés.

À titre de comparaison avec l'état de l'art, on peut citer les résultats du modèle *IMS+Word2vec* pour la tâche de désambiguïsation lexicale ciblée (*WSD Lexical sample task*) : le modèle obtient en 2016 un score de 89.4% sur un corpus constitué de plus de 15 000 noms et 10 000 verbes, avec un inventaire de sens issu de WordNet [37].

4.4 Analyse qualitative

Dans la seconde partie de mon stage, nous nous sommes intéressés de plus près aux précisions effectuées par les différentes méthodes sur un petit ensemble d'énoncés.

Données retenues - Nous avons sélectionné 4 lemmes qu'il nous semblait intéressant d'observer en particulier :

- ÉCLAIR, couple d'**homonymes** avec les meilleurs résultats
  - $éclair^{1}_{I.1}$: Il y a eu des *éclairs* pendant l'orage. (phénomène atmosphérique)
  - $éclair^{2}$: J'aime les *éclairs* au chocolat. (pâtisserie)
- ALLER, couple en lien **métaphorique** avec les pires résultats
  - $aller_{I.1.a}$: Je *vais* à Paris. (déplacement)
  - $aller_{II.1}$: Je *vais* au cinéma. (déplacement + action associée à la destination)
- SORTIR, couple en lien **métaphorique** difficile à désambiguïser pour les classifieurs mais facile pour l'humain
  - $sortir_{I.1}$: Je *sors* de ma chambre. (X sort de Y)
  - $sortir_{II.1.a}$: Je *sors* le stylo de ma trousse (X sort Y de Z)
- SOLEIL, couple en lien **métonymique**, bien connu car étudié en 2018 dans le BEL-RL-fr pour des tâches de désambiguïsation manuelle
  - $soleil_{I.a}$:Le *soleil* se lève à l'est. (astre)
  - $soleil_{II.1}$:Le *soleil* d'août est étouffant (ensoleillement, beau temps)

Nous avons constitué un corpus de dev aléatoire regroupant 10% des données de la base. Puis, nous avons relevé les prédictions de toutes les méthodes sur ce dev,pour ces 4 lemmes, ce qui fait en tout **50 énoncés** (14 pour SORTIR, 12 pour les 3 autres).

Tous ces énoncés ont été relus avec attention. Nous avons observé et comparé les prédictions des différentes méthodes, mais également vérifié manuellement la qualité de l'analyse syntaxique effectuée en amont par UDPipe. Les voisins RL-fr, les cooccurrents ainsi que les voisins syntaxiques fréquents ont également été

37. voir `https://nlpprogress.com/english/word_sense_disambiguation.html` et (Iacobacci *et al.*, 2016). L'article ne mentionne pas le taux de polysémie moyen des lemmes désambiguïsés.

recherchés dans les énoncés, ce qui nous a permis de mieux comprendre nos résultats, et de dégager des pistes d'amélioration.

Constats sur les données - La lecture des 50 énoncés sélectionnés a révélé la présence d'imperfections dans les données elles-mêmes, avant tout pré-traitement : des fautes d'orthographe ou de grammaire, des erreurs de typographie et des tournures maladroites ou ambiguës ont été relevées dans 9 énoncés. Les fautes concernent les données issues du frTenTen17, dont on savait qu'elles étaient moins propres que celles du RL-fr ; cependant un nettoyage plus poussé des données pourrait éviter aux erreurs de se propager dans l'analyse UDPipe, et donc améliorer les résultats.

Par ailleurs, un des énoncés nous a finalement semblé ambigu quant au sens de la cible réellement employé : *"J'aime apprendre de nouvelles choses, essayer de nouvelles choses et **aller** à de nouveaux endroits."* ($aller_{I.1.a}$ ou $aller_{II.1}$?) ; son classement dans $aller_{I.1.a}$ dans nos données est donc questionnable.

Ces points doivent donc être pris en compte comme une source d'erreurs pour nos résultats.

