Modèles neuronaux joints de désambiguïsation lexicale et de traduction automatique

Loïc Vial

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Loïc VIAL

Thèse dirigée par **Benjamin Lecouteux**
et codirigée par **Didier Schwab**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG), équipe GETALP**
et de **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (ED MSTII)**

Modèles neuronaux joints de désambiguïsation lexicale et de traduction automatique

Thèse soutenue publiquement le **20 juillet 2020**
devant le jury composé de :

M. Frédéric Béchet
Professeur des universités, Université Aix-Marseille, examinateur
Président du jury

M. Laurent Besacier
Professeur des universités, Université Grenoble Alpes, examinateur

M. Pierre Zweigenbaum
Directeur de recherche, CNRS Île-de-France Gif-sur-Yvette, rapporteur

M. Mathieu Lafourcade
Maître de conférences HDR, Université de Montpellier, rapporteur

M. Didier Schwab
Maître de conférences, Université Grenoble Alpes, examinateur

M. Benjamin Lecouteux
Maître de conférences, Université Grenoble Alpes, directeur de thèse

Résumé

La désambiguïsation lexicale (DL) et la traduction automatique (TA) sont deux tâches centrales parmi les plus anciennes du traitement automatique des langues (TAL). Bien qu'ayant une origine commune, la DL ayant été conçue initialement comme un problème fondamental à résoudre pour la TA, les deux tâches ont par la suite évolué très indépendamment. En effet, d'un côté, la TA a su s'affranchir d'une désambiguïsation explicite des termes grâce à des modèles statistiques et neuronaux entraînés sur de grandes quantités de corpus parallèles, et de l'autre, la DL, qui est confrontée à certaines limitations comme le manque de ressources unifiées et un champ d'application encore restreint, reste un défi majeur pour permettre une meilleure compréhension de la langue en général.

Aujourd'hui, dans un contexte où les méthodes à base de réseaux de neurones et les représentations vectorielles des mots prennent de plus en plus d'ampleur dans la recherche en TAL, les nouvelles architectures neuronales et les nouveaux modèles de langue pré-entraînés offrent non seulement de nouvelles possibilités pour développer des systèmes de DL et de TA plus performants, mais aussi une opportunité de réunir les deux tâches à travers des modèles neuronaux joints, permettant de faciliter l'étude de leurs interactions.

Dans cette thèse, nos contributions porteront dans un premier temps sur l'amélioration des systèmes de DL, par l'unification des données nécessaires à leur mise en œuvre, la conception de nouvelles architectures neuronales et le développement d'approches originales pour l'amélioration de la couverture et des performances de ces systèmes. Ensuite, nous développerons et comparerons différentes approches pour l'intégration de nos systèmes de DL état de l'art et des modèles de langue, dans des systèmes de TA, pour l'amélioration générale de leur performance. Enfin, nous présenterons une nouvelle architecture pour l'apprentissage d'un modèle neuronal joint pour la DL et la TA, s'appuyant sur nos meilleurs systèmes neuronaux pour l'une et l'autre tâche.

Mots-clés : désambiguïsation lexicale, traduction automatique, réseaux de neurones, modèles joints

Abstract

Word Sense Disambiguation (WSD) and Machine Translation (MT) are two central and among the oldest tasks of Natural Language Processing (NLP). Although they share a common origin, WSD being initially conceived as a fundamental problem to be solved for MT, the two tasks have subsequently evolved very independently of each other. Indeed, on the one hand, MT has been able to overcome the explicit disambiguation of terms thanks to statistical and neural models trained on large amounts of parallel corpora, and on the other hand, WSD, which faces some limitations such as the lack of unified resources and a restricted scope of applications, remains a major challenge to allow a better understanding of the language in general.

Today, in a context in which neural networks and word embeddings are becoming more and more important in NLP research, the recent neural architectures and the new pre-trained language models offer not only some new possibilities for developing more efficient WSD and MT systems, but also an opportunity to bring the two tasks together through joint neural models, which facilitate the study of their interactions.

In this thesis, our contributions will initially focus on the improvement of WSD systems by unifying the ressources that are necessary for their implementation, constructing new neural architectures and developing original approaches to improve the coverage and the performance of these systems. Then, we will develop and compare different approaches for the integration of our state of the art WSD systems and language models into MT systems for the overall improvement of their performance. Finally, we will present a new architecture that allows to train a joint model for both WSD and MT, based on our best neural systems.

Keywords : Word Sense Disambiguation, Machine Translation, Neural Networks, Joint Models

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé dans la réalisation de cette thèse. En premier lieu, je remercie mes directeurs de thèse, Benjamin Lecouteux et Didier Schwab, pour m'avoir apporté leurs connaissances scientifiques ainsi qu'un suivi constant, tout en m'ayant offert des conditions de travail idéales permettant de m'épanouir pleinement et librement dans ma recherche.

Je remercie spécialement Benjamin pour m'avoir toujours poussé vers le haut en m'apportant de nouvelles idées, un point de vue différent et une approche scientifique complémentaire à la mienne.

Je remercie aussi tout particulièrement Didier pour m'avoir transmis non seulement ses connaissances, mais aussi son attrait pour la recherche et les nombreux aspects du TAL, et pour m'avoir chaleureusement accompagné depuis mon premier stage de master, à la fois sur le plan professionnel et sur le plan amical.

Je remercie aussi l'ensemble de l'équipe GETALP du laboratoire d'informatique de Grenoble, les membres permanents comme les non-permanents, pour la qualité du cadre et de l'ambiance de travail qu'ils m'ont offerts.

Merci à tous ceux qui m'ont aidé à relire et à corriger ce manuscrit : mes parents, Véronique et Jean-Rémy, Clémentine, Béatrice et Pierre-Claude, Elliot et Jérémy.

Enfin un grand merci à ma famille, à mes amis et bien sûr à Clémentine pour m'avoir soutenu pendant ces quatre dernières années.

Table des matières

Liste des tableaux

Liste des figures

Introduction générale

La traduction automatique (TA), qui vise à traduire un texte depuis une langue source vers une langue cible, et la désambiguïsation lexicale (DL), qui consiste à reconnaître les sens des mots employés en fonction de leur contexte, sont deux des tâches les plus anciennes et les plus importantes de la recherche en traitement automatique des langues (TAL).

En effet, d'un côté la TA est théorisée dans les années 1950, puis elle est poussée par une volonté des États-Unis de faciliter la traduction de textes russes en anglais, dans le contexte de la guerre froide (Hutchins, 1986). Et de l'autre côté, le problème de polysémie des mots, qui est au cœur de la recherche en désambiguïsation lexicale, est vu par certains pionniers de la traduction automatique, par exemple Weaver (1955), comme une tâche fondamentale à résoudre pour la TA. Dans les années suivantes, ce problème d'ambiguïté des mots freine énormément la recherche en TA, et la désambiguïsation lexicale devient rapidement un champ de recherche à part entière du TAL (Ide et Véronis, 1998; Navigli, 2009).

Aujourd'hui, plus de 60 ans plus tard, de nombreux systèmes de TA populaires commettent toujours des erreurs dans la traduction de phrases ambigües simples : Google Translate [1] par exemple, traduisait jusqu'à récemment la phrase « *Je mange un avocat.* » par « *I eat a lawyer.* » [2] au lieu de « *I eat an avocado.* », mésinterprétant le sens du mot *avocat*. Cet exemple permet de se rendre compte que ce genre d'erreur, en apparence simple à résoudre, est toujours présent dans les systèmes de TA actuels.

Si j'ai décidé de travailler sur cette thématique de recherche, c'est notamment après avoir découvert, lors de mon premier stage de master, la désambiguïsation lexicale, à travers un exemple simple : comment faire comprendre à une machine la différence entre une souris, l'animal, et une souris d'ordinateur ? Durant mon second stage, j'ai associé la traduction automatique comme une application concrète

1. https://translate.google.com
2. Erreur encore présente en juin 2018.

sur laquelle pratiquer la désambiguïsation lexicale, et pour laquelle il me semblait fondamental de résoudre ce genre de problème. Ensuite, il me semblait évident d'aller plus loin et d'étudier les interactions entre les deux tâches, et c'est à travers les modèles neuronaux, tout juste émergents dans le TAL, que j'ai axé ma recherche.

Cette thèse est ainsi motivée par la volonté de contribuer à la résolution du problème d'ambiguïté dans la TA, mais surtout plus généralement de contribuer à une meilleure intégration de la désambiguïsation lexicale dans le TAL. C'est pourquoi, après avoir étudié l'état de l'art de la TA, de la DL et des nouvelles approches neuronales pour le TAL, nous apporterons de multiples contributions permettant l'amélioration des systèmes état de l'art de la DL, ainsi que de nouvelles méthodes d'intégration d'informations sémantiques dans les systèmes de TA neuronaux. Enfin, nous proposerons une nouvelle architecture neuronale jointe permettant de travailler à la fois sur la TA et sur la DL.

Dans les débuts de la recherche en TA, les systèmes reposaient sur un ensemble de règles et des dictionnaires statiques créés manuellement. Depuis les années 1990 cependant, et notamment grâce aux travaux de Brown et al. (1990), un groupe de chercheurs chez IBM, les systèmes actuels sont maintenant fondés sur des modèles statistiques couplés à des méthodes d'apprentissage automatique qui s'appuient sur des ensembles de textes déjà traduits par des traducteurs humains, qu'on appelle « corpus parallèles ».

Dans les premiers systèmes de TA statistiques, cet apprentissage s'effectuait sur quelques dizaines de milliers de phrases parallèles provenant de textes issus du parlement canadien (Brown et al., 1990, 1991). Aujourd'hui, les systèmes les plus performants sont entraînés sur plusieurs centaines de millions de phrases parallèles issues de sources diverses : textes du parlement européen, extraction automatique depuis le Web, commentaires d'actualité, etc. (Ng et al., 2019).

Ainsi, dans le paradigme actuel, concernant la mauvaise traduction de certaines phrases ambigües, les systèmes de TA peuvent toujours se tromper de sens dans la traduction d'un mot lorsque les corpus parallèles disponibles ne couvrent pas tous les sens de ce mot, ou que le sens attendu n'est pas assez représenté.

Les approches statistiques de TA contournent généralement ce genre de problème en apportant toujours plus de données parallèles. D'autres approches ont cherché à intégrer directement les prédictions d'un système de désambiguïsation lexicale dans des systèmes statistiques, notamment dans la thèse de Carpuat (2008). Cependant, l'autrice ne constate que très peu d'améliorations et ne conclut pas à un intérêt clair à désambiguïser explicitement les termes pour améliorer la traduction des systèmes de TA.

De son côté, la recherche en DL, en raison du manque d'inventaire de sens standard et de ressources disponibles, ne connaît un réel essor qu'à partir des années 1980-1990, avec le développement de ressources lexicales numériques, et notamment le Princeton WordNet (Miller et al., 1990). Aujourd'hui, la grande majorité des études en DL s'articule autour de WordNet, qui devient ainsi l'inventaire de sens standard *de facto* dans la recherche (Navigli, 2009).

Si en TA, les approches qui s'appuient principalement sur des ressources linguistiques ont été aujourd'hui largement écartées au profit des approches statistiques et neuronales, en DL, au contraire, deux grandes catégories d'approches continuent encore de se développer en parallèle : les approches supervisées et les approches à base de connaissances.

Les approches à base de connaissances s'appuient sur des bases de données lexicales et sémantiques comme WordNet, et des mesures à base de graphes ou de similarité sémantique. Les approches supervisées sont, elles, à l'instar des approches statistiques en TA, fondées sur l'apprentissage automatique à partir d'exemples créés manuellement, que l'on appelle « corpus annotés en sens ».

En TA comme en DL, les modèles statistiques classiques ont été, ces dernières années, écartés au profit des réseaux de neurones artificiels, une catégorie de systèmes fondés sur la notion de « neurones », des ensembles de fonctions mathématiques simples, qui, une fois combinées, permettent la résolution de problèmes plus complexes.

En effet, en traitement automatique des langues, les réseaux de neurones ont été popularisés par plusieurs travaux majeurs, notamment, d'une part, les vecteurs Word2Vec de Mikolov et al. (2013), qui sont une manière de représenter les mots d'une langue sous forme de vecteurs, et d'autre part, les premières architectures neuronales qui accomplirent des résultats état de l'art dans diverses tâches, par exemple en traduction automatique avec le modèle « séquence à séquence » de Sutskever et al. (2014) et le modèle d'attention de Bahdanau et al. (2015).

Aujourd'hui, les vecteurs contextualisés, ou modèles de langue tels que BERT (Devlin et al., 2019) sont les successeurs de ces représentations vectorielles issues des réseaux de neurones, et permettent d'avoir une représentation plus précise des mots en prenant en compte leur contexte.

Avec ces derniers modèles de langue, une certaine logique d'unification des tâches, outils et modèles neuronaux pour le TAL se dessine. En effet, les auteurs de BERT proposent et suggèrent, à travers leur outil en ligne [3] d'utiliser leur modèle pour un grand nombre de tâches, soit directement, soit en combinaison à d'autres

3. https://github.com/google-research/bert

réseaux de neurones spécifiques, par apprentissage fin (*fine-tuning*).

On peut aussi retrouver cette volonté d'unification à travers des tâches d'évaluation comme GLUE (Wang et al., 2018), qui permettent d'évaluer un même modèle de langue sur de nombreuses tâches de compréhension de la langue à la fois : similarité sémantique, question-réponse, paraphrase, etc.

Pour finir, des bibliothèques comme transformers [4] permettent, en quelques lignes de code, de télécharger et d'utiliser plus de 400 modèles de langue différents [5] pour faire entre autres de l'analyse de sentiments, du résumé automatique, de la classification de texte, et même de la traduction automatique.

Dans ce contexte, nous faisons d'abord le constat que, bien qu'il puisse sembler intuitif que la désambiguïsation explicite des termes soit essentielle afin de traduire correctement, en pratique, l'intérêt n'est aujourd'hui pas clairement prouvé. Il en résulte que la DL n'a aujourd'hui pas vraiment d'application directe, et que dans l'ensemble, la recherche dans ce domaine suscite moins d'intérêts que la TA. [6]

Dans cette thèse, nous partons ainsi de l'hypothèse que les systèmes de DL ont d'abord besoin d'être plus performants afin de réellement constater un effet positif, une fois associés à d'autres tâches.

Cependant, la recherche en DL est confrontée à de nombreux obstacles. Par exemple, l'étalon du sens le plus courant, c'est-à-dire les performances d'un système de DL qui sélectionnerait toujours et donc naïvement le sens le plus fréquent de tous les mots, obtient déjà une précision de l'ordre de 70% à 80% de bonnes réponses, et cette base de référence était, jusqu'aux débuts des travaux de cette thèse, très difficile à dépasser.

Ensuite, concernant les données annotées en sens, en anglais, langue qui comporte la plus grande quantité de données disponibles, le plus gros corpus annoté manuellement en sens WordNet disponible contient seulement 37 176 phrases et couvre seulement environ 16% des sens de WordNet. Dans d'autres langues, la situation est pire, car même s'il existe de nombreux inventaires de sens reliés ou construits autour de WordNet [7], la quantité de données disponibles est bien moins grande et la recherche est encore plus compliquée (voir par exemple Hadj Salah (2018)).

4. https://github.com/huggingface/transformers
5. https://huggingface.co/models
6. Par exemple, en date du 11/04/20, une recherche sur ACL Anthology (https://www.aclweb.org/anthology/) avec les mots-clés *machine translation* renvoie 50 000 résultats, et une recherche avec les mots-clés *word sense disambiguation* renvoie 15 000 résultats.
7. http://globalwordnet.org/resources/wordnets-in-the-world/

Enfin, nous constatons un manque d'homogénéité dans la création et la distribution des corpus annotés en sens. Ces ressources sont pourtant fondamentales pour la création de systèmes de DL supervisés performants, mais chaque corpus possède son propre format, parfois complexe et difficile à analyser, ce qui ralentit d'autant plus la recherche.

Nous observons ainsi une marge de progression importante pour la recherche en DL, notamment grâce aux nouvelles approches fondées sur les architectures neuronales et à des ressources lexicales qui ne sont pas encore assez exploitées, et qui peuvent permettre de confronter ces limitations. Pour ces dernières, il peut s'agir de corpus annotés en sens, mais aussi d'informations sémantiques contenues dans WordNet. Dans cette thèse, nous apportons plusieurs contributions concernant la DL : modernisation d'une approche à base de connaissances, unification des corpus annotés en sens, nouvelle architecture neuronale, et enfin une méthode originale pour l'amélioration de la couverture et des performances des systèmes supervisés.

Ensuite, nous proposerons de nouvelles méthodes d'intégration de sens dans les systèmes de TA neuronaux de l'état de l'art, en nous appuyant sur les dernières avancées en termes d'architectures neuronales, les tout récents modèles de langue et nos systèmes de DL état de l'art. Nous mesurerons ainsi de nouvelles configurations couplées à des systèmes de DL plus performants pour améliorer les performances générales des systèmes de TA.

Finalement, du fait de notre attachement à l'unification et à la simplification des tâches de TAL, nos contributions spécifiques à la DL auront aussi pour objectif de développer des systèmes neuronaux état de l'art combinant les avantages de la dernière architecture Transformer et du modèle de langue BERT, aussi proches que possible des systèmes état de l'art en TA. Nous proposons ainsi comme dernier élément d'étude de cette thèse, un unique modèle neuronal joint, permettant de traduire et de désambiguïser du texte en même temps. Nous analyserons ainsi les capacités de ce modèle joint, en termes de performances sur l'une et l'autre tâche.

Plan du manuscrit

Ce manuscrit de thèse est séparé en deux parties : (1) l'état de l'art détaillé de la désambiguïsation lexicale et de la traduction automatique neuronale, puis (2) nos contributions dans ces deux domaines. Nous résumons chacune d'elles ci-dessous.

Première partie - État de l'art

Chapitre 1.

Ce chapitre consiste en l'état de l'art de la désambiguïsation lexicale. Nous parlons de l'historique et des enjeux de la DL, puis nous détaillons les ressources nécessaires ainsi que les différentes approches qui sont mises en œuvre. Enfin, nous décrivons les différentes méthodes d'évaluation de la DL.

Chapitre 2.

Ce chapitre consiste en l'état de l'art de la traduction automatique neuronale. Nous présentons d'abord l'historique et le fonctionnement des systèmes de TA statistique et le basculement vers les méthodes neuronales. Puis nous décrivons les principales architectures neuronales de TA, ainsi que les méthodes pour la gestion du vocabulaire. Nous décrivons ensuite les ressources nécessaires à la TA neuronale puis les méthodes d'évaluation des systèmes. Enfin, nous consacrons une section aux travaux mêlant la TA et la DL.

Deuxième partie - Contributions

Chapitre 3.

