Désambiguisation lexicale semi-supervisée et auto-entraînement

Jules Cauzinille, Léo Labat, Mathilde Wang Castejon,

Jeudi 17 juin 2021

1 Introduction

La désambiguisation lexicale (*Word Sense Disambiguation* ou *WSD* en anglais) est une tâche de traitement automatique des langues demandant de déterminer, dans un contexte spécifique, le sens d'un mot polysémique à partir d'une liste déterminée de sens discrets. Dans la phrase "l'avion vole" le verbe voler est compris par un humain dans son sens "se mouvoir dans les airs" plutôt que "dérober quelque chose" et cette décision est prise en fonction du contexte d'apparition du verbe en question. C'est donc le contexte d'un terme polysémique qui va permettre sa désambiguïsation par un programme informatique. Cette tâche est primordiale en TAL puisqu'elle peut permettre de faciliter certains exercices tels que la reconnaissance de la parole, l'extraction de connaissances ou bien la traduction automatique. C'est notamment dans cette dernière application qu'elle prend tout son sens : un terme polysémique en français ne le sera peut-être pas dans d'autres langues. Sa traduction demande donc une forme de classification du mot et d'extraction de son sens en contexte afin d'en trouver un équivalent exact dans la langue cible (ici, "voler" peut se traduire en anglais par "fly" ou par "steal" selon son contexte d'apparition).

Un programme informatique peut permettre de répondre à cette problématique de plusieurs manières : par classification supervisée, semi-supervisée voire même non-supervisée.

La version supervisée demande un corpus annoté en sens permettant d'entraîner un algorithme de classification classique. Or, la constitution d'un tel corpus, qui se doit d'être relativement grand pour obtenir des résultats significatifs, peut s'avérer très coûteuse.

À l'inverse, la version non-supervisée ne demande pas de corpus annoté. Il suffit d'avoir accès à un certain nombre de phrases contenant des occurrences du mot à désambiguïser pour les classifier. En revanche, cette approche présente des performances bien moins intéressantes puisque l'algorithme naïf qu'elle emploie n'a accès qu'à une quantité très réduite d'informations (il ne connaît même pas le nombre de sens que peut prendre le mot cible, ce qui peut s'avérer handicapant pour des algorithmes de clustering notamment).

Une solution qui pourrait potentiellement résoudre à la fois ces deux problématiques est la version semi-supervisée de l'algorithme de classification.

1.1 Hypothèses

Nous avons voulu tester l'hypothèse selon laquelle un classifieur semi-supervisé permettrait de palier au manque de données annotées pour afficher une meilleure performance qu'un classifieur supervisé (de type Support Vector Machine ou Multi-Layer Perceptron). Tout le travail effectué est basé sur deux corpus : l'un annoté en sens (désambiguïsé) et lemmatisé (FSE) l'autre simplement lemmatisé, sans annotation en sens (corpus l'Est Républicain). Nous avons également décidé de nous borner à une représentation vectorielle word2vec statique (*frozen embeddings*) tirée du corpus frWac en CBOW (sans lemmes ni postag) de taille 200. Les paramètres que nous avons voulu tester sont : la taille de la fenêtre autour du mot cible à désambiguiser (contexte représenté par addition des embeddings de mots autour du mot cible), différentes architectures d'algorithmes de classification pour la partie superivisée (MLP et SVM), une version semi-supervisée du K-Means et, enfin, le principe d'auto-entraînement à comparer à la version simplement supervisée à laquelle nous avons ajouté nos "nouvelles" phrases d'exemples classifiées par notre algorithme de partitionnement.

2 Préparation du corpus

2.1 Corpus annoté

Le corpus annoté "FSE" qui nous a été fourni contient, en plus du sens du mot à désambiguïser, des informations sur chaque mot telles que son lemme ou sa catégorie morpho-syntaxique. Néanmoins, il ne nous a pas semblé nécessaire d'utilise toutes les informations disponibles (nous travaillons pas sur une forme "feature based" de WSD mais plutôt sur les problématiques liées à la taille de corpus).
Le parsing de ce premier corpus se fait avec Python, sur Google Colab. Nous avons choisi de représenter chaque exemple comme un tuple contenant la phrase en string puis la classe associée au sens du mot cible. Par exemple, pour le verbe "abattre" on a :

```
1 ('Malheureusement les ruines furent abattues en 2004 . ', '1')
2
```

Ces listes de phrases sont ensuite "contextualisées" dans le sens où l'on sélectionne les n mots entourant la forme fléchie du lemme cible, on les vectorise à partir de notre dictionnaire d'embeddings, puis on additionne les 2n+1 embeddings ainsi extraits. Cet ensemble d'exemple permet alors d'entraîner les différents algorithmes implémentés. Il est tout de même important de signaler que les ensembles d'exemples annotés ne représentent qu'environ 50 phrases par lemme dans les meilleurs cas... Nous rejoignons alors la problématique soulevée

par notre hypothèse, à savoir que la disponibilité des données annotées permettant d'entraîner des algorithmes supervisés est bien trop faible pour obtenir des résultats satisfaisants et qu'une approche semi-supervisée pourrait potentiellement permettre de palier à ce manque de données.