Analyse UDPipe - J'ai relu les analyses UDPipe des énoncés et dénombré les erreurs de 3 types : les erreurs de tokénisation (ex : tokéniser *l'éclair* en L+'+ÉCLAIR au lieu de L'+ÉCLAIR), de lemmatisation (ex : lemmatiser *sors* en SORS au lieu de SORTIR) et d'étiquetage grammatical ou Part-of-Speech (POS) (ex : étiqueter *sors* en ADV au lieu de VERB).

type de défaut	**éclair**	**aller**	**sortir**	**soleil**
tokénisation	5	2	4	0
lemmatisation	3	0	17	1
POS	4	2	13	4
total	12	4	34	5

FIGURE 31 – Détail des erreurs relevées dans l'analyse UDPipe (par lemme)

On peut supposer que de telles erreurs se produisent également dans les données d'apprentissage, ce qui est une source d'erreur pour nos méthodes : un voisin (lexical ou syntaxique) mal lemmatisé ne sera pas reconnu, un mauvais étiquetage grammatical biaise le décompte des POS dans le contexte local autour de l'occurrence-cible, etc... Il est possible que les mauvais résultats pour SORTIR s'expliquent en partie par de telles erreurs : il faudrait voir si les données pour ce lemme sont de moins bonne qualité. On pourrait aussi éventuellement essayer un autre outil qu'UDPipe.

> **Remarque:** Il faut garder à l'esprit que certaines erreurs sont liées entre elles : une erreur de lemmatisation est souvent corrélée à une erreur de POS.

D'un autre côté, il reste des informations fournies par UDPipe que nos méthodes n'exploitent pas : c'est le cas des traits morpho-syntaxiques des tokens. Considérons par exemple les énoncés *"Saupoudrez un peu de thé sur le dessus de l'**éclair**"* et *"Mettez la crème pâtissière dans une poche à douille de 11 cm de diamètre et remplissez les **éclairs**."*. Dans les deux cas, la présence de verbes au mode impératif est un indice du genre "recette de cuisine", qui est lui-même un indice fort pour désambiguïser ÉCLAIR. Ce trait a été correctement identifié par UDPipe, il pourrait donc s'agir d'une piste pour de nouveaux features dans notre représentation de données ; ici la régression logistique et random forest échouent à prédire le bon sens.

Voisins RL-fr (version sans aléatoire)- Si la méthode des voisins obtient une excellente précision pour les 4 lemmes testés, nous avons constaté que certaines prédictions étaient correctes "par hasard". Par exemple, un énoncé d'ALLER commence par *"Si l'on veut bien se **rendre** compte de la grandeur de cet ouvrage [...]"*, avec *rendre* identifié comme voisin du sens $aller_{I.1.a}$, ce qui permet ici d'effectuer la bonne catégorisation de l'énoncé. Cependant, dans le RL-fr, ce n'est pas avec *rendre* qu'$aller_{I.1.a}$ est en relation, mais avec *se rendre* (au sens du déplacement, et non pas de *se rendre compte* !). Cette erreur est possible car, en l'état du code, je n'ai conservé aucune autre information concernant les arcs lexie-voisin que le lemme des voisins en question.

> **Exemple:** En d'autres termes, de l'information *"relation de Synonymie avec intersection de 'aller$_{I.1.a}$' vers 'se [préfixe] rendre $_{II}$ (VERB)'*, je ne conserve que *"relation entre 'aller$_{I.1.a}$' et 'rendre'"*.

Pour éviter de capter de mauvaises occurrences, il serait intéressant de conserver également les préfixes et suffixes associés aux voisins, ainsi que leur catégorie grammaticale (ce qui éviterait par exemple de confondre la CCONJ *car* avec le NOUN *car*).

Par ailleurs, la méthode des voisins (sans aléatoire) ne prédit jamais plus de 58% des énoncés (pour ÉCLAIR) ; seul un énoncé sur 12 est prédit pour ALLER et SORTIR. Les énoncés non-prédits correspondent pour la plupart à des cas où **aucun voisin** n'apparaît, alors que des indices lexicaux peuvent être dégagés par le locuteur. Nous avons donc tenté de voir si la méthode des voisins pouvait être améliorée sur ces 4 lemmes en prenant aussi en compte des **voisins de niveau 2**, c'est-à-dire les termes liés à nos lexies dans le RL-fr par un chemin de deux relations successives. Nous avons groupé ces voisins de niveau 2 en deux catégories : les **paradigmatiques** et **syntagmatiques** (cf section. 3.1.3), selon la nature des relations qui les lient aux sens-cibles. En faisant gonfler le nombre de voisins, on espère donc faire plus de prédictions correctes.

nombre de voisins	éclair	aller	sortir	soleil
niveau 1 - sens 1	30	18	9	8
niveau 1 - sens 2	4	6	6	11
niveau 2 syntagmatiques - sens 1	66	51	127	0
niveau 2 syntagmatiques - sens 2	0	31	40	5
niveau 2 paradigmatiques - sens 1	30	44	15	27
niveau 2 paradigmatiques - sens 2	28	6	2	88