Ce chapitre décrit nos premiers travaux effectués au cours de cette thèse, sur une nouvelle méthode de création de vecteurs de sens qui exploite et compare différents vecteurs de mot tels que Word2Vec (Mikolov et al., 2013) et GloVe (Pennington et al., 2014) pour l'amélioration d'une méthode de DL à base de connaissances, fondée sur l'algorithme de Lesk (Lesk, 1986). Ces travaux sont dans la continuité des précédents travaux de master (Vial, 2016) et ont donné lieu à deux articles de conférence : Vial et al. (2017c) (en français) et Vial et al. (2017a) (en anglais). Nos vecteurs de sens en résultant sont disponibles librement. [8]

8. https://github.com/getalp/WSD-IWCS2017-Vialetal

Chapitre 4.

Ce chapitre décrit notre ressource UFSAC (*Unification of Sense Annotated Corpora*). Nous réalisons dans un premier temps l'inventaire des corpus annotés en sens WordNet existants et nous évoquons la problématique du manque d'uniformisation des formats et des inventaires de sens dans ces corpus, qui freine la recherche dans les systèmes de DL supervisés. Nos travaux visent ainsi à regrouper dans une même ressource et sous un même format tous ces corpus. Nous fournissons aussi des outils pour la création et la manipulation des corpus UFSAC. Ces travaux ont donné lieu à deux articles de conférence : Vial et al. (2017b) (en français) et Vial et al. (2018b) (en anglais). Nos corpus et outils sont disponibles librement. [9]

Chapitre 5.

Ce chapitre décrit nos travaux autour des systèmes de DL supervisés neuronaux. Nous proposons une analyse de l'ensemble des corpus UFSAC pour l'entraînement de systèmes supervisés, ainsi qu'une nouvelle architecture neuronale s'appuyant sur l'architecture Transformer (Vaswani et al., 2017) ainsi que BERT (Devlin et al., 2019). Nos expériences comparent ainsi différents corpus d'entraînement issus de notre ressource UFSAC ainsi que différents paramètres pour notre architecture. Nous obtenons ainsi des résultats état de l'art sur toutes les tâches de DL. Ces travaux ont donné lieu à un article de conférence : Vial et al. (2018c) (en français) et un article de journal : Vial et al. (2019b) (en français). Nos systèmes de DL neuronaux sont disponibles librement. [10]

Chapitre 6.

Ce chapitre décrit nos travaux autour d'une nouvelle méthode que l'on nomme « compression de vocabulaire de sens », qui tire parti des relations sémantiques de WordNet (hyperonymie, méronymie...) pour augmenter la couverture et les performances des systèmes de DL supervisés. Nous appliquons cette méthode à notre meilleur système neuronal décrit dans le chapitre précédent, et nous obtenons de nouveaux résultats état de l'art. Ces travaux ont donné lieu à deux articles de conférence : Vial et al. (2019c) (en français) et Vial et al. (2019a) (en anglais), et nous avons obtenu le prix du meilleur article à la conférence francophone.

9. https://github.com/getalp/UFSAC
10. https://github.com/getalp/disambiguate

Chapitre 7.

Ce chapitre présente nos travaux sur l'intégration d'informations sémantiques à des systèmes de TA neuronaux. Nous proposons trois méthodes d'intégration de ces informations : sous forme d'étiquettes de sens WordNet ajoutées en entrée du système, en ajoutant directement les représentations de BERT, et par apprentissage guidé avec les sens WordNet décodés conjointement avec les mots en sortie du système. Dans nos expériences, nous comparons ces méthodes entre elles et en les combinant. Nous montrons ainsi une amélioration des performances d'un système de TA neuronal fondé sur l'architecture Transformer. Nous avons ainsi participé à la tâche de traduction anglais-tchèque de la campagne d'évaluation IWSLT 2019 (Vial et al., 2019d) avec notre système utilisant BERT en entrée.

Chapitre 8.

Enfin, ce chapitre décrit nos travaux sur la création d'un modèle neuronal joint de TA et de DL. En effet, grâce aux travaux présentés précédemment, notre système état de l'art en DL ainsi que notre meilleur système de TA reposent tous les deux sur l'architecture Transformer et le modèle de langue BERT en entrée du système. Nous étudions ainsi les capacités d'un modèle neuronal unique réalisant les deux tâches à la fois. Dans nos expériences, nous réalisons ainsi un apprentissage joint sur des corpus à la fois annotés en sens et bilingues, et nous comparons ainsi l'apport de notre architecture dans chacune des deux tâches, grâce à un premier apprentissage général et à un deuxième apprentissage fin de nos modèles. Notre système permettant la DL et la TA, séparément ou conjointement, est ainsi disponible librement. [11]

11. https://github.com/getalp/disambiguate-translate

Première partie

État de l’art

Chapitre 1

Désambiguïsation lexicale

1.1 Introduction

Dans la communication écrite ou parlée, les mots que l'on emploie peuvent avoir plusieurs sens, qui sont déduits généralement de leur contexte. Par exemple, dans la phrase « *La souris mange le fromage.* », on pense tout de suite, pour le mot *souris*, au sens du petit rongeur, plutôt que celui du périphérique informatique.

En traitement automatique des langues (TAL), savoir quels sont les sens véhiculés par les mots qui sont manipulés peut s'avérer crucial afin de construire des systèmes plus justes et pertinents. Par exemple, on attend d'un système de traduction automatique français-anglais qu'il traduise correctement le mot *avocat* dans la phrase « *Je mange un avocat.* » par *avocado* et non pas par *lawyer*.

La désambiguïsation lexicale (DL), en anglais *Word Sense Disambiguation* (WSD), est ainsi une tâche centrale du TAL qui vise à résoudre cette ambiguïté en assignant aux mots contenus dans un texte leur sens le plus probable, à partir d'un inventaire de sens prédéfini.

Dans ce chapitre, nous allons d'abord faire un rapide historique de la tâche et décrire les enjeux, avant de voir quelles sont les différentes approches pour la DL ainsi que les ressources nécessaires à leur mise en œuvre. Nous décrirons ainsi les principaux systèmes de l'état de l'art, et enfin les méthodes pour leur évaluation.

1.1.1 Historique

La désambiguïsation lexicale, comme elle concerne une notion fondamentale qui est le sens, est un élément central dans de nombreux champs du TAL et pour la compréhension de la langue en général. Comme l'écrit Navigli (2009), elle apparaît d'abord comme une des tâches principales pour la traduction automatique dès les années 1950 (Weaver, 1955), et elle a ensuite été considérée très rapidement comme un problème à part entière au vu de sa grande difficulté.

En effet, même si la DL peut être vue comme un simple problème de classification (tel sens est assigné à tel mot dans tel contexte), elle soulève en fait de nombreuses questions fondamentales qu'il est nécessaire de se poser. Par exemple : comment définir précisément un sens ? Est-ce que l'existence d'un sens dépend d'une langue ou d'un domaine d'application ? Quel niveau de granularité est souhaité pour distinguer des sens proches ? etc.

Dans les premiers travaux sur la DL, cette difficulté était accrue par le manque de ressource sémantique disponible. Sans inventaire de sens standard, base de données lexicale ou campagne d'évaluation, la comparaison des systèmes n'était pas évidente, et cette situation est d'ailleurs toujours d'actualité pour la majorité des langues autres que l'anglais (voir par exemple les travaux de Hadj Salah et al. (2018) sur la DL de l'arabe).

C'est dans les années 1990 qu'arrivent les premieres bases de données lexicales, et notamment WordNet (Miller et al., 1990), l'inventaire de sens pour l'anglais de référence, toujours utilisé de nos jours dans la plupart des travaux de la DL. Dans la même période est créé le premier corpus annoté en sens à grande échelle, le SemCor (Miller et al., 1993), puis les premières campagnes d'évaluation comme SensEval (Kilgarriff, 1998), qui permettent finalement la création et l'évaluation de systèmes de DL de façon automatique et à grande échelle.

Il existe aujourd'hui une multitude d'approches pour la DL, qu'on différencie principalement par la nature et par la quantité des données sur lesquelles elles s'appuient. On trouve ainsi principalement d'un côté les approches supervisées, et de l'autre, les approches à base de connaissances. Les approches supervisées reposent sur de grandes quantités de données manuellement annotées, couplées à des méthodes d'apprentissage automatique, afin de prédire le sens d'un mot en fonction des exemples observés. Les approches à base de connaissances, elles, s'appuient sur des ressources lexicales telles que des dictionnaires, des thésaurus ou des réseaux lexicaux, couplées à des méthodes telles que des calculs de similarité ou de parcours de graphe.

1.1.2 Enjeux et applications

La DL a des applications qui semblent évidentes, par exemple en traduction automatique (TA) français-anglais, où les systèmes doivent être capables de distinguer les deux sens principaux du mot *avocat* en fonction du contexte pour le traduire correctement. Pour la recherche d'information (RI), comprendre la véritable intention de l'utilisateur lorsqu'il cherche par exemple l'adresse d'un avocat, ou lorsqu'il cherche une recette de cuisine avec de l'avocat, semble aussi fondamental pour obtenir des résultats pertinents.

Pourtant en pratique, force est de constater que la désambiguïsation explicite des termes n'est quasiment jamais employée dans les systèmes état de l'art de TA ou de RI. Il y a bien sûr eu des travaux qui ont intégré ces informations à des systèmes de RI ou de TA, mais l'apport réel de cette information sémantique n'est cependant pas toujours évident.

En effet, on trouve d'un côté des travaux comme l'article de Sanderson (1994), dans lequel l'auteur montre que la DL n'est pas utile aux systèmes de RI, et qu'elle dégrade même leurs performances si la précision du système de DL utilisé est en dessous du seuil de 90%. De même, Carpuat et Wu (2005) intègrent des étiquettes de sens dans un système de TA statistique et ne constatent aucun effet significatif. Dans l'étude de Resnik (2006), l'auteur reconnait ainsi qu'il n'y a pas d'exemple clair d'applications concrètes dans lesquelles la DL aurait prouvé son utilité.

Cependant, on trouve aussi d'autres travaux comme l'article de Chan et al. (2007a), qui montre cette fois une amélioration d'un système de TA grâce aux prédictions d'un système de DL. De même, dans l'article de Zhong et Ng (2012), leur méthode d'intégration de la DL dans un système de RI améliore significativement ses résultats. Plus récemment, les travaux de Rios et al. (2017), Liu et al. (2018) et Hadj Salah (2018) ont mis en avant les difficultés pour un système de traduction automatique neuronale de faire un choix lorsqu'il est confronté à un mot polysémique, et ils ont intégré avec succès les prédictions d'un système de DL pour améliorer ses performances.

Finalement, nous pensons que l'apport de la DL à d'autres tâches du TAL dépend de plusieurs critères, notamment :

— de la méthode d'intégration des sens pour la tâche, qui doit permettre aux sens d'agir comme une information additionnelle aux mots, plutôt qu'en remplacement de ces mots ;

— des performances du système de DL utilisé, qui ont énormément évolué ces dernières années (voir section 1.5.1.3) ;

— de la granularité des sens, qui dépend de l'inventaire de sens utilisé ;

— de la quantité de données exploitées par le système de base, en effet l'impact de la DL sur un système de TA exploitant déjà des milliers de documents sera forcément moins important que sur un système de TA n'ayant accès qu'à peu de ressources ;

— de l'utilisation ou non d'autres méthodes de désambiguïsation plus implicites, comme des vecteurs de mot et les modèles de langue pré-entraînés.

1.2 Inventaires de sens

En désambiguïsation lexicale, toutes les méthodes ont besoin, à minima, d'un inventaire de sens qui liste les différents sens des mots de la langue cible. [1]

Cet inventaire de sens peut être, par exemple, un dictionnaire classique tel que le *Petit Larousse* ou le *Robert*. Mais ce sont plus souvent des bases de données lexicales comme le Princeton WordNet, BabelNet ou Wikipedia qui sont utilisées en pratique, car elles sont à la fois facilement accessibles en ligne, et elles offrent, en plus de l'inventaire de sens, une vaste quantité d'informations qui peuvent être utiles à certaines méthodes de DL.

L'inventaire de sens est donc la première ressource nécessaire à la mise en œuvre d'un système de DL. Son choix est fondamental car non seulement il a un impact sur les autres ressources sur lesquelles vont pouvoir s'appuyer le système de DL, mais en plus il peut être difficile de comparer les performances de deux systèmes de DL s'appuyant sur deux inventaires de sens différents.

Par exemple, le nom *souris* a cinq sens dans le dictionnaire Larousse en ligne [2] : (1) le mammifère rongeur (souris grise), (2) les autres rongeurs similaires à la souris grise, (3) une jeune femme, (4) la partie du gigot de mouton et (5) le dispositif informatique. Le Wiktionary, lui, liste dix sens [3] : en plus des sens (3), (4) et (5) du Larousse, les sens (1) et (2) ont été regroupés, et on retrouve d'autres sens moins communs comme par exemple le cartilage des naseaux des chevaux, ou encore une espèce de papillon de nuit.

On voit ainsi que non seulement un inventaire de sens n'est pas forcément aussi complet qu'un autre, mais en plus ils peuvent différer par leur granularité. En effet,

1. Dans cette thèse, nous faisons la distinction entre la désambiguïsation lexicale, qui nécessite un inventaire de sens, et l'induction de sens, qui consiste à induire les différents sens des mots en observant simplement leur usage, de manière totalement non supervisée.

2. https://www.larousse.fr/dictionnaires/francais/souris/73746. Consulté le 29/04/19.

3. https://fr.wiktionary.org/wiki/souris. Consulté le 29/04/19.

on parle de granularité fine lorsque l'inventaire de sens distingue plus finement les sens. C'est le cas dans l'exemple précédent, où, bien que le Larousse contienne moins de sens pour le nom *souris*, il distingue plus finement les sens (1) et (2) que le Wiktionary qui les a groupés ensemble dans une même entrée sémantique.

En pratique, depuis la construction du Princeton WordNet (Miller et al., 1990), une base de données lexicale développée à l'université de Princeton, la grande majorité des travaux sur la DL de l'anglais utilise cet inventaire de sens, et ce sont souvent des inventaires de sens reliés à WordNet qui sont utilisés pour d'autres langues (par exemple l'Arabic WordNet (Black et al., 2006) pour l'arabe, ou BabelNet (Navigli et Ponzetto, 2010) qui est multilingue). En effet, le Princeton WordNet a la particularité de posséder un riche réseau lexical et sémantique reliant tous les mots et les sens du dictionnaire entre eux. De plus, un ensemble de logiciels permettant d'interroger sa base de données est aussi fourni, depuis sa première version jusqu'à aujourd'hui.

1.3 Ressources

Selon l'approche employée, diverses ressources sont nécessaires pour la DL. Les deux grandes catégories de ressources utilisées sont les corpus annotés en sens, indispensables pour les approches supervisées, et les bases de données lexicales, matière première des approches à base de connaissances. On trouve aussi des ressources plus génériques comme des corpus bruts (non annotés) ou des vecteurs de mot en complément dans certaines approches.

1.3.1 Bases de données lexicales

On parle généralement de base de données lexicale pour toute source de connaissances structurée qui apporte des informations sur les mots et les sens d'une ou plusieurs langues et qui est accessible par logiciel. On retrouve parmi ces bases de données des dictionnaires, des thésaurus, des ontologies, des graphes de connaissances, etc. Dans cette section, nous présentons deux principales bases de données lexicales qui sont utilisées en DL : WordNet et BabelNet.

1.3.1.1 WordNet

Le Princeton WordNet (Miller et al., 1990), souvent appelé simplement WordNet, est une base de données lexicale créée au laboratoire des sciences cognitives de l'université de Princeton, qui a la particularité de regrouper les différents sens des mots de la langue anglaise en ensembles de synonymes appelés *synsets* (contraction de *synonym sets*). Ainsi, pour le nom *mouse* qui possède par exemple quatre sens, son deuxième sens (un *mouse* au sens d'un œil au beurre noir) partage le même *synset* que le premier sens du nom *shiner* et le premier sens du groupe nominal *black eye*.

WordNet repose sur des principes psycholinguistiques, qui ont pour conséquence une granularité très fine dans la distinction des sens. Par exemple, le premier et le deuxième sens du mot *snow* se distinguent par le fait que le premier (*precipitation falling from clouds in the form of ice crystals*) se réfère à la neige qui tombe du ciel, tandis que le second (*a layer of snowflakes (white crystals of frozen water) covering the ground*) se réfère à la neige qui couvre le sol. Les mots peuvent ainsi avoir un nombre de sens très grand, allant jusqu'à 59 sens différents pour le verbe *break*.

La richesse de cette base lexicale, en plus de sa granularité fine, vient aussi de son vaste réseau sémantique. En effet, les *synsets* sont reliés entre eux par des relations sémantiques telles que l'antonymie, l'hyperonymie, la méronymie, etc.

Le réseau sémantique de WordNet classe les mots en quatre parties du discours : les noms, les verbes, les adjectifs et les adverbes. Les principales relations sémantiques possibles entre les sens sont :

— la synonymie, qui regroupe les sens en *synsets* ;

— l'antonymie, qui définit deux sens opposés (par exemple les adjectifs *petit* et *grand*) ;

— l'hyperonymie et l'hyponymie, qui définissent respectivement la généralisation et la spécialisation d'un sens (par exemple *voiture* est un hyponyme de *véhicule*, et à l'inverse *animal* est un hyperonyme de *chat*) ;

— la méronymie et l'holonymie, qui définissent la « partie de » (par exemple *main* est à la fois un méronyme de *bras* et un holonyme de *doigt*).

La figure 1.1 illustre ces relations entre quelques *synsets*.

WordNet s'est rapidement imposé comme un standard *de facto* en DL de l'anglais, si bien que son inventaire de sens est utilisé pour l'annotation de la grande majorité des corpus anglais, tant pour l'apprentissage que pour l'évaluation des systèmes. Il est aussi au cœur de nombreuses approches à base de connaissances.

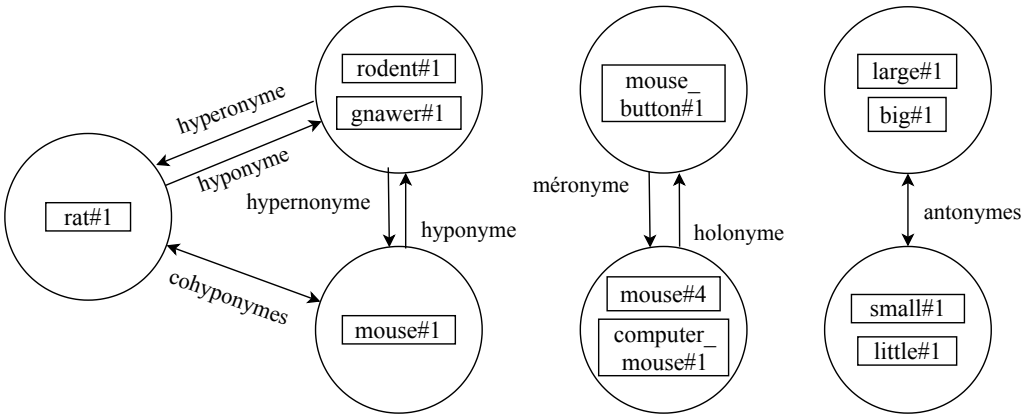


FIGURE 1.1 – Relations sémantiques entre différents *synsets* de WordNet.

La dernière version du Princeton WordNet, la version 3.1, est sortie en 2013. Elle contient ainsi 155 287 mots et 206 941 sens répartis en 117 659 *synsets* [4]. Tous les mots communs de la langue anglaise y sont présents, et on retrouve aussi certaines entités nommées connues (Obama, Einstein, etc.).