2.2 Corpus non annoté

Notre corpus non-annoté, qui permettra de "créer" de nouveaux exemples d'apprentissage par clustering semi-supervisé en se basant sur le corpus annoté décrit plus haut, provient d'un ensemble de phrases lemmatisées du journal l'Est Républicain. La segmentation en phrases puis en mot se fait de la même manière que pour le corpus annoté à la différence que l'on ne récupère que les phrases contenant des occurences fléchies de nos lemmes cibles et que chaque phrase ne sera pas associée au sens du mot en question. Ce corpus, contrairement au premier, propose plusieurs milliers d'occurences de chacun de nos mots cible (environ 2000 occurrences par mot pour les verbes sur lesquels nous allons nous concentrer par la suite).

3 Version supervisée

La tâche de désambiguïsation lexicale supervisée a été longuement explorée en TAL et atteint aujourd'hui des performances tout à fait acceptables, dépassant généralement les 90% de précision peu importe l'homonyme classifié, pourvu que les données soient suffisantes. La disponibilité des données est cruciale, et comme pour bien d'autres tâches de TAL, l'anglais est favorisé par rapport à de nombreuses autres langues. Concernant le français sur lequel nous travaillons, les données restent très éparses et insuffisantes, car cette approche demande en effet un algorithme de classification et des données manuellement annotées pour l'entraîner. Toute la complexité de l'exercice se situe donc dans la constitution de ces données et dans le choix des caractéristiques à représenter. On utilise en général des plongements lexicaux représentant les contextes d'apparition du mot à désambiguïser auxquels on pourrait ajouter quelques features supplémentaires (postag, bigrammes, ordre des mots du contexte, terminaisons, etc.), sans gain significatif. La grande dimensionnalité de ces données (vecteurs de plusieurs centaines de composantes) fait des Support Vector Machines de très bons algorithmes, relativement populaires dans la résolution des problèmes de désambiguïsation lexicale. C'est cette architecture, à laquelle nous avons ajouté un Multi-layer Perceptron en guise de comparaison, qui nous permet d'établir une baseline relative à l'utilisation de classifieur supervisé classique.

3.1 Méthode et résultats observés

Après avoir récupéré nos données d'apprentissage (corpus FSE, cf. 4.1), nous avons construit, en python sur Google Colab, un SVM linéaire avec le package sklearn. Ce classifieur prend en entrée une collection de contextes du mot à

désambiguïser, sous forme d'addition des embeddings des n mots de contexte, associés au sens de l'occurrence correspondante. Nos données annotées étant très petites (entre 20 et 50 phrases par lemme à désambiguïser) le moyen le plus efficace de mesurer la précision comme exactitude (*accuracy*) de l'algorithme est de l'entraîner plusieurs fois sur des données aléatoirement mélangées (à la fois pour l'ensemble de train et celui de test). On obtient ainsi une distribution de scores dépendant de l'ordre dans lequel les phrases ont été vue à l'apprentissage, de la distribution des classes dans l'ensemble de test et de l'initialisation aléatoire des poids appris par l'algorithme. Ainsi, pour un mot tel que "affecter" dont il existe 3 sens dans le corpus FSE, on obtient un score maximal de 96% de précision et un score moyen de 84% avec une variance de 0.005 pour un contexte de taille 4 de chaque côté du mot cible (4 mots avant, 4 mots après). La baseline, donc le pourcentage du sens le plus représenté dans le corpus, étant de 66% (pour le sens numéro 1, avec 33% pour le sens numéro 3), on peut estimer que ces résultats sont déjà satisfaisants. Voilà les résultats pour les trois verbes "abattre", "aborder" et "affecter", avec la fenêtre représentant le nombre de mots observés avant et après le mot cible, pour le SVM linéaire (5000 epoch) et pour le MLP (5000 epoch, couche cachée de taille 100, activation tanh) sur un corpus de test représentant 20% des données et sur 100 entraînements différents avec mélange aléatoire des exemples du corpus (ce qui correspond au protocole de tous les tableaux présentés dans ce compte-rendu).

Résultats SVM sur 100 entraînements					
mot / baseline	fenêtre	max	min	moyenne	variance
Abattre / 68%	2	88%	52%	69%	0.008
	4	92%	56%	72%	0.005
Aborder / 83%	2	92%	60%	76%	0.005
	4	95%	70%	83%	0.002
Affecter / 66%	2	96%	56%	78%	0.006
	4	96%	64%	84%	0.005

On remarque des résultats satisfaisants que l'on pourrait probablement augmenter en modifiant certains paramètres. Les fenêtres plus grandes ont tendance à donner des résultats légèrement meilleurs jusqu'à un certain point et l'ajout de données annotées serait également très bénéfique. On a souvent une accuracy moyenne supérieure à la baseline avec une fenêtre de 4 (dans une moindre mesure pour "aborder" puisque le sens 1 représente déjà 83% des occurences). En réalité, on peut se rendre compte, en multipliant le nombre d'entraînements, que les résultats sont très, voire trop variables et qu'un classifieur supérvisé de type SVM efficace dans la tâche de désambiguisation nécessiterait beaucoup plus que 150 phrases d'exemple à l'entraînement. Pour ce qui est du choix de la représentation du contexte par l'addition des embeddings plutôt que leur concaténation, il semble évident puisque la prise en compte de l'ordre des mots donnerait probablement de moins bons résultats sur des exemples non vus à l'apprentissage et qu'un programme de désambiguïsation lexicale se doit, par principe, d'être très général et efficace sur tout type de texte.