FIGURE 32 – Détail du nombre de voisins (niveau/catégorie) par lemme

Chacun des énoncés a été relu, et nous avons effectué à la main les prédictions de la méthode des voisins prenant en compte les voisins de niveau 2 (d'abord uniquement syntagmatiques, puis uniquement paradigmatiques, et enfin les deux ensembles). Le tableau 33 récapitule, pour chacune de ces configurations et pour chacun des lemmes, le nombre de prédictions correctes effectuées, accompagné du nombre de prédictions effectuées par la méthode. Rappelons que le nombre total de prédictions attendues par lemme est de 12 (et de 14 pour SORTIR).

nb de prédictions correctes / effectuées	éclair	aller	sortir	soleil
niveau 1 seulement	7/7	1/1	1/1	4/5
niveau 2 syntagmatiques seulement	5/5	5/8	6/12	1/1
niveau 2 paradigmatiques seulement	10/10	1/2	2/2	7/12
niveau 2 syntagmatiques & paradigmatiques	**11/11**	5/9	6/12	7/12

FIGURE 33 – Evolution des prédictions en fonction des voisins considérés

On constate :

— une grande **amélioration des résultats pour ÉCLAIR** avec le niveau 2 paradigmatique et syntagmatique : la précision de 100% est conservée, mais on prédit bien plus d'énoncés ; il semblerait que, du fait de la relation **homonymique** entre les deux lexies du couples, leurs contextes d'apparition soient assez distincts pour que les voisins de niveau supérieur continuent d'apporter de la précision lexicale sans se recouvrir mutuellement

— une plus forte apparition du niveau 2 syntagmatique pour ALLER et SORTIR (relation de copolysémie métaphorique), et du niveau 2 paradigmatique pour SOLEIL (relation de copolysémie métonymique). Cependant, dans chacun des cas, **la précision finale est dégradée**, et proche de ce qu'on obtiendrait avec des prédictions aléatoires.

En réalité, sauf pour ÉCLAIR, les nouveaux voisins qui apparaissent dans les énoncés ne sont quasiment jamais des mots considérés comme des indices lexicaux par un lecteur humain : ce sont des mots de classe grammaticale fermée (comme la préposition *de*, très fréquente en contexte) ou des verbes supports fréquents comme *être* ou *aller*.

C'est alors le sens auquel sont associés ces voisins qui se voit attribuer l'ensemble des énoncés que parcourt la méthode, d'où un résultat proche de l'aléatoire (puisque les énoncés se répartissent de manière équitable entre les deux sens). La solution ne semble pas être de retirer ces mots des voisins, puisque nous avons dit que les autres voisins n'apparaissaient pas non plus - en retirant les classes fermées et les verbes supports fréquents, on revient à des résultats équivalents à ceux obtenus avec les seuls voisins de niveau 1. C'est donc sans succès que nous avons essayé d'améliorer la méthode des voisins pour les couples de sens qui ne sont pas des homonymes.

Voisinage syntaxique et cooccurrents - L'examens des voisins syntaxiques et des cooccurrents a permis de repérer quelques imperfections dans les fonctions qui calculent ces ensembles de mots, et qui ont donc pu être corrigées (par exemple, on ne souhaite pas extraire de symboles). L'absence de certains indices lexicaux dans les ensembles de cooccurrents peut également s'expliquer par la méthode de calcul employée : le **tf-idf est "agressif"** en cela qu'il élimine d'office tout terme apparaissant dans les corpus d'apprentissage des deux sens [38]. Une amélioration pourrait consister à revoir cette méthode de calcul. L'examen des prédictions des différents énoncés n'a pas permis de mettre en évidence des différences notables entre les couples de lemmes selon leur relation de copolysémie ou leur catégorie grammaticale.

Features importants - Enfin, il m'a semblé intéressant de me questionner sur la manière dont la représentation des données choisie pour la Régression Logistique et pour Random Forest (cf section 4.3.4) a été exploitée par ces méthodes : pour chaque lemme, **quels features ont été importants ?** Les features discriminants sont-ils les mêmes pour chacune de ces méthodes ? Pour chacun des lemmes ?

La bibliothèque *sklearn* permet un accès direct aux features importants de ses classifieurs de type Random Forest [39]. Cette fonctionnalité n'existe pas pour le classifieur de type Régression Logistique, j'ai donc écrit une fonction qui permet de classer les features par importance pour cette méthode [40]. Pour les 4 lemmes ÉCLAIR, ALLER, SORTIR et SOLEIL, j'ai consulté les 10 features les plus discriminants par méthode au cours de l'exécution étudiée pour cette analyse qualitative.