Il est aussi à noter qu'en 2019, le projet Open English WordNet, aussi simplement appelé English Wordnet, voit le jour. Ce projet se situe dans la continuité du Princeton WordNet et il est soutenu par la Global WordNet Association. Il marque cependant une rupture avec ce dernier car il suit une approche plus collaborative dans laquelle n'importe qui peut contribuer en proposant des modifications [5].

Dans les autres langues que l'anglais, de nombreuses bases de données lexicales reproduisent la structure et la méthode de conception du Princeton WordNet, si bien qu'ils font partie d'un ensemble de « WordNets » dont la liste est maintenue par la Global WordNet Association [6]. On peut par exemple citer le WOLF (WOrdNet Libre du Français) pour le français, l'Arabic WordNet pour l'arabe, etc. Cependant, la grande majorité des travaux et des ressources pour la désambiguïsation lexicale concerne uniquement la langue anglaise (en particulier les corpus annotés en sens et les campagnes d'évaluation), c'est pourquoi ces autres bases de données sont bien moins connues et utilisées que le Princeton WordNet.

4. https://wordnet.princeton.edu/documentation/wnstats7wn
5. https://github.com/globalwordnet/english-wordnet
6. http://globalwordnet.org/resources/wordnets-in-the-world/

1.3.1.2 BabelNet

BabelNet (Navigli et Ponzetto, 2010) est une base de données lexicale multilingue, créée à l'université La Sapienza de Rome, qui repose sur le concept de *Babel synsets*. Un *Babel synset* représente un concept et regroupe des entrées lexicales provenant de multiples bases de données lexicales telles que Wikipedia, Wiktionary, WordNet, etc.

Cette ressource est, à la différence de WordNet, générée de façon automatique, à l'aide d'un algorithme qui aligne les différents inventaires de sens en s'appuyant sur différentes informations propres à la base de données cible (titres et liens entre les pages Wikipedia, relations sémantiques entre les *synsets* de WordNet, etc.). Le résultat est un vaste réseau sémantique mis à jour continuellement. Il couvre, dans sa version 4.0, 284 langues et près de 16 millions de *Babel synsets* [7].

On retrouve des utilisations notables de BabelNet dans certains travaux comme l'algorithme de Babelfy (Moro et al., 2014) qui exploite le graphe de BabelNet pour permettre de désambiguïser du texte dans n'importe quelle langue présente dans la base de données. Les campagnes d'évaluation SemEval 2013 (Navigli et al., 2013) et SemEval 2015 (Moro et Navigli, 2015) ont aussi consacré une tâche de désambiguïsation lexicale multilingue utilisant l'inventaire de sens de BabelNet.

1.3.2 Corpus annotés en sens

Selon Habert et al. (1998), un corpus est « une collection de données langagières qui sont sélectionnées et organisées selon des critères linguistiques explicites pour servir d'échantillon du langage ». Un corpus regroupe généralement plusieurs documents qui peuvent contenir plusieurs millions de mots, lesquels peuvent être annotés avec plusieurs informations (lemmes, parties du discours, sens, etc.).

Parmi les corpus « bruts », c'est-à-dire sans annotation, on trouve par exemple le *Brown Corpus* (Francis et Kučera, 1964) (un million de mots), le *British National Corpus* (Burnard, 1998) (100 millions de mots) ou encore l'*American National Corpus* (Ide et Macleod, 2001) (20 millions de mots). Les textes proviennent de diverses sources comme des journaux, des livres, des encyclopédies ou du Web.

Un corpus annoté en sens est un corpus dans lequel tous les mots ou seulement une partie d'entre eux sont annotés avec une étiquette de sens issue d'un inventaire de sens particulier. Par exemple, le corpus de la tâche 7 de SemEval 2007 (Navigli et al., 2007) est annoté avec des identifiants de sens de WordNet 2.1 tandis que le

7. https://babelnet.org/stats. Consulté le 24/12/19.

corpus anglais de la tâche 13 de SemEval 2015 (Moro et Navigli, 2015) est annoté avec des identifiants de sens de trois bases lexicales : WordNet 3.0, BabelNet 2.5.1 et Wikipedia.

Comme nous l'évoquons dans un de nos articles (Vial et al., 2018b), on peut trouver au moins trois raisons de vouloir créer un corpus annoté en sens :

1. Estimer la distribution des sens. C'est à cette fin qu'a été annoté le SemCor par exemple. Les sens de WordNet sont principalement ordonnés en fonction de la répartition des sens dans ce corpus depuis la version 1.7.
2. Construire un système de désambiguïsation lexicale à partir des exemples contenus dans le corpus annoté. C'est le cas du DSO, de l'OMSTI et du Train-O-Matic par exemple.
3. Évaluer des systèmes de désambiguïsation lexicale, en comparant les annotations produites par un système avec une référence. C'est le cas, cette fois, des corpus créés dans le cadre des campagnes d'évaluation SensEval/SemEval.

Les corpus annotés en sens sont nombreux, et ils sont au cœur d'une de nos contributions (voir chapitre 4). Pour cette raison, nous présentons dans cette section tous les principaux corpus annotés en sens utilisés en DL dont nous ferons usage dans cette contribution.

1.3.2.1 SemCor

Le SemCor (Miller et al., 1993), contraction de *Semantic Concordance* est un corpus construit par la même équipe que celle qui a développé WordNet. Il est constitué de 352 documents provenant du *Brown Corpus* (Francis et Kučera, 1964), dans lesquels tous les mots ont été annotés en sens avec l'inventaire de WordNet 1.6.

Ce corpus a été développé initialement avec l'idée d'améliorer WordNet en lui donnant des exemples d'utilisation pour la plupart des mots de son lexique et en permettant d'estimer la distribution des sens dans un texte représentatif de l'anglais américain, afin de les ordonner dans le dictionnaire. C'est grâce à ce travail que depuis la version 1.7 de WordNet, et jusqu'à la dernière à ce jour, l'ordre des sens suit la plupart du temps l'ordre décroissant de fréquence d'apparition du sens dans le SemCor. Le premier sens de chaque mot se voulant être le sens le plus utilisé pour ce mot.

Cependant, même si le SemCor a une assez bonne couverture, seuls 43,3% des mots polysémiques de WordNet et 28,6% des sens de mots polysémiques y

sont présents, certains mots et sens restent donc absents du corpus. Par exemple, la souris en tant que dispositif de pointage a été présentée pour la première fois en 1968, il ne peut donc pas y avoir de mention de ce sens dans le *Brown Corpus* qui date de 1964.

Il en résulte que pour plus de la moitié des mots de WordNet, l'ordre des sens ne suit pas leur fréquence d'utilisation dans le SemCor. Pour ces mots-là, les concepteurs de WordNet les ont arbitrairement classés.

Aujourd'hui, le SemCor est indéniablement le corpus le plus utilisé en tant que données d'entraînement pour les systèmes supervisés. Il est en effet la source principale de données annotées des meilleurs systèmes depuis la création des premières campagnes d'évaluation (Hoste et al., 2001; Decadt et al., 2004) jusqu'aux systèmes les plus récents (Yuan et al., 2016; Huang et al., 2019; Vial et al., 2019a).

1.3.2.2 DSO

Le corpus DSO (Ng et Lee, 1996), qui tient son nom du groupe de recherche *Defence Science Organisation* est un corpus non disponible librement constitué de phrases du *Brown Corpus* ainsi que d'articles du *Wall Street Journal* dans lequel un grand nombre d'instances de seulement 121 noms et 70 verbes différents ont été annotés en sens avec la version 1.5 de WordNet.

Ces mots ont été choisis parmi les plus fréquents et les plus ambigus de l'anglais, afin de fournir des exemples d'utilisation pour améliorer les systèmes supervisés dans les cas les plus difficiles à distinguer. On retrouve ainsi par exemple 1 500 instances des noms *problem* (3 sens différents dans WordNet 3.1) et *way* (12 sens), et des verbes *take* (42 sens) et *come* (21 sens) dans différents contextes. Au final, ce sont plus de 200 000 instances de mots qui sont annotées.

Ce corpus est notamment utilisé en plus du SemCor dans le système NUS-PT (Chan et al., 2007b), le meilleur système de la campagne d'évaluation SemEval 2007 (Navigli et al., 2007). Cependant, son utilisation est assez peu présente dans d'autres travaux, car non seulement l'inventaire de sens utilisé est ancien, mais en plus le corpus est distribué de manière payante [8].

8. https://catalog.ldc.upenn.edu/LDC97T12

1.3.2.3 Princeton WordNet Gloss Corpus

En 1999, motivé par la volonté d'enrichir WordNet en ajoutant des annotations de lemmes, de parties du discours et de sens aux mots des définitions, le projet eXtended WordNet voit le jour à l'université de Dallas au Texas (Harabagiu et al., 1999). Le projet s'appuie sur une combinaison de multiples méthodes de désambiguïsation et a pour objectif d'annoter tous les mots des définitions de WordNet avec une précision de plus de 90% (Mihalcea et Moldovan, 2001). Il se poursuit jusqu'en 2004 pour aboutir à la version appelée *eXtended WordNet 2.0* [9] atteignant finalement une précision estimée à 86% (Moldovan et Novischi, 2004).

En 2006, à l'occasion de la sortie de WordNet 3.0, le contenu du projet *eXtended WordNet* est officiellement intégré à la base de données lexicale, les annotations sont manuellement vérifiées, et le corpus des définitions de WordNet annotés en sens est maintenant distribué sous le nom de Princeton WordNet Gloss Corpus [10].

Le corpus est finalement très peu utilisé dans l'état de l'art, mais il permet à nos systèmes récents (Vial et al., 2019a) d'obtenir les meilleurs résultats de l'état de l'art, en utilisant ces définitions annotées comme des exemples pour un apprentissage supervisé.

1.3.2.4 OMSTI

Le corpus OMSTI *(One Million Sense-Tagged Instances)* développé par Taghipour et Ng (2015b) est un corpus d'environ un million de phrases, annotées semi-automatiquement grâce à une méthode d'alignement appliquée à des corpus parallèles anglais-chinois. La méthode employée est la suivante :

— D'abord, les auteurs assignent manuellement une traduction chinoise pour chacun des sens de WordNet 1.7.1, pour les 60% de mots les plus fréquents dans le corpus Brown.

— Ensuite, un aligneur de mots (GIZA++) est utilisé sur une grande quantité de textes parallèles anglais-chinois provenant du corpus MultiUN (Eisele et Chen, 2010), ce qui permet d'assigner une ou plusieurs traductions chinoises possibles pour un terme anglais donné.

— Enfin, pour tous les cas où l'aligneur a determiné qu'un mot anglais était traduit par un mot chinois correspondant à une des traductions manuelles, alors ce mot est annoté avec le sens correspondant.

9. http://www.hlt.utdallas.edu/~xwn/downloads.html
10. http://wordnet.princeton.edu/glosstag.shtml

À sa sortie, l'OMSTI était le plus grand corpus annoté en sens disponible publiquement. Il a plus tard été converti avec des sens de WordNet 3.0, puis a été utilisé en tant que corpus d'apprentissage dans plusieurs travaux notables comme ceux de Iacobacci et al. (2016) et Yuan et al. (2016).

1.3.2.5 MASC

Le MASC (*Manually Annotated Sub-Corpus*) développé par Ide et al. (2008) est un sous-corpus de l'OANC (*Open American National Corpus*) contenant environ 500 000 mots provenants de sources diverses (transcription de débats oraux, courriels, blogs, etc.), et annoté manuellement avec de nombreuses informations linguistiques (parties du discours, sens, entités nommées, co-références, etc.).

En ce qui concerne les annotations en sens, 114 mots polysémiques ont été choisis et toutes leurs occurrences ont été annotées avec les sens de WordNet 3.0.

La première utilisation du MASC dans les travaux sur la DL, à notre connaissance, se trouve dans les travaux de Yuan et al. (2016). Cependant, les auteurs ont entièrement ré-annoté l'ensemble du MASC à travers une plateforme de *crowdsourcing* en utilisant un autre inventaire de sens, le NOAD, et en fournissant une correspondance avec les sens de WordNet 3.0.

1.3.2.6 Ontonotes

Ontonotes *Release* 5.0, qu'on appelera simplement Ontonotes (Hovy et al., 2006), est la version finale du projet Ontonotes qui a réuni plusieurs groupes de recherche américains dans le but d'annoter avec des informations structurelles et sémantiques une grande quantité de textes provenant de sources diverses (écrites et orales) dans trois langues : l'anglais, le chinois et l'arabe.

La ressource comporte plus d'un million de mots pour sa partie anglaise, et la méthode suivie pour les annotations vise à assurer un accord inter-annotateurs supérieur à 90%. L'inventaire de sens utilisé par les annotateurs est propre à Ontonotes. Il se fonde sur WordNet 3.0, mais afin de simplifier la tâche d'annotation et de s'assurer d'un bon accord inter-annotateurs, les sens dont les distinctions sont très fines ont été regroupés. Enfin, une correspondance entre les deux inventaires de sens est fournie.

1.3.2.7 Train-O-Matic

Le corpus Train-O-Matic (Pasini et Navigli, 2017) est issu d'une volonté de créer des données annotées en sens pour l'apprentissage de systèmes de DL supervisés de façon complètement automatique, et sans besoin de corpus parallèles.

La méthode de désambiguïsation automatique repose sur une méthode d'extraction automatique de phrases depuis des corpus monolingues, et sur une approche à base de connaissances exploitant le réseau sémantique de BabelNet pour trouver les exemples les plus fiables pour la plupart des sens de WordNet. Les auteurs fournissent ainsi un corpus issu des données de Wikipedia et des corpus des Nations unies (Ziemski et al., 2016).

Le corpus est ensuite utilisé comme données d'apprentissage dans un système supervisé (IMS), et il est comparé au SemCor et à l'OMSTI utilisés dans les mêmes conditions. Les résultats sont nettement inférieurs en utilisant le Train-O-Matic plutôt que le SemCor, mais ils sont comparables à ceux utilisant l'OMSTI.

1.3.2.8 Corpus des campagnes d'évaluation

Au cours des campagnes d'évaluation SensEval puis SemEval, plusieurs corpus ont été manuellement annotés en sens afin de comparer les systèmes de désambiguïsation lexicale. Ces corpus sont composés de peu de textes et dépassent rarement les 5 000 mots.

Dans la première campagne d'évaluation SensEval (Kilgarriff, 1998), plus de 100 instances de 60 mots différents ont été annotées avec un inventaire de sens nommé Hector, non lié à WordNet.

Par la suite, les campagnes d'évaluation SensEval 2 (Edmonds et Cotton, 2001) puis SensEval 3 (Snyder et Palmer, 2004) ont utilisé respectivement WordNet 1.7 et WordNet 1.7.1 pour annoter plusieurs ensembles de corpus, tout en introduisant la distinction entre les tâches de désambiguïsation « tous mots », dans lesquelles tous les mots ayant une entrée dans l'inventaire de sens sont annotés, et les tâches « échantillon lexical », dans lesquelles seules des instances d'un même mot ou d'un petit ensemble de mots doivent être annotées.

Depuis 2007, SensEval a été renommé en SemEval (Navigli et al., 2007), c'est lors de cette campagne que la première tâche de désambiguïsation « gros grain » a vu le jour, dans laquelle plusieurs sens proches sont acceptés pour une même instance d'un mot. L'inventaire de sens utilisé pour cette tâche ainsi que pour la tâche « grain fin » est WordNet 2.1.

Enfin, à l'occasion des éditions 2013 et 2015 (Navigli et al., 2013; Moro et Navigli, 2015), des corpus annotés avec l'inventaire de sens de BabelNet en plus de WordNet 3.0 ont été créés, afin de proposer des tâches de désambiguïsation multilingues. Il est à noter cependant que seuls les noms sont à désambiguïser dans la tâche de DL de SemEval 2013.

1.3.2.9 Synthèse, unification et analyse des corpus

Comme on peut le voir, il existe de nombreux corpus annotés en sens. Cependant, parce qu'ils ont été créés avec des intentions différentes et à des années différentes, les inventaires de sens utilisés dans chacun d'eux sont très variables, de même que leur format de distribution qui est souvent propre à chacun. En conséquence, certains corpus sont individuellement difficiles à exploiter, et ce manque d'homogénéité freine aussi leur utilisation et leur étude à plus grande échelle.

Toutes ces problématiques nous ont conduit à réaliser une de nos contributions, la ressource UFSAC, qui vise l'unification de tous les corpus annotés en sens présentés précédemment, à la fois dans un format unique mais aussi dans un inventaire de sens unique. Plus de détails se trouvent dans le chapitre 4. Dans le tableau 1.1, nous récapitulons ainsi les statistiques principales de ces corpus (nombre de phrases, nombre de mots annotés, etc.) une fois convertis au format UFSAC.

Pour finir, nous avons aussi mené une analyse de ces corpus dans le but de leur exploitation pour l'apprentissage d'un système supervisé qui se trouve dans le chapitre 5.

1.3.3 Ressources génériques

En plus des bases de données lexicales et des corpus annotés en sens, d'autres ressources plus génériques sont parfois exploitées dans les systèmes de DL. On retrouve notamment :

— des corpus parallèles (bilingues) dans les travaux de Chan et al. (2007b) et Taghipour et Ng (2015a), où les auteurs assignent une traduction chinoise aux différents sens de certains mots anglais, puis utilisent un grand corpus parallèle anglais-chinois et un aligneur de mots afin de « désambiguïser » l'anglais grâce aux traductions (ce qui a conduit au développement de l'OMSTI) ;

— des corpus bruts (c'est-à-dire non annotés), qui peuvent servir à étendre des corpus annotés en sens dans les approches telles que Yuan et al. (2016) ou Vial et al. (2019b) ;

Corpus	Année de sortie	Inventaire de sens original	Nombre de phrases	Nombre de mots	Nombre de mots annotés en sens
SemCor	1993	WordNet 1.6	37 176	778 587	229 533
DSO	1997	WordNet 1.5	101 004	2 705 190	176 197
WNGC	2007	WordNet 3.0	117 659	1 634 691	496 776
OMSTI	2015	WordNet 3.0	863 648	36 636 675	1 109 147
MASC	2008	WordNet 3.0	31 759	585 353	113 518
Ontonotes	2013	Ontonotes	124 852	2 475 987	233 616
Train-O-Matic	2017	WordNet 3.0	788 888	31 708 188	834 455
SensEval 2 (t.m.)	2001	WordNet 1.7	252	5 766	2 282
SensEval 2 (é.l., entr.)	2001	WordNet 1.7	8 611	251 772	8 451
SensEval 2 (é.l., éval.)	2001	WordNet 1.7	4 328	125 477	4 239
SensEval 3 (t.m.)	2004	WordNet 1.7.1	352	5 541	1 850
SensEval 3 (é.l., entr.)	2004	WordNet 1.7.1	7 860	241 565	7 819
SensEval 3 (é.l., éval.)	2004	WordNet 1.7.1	3 944	122 131	3 849
SemEval 2007 - 07	2007	WordNet 2.1	245	5 637	2 261
SemEval 2007 - 17	2007	WordNet 2.1	135	3 201	455
SemEval 2013	2013	BabelNet 1.1.1	306	8 391	1 644
SemEval 2015	2015	BabelNet 2.5	138	2 604	1 022

TABLE 1.1 – Statistiques générales des corpus annotés en sens. « é.l. » correspond aux tâches « échantillon lexical », « t.m. » correspond aux tâches « tous mots », « entr. » correspond aux corpus d'« entraînement » et « éval. » aux corpus d'« évaluation ».