Résultats MLP sur 100 entraînements					
mot / baseline	fenêtre	max	min	moyenne	variance
Abattre / 68%	2	88%	44%	72%	0.006
	4	92%	56%	76%	0.007
Aborder / 83%	2	100%	68%	82%	0.004
	4	95%	77%	87%	0.001
Affecter / 66%	2	100%	60%	80%	0.007
	4	96%	68%	84%	0.004

Le MLP semble donner des résultats très proches de ceux du SVM, si ce n'est légèrement supérieurs. En revanche, il demande un temps d'entraînement bien plus important et qui ne baisse pas proportionellement au nombre d'epoch. Ce temps de traitement peut s'avérer gênant sur les 100 entraînements (environ 50 secondes contre les 5 secondes du SVM) mais reste négligeable sur un seul.

4 Version semi-supervisée

La version semi-supervisée (par seeding) de la tâche de désambiguisation lexicale emploie un algorithme de clustering (le k-means) qui utilise des données annotées en sens comme point de départ pour ensuite associer, à chaque nouvelle donnée non-annotée, l'un des sens possibles en fonction des clusters déjà présents.

4.1 K-means

L'algorithme que nous utilisons pour le clustering est le K-moyennes (K-means). Cet algorithme se base sur un hyperparamètre K censé représenter le nombre de clusters dans lesquels les données seront séparées. Sa version non-supervisée initialise K centroïdes de manière aléatoire, puis à chaque donnée fournie, calcule la distance entre la représentation vectorielle de cette dernière et les K centroïdes initiaux. La plus petite distance détermine alors l'appartenance de la donnée en question à l'un des K centroïdes (donc à l'un des K clusters). Une fois que toutes les données sont ainsi réparties, on calcule la moyenne de chaque cluster pour déterminer un nouveau centroïde. À partir de ce nouveau centroïde, la tâche est alors répétée : les distances sont calculées entre les centroïdes et les données qui sont ensuite réassignées à tel ou tel cluster en fonction. La moyenne des clusters est encore une fois calculée, et ainsi de suite jusqu'à ce que les données soient efficacement réparties et que les centroïdes ne bougent plus à la mise à jour. On peut alors répéter cette tâche plusieurs fois en calculant la variance obtenue pour choisir la distribution la plus performante (correspondant à une variance minimisée).

4.2 Modification du K-Means

Un tel algorithme ne tire pas parti des données annotées que nous avons à disposition (dans notre cas, il assigne les contextes aux clusters donnant une

distance moyenne minimale entre ceux-ci). On peut alors définir un algorithme légèrement différent capable d'initialiser les centroïdes non pas aléatoirement mais sur la base de données annotées. On définit ainsi des "graines" (*seeds*) correspondant aux coordonnées de centroïdes issus du calcul de la moyenne des valeurs de toutes les données annotées pour un cluster donné. Ainsi les données annotées servent de base à un algorithme de type K-Means, puisqu'ils servent à initialiser les centroïdes des clusters. Mais ces données annotées sont conservées pour le calcul des centroïdes, et font partie intégrante de l'ensemble des données. En plus de cette différence dans l'initialisation (le K-Means classique tire aléatoirement K données et s'en sert de point de départ pour les centroïdes), on ajoute une condition dans l'algorithme au niveau de l'assignation des données : au moment du calcul des distances avec les centroïdes à une itération donnée, si la donnée considérée fait partie des données annotées, les distances ne sont pas calculée et l'assignation correspond à l'annotation de référence. Cela implique une confiance accordée par hypothèse aux données annotées, puisque l'algorithme ainsi conçu ne pourra jamais corriger une erreur d'annotation de référence. Néanmoins, l'objectif est de capturer, grâce à la représentation vectorielle par plongements lexicaux (ou *embeddings*) des similitudes entre les données annotées et les données non annotées fournies au K-Means modifié en espérant agréger, au prix d'un bruit plus ou moins important, des effets de collocation et de coprésence sémantique.

4.3 K-means semi-supervisé en WSD

L'implémentation effective d'un tel algorithme à nos données aboutit à des résultats contrastés qui ne permettent pas, dans l'état actuel de nos expériences, de conclure qu'un classifieur semi-supervisé permet de mieux classer les données qu'un classifieur supervisé classique, quand bien même celui-ci aurait été entraîné sur un petit corpus. En effet, le classifieur semi-supervisé obtient systématiquement des performances inférieures à celles obtenues par classification supervisée classique décrite plus haut. De même, la baseline consistant à classer systématiquement les occurrences de mots-formes d'un lemme dans la classe correspondant au sens le plus fréquent permet de révéler que les résultats, même les meilleurs comme ceux que nous présentons dans le tableau suivant, sont peu éloignés de ce seuil de référence (*Most Frequent Sense baseline* en anglais).