A l'ordre près, les 10 features les plus importants coïncident fortement entre les deux méthodes. Pour le lemme ÉCLAIR, 3 features représentent à eux seuls plus d'un quart du poids de la prédiction Random Forest : il s'agit du nombre de voisins lexicaux de chaque sens dans le contexte global, ainsi que de la présence du mot *chocolat* (d'ailleurs inclus dans les voisins lexicaux de *éclair*^{2}). ÉCLAIR étant en effet

38. En effet, si un lemme apparaît dans les deux corpus d'apprentissage, son idf vaut $log(2/2) = log(1) = 0$, d'où $tf - idf_{sens_i} = 0$ (voir fig. 14)

39. voir documentation en ligne

40. L'importance d'un feature est assimilée à la valeur (absolue) du poids qui lui est associé par le classifieur (les features ayant préalablement été ramenés à des valeurs comprises entre 0 et 1).

déjà prédit très efficacement par la méthode des voisins (plus de 90% de précision en phase de développement), on voit que l'apport de notre représentation de données est moindre.

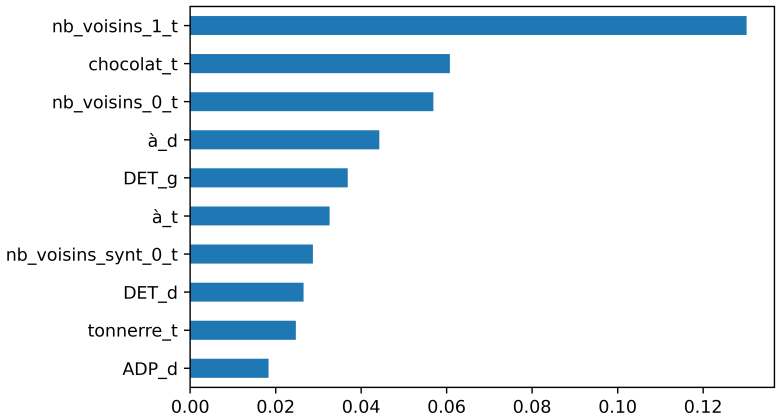


FIGURE 34 – 10 features les plus importants pour ÉCLAIR (Random Forest)

Pour les autres lemmes cependant, les features qui apparaissent parmi les plus discriminants concernent des critères introduits par notre représentation de données, comme le voisinage syntaxique ou les catégories grammaticales dans le contexte local. Pour ALLER et SORTIR notamment, la précision des prédictions augmente considérablement en Régression Logistique / Random Forest par rapport aux meilleures méthodes précédentes. On observe également pour ALLER un apport des entités nommées, avec le nombre d'entités de localisation dans le contexte droit comme critère discriminant (*ex : aller à Paris*).

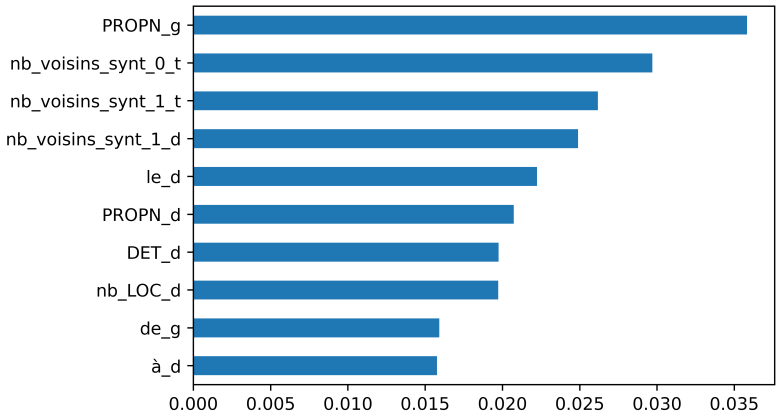


FIGURE 35 – 10 features les plus importants pour ALLER (Random Forest)

De manière générale, on observe une grande variation des types de features importants d'un lemme à l'autre. Ceux pour lesquels les méthodes précédentes fournissaient déjà de bons résultats conservent les critères du voisinage lexical et du

vocabulaire parmi les plus importants en Régression Logistique et Random Forest ; tandis que les nouveaux critères sont plus déterminants pour les autres.

Par la suite, il serait intéressant de généraliser cette observation des features importants pour tenter d'améliorer la représentation des données concernant une certaine catégorie grammaticale ou une certaine relation de copolysémie.