— des vecteurs de mot (p. ex. Word2Vec, GloVe...) ou des modèles de langue (p. ex. ELMo, BERT...) pré-entraînés, d'abord employés pour améliorer certaines approches pré-neuronales (Chen et al., 2014; Taghipour et Ng, 2015a; Iacobacci et al., 2016; Vial et al., 2017c,a) puis finalement très largement utilisés dans les systèmes neuronaux actuels (Luo et al., 2018a,b; Peters et al., 2018; Loureiro et Jorge, 2019; Vial et al., 2018c, 2019b,c).

Dans la section suivante, nous allons détailler un peu plus cette dernière catégorie de ressources, car elles prennent de plus en plus d'ampleur dans les approches récentes de DL.

1.3.3.1 Vecteurs de mot

On peut retracer les méthodes de représentations vectorielles de mots dès la fin des années 1960 avec le modèle vectoriel (Salton, 1968) appliqué à la recherche de documents. Plusieurs approches ont ensuite vu le jour, comme celles fondées sur l'hypothèse distributionnelle (Harris et al., 1989) ou bien celles fondées sur l'hypothèse componentielle (Schwab, 2005).

En 2013, Mikolov et al. (2013) présentent une nouvelle méthode de création de vecteurs de mot nommée Word2Vec, et ces vecteurs se démarquent notamment pour leur adaptibilité à un grand nombre de tâches. En effet, ils donnent de bons résultats :

— dans la tâche de similarité de mots, dans laquelle la simple mesure cosinus entre deux vecteurs corrèle fortement avec les scores attribués par des humains (p. ex. *vache* et *lait* sont plus proches que *vache* et *Minitel*) ;

— dans la tâche d'analogie de mots, dans laquelle on doit trouver un mot X qui répond à une analogie du type « A est à B ce que C est à X », là aussi les opérations « + » et « - » s'appliquent de façon très intuitive aux vecteurs ($X = B - A + C$) ;

— utilisés en tant que trait vectoriel dans un classifieur (linéaire, réseau de neurones...), grâce à leur représentation dense qui permet à la fois d'éviter l'utilisation de vecteurs *one hot* pour représenter les mots et de se passer parfois de l'étape de sélection de traits linguistiques particuliers (parties du discours, lemmes, etc.).

De plus, Word2Vec se démarque aussi par sa facilité d'utilisation. En effet, les auteurs distribuent sur leur site Web [11] différents modèles de vecteurs pré-entraînés, le code et des tutoriels pour appliquer les vecteurs, les reproduire ou

11. https://code.google.com/archive/p/word2vec/

les entraîner sur de nouvelles données. Deux formats de fichiers sont aussi créés, un format textuel et un format binaire.

Par la suite, d'autres modèles de vecteurs de mot ont vu le jour. On peut citer les vecteurs de Levy et Goldberg (2014) qui reposent sur les dépendances syntaxiques entre les mots, les vecteurs de Baroni et al. (2014), qui fournissent des vecteurs fondés sur le comptage en plus des vecteurs fondés sur les prédictions, et les vecteurs de Pennington et al. (2014) (GloVe), qui s'appuient sur une matrice de co-occurrences.

Pour finir, Bojanowski et al. (2017) proposent FastText, une méthode permettant d'avoir un vocabulaire ouvert, en générant de nouveaux vecteurs de mot à partir de n-grammes de caractères. De plus, les auteurs distribuent sur leur page Web [12] des vecteurs dans 157 langues, entraînés sur les corpus Common Crawl et Wikipedia.

Ces vecteurs ont ainsi été exploités à de nombreuses reprises dans divers travaux de DL (Chen et al., 2014; Taghipour et Ng, 2015a; Iacobacci et al., 2016; Vial et al., 2017c,a), mais, bien qu'ils apportent clairement une information lexicale et sémantique utile, l'ambiguïté des mots est toujours « encodée » dans ces représentations vectorielles. C'est ainsi que le vecteur du mot *souris* dans FastText se retrouve à la fois proche de *mulot*, *rat*, *curseur* et *molette*.

Cette ambiguïté est une des raisons qui a poussé à la création des vecteurs « contextuels », dont la représentation dépend du contexte, comme nous allons le voir dans la section suivante.

1.3.3.2 Vecteurs contextuels et modèles de langue

McCann et al. (2017) proposent une méthode d'apprentissage par transfert, dans laquelle un modèle neuronal, constitué d'un encodeur et d'un décodeur est d'abord pré-entraîné sur une tâche de traduction automatique. Ensuite, l'encodeur, lui-même constitué d'une couche de vecteurs de mot et de couches récurrentes, est réutilisé en entrée d'autres modèles accomplissant diverses tâches (classification, question-réponse, etc.).

La couche de vecteurs de mot de l'encodeur est initialisée avec les vecteurs de GloVe, et les auteurs indiquent qu'avec cette méthode, l'encodeur pré-entraîné, qu'ils nomment CoVe (pour *Context Vectors*), permet ainsi de contextualiser les vecteurs, ce qui apporte une meilleure représentation que GloVe pour les tâches sur lesquelles ils sont évalués.

12. https://fasttext.cc/

Par la suite, l'article de Peters et al. (2018) présente ELMo (*Embeddings from Language Models*), une méthode similaire à CoVe, à la principale exception que le modèle n'est plus pré-entraîné sur une tâche de traduction automatique, mais sur un objectif non supervisé de prédiction d'un mot en fonction des mots voisins, qui est appellé modèle de langue.

Plus précisément, les auteurs décrivent leur modèle de langue de la façon suivante. Soit une séquence de N mots $t_1, t_2, \ldots, t_N$, un premier modèle *forward* calcule la probabilité d'un mot t_k en fonction des mots qui le précèdent $t_1, \ldots, t_{k-1}$, puis un modèle *backward* calcule la probabilité de t_k en fonction des mots qui le suivent $t_{k+1}, \ldots, t_N$. Enfin, le modèle de langue complet va chercher à prédire t_k en fonction des informations des modèles *forward* et *backward*.

Les auteurs calculent les vecteurs de mot à partir de convolutions de caractères afin d'avoir un vocabulaire ouvert, puis ils utilisent plusieurs couches de cellules LSTM (Hochreiter et Schmidhuber, 1997) pour les fonctions *forward* et *backward*. ELMo peut ensuite être intégré à des modèles pour d'autres tâches, de la même manière que CoVe.

Le plus grand intérêt d'ELMo par rapport à CoVe se situe au niveau du pré-entraînement. En effet, la tâche de prédiction d'un mot peut se faire de manière complètement non supervisée, avec seulement du texte monolingue, contrairement à la tâche de traduction automatique utilisée par CoVe qui nécessite des corpus bilingues alignés.

BERT et ses dérivés

Devlin et al. (2019) présentent BERT (*Bidirectional Encoder Representations from Transformers*), un modèle qui repose sur le même principe de modèle de langue pré-entraîné sur des données monolingues. À la différence d'ELMo, qui utilise des couches de cellules LSTM, BERT repose sur l'architecture Transformer (Vaswani et al., 2017) (plus de détails dans la section 2.2.4). Les auteurs proposent aussi deux nouveaux objectifs pour le pré-entraînement : le MLM (*Masked Language Model*) qui consiste à masquer le mot à prédire k_t en le remplaçant par un symbole spécial `[MASK]`, et le NSP (*Next Sentence Prediction*) qui consiste à prédire la phrase suivante à partir de la phrase actuelle.

BERT est aussi entraîné sur plus de données qu'ELMo (environ 3,3 milliards de mots contre un milliard pour ELMo), et la gestion du vocabulaire est différente. En effet, les auteurs de BERT ont choisi d'avoir un vocabulaire fixe de 30 000 sous-unités de mots (plus de détails dans la section 2.3.2). Les auteurs rapportent ainsi

des résultats état de l'art sur de nombreuses tâches : compréhension de la langue, question-réponse, etc.

Au cours de l'année 2019, de multiples variations et améliorations de BERT ont été présentées. Les plus notables sont :

— XLM (*Cross-lingual Language Model*) (Lample et Conneau, 2019), une variante qui combine plusieurs nouveaux objectifs en plus du MLM : le CLM et le TLM. Le CLM (*Causal Language Model*) vise à prédire un mot en fonction des mots précédents (ce qui permet notamment la génération de textes comme les modèles GPT et GPT2 d'OpenAI (Radford et al., 2018, 2019)), et le TLM (*Translation Language Model*) vise à prédire des mots masqués dans une langue grâce à leur traduction dans une autre langue ;

— XLNet (Yang et al., 2019), qui s'appuie sur une architecture neuronale différente de BERT, s'affranchit de sa phase de pré-traitement utilisant un segmenteur externe, et qui est entraîné sur plus de 30 milliards de mots (soit 10 fois plus que BERT) ;

— RoBERTa (Liu et al., 2019), qui, à l'instar de XLNet, vise aussi à optimiser BERT sur plusieurs aspects dont la segmentation des données et la quantité de données d'apprentissage.

On peut aussi citer Albert (Lan et al., 2020) et DistilBERT (HuggingFace, 2019), des modèles ayant pour objectif de réduire la taille de BERT (en termes de nombre de paramètres) tout en gardant des performances comparables (voire supérieures pour Albert).

Le point commun notable entre tous ces modèles est qu'ils sont sont tous fondés sur l'architecture Transformer issue de l'article de Vaswani et al. (2017). Des outils disponibles librement comme *transformers* de la société Hugging Face [13] permettent leur utilisation et leur intégration dans un réseau neuronal très facilement.

Les modèles de langue pour la désambiguïsation lexicale

Dans le contexte de la désambiguïsation lexicale, les modèles de langue pré-entraînés sont devenus une ressource centrale utilisée dans de nombreux travaux (Loureiro et Jorge, 2019; Wiedemann et al., 2019; Huang et al., 2019; Kumar et al., 2019; Vial et al., 2019c,a).

En effet, on peut considérer le principe même d'avoir un vecteur différent pour un mot en fonction de son contexte comme une désambiguïsation implicite. Si

13. https://github.com/huggingface/transformers

une étiquette provenant d'un inventaire de sens n'est pas directement attribuée, les vecteurs sont quand même situés à des endroits différents de l'espace, ce qui facilite grandement leur classification par rapport aux vecteurs fixes et ambigus type GloVe.

Les travaux de Coenen et al. (2019) et de Wiedemann et al. (2019) confirment d'ailleurs cette intuition, en permettant la visualisation de vecteurs de BERT pour plusieurs sens d'un même mot dans un espace à deux dimensions. On observe que certains sens forment effectivement des groupes de points proches dans l'espace (voir figure 1.2).

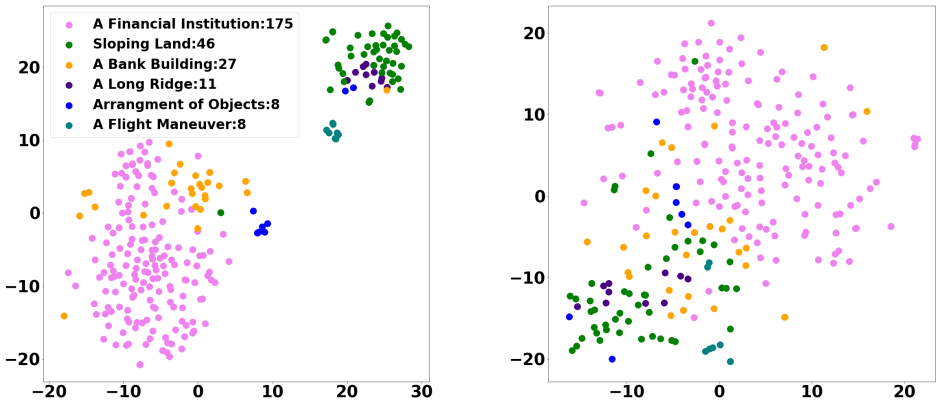


FIGURE 1.2 – Visualisation en deux dimensions des vecteurs de BERT (à gauche) et d'ELMo (à droite) pour un même mot (bank) dans différents sens. Figure issue de l'article de Wiedemann et al. (2019).

1.4 Approches

Les approches pour la désambiguïsation lexicale sont multiples et sont généralement classées en plusieurs catégories en fonction de la nature des ressources utilisées et de leur quantité (Navigli, 2009; Schwab et al., 2013; Raganato et al., 2017b).
On distingue ainsi trois grandes catégories d'approches :

— Les approches à base de connaissances, qui s'appuient principalement sur des ressources telles que des dictionnaires, des réseaux lexicaux ou des graphes sémantiques par exemple. On retrouve ici les approches à base de

mesures de similarité sémantique comme celles exploitant l'algorithme de Lesk (Lesk, 1986; Banerjee et Pedersen, 2002; Vial et al., 2016, 2017d,a,c), ou les mesures exploitant un graphe comme celui de BabelNet (Navigli et Ponzetto, 2010), par exemple Babelfy (Moro et al., 2014).

— Les approches supervisées, qui exploitent un grand nombre d'exemples de mots annotés manuellement ou automatiquement en sens, et qui sont généralement liées à une méthode d'apprentissage automatique telle qu'un classifieur linéaire (Chan et al., 2007b; Zhong et Ng, 2010) ou plus récemment un réseau de neurones récurrents (Yuan et al., 2016; Raganato et al., 2017b; Vial et al., 2018c, 2019a).

— Les approches non supervisées, qui réalisent de l'induction de sens, c'est-à-dire que seuls des textes non annotés sont utilisés, et les différents sens des mots sont induits en fonction de leurs contextes ou de leur traduction (Brody et Lapata, 2008; Yarowsky, 1995).

Bien sûr, les frontières entre ces catégories sont parfois floues. On retrouve par exemple des approches à base de connaissances améliorées à l'aide de corpus annotés en sens (Vial et al., 2016), ou à l'inverse des méthodes supervisées améliorées grâce à des connaissances (Luo et al., 2018a,b; Vial et al., 2019a). Certaines méthodes se définissent comme « semi-supervisées » car elles utilisent d'autres ressources en plus de corpus annotés en sens (Chen et al., 2014; Yuan et al., 2016). On trouve aussi une catégorie de méthodes dites « faiblement supervisées » qui visent l'utilisation d'un minimum de données annotées en sens, etc.

Dans cette section, nous proposons un tour d'horizon de ces différentes approches que nous classons dans les deux grandes catégories que sont les approches supervisées d'une part, et les approches à base de connaissances d'autre part (regroupant les approches à base de similarité et de graphes). Nous clarifions le statut des approches semi-supervisées dans un troisième temps. Enfin, nous proposons dans la figure 1.5 un diagramme récapitulatif de ces différentes approches sous la forme d'un arbre.

On ne parlera cependant pas des approches non supervisées car ce chapitre est consacré à la désambiguïsation lexicale et non à l'induction de sens. Bien que les deux problèmes soient fortement liés, la désambiguïsation lexicale repose sur la connaissance préalable d'un inventaire de sens, ce que les approches non supervisées n'ont pas.

1.4.1 Approches à base de connaissances

Les approches à base de connaissances, comme nous l'avons dit précédemment, s'appuient sur des connaissances explicites telles que des bases de données lexicales, des thésaurus ou des graphes sémantiques par exemple.

D'une manière générale, l'avantage principal des méthodes de cette catégorie est qu'elles offrent une bonne couverture, étant donné que les bases de connaissances utilisées sont souvent directement liées à l'inventaire de sens. De plus, elles sont facilement généralisables d'une langue à une autre, du moment que les connaissances sur lesquelles elles s'appuient sont disponibles dans la langue ciblée.

On peut classer les approches à base de connaissances en deux sous-catégories : les approches à base de similarité sémantique, et les approches à base de graphes.

1.4.1.1 Approches à base de similarité sémantique

Les approches à base de similarité sémantique sont les héritières d'une des premières méthodes pour la désambiguïsation lexicale : l'algorithme de Lesk (Lesk, 1986). Le fonctionnement de ce type d'approche repose sur deux composants [14] :

— Une mesure de similarité sémantique, ou algorithme **local**, qui permet d'attribuer un score de proximité entre deux sens. Par exemple, une bonne mesure donnera un score de proximité vraisemblablement plus élevé entre la paire de sens (fromage/nourriture, souris/animal) qu'entre la paire de sens (fromage/nourriture, souris/ordinateur).

— Une méthode de propagation de l'algorithme local, ou algorithme **global**, qui permet de désambiguïser tous les mots dans un contexte donné (phrase, document...) grâce à la mesure locale. Par exemple, attribuer, pour tous les mots d'une phrase, le sens qui obtient le plus haut score de similarité avec les sens des mots voisins dans une fenêtre de trois mots.

Une de nos contributions s'appuie principalement sur l'algorithme de Lesk (voir chapitre 3), ainsi que plusieurs de nos travaux antérieurs (Vial et al., 2016; Vial, 2016; Vial et al., 2017d). Nous allons ici présenter cet algorithme et sa principale amélioration, l'algorithme de Lesk étendu. Enfin, nous aborderons d'autres méthodes à base de similarité inspirées de l'algorithme de Lesk.

14. Découpage qu'on peut retrouver par exemple dans les travaux de Schwab et al. (2011).

L'algorithme de Lesk

Lesk (1986) présente un des tout premiers algorithmes pour la désambiguïsation lexicale utilisant uniquement un dictionnaire électronique contenant des mots, des sens et leurs définitions.

Pour cela, sa méthode repose sur l'idée générale que le sens d'un mot, pris dans un contexte, partage des caractéristiques communes avec les sens des autres mots de ce même contexte. Ainsi, pour désambiguïser un mot, l'algorithme va chercher parmi tous ses sens possibles et lui attribuer celui qui aura la plus grande similarité avec les sens de ses mots voisins.

Afin de mesurer cette similarité entre deux sens, Lesk propose de simplement compter le nombre de mots en commun dans leur définition. Plus formellement, si nous notons $D(S) = \{w_1, w_2, \ldots, w_n\}$ la définition de S, alors on peut décrire la mesure de similarité de Lesk entre deux sens S_1 et S_2 de la manière suivante :

$$Lesk(S_1, S_2) = |D(S_1) \cap D(S_2)|$$

Pour donner un exemple du fonctionnement de l'algorithme de Lesk, prenons la phrase « *Je pose la fourchette à côté de la cuillère.* » et cherchons à attribuer le sens correct du mot *fourchette*. Dans le Larousse en ligne [15], les sens 1 et 2 des mots *fourchette* et *cuillère* ont les définitions suivantes :

Sens	Définition
fourchette#1	« *Ustensile de table dont le manche se termine par des dents [...]* »
fourchette#2	« *Écart entre deux valeurs, deux possibilités extrêmes [...]* »
cuillère#1	« *Ustensile de table ou de cuisine composé d'un manche [...]* »
cuillère#2	« *Contenu d'une cuillère ; cuillerée.* »

15. https://www.larousse.fr/dictionnaires/francais/. Consulté le 04/06/19.