All resultats trial 100								
lemma	moyenne	max	min	deviation	taille	total	nbr sens	MFS baseline
situer	92.7%	100%	50%	0.0105	40	50	2	('1', 94.0)
dater	91.8%	100%	60%	0.09	39	49	2	('1', 85.71)
témoigner	89.4%	100%	70%	0.091	40	50	2	('2', 90.0)
interpeller	87%	100%	40%	0.013	40	50	2	('5', 96.0)
détenir	80%	100%	54%	0.011	42	53	2	('1', 54.72)
indiquer	79.1%	100%	20%	0.017	40	50	3	('2', 84.0)

4.4 Auto-entraînement

Le K-means semi-supervisé peut être vu comme un classifieur en soi : il permet d'assigner un sens à une occurrence de mot en fonction d'un ensemble d'apprentissage annoté auquel a été ajouté l'ensemble des données non-annotées. Il peut également être vu comme un générateur d'exemples annotés. Disposant de trop peu de données pour nos classifieurs supervisés (SVM et MLP dont nous avons discuté plus haut), nous sommes en droit de nous demander comment évolueraient leurs performances à l'ajout de ces "nouvelles" données malgré le fait qu'elles ne soient pas 100% fiables. Pour prendre en compte le fait que notre classifieur KMeans fait des erreurs qui seront considérées comme aussi valides que des données annotées de référence, nous nous sommes concentrés sur les verbes que notre classifieur KMeans a visiblement bien classés, puisque sa précision moyenne est dans leur cas supérieure à la baseline définie par le sens le plus fréquent. Il s'agit de 19 verbes sur lesquels nous allons comparer les différentes performances de nos classifieurs, dont voici les résultats : Figure 1 donne les résultats d'un classifieur SVM supervisé entraîné sur un nombre réduit de données (le corpus FSE), qui sont globalement systématiquement supérieurs à ceux de Figure 2, qui donne les résultats d'un classifieur SVM entraîné sur un corpus plus important (d'environ 2000 phrases par verbe). Ce dernier est issu d'une classification antérieure par un classifieur KMeans modifié par nos soins partant de "graines" constituées par le corpus d'apprentissage du SVM entraîné et évalué à la Figure 1.

FIGURE 1 – Résultats d'un SVM entraîné sur le petit corpus FSE

lemma / baseline	window	max	min	mean	écart type
traduire / 52.0	4	100.0%	50.0%	80.0%	0.1342
viser / 56.0	4	100.0%	70.0%	86.0%	0.0917
prononcer / 32.69	4	91.67%	41.67%	66.67%	0.1491
convenir / 32.69	4	66.67%	33.33%	50.83%	0.1083
achever / 62.0	4	90.0%	50.0%	80.0%	0.1265
adapter / 42.0	4	80.0%	50.0%	67.0%	0.1005
alimenter / 62.0	4	90.0%	70.0%	76.0%	0.0663
coûter / 31.91	4	50.0%	10.0%	37.0%	0.1345
relancer / 60.0	4	90.0%	40.0%	66.0%	0.1356
appuyer / 69.39	4	100.0%	70.0%	84.0%	0.1114
chuter / 72.97	4	100.0%	77.78%	93.33%	0.0737
confirmer / 38.0	4	80.0%	20.0%	56.0%	0.196
manifester / 55.1	4	80.0%	50.0%	65.0%	0.1025
conduire / 34.62	4	83.33%	50.0%	60.83%	0.1057
aider / 34.0	4	60.0%	30.0%	44.0%	0.102
détenir / 54.72	4	100.0%	76.92%	84.62%	0.0769
contenir / 42.86	4	100.0%	60.0%	78.0%	0.1166
dater / 85.71	4	100.0%	100.0%	100.0%	0.0
confier / 50.0	4	90.0%	70.0%	77.0%	0.0781

FIGURE 2 – Résultats d'un SVM entraîné sur un corpus issu d'une classification semi-supervisée

lemma / baseline	window	max	min	mean	écart type
traduire / 52.0	4	90.0%	50.0%	64.0%	0.12
viser / 56.0	4	80.0%	40.0%	68.0%	0.14
prononcer / 32.69	4	75.0%	25.0%	53.33%	0.159
convenir / 32.69	4	66.67%	25.0%	48.33%	0.1384
achever / 62.0	4	100.0%	80.0%	93.0%	0.09
adapter / 42.0	4	80.0%	20.0%	46.0%	0.1744
alimenter / 62.0	4	80.0%	50.0%	63.0%	0.0781
coûter / 31.91	4	90.0%	30.0%	51.0%	0.1814
relancer / 60.0	4	80.0%	30.0%	64.0%	0.1497
appuyer / 69.39	4	100.0%	80.0%	95.0%	0.0671
chuter / 72.97	4	100.0%	55.56%	81.11%	0.1319
confirmer / 38.0	4	80.0%	50.0%	68.0%	0.0872
manifester / 55.1	4	80.0%	50.0%	59.0%	0.0943
conduire / 34.62	4	75.0%	41.67%	58.33%	0.1054
aider / 34.0	4	80.0%	30.0%	56.0%	0.1428
détenir / 54.72	4	92.31%	61.54%	76.15%	0.1
contenir / 42.86	4	90.0%	40.0%	69.0%	0.1513
dater / 85.71	4	100.0%	80.0%	96.0%	0.0663
confier / 50.0	4	80.0%	40.0%	59.0%	0.1375