5 Conclusion et perspectives

Durant ce stage, j'ai pu mettre en place plusieurs méthodes de désambiguïsation lexicale automatiques, utilisant des corpus d'apprentissage et de test partiellement annotés en sens. Cette annotation repose sur l'inventaire de sens d'une ressource lexicale de nouvelle génération, le RL-fr, dont l'ossature relationnelle très riche (plusieurs centaines de relations lexicales décrites) la distingue des ressources ontologiques traditionnellement utilisées en TAL pour la désambiguïsation. J'ai donc pu largement enrichir mes connaissances quant aux classifieurs de référence du TAL, et avoir un aperçu des différentes manières dont on peut décrire le lexique d'une langue. En particulier, j'ai pu me documenter sur la théorie Sens-Texte qui sous-tend le RL-fr (Mel'čuk et Polguère, 2021) et sur les notions fondamentales de la lexicologie (Polguère, 2019).

L'étude comparative des différentes méthodes implémentées a permis de mettre en avant l'apport d'une description lexicale fine des lexies ciblées pour la tâche de désambiguïsation, notamment avec la prise en compte de leurs voisins lexicaux. Combinée à d'autres critères, cette approche a permis de prédire avec une précision proche de 89% [41] le sens des occurrences étudiées. Si la comparaison avec les résultats obtenus par les méthodes à base de réseaux de neurones menées dans le cadre du projet BEL-RL-fr en 2020-2021 reste limitée (à la fois par la petite taille des données traitées pendant mon stage, et par la simplification du problème que mon approche constitue), elle met en regard de ces derniers résultats les performances de méthodes plus basiques, et donc plus interprétables.

L'analyse qualitative sur une partie des données a en effet permis de dégager des critères de désambiguïsation pertinents ainsi que des pistes d'amélioration quant à l'exploitation des données du RL-fr. Au cours de mes essais, les couples de sens en relation homonymique ont globalement été parmi les mieux prédits, en raison de la différence marquée entre leurs contextes d'apparition respectifs. La poursuite ultérieure de notre analyse qualitative pourrait permettre de prendre en compte les particularités des autres relations de copolysémie dans la désambiguïsation ; ce point constitue un des questionnements du projet BEL-RL-fr. Le travail amorcé pendant mon stage permettra également d'alimenter la réflexion portée sur l'enrichissement et l'exploration du RL-fr, et sur le phénomène de l'ambiguïté lexicale.

Si la découverte du milieu de la recherche m'a paru très intéressante, j'ai été un peu déstabilisée au début de mon stage de ne pas savoir avec exactitude ce que nous allions faire, ni à quel rythme. Une question peut en effet en appeler une autre, et nous n'avions pas défini à l'avance précisément quelles méthodes allaient être testées ou non. De plus, le BEL-RL-fr est un projet encore récent, et la réflexion sur son

41. pour la Régression Logistique, cf fig.28

exploitation dans les tâches de désambiguïsation n'en est qu'à son commencement. C'est en m'entretenant régulièrement avec mes maîtres de stage et en leur posant mes questions que j'ai par la suite mieux compris le cadre dans lequel s'inscrivait mon travail. Cela m'a aussi permis de clarifier la documentation que j'ai rédigée pour expliquer mon code et mes résultats tout au long du stage.

D'un point de vue technique, ce stage m'a permis d'améliorer mes compétences en Python et de découvrir des librairies de référence pour le TAL. J'ai pu apprendre à utiliser l'IDE Visual Studio Code et me familiariser davantage avec les Jupyter Notebooks, grâce auxquels j'ai pris l'habitude de documenter mon travail de la manière la plus claire possible, tout en travaillant à adopter des bonnes pratiques de code (commentaires systématiques, définition de classes Python pour les objets fonctionnant comme des entités). J'ai aussi appris à utiliser GitLab et quelques unes de ses nombreuses fonctionnalités destinées à faciliter la gestion de projets (dépôt des données et gestion des différentes branches, tickets de discussion, wiki du projet...). Enfin, la rédaction de ce rapport a été l'occasion de m'initier à la rédaction en LaTeX et à la gestion d'une bibliographie.

En conclusion, ce stage m'a permis d'acquérir de nouvelles compétences et d'enrichir mes connaissances au sujet de la désambiguïsation lexicale ; il constitue à mes yeux une bonne transition vers mon orientation en master TAL pour la rentrée prochaine.

Bibliographie

Références

Audibert, L. (2004). Word sense disambiguation criteria : a systematic study. *In {COLING} 2004 : Proceedings of the 20th International Conference on Computational Linguistics*, pages 910–916, Genève, Switzerland.