En calculant la similarité entre toutes les paires de sens possibles, et en excluant les mots vides (le, de, et, à...) [16] on a :

S_1	S_2	$D(S_1) \cap D(S_2)$	$Lesk(S_1, S_2)$
fourchette#1	cuillère#1	{*ustensile*, *table*, *manche*}	3
fourchette#1	cuillère#2	{}	0
fourchette#2	cuillère#1	{}	0
fourchette#2	cuillère#2	{}	0

Le score le plus élevé concerne ainsi la paire de sens (fourchette#1, cuillère#1), on attribuera donc ces sens-là à ces deux mots.

Dans son article, Lesk a essayé trois dictionnaires anglais classiques (Oxford, Collins et Webster). Il constate une précision de l'ordre de 50% à 70% sur quelques exemples qu'il a extraits d'un roman et d'un article de presse.

Lesk étendu et adapté pour WordNet

Quelques temps après la popularisation de WordNet et de la campagne d'évaluation SensEval, Banerjee et Pedersen (2002) proposent une implémentation qui adapte l'algorithme de Lesk à l'inventaire de sens et aux définitions de WordNet. De plus, ils proposent une extension de la mesure de Lesk classique en exploitant les relations sémantiques présentes entre les sens dans la base de données lexicale.

L'extension fonctionne de la manière suivante : en plus de compter le nombre de mots en commun dans les définitions des sens dont on veut calculer la similarité, on compte aussi les mots en commun dans les définitions des sens reliés sémantiquement aux sens d'origine.

Plus formellement, si l'on note $rel(S)$ l'ensemble des sens reliés à S à travers un lien explicite dans WordNet, alors la mesure de Lesk étendue entre les sens S_1 et S_2 notée $ExtLesk(S_1, S_2)$ est la suivante :

$$ExtLesk(S_1, S_2) = \left| \left((D(S_1) \bigcup_{r \in rel(S_1)} D(r) \right) \bigcap \left(D(S_2) \bigcup_{r \in rel(S_2)} D(r) \right) \right|$$

Banerjee et Pedersen (2002) évaluent ensuite leur système sur la tâche d'échantillon lexical de SensEval 2. Avec l'approche de Lesk classique, ils obtiennent une précision de 16% et grâce à leur extension, ils obtiennent une précision de 32%.

16. Lesk (1986) exclut en effet certains mots de ses calculs comme *the* ou *of* sans donner de liste précise, de même dans l'adaptation pour WordNet de Banerjee et Pedersen (2002).

Autres mesures de similarité sémantique

Patwardhan et al. (2003) s'intéressent à la mesure de Lesk étendue et comparent ses performances à d'autres mesures de similarité sémantique. Ils essaient ainsi plusieurs mesures en remplacement de la mesure de Lesk :

— La mesure de Leacock-Chodorow (Leacock et Chodorow, 1998) mesure la similarité entre deux sens de noms en calculant le plus court chemin qui les sépare dans le graphe des relations d'hyperonymie et d'hyponymie de WordNet.

— La mesure de Resnik (Resnik, 1995) attribue une valeur de « quantité d'informations » (*information content*) à chaque sens en estimant la variabilité des contextes dans lesquels il se trouve dans des corpus non annotés en sens. Puis, elle considère que deux sens sont proches s'ils partagent une quantité d'informations proche.

— La mesure de Jiang-Conrath (Jiang et Conrath, 1997) améliore la mesure de Resnik en considérant aussi la longueur du chemin entre les deux sens dans le graphe de WordNet.

— La mesure de Lin (Lin, 1998) est une autre mesure similaire à celle de Jiang-Conrath.

— La mesure de Hirst-St. Onge (Hirst et St-Onge, 1998) utilise non seulement les relations d'hyperonymie et d'hyponymie dans son calcul de proximité, mais aussi toutes les autres offertes par WordNet (antonymie, méronymie, etc.).

Les tests sont menés sur la tâche « échantillon lexical » de SensEval 2, et les résultats montrent que la mesure de Lesk étendue reste la meilleure, suivie de près par la mesure de Jiang-Conrath.

Autres méthodes de propagation de la mesure de similarité

Dans tous les travaux que nous avons vus précédemment (Lesk, 1986; Banerjee et Pedersen, 2002; Patwardhan et al., 2003), le cœur des études est la mesure de similarité sémantique, ou algorithme local. La méthode de propagation de cette mesure, ou algorithme global, est systématiquement la même. Elle consiste à évaluer toutes les combinaisons de sens possibles dans une fenêtre de contexte de cinq à dix mots, et de choisir les sens qui maximisent la mesure de similarité locale.

Dans d'autres travaux conduits en parallèle tels que Cowie et al. (1992) et Gelbukh et al. (2003), d'autres stratégies de propagation sont étudiées. En effet, Gelbukh et al. (2003) constatent que l'utilisation d'une petite fenêtre fixe de mots,

comme dans la méthode de Lesk originale, peut mener à de fortes incohérences au sein d'une même phrase ou d'un document, parce que la désambiguïsation des mots dans une fenêtre de contexte ne prend pas du tout en compte les résultats de la désambiguïsation des mots au-delà de cette fenêtre.

Partant d'un principe démontré par Gale et al. (1992) que le sens d'un même mot varie très rarement au sein d'un même discours, ils proposent alors un algorithme global permettant de prendre en compte l'intégralité d'un document pour désambiguïser un mot. Les auteurs constatent que pour un texte de 500 mots à désambiguïser et ayant chacun en moyenne 3 sens [17], il existe 3^{500} combinaisons de sens possibles, soit environ 4×10^{238} combinaisons. Dans ces conditions, il est impossible de calculer la similarité entre toutes les paires possibles de sens. C'est pourquoi les auteurs implémentent une heuristique à base d'algorithmes génétiques capables d'approcher la meilleure solution sans essayer toutes les possibilités.

Certains travaux explorent d'autres heuristiques pour l'algorithme global. Cowie et al. (1992) implémentent par exemple l'algorithme du recuit simulé et Schwab et al. (2011) implémentent un algorithme à base de colonies de fourmis. Dans une de nos contributions (Vial et al., 2017d) [18], nous comparons les performances de certains de ces différents algorithmes globaux et d'autres, comme l'algorithme des chauve-souris et l'algorithme des coucous. Plus de détails peuvent aussi se trouver dans les travaux de Tchechmedjiev (2012).

Finalement, il est à noter qu'une de nos contributions dans cette thèse porte sur l'amélioration de l'algorithme de Lesk grâce à des vecteurs de mot (voir chapitre 3). De plus, une autre contribution, précédant la thèse, porte sur l'amélioration de l'algorithme de Lesk grâce à des corpus annotés en sens (Vial et al., 2016).

Algorithme de Lesk simplifié

Une autre variante de l'algorithme de Lesk qu'on peut retrouver dans Kilgarriff et Rosenzweig (2000) et dans Vasilescu et al. (2004) consiste à compter le nombre de termes en commun entre la définition d'un sens d'un mot cible et directement les mots du contexte autour de lui. On appelle cet algorithme « algorithme de Lesk simplifié » parce qu'il ne nécessite plus de calculer la mesurer de similarité pour toutes les combinaisons possibles de sens, mais seulement une fois par sens pour

17. Ce qui correspond à une taille de document typique et au nombre de sens moyen dans WordNet pour les mots polysémiques.
18. Article poursuivant mes travaux de master (Vial, 2016).

chaque mot. Le nombre de fois où l'on exécute l'algorithme local passe ainsi de 3^{500} à 3×500 si on reprend l'exemple vu précédemment.

L'algorithme de Lesk simplifié a cependant un désavantage important par rapport à l'algorithme de Lesk original. En effet, dans son utilisation, si plusieurs sens d'un mot cible obtiennent le même score de similarité avec leur contexte, c'est l'ordre des sens dans l'inventaire utilisé qui détermine le sens choisi. Or cet ordre des sens, s'il est effectivement donné dans WordNet, est en partie issu de la fréquence de ces sens dans des corpus annotés en sens. Ainsi, l'algorithme de Lesk simplifié, contrairement à sa version classique, repose fortement sur des données qui ne sont pas disponibles pour n'importe quelle langue ou dans n'importe quel inventaire de sens.

Le Lesk simplifié connaîtra aussi quelques améliorations. Par exemple, Basile et al. (2014) mesurent la similarité grâce à la mesure cosinus entre un vecteur calculé à partir de mots de la définition d'un sens et un vecteur calculé à partir des mots du contexte du mot à désambiguïser. Ils utilisent cependant, en plus de la fréquence d'apparition des sens, de nombreuses autres informations sémantiques issues de BabelNet.

1.4.1.2 Méthodes à base de graphes

Une autre branche de méthodes pour la DL à base de connaissances se compose des méthodes fondées sur les graphes. Ces méthodes cherchent, comme pour les méthodes à base de similarité sémantique, une cohérence globale au niveau du texte, mais cette fois en exploitant des techniques de parcours d'arbres ou de graphes plus générales, qu'on peut retrouver dans d'autres domaines, notamment l'algorithme du PageRank de Brin et Page (1998) et ses dérivés.

Le premier algorithme pour la DL à base de graphes a été proposé par Mihalcea et al. (2004). Dans leur article, l'algorithme du PageRank (Brin et Page, 1998) qui est normalement utilisé pour mesurer la pertinence d'une page Web en fonction des liens pointant depuis/vers la page, est ainsi appliqué au problème de la désambiguïsation lexicale. En effet, en considérant les *synsets* de WordNet comme des nœuds et les relations sémantiques de WordNet comme des liens, on peut calculer un score de PageRank pour chacun des nœuds avec une formule de la forme suivante :

$$S(V_i) = \sum_{j \in In(V_i)} \frac{S(V_j)}{|Out(V_j)|}$$

avec V_i le nœud dont on veut calculer le score, $In(V_i)$ l'ensemble de nœuds pointants vers V_i et $Out(V_j)$ l'ensemble des nœuds dont V_j fait un lien sortant. Dans

l'algorithme de DL en tant que tel, on construit d'abord le sous-graphe de WordNet composé des sens des mots présents dans le texte à désambiguïser. On assigne ensuite initialement un score de 1 à chaque nœud du graphe. Enfin, on exécute le calcul des scores de PageRank pour chaque nœud jusqu'à convergence des scores, et on assigne aux mots du texte les sens qui ont le plus grand score.

De multiples variantes de cet algorithme ont ensuite été proposées, toujours en utilisant WordNet comme graphe de connaissances. Par exemple, on peut citer Agirre et Soroa (2009), Agirre et al. (2014) et les travaux de Moro et al. (2014). Ces derniers présentent notamment Babelfy, un système de DL multilingue à base de graphes utilisant le réseau sémantique de BabelNet.

1.4.2 Approches supervisées

Les approches supervisées sont celles qui s'appuient sur des données annotées en sens comme ressource principale pour leurs méthodes. Dominant généralement les campagnes d'évaluation, les approches supervisées nécessitent cependant une quantité de corpus annotés en sens très coûteuse afin d'obtenir une bonne couverture. De plus, elles sont difficilement applicables à d'autres langues que l'anglais car c'est la langue largement la plus dotée en quantité de textes annotés.

Avec la récente popularité des vecteurs de mot puis des architectures neuronales dans le traitement automatique des langues, les méthodes supervisées ont énormément évolué. C'est pourquoi dans cette section, nous allons passer en revue les principales approches supervisées, en décrivant d'un côté les méthodes pré-neuronales et de l'autre celles à base de réseaux de neurones. De plus, nous allons traiter d'un côté les approches neuronales à classification directe, et de l'autre côté, celles fondées sur la méthode des k plus proches voisins.

1.4.2.1 Approches supervisées pré-neuronales

On peut retrouver les débuts des méthodes supervisées aux travaux de Yarowsky (1992), et de Leacock et al. (1993) par exemple, qui construisent des systèmes statistiques entraînés sur des exemples afin de discriminer automatiquement les sens d'un mot. Dans ces travaux cependant, seule une poignée de mots polysémiques est annotée en sens et l'évaluation des performances de leurs systèmes se fait sur leurs propres exemples, ce qui rend difficile leur comparaison avec d'autres méthodes.

Plus tard, les travaux de Ng et Lee (1996) puis Ng (1997) deviennent les prémices de la construction des systèmes supervisés à grande échelle, en entraînant

un système sur une version préliminaire du corpus DSO (Ng et Lee, 1997) comprenant plus de 192 000 annotations en sens pour 121 noms et 70 verbes différents. Leur système utilise un ensemble de traits linguistiques tels que les parties du discours et l'ensemble non ordonné des mots entourant une instance afin de classifier le sens d'un mot. Les auteurs comparent ensuite les performances de plusieurs classifieurs tels que la méthode des k plus proches voisins et le modèle bayésien naïf.

À la campagne d'évaluation SemEval 2007, le système NUS-PT (Chan et al., 2007b) arrive en tête de la tâche de DL « gros grain » avec un score F1 de 82,50 %. Son fonctionnement, qui est similaire à celui de Lee et Ng (2002) repose sur la classification d'un mot en sens en fonction des ensembles de traits linguistiques suivants :

— Les collocations locales, composées de 11 traits : $C_{-1,-1}$, $C_{1,1}$, $C_{-2,-2}$, $C_{2,2}$, $C_{-2,-1}$, $C_{-1,1}$, $C_{1,2}$, $C_{-3,-1}$, $C_{-2,1}$, $C_{-1,2}$, et $C_{1,3}$, où $C_{i,j}$ correspond à la suite ordonnée de mots dans le contexte local du mot cible. i et j dénotent ainsi les positions de début et de fin de la séquence, relatives au mot cible.

— Les parties du discours, avec 7 traits : P_{-3}, P_{-2}, P_{-1}, P_0, P_1, P_2, P_3, où P_i est la partie du discours du mot à la position i relative au mot cible.

— Les mots uniques du contexte, avec un trait pour chaque lemme trouvé dans la même phrase que le mot cible pendant l'entraînement. La valeur de ces traits vaut ensuite 1 si le lemme est présent dans la même phrase que le mot cible, et 0 sinon.

Pour chacun des mots du lexique de WordNet, un classifieur linéaire de type « séparateur à vaste marge » (SVM) est entraîné pour assigner un sens à un mot en fonction de ces traits. Le système est entraîné sur les corpus SemCor et DSO, et une méthode permettant d'ajouter des exemples depuis un corpus parallèle est utilisée en exploitant des corpus parallèles anglais-chinois.

À la suite de ces travaux, Zhong et Ng (2010) proposent un système supervisé nommé « It Makes Sense » (IMS), qui reprend tous les éléments de NUS-PT, mais sous la forme d'une implémentation en Java qui est libre et extensible. Le système IMS servira de base et de référence à de nombreux systèmes supervisés comme par exemple Taghipour et Ng (2015a) et Iacobacci et al. (2016). Ces deux systèmes améliorent IMS en l'entraînant sur de nouveaux corpus d'entraînements (l'OMSTI en particulier) en en intégrant de nouveaux traits à base de vecteurs de mot comme ceux de Word2Vec (Mikolov et al., 2013).

Plus tard, les systèmes supervisés qui s'appuient sur des traits linguistiques sont délaissés au profit des systèmes neuronaux. Ces derniers permettent d'extraire et d'apprendre automatiquement les informations dont ils ont besoin pour accomplir

la tâche, à partir de vecteurs de mot appris conjointement avec le modèle, ou à partir de vecteurs pré-entraînés. Les systèmes neuronaux ont évolué en deux branches distinctes : d'une part, les approches à classification directe et d'autre part, celles qui s'appuient sur la méthode des k plus proches voisins.

1.4.2.2 Approches neuronales à classification directe

Dans cette première catégorie d'approches neuronales, un réseau neuronal classifie et attribue directement un sens à chaque mot donné en entrée à l'aide d'une distribution de probabilité calculée par la fonction *softmax*. Ces systèmes sont fondamentalement dans la continuité des systèmes pré-neuronaux, mais nécessitent moins de ressources que ces derniers car aucun trait linguistique explicite n'est nécessaire.

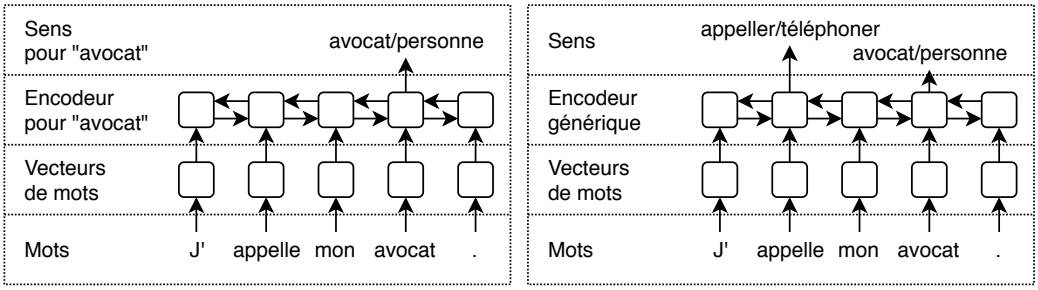


FIGURE 1.3 – Approches neuronales à classification directe. À gauche, l'architecture nécessite un encodeur par lemme (Kågebäck et Salomonsson, 2016). À droite, l'encodeur est capable d'assigner un sens à tous les mots à la fois (Raganato et al., 2017b; Luo et al., 2018a,b; Vial et al., 2019b).

Kågebäck et Salomonsson (2016) sont les premiers à mettre en œuvre un réseau de neurones pour la DL. Leur architecture, illustrée dans la figure 1.3 a trois couches de cellules : la première convertit les mots en vecteurs à l'aide de vecteurs pré-entraînés (ceux de GloVe), la deuxième est une couche de cellules récurrentes *LSTM*, et la troisième assigne un sens à chaque mot grâce à la fonction *softmax*.

Dans leur système, à l'instar des systèmes pré-neuronaux, un modèle n'est capable de prédire que le sens d'un seul lemme du dictionnaire et donc chaque lemme a son propre modèle de classification qui est entraîné séparément.

Les couches cachées de cellules *LSTM* sont ainsi très petites, avec seulement deux couches de taille 74 chacune. Il est cependant peu aisé d'entraîner ce système à annoter tous les mots d'un document car chaque lemme doit avoir son propre modèle.

Leur système est ainsi évalué sur les tâches « échantillon lexical » des campagnes d'évaluation SensEval 2 et SensEval 3 dans lesquelles plusieurs instances d'un faible nombre de lemmes distincts sont à annoter en sens, mais il n'est pas évalué sur les tâches de désambiguïsation lexicale « tous mots » où tous les mots d'un document doivent être annotés en sens.

Plus tard, Raganato et al. (2017b) proposent également un modèle à base de *LSTM* mais qui permet cette fois de prédire une étiquette pour chacun des mots donnés en entrée. Cette étiquette fait partie d'un ensemble comprenant tous les sens du dictionnaire utilisé ainsi que tous les mots observés pendant l'entraînement. Ils augmentent ensuite leur modèle avec une couche d'attention et ils effectuent un entraînement multi-tâches dans lequel leur réseau prédit à la fois un sens ou un mot, une étiquette de partie du discours et une étiquette de catégorie sémantique.