Les résultats ainsi reproduits présentent quelque intérêt, dans la mesure où ils ne sont pas univoques quant à l'échec ou au succès de l'épreuve de falsification à laquelle nous souhaitions soumettre notre hypothèse de départ. En effet, nous cherchions à tester l'idée selon laquelle un corpus annoté par classification semi-supervisée pourrait améliorer les résultats d'un classifieur supervisé tel que le SVM, pour palier le manque de données dont souffre le français pour la tâche de désambiguïsation lexicale. Or, si sur ces 19 verbes, la performance du SVM entraîné sur un corpus plus important est inférieure pour la majorité d'entre eux à la performance du SVM entraîné sur données annotées, une proportion non négligeable (5/19) fournissent des résultats plus satisfaisant à l'évaluation : il s'agit d'"achever" (93% pour le SVM entraîné par corpus issu d'auto-entraînement contre 80% pour le SVM entraîné sur un corpus de petite taille mais de référence), "coûter" (51% contre 37%), "appuyer" (95% contre 84%), "confirmer"(68% contre 56%) et "aider" (56% contre 44%).

4.5 Proposition d'hypothèses explicatives

Pour expliquer de tels résultats, de nombreuses pistes pourraient être explorées. Tout d'abord, il faut garder à l'esprit que l'apprentissage réalisé par un SVM "auto-entraîné" repose sur un corpus bruité, c'est-à-dire que les données prises pour données de référence ne sont pas aussi fiables que celles fournies par des annotateurs humains. Les erreurs sont intégrées et apprises par le classifieur comme des données valides et porteuses d'informations importantes pour

la classification, quand bien même celles-ci l'induiraient en erreur. Il suffit donc que pour de nombreux lemmes, les données disponibles ne permettent pas une classification de proche en proche suffisamment satisfaisante, c'est-à-dire que le KMeans modifié ne permette pas de capturer les effets de similitude de contexte (il repose en effet sur une simple distance euclidienne dans l'espace vectoriel censé représenter, au moins en partie, l'espace sémantique). Il faut garder à l'esprit que les performances du classifieur auto-entraîné sont établies à partir d'un corpus de test issu uniquement des données annotées, qui forment un corpus bien particulier, avec ses caractéristiques d'ensemble qui définissent ce qu'on appelle communément son "domaine". Notre classifieur auto-entraîné contient de nombreux exemples différents issu d'un domaine différent qui tient à la nature du corpus annoté automatiquement, celui de l'Est Républicain. On pourrait imaginer dès lors un effet semblable à celui d'un *overfit* : la comparaison sur les données annotées étant similaire, le fait que le premier classifieur entièrement supervisé soit uniquement entraîné sur des données issues du même domaine que celles sur lesquelles les tests sont réalisés doit induire des effets de proximité : les données annotées utilisées comme "graines" de l'autre algorithme semi-supervisé se trouvent peut-être ainsi comme "diluées" dans de nombreux autres exemples issus du corpus de l'Est Républicain, ce qui rend la classification réalisée a posteriori moins efficace sur les données de test mais augmente peut-être, sans que nous ayons de moyen de le vérifier à ce stade, la puissance de généralisation.

En outre, il convient d'insister que la représentation vectorielle de nos données ne va pas de soi, et le choix d'additionner des embeddings statiques a des implications, notamment du point de vue de la collocation syntaxique (pour laquelle l'ordre compte) puisque les informations contenues dans l'ordre des mots de contexte sont perdues lors de la vectorisation des données. Néanmoins, nous avons fait ce choix eu égard au caractère épars des données et en gardant à l'esprit que d'un point de vue sémantique, c'est la co-présence de lexèmes qui nous semble par hypothèse la plus importante, ce qui implique une importance moindre de l'ordre des mots (pour une phrase parlant d'un vol, le fait qu'il s'agit du fait d'un délinquant ou d'un avion a des chances d'être capturé dans le contexte quel que soit l'ordre dans lequel les mots qui le constituent est agencé). Toutefois, ce choix d'une fenêtre de contexte ignorant l'ordre des mots pour représenter vectoriellement nos occurrences de mots-formes discrimine sans doute les verbes pour lesquels la désambiguïsation est plus facile par échantillonnage de lexèmes, c'est-à-dire les verbes pour lesquels il est fréquent que le simple contexte ait un fort effet de désambiguïsation. Ainsi, l'agrégation d'erreur lors de l'enrichissement du corpus d'entraînement par notre KMeans modifié est compensée par un fort effet de désambiguïsation lié à l'augmentation du nombre de phrases du corpus contenant des items lexicaux discriminants, qui jouent autant lors de la classification semi-supervisée (qui s'appuie sur des distances euclidiennes) que sur l'apprentissage complètement supervisé réalisé sur la sortie de celle-ci. A ce titre, il peut être intéressant de noter que les cinq verbes sur lesquels nos performances auto-entraînées sont meilleures que celles d'un classifieur classique ont tous au moins trois sens (3 pour "achever", 4 pour "coûter", "appuyer" et