Audibert, L. (2007). Désambiguïsation lexicale automatique : sélection automatique d'indices. *In* et Philippe Muller, N. H., éditeur : *Traitement Automatique des Langues Naturelles (TALN-2007)*, pages 13–22, Toulouse, France. IRIT Press.

Firth, J. R. (1957). *Papers in Linguistics, 1934-1951*. Oxford University Press, London.

Iacobacci, I., Pilehvar, M. T. et Navigli, R. (2016). Embeddings for word sense disambiguation : An evaluation study. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.

Jurafsky, D. et Martin, J. H. (2022). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition, 3rd edition draft chapters.*

Mel'čuk, I. et Polguère, A. (2021). Les fonctions lexicales dernier cri. *In* Marengo, S., éditeur : *La Théorie Sens-Texte. Concepts-clés et applications*, Dixit Grammatica, pages 75–155. L'Harmattan, Paris.

Navigli, R. (2009). Word sense disambiguation : A survey. *ACM Comput. Surv.*, 41(2).

Polguère, A. (2014). Principes de modélisation systémique des réseaux lexicaux. *In TALN 2014*, pages 79–90, Marseille, France.

Polguère, A. (2019). *Lexicologie et sémantique lexicale : Notions fondamentales*. Paramètres. Presses de l'Université de Montréal, Montréal. Code : Lexicologie et sémantique lexicale : Notions fondamentales Publication Title : Lexicologie et sémantique lexicale : Notions fondamentales Reporter : Lexicologie et sémantique lexicale : Notions fondamentales Series Title : Paramètres.

Veronis, J. (2001). Sense tagging : does It make sense ? *In Corpus Linguistics'2001 Conference.*

Annexes

Liste des catégories grammaticales

Cette liste correspond aux catégories universelles utilisées par UDPipe.

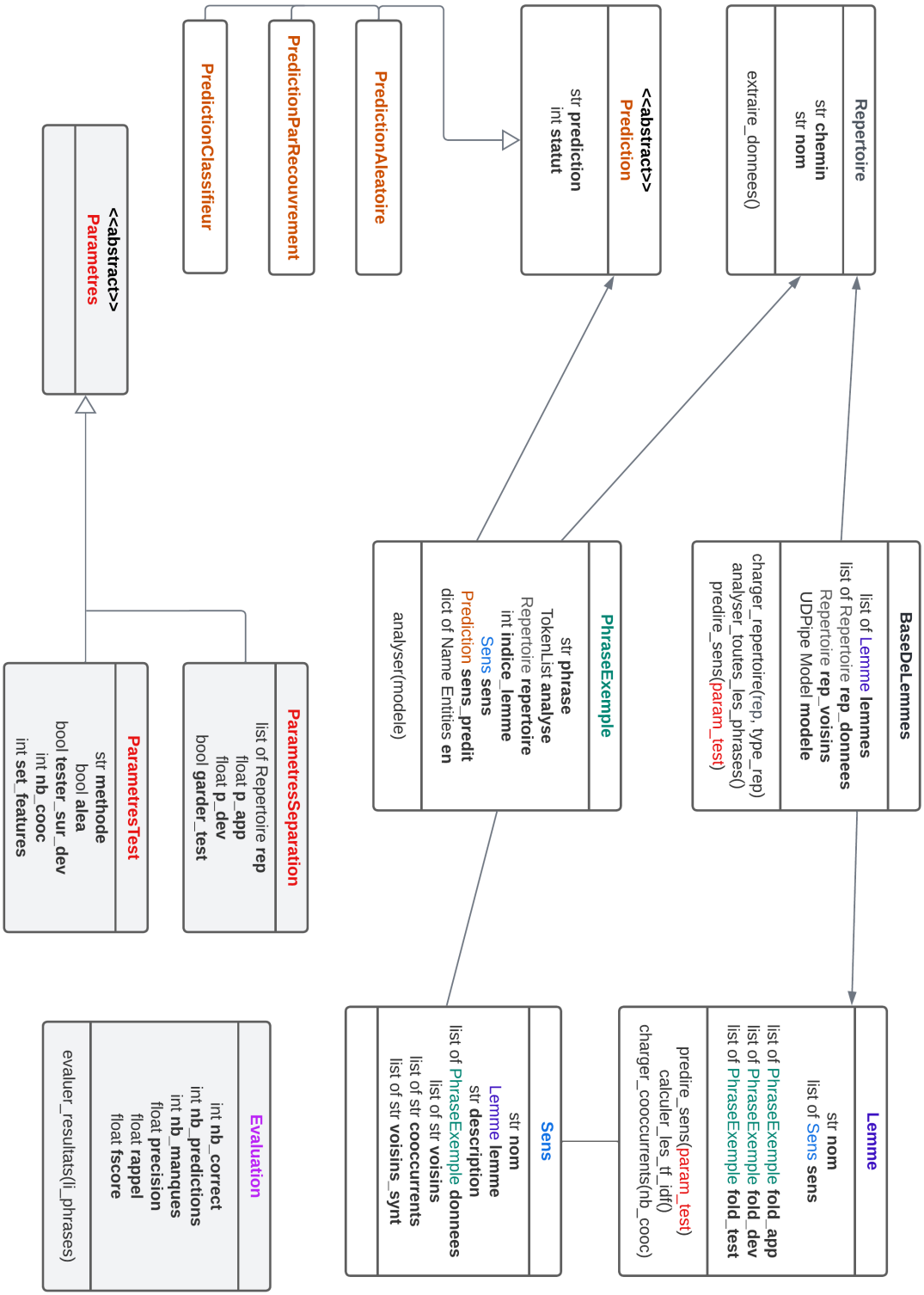
- classes ouvertes
  - ADJ : adjectif
  - ADV : adverbe
  - INTJ : interjection
  - NOUN : nom commun
  - PROPN : nom propre
  - VERB : verbe
- classes fermées
  - ADP : préposition
  - AUX : verbe auxiliaire
  - CCONJ : conjonction de coordination
  - DET : déterminant
  - NUM : numéral
  - PART : particule
  - PRON : pronom
  - SCONJ : conjonction de subordination
- Autres catégories
  - PUNCT : ponctuation
  - SYM : symbole
  - X : autre