Cette architecture permet cette fois d'annoter tous les mots d'une séquence en une passe. En effet, leur réseau associe à un mot en entrée une étiquette appartenant à l'ensemble des sens de leur inventaire de sens ainsi que l'ensemble des mots observés pendant l'entraînement. Cette approche permet à leur modèle d'apprendre à prédire une étiquette de sens lorsque le mot est annoté dans le corpus d'entraînement et une étiquette de mot lorsque le mot n'est pas annoté (si c'est un mot outil par exemple).

L'inconvénient de leur approche est qu'elle n'est pas applicable lorsque l'on veut réaliser l'apprentissage sur un corpus partiellement annoté en sens. En effet pour ce type de corpus, leur modèle va apprendre à recopier des mots non annotés, au lieu d'essayer de les annoter, alors qu'ils sont potentiellement porteurs de sens.

Dans deux autres articles, Luo et al. (2018a,b) ont proposé une amélioration des performances obtenues par ce type de système, en calculant une attention entre le contexte d'un mot cible et les définitions de ses différents sens. Leur travail est ainsi le premier à incorporer les connaissances de WordNet dans un système de désambiguïsation neuronal.

Enfin, notre travail autour d'une nouvelle architecture neuronale (voir le chapitre 5) s'appuie aussi sur ce type d'approche, simplifiant grandement l'architecture et permettant d'exploiter pour la première fois des modèles de langue pré-entraînés (ELMo, BERT...) dans ce type de système. De plus, notre méthode de compression de vocabulaire de sens (voir le chapitre 6) nous permet d'augmenter la couverture de ces systèmes tout en réduisant leur nombre de paramètres.

Ce qu'on peut remarquer, c'est qu'il existe ainsi deux branches distinctes de ces types de réseaux neuronaux :

1. Ceux dans lesquels il y a un réseau neuronal distinct et spécifique à chaque

lemme du dictionnaire (Kågebäck et Salomonsson, 2016). Comme pour les approches pré-neuronales, chaque réseau est capable de gérer un lemme particulier avec ses sens. Par exemple, l'un des classifieurs est spécialisé dans le choix entre les quatre sens possibles du nom *souris*. Ce type d'approche est particulièrement adapté aux tâches « échantillon lexical », où seul un petit nombre de mots distincts et très ambigus doit être annoté dans plusieurs contextes. Mais ils nécessiteraient plusieurs milliers de réseaux différents [19] pour pouvoir aussi être utilisés dans les tâches de désambiguïsation lexicale « tous mots », dans lesquelles tous les mots d'un document doivent être annotés en sens.

2. Ceux dans lesquels il y a un seul réseau neuronal, plus grand et capable de gérer tous les lemmes du lexique, qui attribuent à un mot un sens issu de l'ensemble de tous les sens de l'inventaire de sens utilisé (Raganato et al., 2017b; Luo et al., 2018a,b; Vial et al., 2019b).

L'avantage de la première branche des approches est que pour désambiguïser un mot, il est beaucoup plus facile de limiter notre choix à l'un de ses sens possibles que de chercher parmi tous les sens de tous les mots du lexique. Pour en donner une idée, le nombre moyen de sens des mots polysémiques dans WordNet est d'environ 3, alors que le nombre total de sens en considérant tous les mots est 206 941. [20]

La seconde approche a cependant une propriété intéressante : tous les sens résident dans le même espace vectoriel et partagent donc des caractéristiques dans les couches cachées du réseau. Cela permet au modèle de prédire un sens identique pour deux mots différents (synonymes), mais aussi de prédire un sens pour un mot non présent dans le dictionnaire (néologisme, faute d'orthographe, etc.).

1.4.2.3 Approches neuronales par la méthode des k plus proches voisins

Une autre catégorie d'approches neuronales supervisées a évolué en parallèle de celles présentées précédemment : celles qui s'appuient sur la méhode des k plus proches voisins.

Dans ce type d'approche, illustrée dans la figure 1.4, le composant principal est un modèle de langue neuronal capable de prédire un mot en tenant compte des mots qui l'entourent, soit un modèle de langue type MLM (*Masked Language Model*) comme on l'a vu dans la section 1.3.3.2. Une fois le modèle de langue entraîné,

19. L'ensemble de WordNet contient par exemple 26 896 mots polysémiques (https://wordnet.princeton.edu/documentation/wnstats7wn)
20. https://wordnet.princeton.edu/documentation/wnstats7wn

il est utilisé pour produire des vecteurs de sens en moyennant les vecteurs de mot prédits par le modèle à l'endroit où ces mots sont annotés avec un sens particulier.

Au moment du test, le modèle de langue est utilisé pour prédire un vecteur en fonction du contexte environnant, et le sens le plus proche du vecteur prédit est attribué à chaque mot.

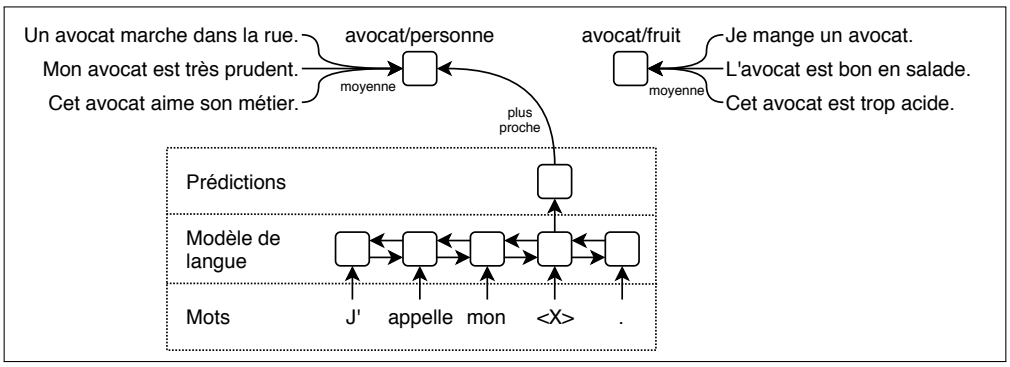


FIGURE 1.4 – Approches neuronales par la méthode des k plus proches voisins.

Dans les premiers travaux qui emploient cette approche, le modèle de langue est entraîné spécialement dans le but d'être utilisé pour le système de DL (Yuan et al., 2016; Le et al., 2018). Par la suite, avec la popularité grandissante des modèles de langue pré-entraînés (ELMo, BERT, etc.), ce sont ces modèles qui sont utilisés pour la création des vecteurs de sens (Loureiro et Jorge, 2019; Wiedemann et al., 2019; Huang et al., 2019; Kumar et al., 2019).

1.4.3 Approches semi-supervisées et repli sur le premier sens

Pour terminer sur les différentes approches pour la DL, nous allons parler des approches semi-supervisées, une catégorie d'approches mêlant l'utilisation de plus ou moins de données annotées en sens avec parfois des données brutes, et parfois des bases de connaissances.

En effet, Navigli (2009) définit les méthodes semi-supervisées comme étant celles nécessitant une supervision humaine *partielle*, par exemple en initialisant un système sur une faible quantité de données annotées en sens, puis en extrapolant sur des données non annotées. C'est effectivement cette méthode qui est utilisée dans les travaux de Yarowsky (1995) par exemple, bien que l'auteur définisse sa méthode comme « non supervisée ». Cependant, le terme d'approche semi-supervisée est aussi appliqué dans les travaux de Yuan et al. (2016), alors

que les auteurs utilisent une grande quantité de données annotées en sens (le SemCor et l'OMSTI), simplement pour indiquer une amélioration de leur système avec des données brutes. C'est aussi ce terme qui est utilisé dans les travaux de Chen et al. (2014) pour indiquer quand leur système utilise une information provenant de corpus annotés en sens : l'ordre de sens dans la base de données lexicale. Dans ce cas là pourtant, à l'inverse des travaux cités précédemmment, aucune donnée brute n'est utilisée pour améliorer un système supervisé.

Il ressort de ces travaux une confusion autour de l'appellation d'approche « semi-supervisée », qui se retrouve utilisée à partir du moment où des données annotées en sens sont utilisées. Cela pose selon nous plusieurs problèmes :

— D'une part, cela peut définir au moins trois types d'approches qui sont radicalement différentes et difficilement comparables car la nature et la quantité de ressources utilisées sont toutes deux très hétérogènes : celles à base de connaissances avec un repli sur le premier sens (Chen et al., 2014), celles qui s'appuient sur des corpus annotés en sens avec une amélioration grâce à des corpus bruts (Yuan et al., 2016) et celles qui s'appuient sur une très faible quantité de données annotées en sens extrapolées à de grandes quantités de données brutes (Yarowsky, 1995).

— D'autre part, avec la prédominance des vecteurs de mot et modèles de langue pré-entraînés de manière non supervisée, et utilisés dans toutes les approches supervisées récentes (entre autres Kågebäck et Salomonsson (2016); Raganato et al. (2017b); Luo et al. (2018a); Vial et al. (2019a)), toutes ces approches pourraient être qualifiées de semi-supervisées, car elles utilisent des ressources similaires à Yuan et al. (2016) (en particulier celles utilisant des modèles de langue qui réalisent une sorte de désambiguïsation non supervisée).

— Enfin, l'utilisation de l'ordre de sens dans WordNet comme une donnée *supervisée* est discutable : certes les auteurs de WordNet ont construit le SemCor pour ordonner les sens de WordNet par leur fréquence d'apparition, mais en même temps, seuls 16% des sens de WordNet sont finalement présents dans le SemCor, ce qui fait que cet ordre est bien souvent laissé au choix des concepteurs de WordNet eux-mêmes.

Pour ces raisons, nous éviterons au maximum dans cette thèse d'utiliser cette appellation, et nous préfèrerons caractériser les approches par les ressources *principales* sur lesquelles elles s'appuient, tout en mentionnant les ressources *secondaires* lorsqu'elles sont pertinentes. Ainsi, le système de Chen et al. (2014) s'appuie principalement sur les connaissances avec un repli possible sur le premier sens. Celui de Yuan et al. (2016) est un système supervisé amélioré avec des corpus bruts (comme notre approche présentée dans le chapitre 5). De même, nos travaux

de 2016 (Vial et al., 2016) décrivent une méthode à base de connaissances, enrichie avec des données annotées en sens. À l'inverse, nos travaux présentés dans le chapitre 6 présentent une méthode d'amélioration des approches supervisées grâce à des connaissances.

1.5 Évaluation

L'évaluation des différents systèmes de DL peut se faire de deux manières : d'un côté l'évaluation *in vivo* consiste à mesurer l'apport de la DL au sein d'une autre tâche, comme par exemple l'amélioration d'un système de traduction automatique ou de recherche d'information, de l'autre côté, l'évaluation *in vitro* consiste à mesurer directement les performances d'un système de DL en comparant ses prédictions avec celles d'un humain sur des corpus d'évaluation.

En pratique, l'évaluation *in vivo* a rarement été utilisée. On peut citer les travaux de Vickrey et al. (2005), qui introduisent la tâche de « traduction de mots », dans laquelle un algorithme semblable à un algorithme de DL doit trouver la bonne traduction d'un mot en fonction du contexte. Les différents « sens » d'un mot sont donc les différentes traductions possibles, et la méthode est évaluée sur un ensemble de corpus parallèles anglais-français. On peut aussi citer la tâche de substitution lexicale (McCarthy, 2002) ou le *Word-in-Context* (Pilehvar et Camacho-Collados, 2019) qu'on décrira plus loin.

Cependant ce type d'évaluation reste très rare, et les systèmes de DL sont donc majoritairement évalués *in vitro* lors de campagnes d'évaluation.

1.5.1 Campagnes d'évaluation

L'évaluation *in vitro* consiste à évaluer les systèmes de DL sur des tâches de DL spécifiques. Elle est rendue possible notamment grâce à la création de la première campagne d'évaluation nommée SensEval (Kilgarriff, 1998), centrée sur la tâche de DL. Après deux autres éditions de SensEval, cette campagne a été renommée en SemEval et touche maintenant à des tâches diverses allant de la reconnaissance d'entités nommées à de l'analyse de sentiments. Les tâches de DL sont pour autant toujours régulièrement présentes. Après le premier SensEval, on retrouve au moins une tâche de DL dans SensEval 2 (Edmonds et Cotton, 2001), SensEval 3 (Snyder et Palmer, 2004), SemEval 2007 (Navigli et al., 2007), SemEval 2013 (Navigli et al., 2013) et SemEval 2015 (Moro et Navigli, 2015).

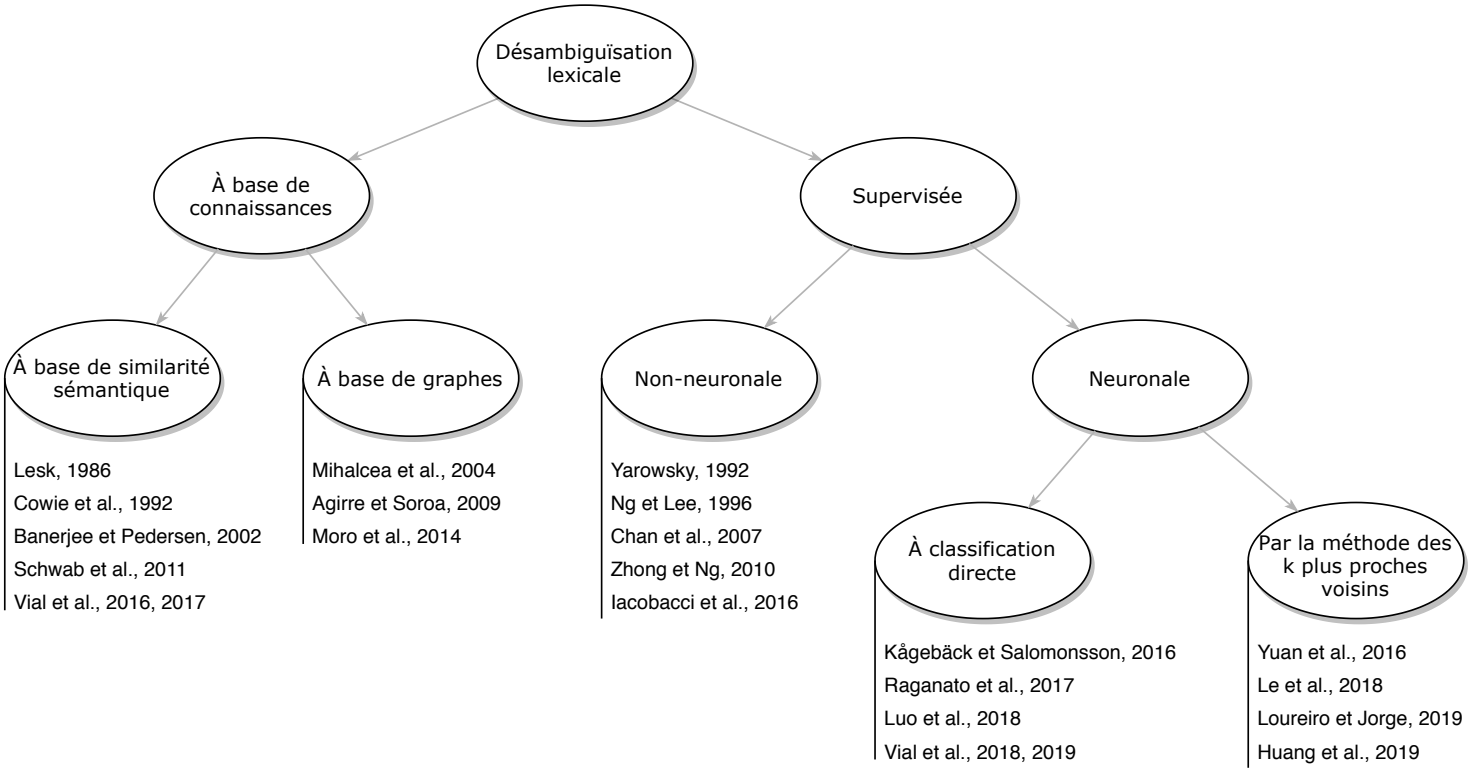


FIGURE 1.5 – Taxonomie des différentes approches pour la désambiguïsation lexicale.

1.5.1.1 Tâches

On distingue deux tâches pour l'évaluation de la désambiguïsation lexicale *in-vitro* :

— La tâche « tous mots », dans laquelle tous les mots dans un ensemble de documents doivent être annotés en sens.

— La tâche « échantillon lexical », dans laquelle de nombreuses instances de seulement quelques lemmes extraits dans un ensemble de phrases doivent être annotées en sens.

Par exemple, la campagne d'évaluation SensEval 2 (Edmonds et Cotton, 2001) comprenait à la fois une tâche « tous mots » dans laquelle 2 301 mots sont à annoter en sens dans 238 phrases, avec un nombre de lemmes uniques de 1 051, et une tâche « échantillon lexical » dans laquelle 4 238 mots sont à annoter dans 4 328 phrases, pour un nombre de lemmes uniques de 233.

La campagne d'évaluation SemEval 2007 (Agirre et al., 2007) a aussi proposé deux tâches « tous mots » distinctes :

— La tâche de DL « grain fin » (tâche 17), dans laquelle les mots sont annotés avec un unique sens, et où aucune approximation n'est possible.

— La tâche de DL « gros grain » (tâche 7), dans laquelle les mots sont annotés pour la plupart avec plusieurs sens possibles, ce qui reflète souvent mieux un réel jugement humain.

La principale langue cible de ces tâches d'évaluation est l'anglais, mais il existe aussi des tâches pour la désambiguïsation d'autres langues comme le français ou l'italien par exemple (Navigli et al., 2013; Moro et Navigli, 2015).

Dans les tâches de DL des campagnes d'évaluation, les corpus sont annotés avec la version la plus récente de WordNet disponible au moment de la campagne (voir la section 1.3.2.8). Étant donné que les étiquettes de sens ne sont pas toujours compatibles d'une version à une autre de WordNet, ces variations peuvent compliquer la comparaison des systèmes entre eux. Cependant, depuis nos travaux sur UFSAC (voir chapitre 4) et les travaux de Raganato et al. (2017a), tous les corpus d'évaluation sont regroupés et utilisent l'inventaire de sens WordNet 3.0.

1.5.1.2 Mesures

Dans ces tâches de DL, les systèmes comparés doivent annoter en sens tous les mots d'un document ou une partie d'entre eux. Les annotations sont ensuite comparées aux références et les performances des systèmes sont mesurées selon les mesures de couverture (C), précision (P), rappel (R) et F-mesure (F1), dont les formules sont :

$$C = \frac{mots\ annotés}{mots\ à\ annoter} \qquad P = \frac{mots\ correctement\ annotés}{mots\ annotés}$$

$$R = \frac{mots\ correctement\ annotés}{mots\ à\ annoter} \qquad F1 = \frac{2 \times P \times R}{P + R}$$

Dans la plupart des articles sur la DL, seule la mesure F1 est utilisée pour comparer des systèmes entre eux. En effet, la majorité des systèmes de DL annotent en sens tous les mots possibles à annoter. On se retrouve avec $mots\ à\ annoter = mots\ annotés$, donc $C = 1$, et enfin $P = R = F1$.