"confirmer" et 5 pour "aider"). Ces sens correspondent peut-être à des sens bien distincts (on imagine bien comment le contexte lexical proximal d'un mot-forme du lemme "achever" peut varier selon qu'il s'agit de la mise à mort de quelqu'un ou de l'aboutissement d'un travail) qui sont ainsi plus facile à choisir que les différents contextes sont nombreux. Enfin, il semble raisonnable de supposer que la rareté des données pour certains verbes (18 exemples annotés pour "dominer") ne permet pas d'agréger convenablement les informations lexico-contextuelles de proche en proche dans la mesure où les données sont trop éparses et ne couvrent pas un nombre suffisant de contextes suffisamment désambiguïsants. Face à un tel constat, on ne peut que s'en remettre à la création à l'avenir de corpus annotés par des humains qui suffiront peut-être un jour à atteindre un seuil par-delà lequel notre protocole d'auto-entraînement pourrait porter ses fruits.

5 Conclusion

5.1 Falsification de notre hypothèse initiale ?

Notre hypothèse de départ consistait à avancer la possibilité d'une amélioration des performances d'un algorithme de classification supervisée en ayant recours, face à la rareté des données de référence annotées, à un algorithme de classification semi-supervisé issu de la famille des algorithmes non-supervisé, pour combiner les avantages de la non-supervision (pas besoin de données annotées) et ne pas négliger les informations déjà disponibles que contiennent les petits corpus annotés auxquels nous avons accès. Nous l'avons vu, les performances obtenues par le classifieur KMeans ne sont pas satisfaisantes eu égard à la baseline définie par le sens le plus fréquent. Face à ce constat, nous nous sommes concentrés sur les 19 verbes pour lesquels notre classifieur KMeans obtenait de meilleures performances, traitant la baseline de fréquence comme un indice de confiance dans la classification automatique obtenue. Les résultats de la classification réalisée après un "auto-entraînement" par clustering ne permettent pas de tirer de conclusion univoque quant à la pertinence de notre démarche : si les performances "auto-entraînées" sont souvent moins bonnes, certaines sont bien meilleures, et cela tient peut-être à la qualité du peu de données annotées mais également à des caractéristiques plus linguistiques tenant aux nombreuses différences sémantiques qui traversent les lexèmes considérés lors de notre étude (différences des contextes et facilité plus ou moins grande du contexte très proche à désambiguïser). Ainsi, si nos résultats ne permettent pas de crier victoire et d'annoncer le triomphe de notre méthode, ils peuvent aussi être considérés comme encourageants, puisqu'ils révèlent qu'une amélioration des performances est possible.

5.2 Perspectives d'approfondissement

Pour prolonger notre travail, on pourrait élaborer un protocole garantissant une meilleure qualité des données annotées de façon automatique en définissant

un seuil de confiance C permettant de ne sélectionner que les phrases dont la classification par le k-means semi-supervisé propose un score $>$ c, c'est-à-dire définir une borne à la distance au centroïde établissant un seuil de confiance dans la classification. Nous avons entraînerions ainsi notre SVM initial sur ces données élargies. On pourrait également accorder une importance plus appuyée au protocole d'évaluation des performances, de sorte que jouent moins des effets tels que ceux de la dispersion par changement de domaine (corpus journalistiques contre corpus internet, etc.), en diversifiant les sources de sens annotés gold, quand bien même celles-ci seraient de taille extrêmement réduite. Ainsi, même si nous n'avons pas obtenu les résultats escomptés, ou même espérés, notre travail suggère qu'une amélioration est possible pour certains verbes et qu'au prix d'enrichissement de corpus annotés, de modifications algorithmiques et d'altérations de notre protocole, notre démarche vaut la peine d'être poursuivie.

Annexe des tableaux complets

A partir de la page suivante...