Liste des lexies étudiées

lemme	POS	lexies	description	relation entre les lexies
AVEC	ADP	avec I.2	synonyme : « en présence de », X avec Y où Y est un être (*avec elle*)	/
		avec I.3.a	synonyme : « au moyen de », faire X avec Y (*avec ardeur*)	
ALLER	VERB	aller I.1.a	témoigne d'un déplacement : X va à Y (*aller à Paris*)	métaphore
		aller II.1	témoigne d'un déplacement + activité : X va à Y [pour …] (*aller aux toilettes*)	
ALLONGER	VERB	allonger[1]	fait de rendre long, X allonge Y (*allonger la pâte*)	homonymes
		allonger[2]	synonyme : « donner », X allonge Y à Z (*allonger la monnaie*)	
CHANTER	VERB	chanter I.2	émettre un certain bruit, pour un animal, X chante (*l'oiseau chante*)	/
		chanter IV	ça chante X – ça plaît à X (*fais ce qui te chante!*)	
COURIR	VERB	courir I.1.a	se déplacer d'une certaine manière, X court (*il court dans la rue*)	/
		courir VI	synonyme : « s'exposer » (*il court un risque*)	
DORMIR	VERB	dormir I.1.a	être dans l'état de sommeil, X dort (*je dors*)	métaphore
		dormir II.2.b	synonyme : « reposer, croupir », X dort dans Y (*les livres dormant dans les archives*)	
MANGER	VERB	manger I.1.a	s'alimenter, X mange Y (*il mange un steak*)	/
		manger VI.1	synonyme : « écraser », X mange Y (*les joueurs ont mangé leurs adversaires*)	
MARCHER	VERB	marcher I.1.a	se déplacer d'une certaine manière, X marche (*il marche dans la rue*)	métaphore
		marcher VI.1.a	synonyme : « fonctionner », X marche (*l'ascenseur marche de nouveau*)	
NETTOYER	VERB	nettoyer I.1.a	synonyme : « laver », action sur une chose physique (*il nettoie les carreaux*)	métaphore
		nettoyer VI.2.a	sens métaphorique (*il nettoie son ordinateur pour libérer de l'espace*)	
SORTIR	VERB	sortir I.1	antonyme : « entrer », X sort de Y (*il sort de son lit*)	métaphore
		sortir II.1.a	action sur une chose physique, X sort Y de Z (*il sort un stylo de sa trousse*)	
VOLER	VERB	voler[1] I.1	se déplacer d'une certaine manière, X vole (*l'oiseau vole*)	homonymes
		voler[2] I	synonyme : « dérober », X vole Y à Z (*ils ont volé la banque*)	
LONG	ADJ	long I.2.a	synonyme : « grand », caractéristique d'une chose physique (*un manteau long*)	/
		long III.2.b	synonyme : « interminable », caractéristique d'une durée temporelle (*une longue journée*)	
AILLEURS	ADV	ailleurs[1]	localisation physique (*il n'y a plus de place, je vais m'installer ailleurs*)	extension de sens
		ailleurs[2]	localisation abstraite (*la liberté est ailleurs*)	
PRES	ADV	près I.1	localisation, X près de Y (*Nancy est près de Metz*)	métaphore
		près II.1	synonyme : « environ » (*ça fait presque 2 mois*)	
ANTENNE	NOM	antenne II.1	objet, synonyme : « capteur/récepteur » (*une antenne téléscopique*)	métonymie
		antenne II.2	entité, synonyme : « direct » (*il est à l'antenne en ce moment*)	
BOUCHER	NOM	boucher I.2	individu qui pratique un métier (*acheter de la viande chez le boucher*)	métaphore