1.5.1.3 Performances des systèmes

Deux mesures de référence sont généralement données dans les tâches d'évaluation :

— L'étalon du sens aléatoire, une mesure théorique qui se calcule comme la moyenne des probabilités d'obtenir un mot correctement annoté en prenant un sens aléatoire dans l'inventaire de sens, pour tous les mots du document.

— L'étalon du premier sens, ou sens le plus fréquent, qui est le score obtenu par un système choisissant systématiquement le premier sens dans l'inventaire de sens (voir section 1.4.3).

Dans le tableau 1.2, on peut voir les performances des meilleurs systèmes sur la tâche 7 de SemEval 2007 depuis sa sortie, avec ces mesures étalons. Comme on peut le voir, parmi les approches détaillées dans la section 1.4, celles qui arrivent généralement en tête sont les approches supervisées. Seule la méthode de Chen et al. (2014) à base de connaissances a égalé une méthode supervisée, jusqu'à l'arrivée des premiers réseaux de neurones et la méthode de Yuan et al. (2016).

Ce que l'on peut aussi remarquer, c'est qu'entre 2007 et 2016, les scores des meilleures méthodes de DL n'ont d'une part pas beaucoup évolué, et d'autre part n'étaient pas très supérieurs à l'étalon du premier sens. Depuis 2018 et notamment

Année	Système	Type	Score F1 (%)
-	Sens aléatoire	-	62,7
-	Premier sens	-	78,9
2007	Chan et al. (2007b)	sup.	82,5
2010	Zhong et Ng (2010)	sup.	82,6
2014	Chen et al. (2014)	conn.	82,6
2016	Yuan et al. (2016)	sup.	84,3
2018	Vial et al. (2018a)	sup.	85,8
2019	Vial et al. (2019a)	sup.	90,6

TABLE 1.2 – Résultats des systèmes état de l'art à la fin de chaque année sur la tâche 7 de la campagne d'évaluation SemEval 2007 (tâche de DL « tous mots » et « gros grain »). « sup. » indique un système supervisé, « conn. » indique un système à base de connaissances.

en 2019, les systèmes état de l'art sont nettement plus performants, et ce, grâce aux modèles de langue pré-entraînés d'une part (voir section 1.3.3.2) et grâce à nos contributions d'autre part (voir chapitre 4, chapitre 5 et chapitre 6).

1.5.2 Autres tâches associées à la désambiguïsation lexicale

Pour finir sur l'évaluation des systèmes de DL, nous souhaitons mettre en lumière et décrire deux tâches qui ne sont pour l'instant que peu ou pas utilisées mais qui sont pourtant en lien avec la DL : la tâche de substitution lexicale et la tâche de *Word-in-Context*. Ces deux tâches sont toutes les deux des alternatives à la méthode « classique » d'évaluation de la DL (c'est-à-dire comparer des étiquettes de sens), leurs auteurs jugeant cette dernière comme n'étant pas assez représentative de l'utilité réelle de la DL.

1.5.2.1 Substitution lexicale

La tâche de substitution lexicale a été présentée par McCarthy (2002) comme une tâche d'évaluation de la DL « orientée application », puis elle a été intégrée à la campagne d'évaluation SemEval 2007 (tâche 10) (McCarthy et Navigli, 2007) et SemDis 2014 (Fabre et al., 2014). L'objectif est, en fonction d'un contexte donné, de remplacer un mot cible par un substitut, et de comparer cette réponse à celles de plusieurs annotateurs.

Par exemple, prenons la phrase « *Je cueille des truffes en forêt.* » et le mot cible *truffe*, l'objectif est de distinguer entre autres la plante de la truffe d'un animal ou de la truffe en chocolat, et donc de remplacer le mot par un autre comme par exemple *champignon*.

De cette manière, aucun inventaire de sens spécifique n'est requis. La tâche permet donc de comparer entre eux par exemple des systèmes de DL et des systèmes non supervisés, et permet plusieurs applications, comme la simplification de textes, le résumé automatique, etc.

Cette tâche souffre cependant de certains inconvénients. Par exemple, les annotateurs ne peuvent pas penser à toutes les alternatives possibles d'un mot et la pertinence des termes de remplacement n'est pas évaluée. Finalement, elle est rarement utilisée dans les travaux sur la DL.

1.5.2.2 Word-in-Context

Word-in-Context ou WiC (Pilehvar et Camacho-Collados, 2019) est une tâche initialement créée afin d'étudier les capacités des vecteurs de mot contextualisés (voir section 1.3.3.2) à distinguer différents sens d'un même mot. Elle a fait partie de la campagne d'évaluation SemDeep 5 (Espinosa-Anke et al., 2019).

Le corpus de la tâche se compose de trois ensembles (entraînement, développement et test) de paires de phrases avec un mot cible en commun dans les deux phrases. L'objectif est d'identifier si le mot cible a le même sens dans les deux phrases ou bien deux sens différents.

Par exemple, pour une phrase A « *Je mange un avocat.* », une phrase B « *J'appelle mon avocat.* » et le mot cible *avocat*, le système est censé indiquer que les sens sont différents.

En pratique, les meilleurs systèmes consistent en un classifieur binaire qui s'appuie sur un modèle de langue pré-entraîné comme BERT. On pourrait cependant très bien imaginer comparer entre eux des systèmes de DL utilisant un inventaire de sens commun, en comparant directement leur sortie.

L'avantage de cette tâche est qu'elle ne repose pas nécessairement sur un inventaire de sens, et permet donc d'évaluer les performances de systèmes de DL non supervisés comme supervisés et à base de connaissances, en plus de mesurer la capacité de désambiguïsation des modèles de langue récents.

1.6 Conclusion

Comme nous l'avons vu, la désambiguïsation lexicale est une tâche centrale du TAL qui questionne en permanence l'existence et la nature d'un aspect fondamental des langues : le sens. L'histoire de cette tâche est riche et complexe. En effet, notamment à cause du fait que les ressources disponibles sont très rares, les approches employées, même les plus récentes, mêlent toujours plus ou moins des connaissances structurées discrètes à des données manuellement annotées.

Bien qu'elle semble essentielle afin de créer des systèmes ayant une meilleure compréhension de la langue et faisant moins d'erreurs de sémantique, les systèmes de DL sont en pratique rarement utilisés dans d'autres tâches. Cette faible utilisation peut être due à une faiblesse des performances des systèmes actuels ou à un manque d'uniformisation des ressources. Cela peut être pourtant en passe de changer, grâce aux progrès notables accomplis ces dernières années.

Dans nos contributions spécifiques à la DL, nous montrons ainsi comment nous avons amélioré les performances d'un système de DL à base de connaissances (voir chapitre 3), uniformisé les données d'entraînement et d'évaluation (voir chapitre 4) puis proposé une nouvelle architecture neuronale pour la DL supervisée (voir chapitre 5) et enfin présenté une méthode pour l'amélioration de la couverture et des performances de ces systèmes (voir chapitre 6).

Chapitre 2

Traduction automatique neuronale

2.1 Introduction

La traduction automatique (TA), en anglais *Machine Translation* (MT), est une tâche importante du traitement automatique des langues qui vise à traduire un texte d'une langue source vers une langue cible. Elle représente un défi majeur et particulièrement complexe, car le passage d'une langue à une autre nécessite de résoudre de nombreuses autres problématiques du TAL, comme la compréhension du langage naturel, l'adaptation au style et au domaine, la résolution de coréférences, la désambiguïsation lexicale, etc.

La TA est une tâche avec des enjeux importants. En effet, elle est directement utilisée au quotidien au travers de systèmes comme Google Translate [1] ou DeepL [2], permettant de traduire des courtes phrases ou des documents entiers. Elle est aussi utilisée par des agences de traduction afin de faciliter le travail des traducteurs qui peuvent post-éditer la sortie d'un système de TA [3] (Lemaire, 2017).

De plus, la TA est aussi un moteur pour le TAL. Elle a en effet contribué à de grandes avancées qui ont aussi impacté d'autres tâches. On peut citer par exemple les modèles neuronaux « séquence à séquence » (Sutskever et al., 2014), les modèles d'attention (Bahdanau et al., 2015) ou encore l'architecture Transformer (Vaswani et al., 2017).

De multiples approches pour la TA existent. On peut les regrouper en trois

1. https://translate.google.com
2. https://www.deepl.com/translator
3. https://www.tradonline.fr/localisation-de-contenus-volumineux-comment-reduire-delais-et-cout/

grandes catégories qui sont (1) les approches à base de règles, qui s'appuient sur des dictionnaires et des données syntaxiques, morphologiques et sémantiques dans les langues source et cible, (2) les approches statistiques, qui s'appuient sur des modèles statistiques entraînés sur des corpus parallèles et monolingues, et enfin (3) les approches neuronales, qui remplacent les modèles statistiques par des réseaux de neurones.

À l'instar de la désambiguïsation lexicale, les approches neuronales sont aujourd'hui prédominantes dans les travaux de recherche sur la TA et dans les campagnes d'évaluation. Dans ce chapitre, nous allons ainsi nous focaliser sur les systèmes de TA neuronaux et les avancées qu'ils ont permises dans les architectures neuronales. Nous parlerons des ressources nécessaires à leur mise en œuvre, et de leur évaluation.

2.1.1 Historique et enjeux

On peut retracer les origines des premières études sur la TA au début des années 1950, avec notamment l'organisation de la première conférence dédiée à cette tâche en 1952, et l'expérience de Georgetown-IBM en 1954, une démonstration publique d'un système de traduction russe-anglais pouvant gérer un vocabulaire de 250 mots et six règles grammaticales (Hutchins, 1986, 2004).

De ces débuts jusqu'aux années 1980, les premiers systèmes de TA reposaient sur des dictionnaires et un ensemble de règles pour une paire de langues en particulier. Ce n'est qu'au début des années 1990 qu'émergent les approches statistiques, s'appuyant elles sur des corpus parallèles, avec notamment les travaux de Brown et al. (1990), un groupe de chercheurs chez IBM.

Par la suite, les travaux de Brown et al. (1991) décrivent une méthode pour l'alignement automatique de phrases dans un corpus, et l'article de Brown et al. (1993) propose un ensemble de cinq modèles statistiques de traduction qu'on appellera « modèles IBM », et dont on expliquera brièvement le fonctionnement dans la section 2.1.2.

Enfin, les modèles neuronaux, que nous allons détailler dans la section 2.2 ont remplacé progressivement les modèles statistiques depuis notamment les travaux de Sutskever et al. (2014). Ces modèles se distinguent par l'utilisation d'un unique réseau « bout en bout » remplaçant les multiples modèles statistiques nécessaires aux systèmes statistiques classiques.

Les enjeux autour de la TA sont nombreux : comme l'écrit Hutchins (1986), les tensions entre les États-Unis et l'URSS ont stimulé les premières recherches

et l'aboutissement du premier système de TA russe-anglais dans les années 1950. Dans les années 1970, ce sont toutes les problématiques de normalisation des documents administratifs, commerciaux et techniques de la Communauté économique européenne qui ont grandement encouragé la recherche. Durant cette période, la TA est aussi utilisée dans certains cadres spécifiques. On peut noter par exemple le système canadien TAUM-METEO (Chevalier et al., 1978) qui permettait de traduire quotidiennement les bulletins météorologiques de l'anglais vers le français.

Aujourd'hui, de nombreux systèmes grand public et accessibles en ligne permettent de traduire gratuitement du texte (Google Translate, Microsoft Translator, DeepL...), et peuvent être utiles dans de nombreux cas (communication à l'étranger, aide à la rédaction...). Cependant, même si la qualité de la traduction des systèmes de TA s'est beaucoup améliorée, notamment grâce aux modèles neuronaux (voir section 2.2), ils souffrent toujours de problèmes importants, notamment en ce qui concerne les langues dites peu dotées (voir section 2.4).

2.1.2 Modèles statistiques

Les modèles de traduction statistiques ont d'abord été théorisés par Weaver (1955) puis repris et popularisés par Brown et al. (1990) de chez IBM. Ils s'appuient sur un ensemble de corpus parallèles, c'est-à-dire des textes manuellement traduits, afin de déterminer, grâce à des modèles statistiques, la meilleure traduction possible pour une phrase source.

Plus formellement, dans les méthodes statistiques, pour traduire une phrase x depuis une langue source, on cherche la phrase $\hat{y}$ dans la langue cible, qui va maximiser la probabilité conditionnelle $p(y|x)$, c'est-à-dire la probabilité que y soit une traduction de x. On a donc la formule suivante :

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left(p(y|x) \right)$$

On s'appuie ensuite sur la formule de Bayes qui est la suivante :

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

Avec $p(x)$ et $p(y)$ respectivement la probabilité des phrases x et y indépendamment de leur traduction.

Comme nous recherchons une phrase y qui maximise cette formule pour une phrase x donnée, alors $p(x)$ est une constante et nous pouvons donc l'ignorer. Au

final, la formule que Brown et al. (1993) appellent « l'équation fondamentale de la traduction automatique » est la suivante :

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left(p(x|y) \cdot p(y) \right)$$

$\hat{y}$ étant ainsi la phrase qui maximise le produit des probabilités $p(y)$ et $p(x|y)$.

L'avantage de cette approche est qu'elle scinde le problème en deux parties : D'un coté, la résolution de $p(x|y)$ se fait à l'aide d'un **modèle de traduction**, dont le rôle est d'attribuer une probabilité que la phrase x soit une traduction possible de la phrase y. De l'autre coté, $p(y)$ se résoud à l'aide d'un **modèle de langue**, qui donne une probabilité que y soit une phrase correcte à part entière.

Dans l'article de Brown et al. (1993), les auteurs proposent ainsi un ensemble de cinq modèles de traduction permettant d'estimer $p(x|y)$, en calculant les alignements les plus probables entre les mots de la langue source et ceux de la langue cible. Ces alignements sont ainsi réalisés grâce à des corpus parallèles.

Une des implémentations les plus connues de ces modèles IBM est GIZA (Al-Onaizan et al., 1999) et son amélioration principale nommée GIZA++ (Och et Ney, 2003). Cette dernière sera notamment intégrée au programme Moses (Koehn et al., 2007), un système complet permettant l'entraînement et l'utilisation de ces modèles statistiques en plus d'avoir d'autres avantages, dont la possibilité d'intégrer un modèle de langue externe et l'implémentation d'un algorithme de recherche par faisceau.

Plus précisément, Moses fonctionne grâce à un modèle log-linéaire, dans lequel n composants vont chacun assigner une probabilité p à une potentielle traduction. Chaque composant a un poids λ et ces probabilités sont combinées selon la formule suivante :

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left(p(y|x) \right) = \underset{y}{\operatorname{argmax}} \left(\sum_{i=0}^{n} \lambda_i \cdot log \left(p_i(x, y) \right) \right)$$

De cette manière, le modèle de langue ainsi que le modèle de traduction sont deux composants du modèle log-linéaire. Moses intègre ensuite d'autres composants, comme :

— une pénalité sur les mots produits, qui compte simplement le nombre de mots dans y afin de décourager les phrases trop longues ;

— une pénalité sur les mots inconnus, qui compte le nombre de mots inconnus (symboles `UNK`) dans y afin de décourager leur trop forte apparition ;

— un modèle de distorsion, qui permet de réordonner les mots de la phrase produite en calculant le coût de réordonnancement de chacun des mots de y par rapport à l'ordre des mots dans x. Plus précisément, le calcul du coût est effectué grâce à la formule suivante : $\sum_{i=0}^{n} d_i$, avec $d_i = abs(p_1 + 1 - p_2)$, où p_1 est la position dans x du mot dont la traduction est à l'indice $i-1$ dans y, et p_2 est la position dans x du mot dont la traduction est à l'indice i dans y, enfin n est la taille de la phrase y.

D'autres composants peuvent ainsi s'ajouter et contribuer au calcul des probabilités. Par exemple, lors de mes travaux de master (Vial, 2016), nous avons intégré un modèle de désambiguïsation lexicale à Moses, qui pénalise les traductions dont les sens ne correspondent pas aux sens des mots désambiguïsés de la phrase source.

À noter que le programme Moses ainsi que des tutoriels et même des liens vers les principaux corpus parallèles sont accessibles sur le site Web de Moses [4].

2.1.3 Vers les représentations continues

Entre les années 2014 et 2016, on observe un basculement progressif des méthodes statistiques pour le TAL utilisant des traits explicites et des représentations discrètes vers des méthodes utilisant des représentations continues dont les traits ne sont plus explicités.

En effet, les travaux de Mikolov et al. (2013) sur les vecteurs de mot pré-entraînés (Word2Vec) ainsi que les améliorations qui ont suivies (Pennington et al., 2014; Peters et al., 2018; Devlin et al., 2019) ont montré qu'il était possible d'apprendre des représentations continues de mots de grande qualité, uniquement à partir de corpus bruts, et sans autre connaissance au préalable.

Dans le chapitre 1, nous avons vu que pour la DL, ces méthodes ont notamment impacté les méthodes supervisées (voir section 1.4.2), qui s'appuient aujourd'hui fortement sur ces vecteurs de mot et modèles de langue pré-entraînés, à la place de traits explicites tels que les parties du discours et les collocations de mots, permettant à la fois de simplifier les modèles et d'améliorer leurs performances.

En TA, en plus des vecteurs de mot comme Word2Vec, ce sont aussi les avancées importantes sur les réseaux de neurones, comme l'architecture « séquence à séquence » de Sutskever et al. (2014), qui ont permis de simplifier et d'améliorer les performances des systèmes statistiques.

En effet, ces modèles permettent de s'affranchir de la définition des nombreux composants qu'on a vus précédemment (modèle de langue, modèle de traduction,

4. http://www.statmt.org/moses/

modèle de distorsion, etc.) pour n'avoir qu'un réseau de neurones unique, apprenant conjointement à encoder une phrase source, décoder en langue cible, aligner les mots... pour des performances nettement supérieures.

Dans la section suivante, nous allons ainsi voir plus en détail ces architectures neuronales, qui sont au cœur de plusieurs contributions de cette thèse.

2.2 Traduction automatique neuronale

La traduction automatique neuronale est une évolution de la traduction automatique statistique dans laquelle tous les différents composants (modèle de langue, modèle de traduction, modèle de distorsion...) sont remplacés par un unique réseau de neurones. Comme pour les systèmes de TA statistique, l'apprentissage de ce réseau de neurones s'effectue sur des corpus parallèles et monolingues.

Dans cette section, nous allons ainsi décrire les principales architectures neuronales pour la TA.

2.2.1 Réseaux de neurones récurrents

Les réseaux de neurones fondés sur des cellules récurrentes sont à la base des premières architectures neuronales parvenant à atteindre les performances des systèmes statistiques classiques. C'est pourquoi nous allons détailler ici leur fonctionnement.