FIGURE 3 – Résultats avec le KMeans semi-supervisée

lemma	moyenne	max	min	écart type	taille	total	nbr sens	MFW baseline
aboutir	0.648	1.0	0.0	0.25	40	50		4 ('3', 82.0)
investir	0.513	0.9	0.1	0.17	40	50		4 ('4', 68.0)
traduire	0.57	1.0	0.1	0.23	40	50		3 ('1', 52.0)
témoigner	0.894	1.0	0.7	0.09	40	50		2 ('2', 90.0)
juger	0.42	0.875	0.0	0.22	32	40		4 ('7', 77.5)
justifier	0.55	1.0	0.0	0.27	40	50		4 ('2', 88.0)
viser	0.765	1.0	0.5	0.12	40	50		2 ('4', 56.0)
prononcer	0.48	0.82	0.09	0.15	41	52		5 ('2', 32.69)
accomplir	0.29	0.64	0.0	0.15	44	55		4 ('2', 80.0)
convenir	0.42	0.73	0.09	0.13	41	52		5 ('1', 32.69)
acquérir	0.30	0.73	0.0	0.17	42	53		5 ('1', 67.92)
achever	0.70	1.0	0.4	0.14	40	50		3 ('1', 62.0)
observer	0.31	0.9	0.0	0.26	40	50		4 ('4', 68.0)
adapter	0.43	0.9	0.1	0.19	40	50		5 ('3', 42.0)
admettre	0.51	0.9	0.1	0.19	38	48		4 ('2', 58.33)
entraîner	0.687	1.0	0.3	0.17	40	50		4 ('4', 80.0)
payer	0.32	0.64	0.0	0.14	42	53		6 ('1', 56.60)
respecter	0.60	1.0	0.3	0.16	39	49		2 ('2', 83.67)
affecter	0.18	0.8	0.0	0.21	40	50		8 ('8', 57.99)
demeurer	0.34	0.9	0.0	0.26	36	46		4 ('4', 73.91)
aggraver	0.62	1.0	0.0	0.24	40	50		2 ('1', 82.0)
agir	0.31	0.73	0.0	0.17	41	52		5 ('7', 42.31)
ajouter	0.244	0.7	0.0	0.18	40	50		4 ('1', 40.0)
alimenter	0.66	1.0	0.3	0.14	40	50		3 ('2', 62.0)
coûter	0.33	0.7	0.0	0.14	37	47		4 ('1', 31.91)
relancer	0.69	1.0	0.4	0.13	40	50		3 ('1', 60.0)
préférer	0.59	0.9	0.2	0.15	40	50		3 ('1', 68.0)
appliquer	0.46	0.8	0.0	0.18	40	50		3 ('3', 56.0)
apporter	0.361	0.7	0.1	0.15	40	50		5 ('1', 44.0)
fonder	0.36	0.82	0.0	0.15	42	53		6 ('4', 58.49)
appuyer	0.721	1.0	0.1	0.22	39	49		4 ('2', 69.39)
changer	0.13	0.6	0.0	0.12	37	47		5 ('1', 36.17)
chuter	0.75375	1.0	0.375	0.14	29	37		2 ('4', 72.97)
soutenir	0.27	0.73	0.0	0.17	43	54		6 ('2', 68.52)
concevoir	0.58	1.0	0.1	0.22	40	50		3 ('2', 92.0)
interroger	0.36	0.9	0.0	0.22	40	50		3 ('1', 72.0)
confirmer	0.44	0.9	0.0	0.18	40	50		4 ('2', 38.0)
manifester	0.62	1.0	0.2	0.16	39	49		3 ('1', 55.10)
interpeller	0.81	1.0	0.4	0.13	40	50		2 ('5', 96.0)
signer	0.294	1.0	0.0	0.35	40	50		5 ('1', 80.0)
rester	0.39	0.7	0.0	0.16	40	50		7 ('1', 52.0)
tuer	0.48	0.8	0.2	0.16	40	50		2 ('1', 70.0)
indiquer	0.791	1.0	0.2	0.17	40	50		3 ('2', 84.0)
conduire	0.52	0.91	0.18	0.15	41	52		6 ('2', 34.62)
situer	0.93	1.0	0.5	0.10	40	50		2 ('1', 94.0)
aider	0.39	0.7	0.0	0.14	40	50		5 ('2', 34.0)
poursuivre	0.33	0.7	0.0	0.19	40	50		5 ('4', 56.0)
profiter	0.634	1.0	0.2	0.19	40	50		3 ('2', 86.0)
détenir	0.80	1.0	0.54	0.12	42	53		2 ('1', 54.72)
lire	0.21	0.8	0.0	0.18	40	50		5 ('1', 70.0)
contenir	0.64	0.9	0.2	0.14	39	49		5 ('2', 42.86)
dominer	0.53	0.82	0.18	0.18	42	53		4 ('6', 73.59)
noter	0.6025	1.0	0.0	0.24	14	18		3 ('3', 88.89)
dater	0.92	1.0	0.6	0.09	39	49		2 ('1', 85.71)
adopter	0.16	0.8	0.0	0.18	40	50		5 ('4', 64.0)
enregistrer	0.39	0.8	0.0	0.15	36	46		4 ('3', 76.09)
intervenir	0.475	0.9	0.1	0.19	40	50		3 ('4', 56.0)
conclure	0.44	0.9	0.1	0.15	39	49		4 ('1a', 46.94)
disputer	0.56	0.91	0.0	0.26	42	53		3 ('5', 94.34)
estimer	0.69	1.0	0.18	0.23	42	53		2 ('3', 73.58)
appartenir	0.34	0.7	0.0	0.16	40	50		4 ('6', 54.0)
arriver	0.30	0.69	0.0	0.15	52	65		6 ('1', 58.46)
chercher	0.37	0.8	0.0	0.16	40	50		4 ('4', 60.0)
composer	0.69	1.0	0.09	0.27	42	53		4 ('1', 83.02)
confier	0.53	1.0	0.1	0.14	49	50		2 ('2', 50.0)