		boucher II	synonyme : « meurtrier » (*un boucher sanguinaire*)	
BOUCHON	NOM	bouchon[1] I.1.b	objet, synonyme : « couvercle » (*le bouchon d'une bouteille*)	métaphore
		bouchon[1] III	situation, synonyme : « embouteillage » (*il y a des bouchons sur l'A4*)	
ECLAIR	NOM	éclair[1] I.1	phénomène atmosphérique (*il y a eu de nombreux éclairs pendant l'orage*)	homonymes
		éclair[2]	préparation alimentaire (*il commande un éclair au chocolat*)	
FAUTEUIL	NOM	fauteuil I	siège (*il s'assoit dans le fauteuil*)	métaphore
		fauteuil II	fonction sociale (*il a quitté son fauteuil de président*)	
LESSIVE	NOM	lessive I	générique : « tâche ménagère » (*le dimanche, c'est jour de lessive*)	métonymie
		lessive III	synonyme : « linge » (*la lessive est quasiment sèche*)	
LIT	NOM	lit I.1.a	meuble (*elle dort dans un grand lit*)	métaphore
		lit V.a	lieu physique (*le lit d'une rivière*)	
MARCHE	NOM	marche[1] I.3.c	synonyme : « randonnée », (*une marche de plus de 3 heures*)	/
		marche[1] V.1.b	synonyme : « escalier », (*elle descend les marches*)	
ROUGE	NOM	rouge I.1.b	générique : « couleur » (*une robe rouge*)	métaphore
		rouge IV	synonyme : « vin rouge » (*il boit du rouge*)	
SOLEIL	NOM	soleil I.a	astre (*le soleil est haut dans le ciel*)	métonymie
		soleil II.1	synonyme : « beau temps » (*le soleil d'août est étouffant*)	
VESTE	NOM	veste[1]	vêtement (*une veste chaude*)	homonymes
		veste[2] 1	échec (*le candidat a pris une veste mémorable*)	

Diagramme de classes

Ce diagramme présente les classes utilisées dans mon code, accompagnée de l'intégralité de leurs attributs. Par souci de clarté, seules les méthodes principales ont été citées.


Repertoire
str chemin
str nom
extraire_donnees()
<<abstract>>
Prediction
str prediction
int statut
PredictionAleatoire
PredictionParRecouvrement
PredictionClassifieur
<<abstract>>
Parametres
BaseDeLemmes
list of Lemme lemmes
list of Repertoire rep_donnees
Repertoire rep_voisins
UDPipe Model modele
charger_repertoire(rep, type_rep)
analyser_toutes_les_phrases()
predire_sens(param_test)
PhraseExemple
str phrase
TokenList analyse
Repertoire repertoire
int indice_lemme
Sens sens
Prediction sens_predit
dict of Name Entities en
analyser(modele)
Lemme
str nom
list of Sens sens
list of PhraseExemple fold_app
list of PhraseExemple fold_dev
list of PhraseExemple fold_test
predire_sens(param_test)
calculer_les_tf_idf()
charger_cooccurrents(nb_cooc)
Sens
str nom
Lemme lemme
str description
list of PhraseExemple donnees
list of str voisins
list of str cooccurrents
list of str voisins_synt
ParametresSeparation
list of Repertoire rep
float p_app
float p_dev
bool garder_test
ParametresTest
str methode
bool alea
bool tester_sur_dev
int nb_cooc
int set_features
Evaluation
int nb_correct
int nb_predictions
int nb_manques
float precision
float rappel
float fscore
evaluer_resultats(li_phrases)