FIGURE 4 – Résultats de 64 mots par SVM supervisé

lemma / baseline	window	max	min	mean	ecart type
aboutir / 82.0	4	90.0%	50.0%	76.0%	0.1356
investir / 68.0	4	100.0%	60.0%	76.0%	0.1497
traduire / 52.0	4	100.0%	70.0%	88.0%	0.1077
témoigner / 90.0	4	100.0%	80.0%	93.0%	0.064
juger / 77.5	4	90.0%	60.0%	72.0%	0.098
justifier / 88.0	4	90.0%	70.0%	76.0%	0.0663
viser / 56.0	4	100.0%	70.0%	90.0%	0.1
prononcer / 32.69	4	75.0%	41.67%	63.33%	0.119
accomplir / 80.0	4	86.67%	53.33%	70.67%	0.1041
convenir / 32.69	4	58.33%	25.0%	46.67%	0.1067
acquérir / 67.92	4	84.62%	69.23%	76.92%	0.0487
achever / 62.0	4	100.0%	70.0%	88.0%	0.0872
observer / 68.0	4	80.0%	60.0%	70.0%	0.0775
adapter / 42.0	4	90.0%	60.0%	70.0%	0.1183
admettre / 58.33	4	80.0%	40.0%	63.0%	0.11
entraîner / 80.0	4	90.0%	60.0%	72.0%	0.098
payer / 56.6	4	69.23%	30.77%	56.15%	0.0976
respecter / 83.67	4	90.0%	70.0%	84.0%	0.0663
affecter / 58.0	4	80.0%	50.0%	61.0%	0.1044
demeurer / 73.91	4	70.0%	40.0%	58.0%	0.098
aggraver / 82.0	4	100.0%	80.0%	93.0%	0.064
agir / 42.31	4	83.33%	50.0%	69.17%	0.0917
ajouter / 40.0	4	60.0%	30.0%	50.0%	0.1
alimenter / 62.0	4	90.0%	40.0%	73.0%	0.1345
coûter / 31.91	4	70.0%	30.0%	47.0%	0.1418
relancer / 60.0	4	90.0%	50.0%	65.0%	0.1025
préférer / 68.0	4	80.0%	40.0%	55.0%	0.1204
appliquer / 56.0	4	70.0%	10.0%	50.0%	0.1844
apporter / 44.0	4	70.0%	10.0%	38.0%	0.1661
fonder / 58.49	4	92.31%	53.85%	70.77%	0.1077
appuyer / 69.39	4	100.0%	40.0%	79.0%	0.17
changer / 36.17	4	60.0%	20.0%	42.0%	0.1249
chuter / 72.97	4	100.0%	77.78%	88.89%	0.0703
soutenir / 68.52	4	92.86%	50.0%	75.0%	0.1437
concevoir / 92.0	4	100.0%	80.0%	94.0%	0.08
interroger / 72.0	4	100.0%	60.0%	80.0%	0.1414
confirmer / 38.0	4	90.0%	40.0%	60.0%	0.1265
manifester / 55.1	4	90.0%	50.0%	65.0%	0.1432
interpeller / 96.0	4	100.0%	90.0%	95.0%	0.05
signer / 80.0	4	100.0%	70.0%	83.0%	0.1187
rester / 52.0	4	90.0%	50.0%	70.0%	0.1095
tuer / 70.0	4	70.0%	40.0%	52.0%	0.098
indiquer / 84.0	4	100.0%	70.0%	82.0%	0.098
conduire / 34.62	4	83.33%	41.67%	58.33%	0.1443
situer / 94.0	4	100.0%	80.0%	97.0%	0.064
aider / 34.0	4	80.0%	30.0%	52.0%	0.1327
poursuivre / 56.0	4	90.0%	60.0%	72.0%	0.1077
profiter / 86.0	4	100.0%	70.0%	86.0%	0.08
détenir / 54.72	4	92.31%	61.54%	82.31%	0.0976
lire / 70.0	4	90.0%	80.0%	82.0%	0.04
dominer / 73.58	4	100.0%	76.92%	86.15%	0.0671
noter / 88.89	4	100.0%	75.0%	87.5%	0.125
dater / 85.71	4	100.0%	100.0%	100.0%	0.0
adopter / 64.0	4	100.0%	50.0%	75.0%	0.1432
enregistrer / 76.09	4	90.0%	60.0%	78.0%	0.098
intervenir / 56.0	4	80.0%	50.0%	66.0%	0.1114
conclure / 46.94	4	90.0%	60.0%	78.0%	0.098
disputer / 94.34	4	100.0%	84.62%	93.85%	0.0576
estimer / 73.58	4	100.0%	84.62%	95.38%	0.0615
appartenir / 54.0	4	90.0%	50.0%	65.0%	0.1285
arriver / 58.46	4	80.95%	47.62%	61.9%	0.1065
chercher / 60.0	4	80.0%	40.0%	60.0%	0.1095
composer / 83.02	4	100.0%	69.23%	86.92%	0.0846
confier / 50.0	4	100.0%	50.0%	75.0%	0.1204