En effet, dans un réseau de neurones classique non récurrent, la sortie d'une cellule dépend généralement uniquement de son vecteur d'entrée, d'une matrice de poids, d'un vecteur de biais, et d'une possible fonction d'activation non linéaire. Plus précisément, soit $x \in \mathbb{R}^n$ un vecteur d'entrée, $A \in \mathbb{R}^{n \times m}$ une matrice de poids, $b \in \mathbb{R}^m$ un vecteur de biais et δ la fonction d'activation, alors la sortie $y \in \mathbb{R}^m$ d'une cellule non récurrente se calcule de la manière suivante :

$$y = \delta(xA + b)$$

Les poids A et le biais b sont appris au sein du réseau de neurones, mais la cellule n'est ainsi capable que de produire un résultat pour une entrée de taille fixe, et sans considération pour les résultats produits précédemment.

Lorsque l'on souhaite encoder des séquences entières comme des phrases, de tailles variables, et obtenir une représentation vectorielle d'un mot qui prenne en

compte les mots vus précédemment, il est nécessaire d'ajouter un mécanisme permettant aux cellules d'avoir une « mémoire » des éléments traités précédemment. Les cellules récurrentes telles que LSTM ou GRU, que nous allons maintenant voir, permettent ainsi d'avoir ce mécanisme de mémoire.

2.2.1.1 Cellules LSTM

Les cellules LSTM, ou *Long Short-Term Memory*, ont été introduites par Hochreiter et Schmidhuber (1997) afin de proposer des cellules récurrentes capables de sélectionner quelles informations conserver et quelles informations « oublier » sur de longues séquences.

Leur formulation est la suivante : soient x_t la valeur de l'élement actuel d'une séquence, $h_{(t-1)}$ la sortie de la précédente cellule LSTM et $c_{(t-1)}$ l'état de la cellule LSTM précédente, on calcule les nouvelles valeurs de la sortie h_t et de l'état c_t grâce aux formules suivantes :

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) & f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) & o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t c_{(t-1)} + i_t g_t & h_t &= o_t \tanh(c_t)
\end{aligned}
$$

La sortie h_t d'une cellule LSTM à l'instant t est donc fonction de son entrée x_t, de ses états précédents $h_{(t-1)}$ et $c_{(t-1)}$, des 8 matrices de poids W_{ii}, W_{hi}, W_{if}, W_{hf}, W_{ig}, W_{hg}, W_{io} et W_{ho} et des 8 vecteurs de biais b_{ii}, b_{hi}, b_{if}, b_{hf}, b_{ig}, b_{hg}, b_{io} et b_{ho}. Ces poids et ces biais sont les paramètres qui vont être appris par le modèle neuronal pendant l'entraînement. Enfin, σ est la fonction sigmoïde. La figure 2.1 illustre ainsi le fonctionnement d'une cellule LSTM.

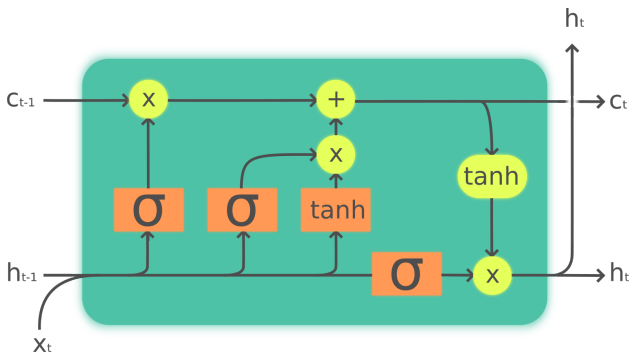


FIGURE 2.1 – Une cellule LSTM. [5]

5. Image de Wikimedia (https://upload.wikimedia.org/wikipedia/commons/3/3b/The_LSTM_cell.png)

2.2.1.2 Cellules GRU

Les cellules GRU, ou *Gated Recurrent Units*, sont une simplification des LSTM introduites par Cho et al. (2014), dont les formules permettant de calculer la sortie h_t en fonction de l'élement actuel de la séquence x et de la sortie précédente de la cellule h_{t-1} sont les suivantes :

$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})) \\
h_t &= (1 - z_t) * n_t + z_t * h_{(t-1)}
\end{aligned}
$$

Cette fois, on n'a plus que six matrices de poids (W_{ir}, W_{hr}, W_{iz}, W_{hz}, W_{in} et W_{hn}, ainsi que six vecteurs de biais b_{ir}, b_{hr}, b_{iz}, b_{hz}, b_{irn} et b_{hn}, ce qui rend les GRU moins coûteuses en mémoire que les LSTM.

La figure 2.2 illustre ainsi le mécanisme de la cellule GRU.

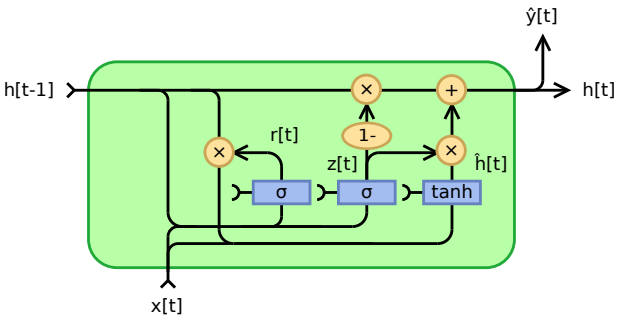


FIGURE 2.2 – Une cellule GRU. [6]

Les GRU sont ainsi utilisés à la place des LSTM dans certains travaux sur la traduction automatique (Cho et al., 2014; Bahdanau et al., 2015; Jean et al., 2015), mais d'autres études (Britz et al., 2017; Weiss et al., 2018) montreront plus tard que dans certaines tâches du TAL, et notamment la TA, les LSTM obtiennent de meilleures performances que les GRU, ce qui fait que ces premières sont, en TA, nettement plus utilisées que les GRU.

6. Image de Wikimedia (https://upload.wikimedia.org/wikipedia/commons/3/37/Gated_Recurrent_Unit%2C_base_type.svg)

2.2.2 Modèles « séquence à séquence »

En s'appuyant sur le fonctionnement des réseaux de neurones récurrents décrits précédemment, les premiers travaux qui présentent un réseau de neurones atteignant des performances proches des systèmes de TA statistiques classiques sont ceux de Sutskever et al. (2014) et Cho et al. (2014).

Dans ces travaux, le modèle neuronal est découpé en deux parties, comme illustré dans la figure 2.3 :

— L'encodeur, dont le rôle est de produire une représentation vectorielle de la phrase en langue source.

— Le décodeur, qui va générer des mots en langue cible, en fonction de la sortie de l'encodeur et des mots qu'il a générés précédemment.

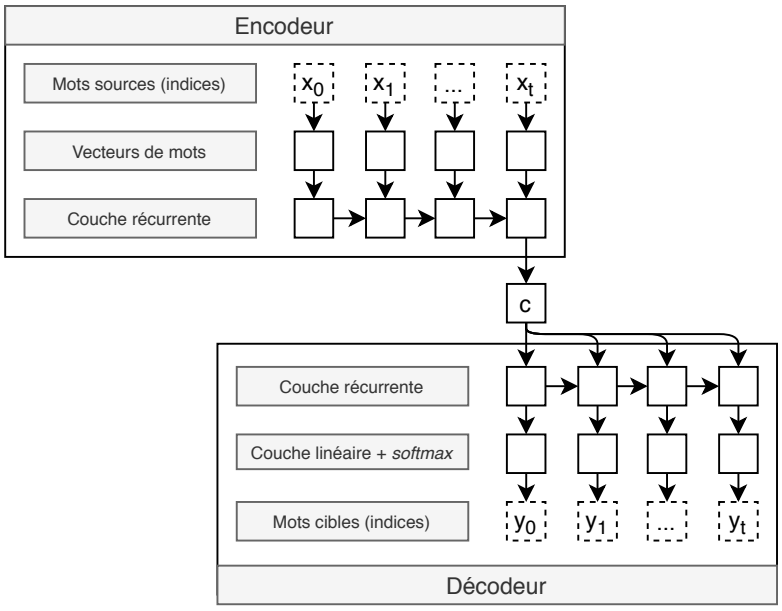


FIGURE 2.3 – Architecture neuronale encodeur-décodeur.

Plus précisément, l'encodeur se charge de transformer une séquence de vecteurs $x = (x_0, x_1, \ldots, x_n)$ en un vecteur c, à l'aide d'une couche de cellules neuronales récurrentes. Dans l'article de Sutskever et al. (2014), les auteurs utilisent des cellules *Long Short-Term Memory*, ou LSTM (Hochreiter et Schmidhuber, 1997) tandis que dans l'article de Cho et al. (2014), les auteurs utilisent des cellules *Gated Recurrent Units*, ou GRU, introduites dans ce même article.

Dans les deux cas, ces cellules récurrentes calculent pour chaque vecteur x_t, un nouveau vecteur h_t qui dépend non seulement de l'entrée, mais aussi de l'état

précédent de la cellule au moment de produire h_{t-1}, et le dernier vecteur produit, h_n est utilisé pour calculer le vecteur final c. Ce vecteur c, qui est donc le résultat de l'encodeur après avoir traité tous les mots de la séquence, est une représentation à lui seul de la phrase source dans son entièreté.

Une fois le vecteur c généré par l'encodeur, le décodeur produit séquentiellement de nouveaux mots en fonction de c et des mots produits précédemment. Pour cela, il s'appuie à nouveau sur des cellules récurrentes, qui prennent à chaque fois le vecteur c en entrée, et qui vont générer un nouveau vecteur h_t en fonction de c et du vecteur produit précédemment h_{t-1}.

Grâce au mécanisme de mémoire des cellules récurrentes du décodeur qui est entraîné avec le modèle, chaque mot ainsi produit dépend à la fois de la phrase à traduire, et des mots produits précédemment. Le décodeur est utilisé pour produire de nouveaux mots jusqu'à l'apparition d'un symbole d'arrêt, généralement un symbole réservé comme `<eos>` qui marque la fin de la phrase.

2.2.3 Mécanisme d'attention

Dans l'architecture encodeur-décodeur de base, le vecteur c produit par l'encodeur est une représentation de taille fixe qui est censée capturer toutes les informations utiles de la phrase source pour permettre au décodeur de la traduire. Cependant, Bahdanau et al. (2015) observent que plus cette phrase source est longue, plus il est difficile pour cet unique vecteur c de contenir toute l'information utile de la phrase.

C'est ainsi que les auteurs suggèrent qu'au lieu de calculer un vecteur c unique à partir du dernier état de la couche récurrente de l'encodeur, il faudrait calculer un nouveau vecteur c_i différent pour chaque mot à produire par le décodeur, à partir de tous les états de l'encodeur.

Le vecteur c_i est ainsi calculé de la manière suivante :

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$$

avec

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})}$$

et

$$e_{ij} = score(s_{i-1}, h_j)$$

La fonction $score$ est une fonction dérivable quelconque intégrée au réseau de neurones, qui est chargée de donner un score d'alignement e_{ij} entre le dernier état du décodeur s_{i-1}, et un état de l'encodeur h_j. α_{ij} est ensuite une normalisation de e_{ij} via la fonction $softmax$, afin d'avoir un score d'alignement compris entre 0 et 1. Enfin, c_i est une somme de toutes les sorties de l'encodeur $h_0, \ldots, h_j$, pondérée par le score d'alignement α_{ij}.

Le résultat est que non seulement l'information de tous les états de l'encodeur peut être exploitée par le décodeur, mais en plus, le vecteur a_{ij} donne un score de l'attention portée par le décodeur au moment de produire le mot y_i, par rapport au mot x_j. C'est pour cela que ce nouveau mécanisme dans l'architecture encodeur-décodeur porte le nom de mécanisme d'attention.

De plus, comme chaque vecteur c_i peut maintenant utiliser tous les états h_j de l'encodeur, et non plus seulement le dernier état h_t, ces états peuvent être calculés avec une couche récurrente bidirectionnelle, c'est-à-dire qui permet de calculer une représentation vectorielle en fonction des contextes à la fois gauches et droits.

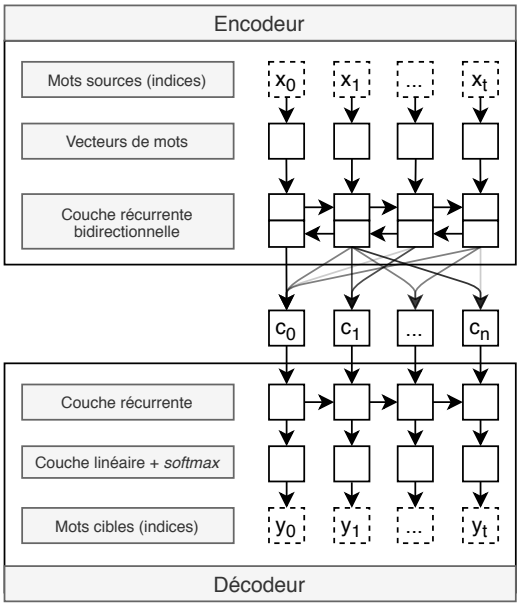


FIGURE 2.4 – Architecture d'un modèle encodeur-décodeur avec mécanisme d'attention.

La figure 2.4 illustre le fonctionnement d'un modèle encodeur-décodeur avec mécanisme d'attention, et dans la figure 2.5, on peut voir un exemple de matrice d'attention (ou matrice d'alignement) en affichant les valeurs de α_{ij}.

Le mécanisme d'attention sera ensuite repris dans de nombreux travaux. Par

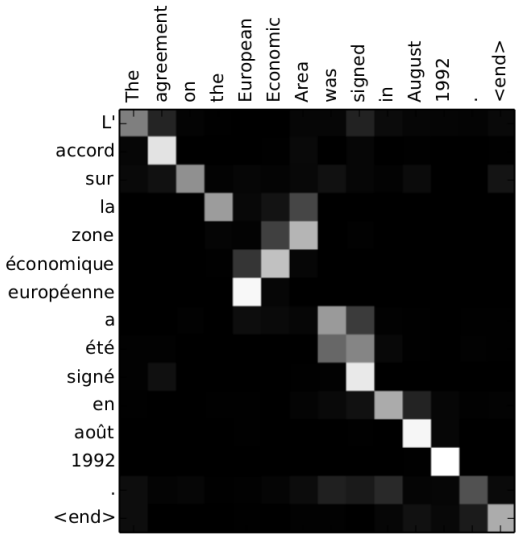


FIGURE 2.5 – Matrice d'alignement entre une phrase source (en anglais) et une phrase cible (en français). Chaque case représente un poids α_{ij}. Plus la valeur est grande (proche de 1) et plus elle est claire. Figure issue de l'article de Bahdanau et al. (2015).

exemple, Luong et al. (2015a) comparent différentes méthodes pour la fonction $score$ et proposent une variante du mécanisme d'attention appelée « attention locale ». Kim et al. (2017) définissent une nouvelle forme d'attention appelée « attention structurée » et ils l'appliquent à des tâches de questions-réponses et de compréhension de la langue.

En TA, Google exploite le mécanisme d'attention dans le premier système qui marquera sa transition vers une approche neuronale (Wu et al., 2016). Ce système s'appuie à la fois sur le mécanisme d'attention et sur un empilement de huit couches de LSTM dans son encodeur et dans son décodeur. En DL, Raganato et al. (2017b) sont les premiers à appliquer un mécanisme d'attention à leur système supervisé.

2.2.4 Architecture Transformer

Vaswani et al. (2017) constatent plusieurs problèmes dans les architectures encodeur-décodeur s'appuyant sur des cellules récurrentes :

— Premièrement, à cause de la nature même de ces cellules récurrentes, qui nécessitent d'abord de calculer h_t pour pouvoir ensuite calculer h_{t+1}, l'encodage d'une séquence ne peut pas s'exécuter en parallèle et nécessite $O(n)$ opérations séquentielles.

— Deuxièmement, bien que les cellules récurrentes soient capables de modéliser des dépendances sur une longue distance, l'information disparaît toujours progressivement à chaque fois qu'elle doit traverser une cellule, ce qui rend difficile l'utilisation d'une information d'un bout à l'autre d'une séquence.

Afin de répondre à ces problèmes, les auteurs proposent une nouvelle architecture de réseau neuronal appelée « Transformer », qui n'utilise aucune récurrence. Elle repose essentiellement sur deux changements importants : (1) un nouveau mécanisme d'attention appelé « attention multi-tête », et (2) une nouvelle façon d'encoder la position d'un mot d'une séquence.

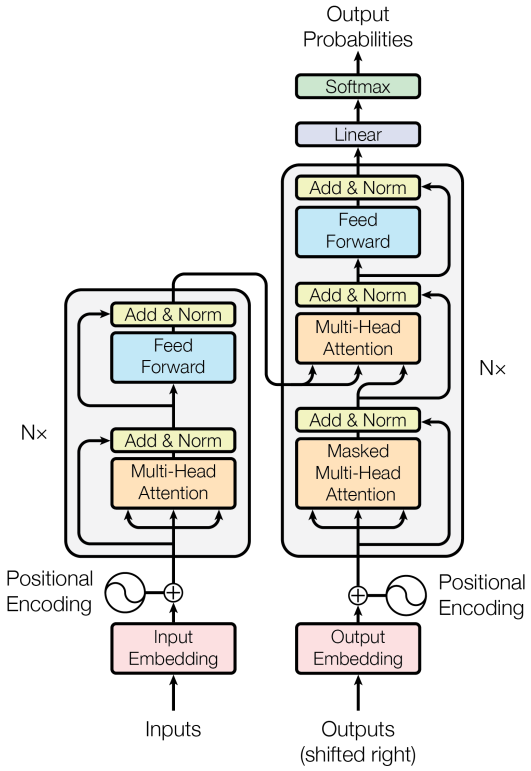


FIGURE 2.6 – Architecture Transformer. Figure issue de l'article de Vaswani et al. (2017).

La figure 2.6 illustre l'architecture Transformer dans son ensemble. Elle est composée d'un encodeur (à gauche) et d'un décodeur (à droite) comme dans les architectures vues précédemment.

Dans les sections suivantes, nous allons détailler le fonctionnement de l'attention multi-tête, du nouvel encodage des positions et des autres particularités du Transformer. Pour finir, nous parlerons des nombreuses applications du Transformer dans les autres disciplines du TAL.

2.2.4.1 Attention multi-tête

Le mécanisme d'attention multi-tête est le composant central de l'architecture Transformer. Illustré dans la figure 2.7, il repose sur une formulation générale de l'attention qui la définit comme un alignement entre une requête (*query* ou Q) et un ensemble de paires de (clé, valeur) (*keys* et *values* ou K et V). Plus précisément, l'attention se calcule comme une somme pondérée des valeurs, où chaque poids associé à une valeur dépend d'une fonction entre la requête et la clé correspondante à cette valeur.

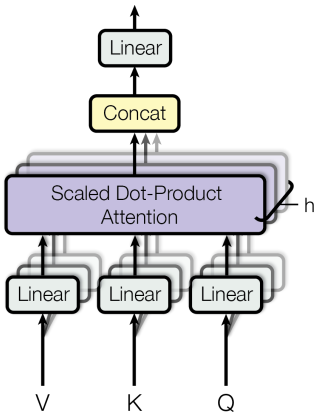


FIGURE 2.7 – Attention multi-tête. Figure issue de l'article de Vaswani et al. (2017).

Dans le cas de l'architecture encodeur-décodeur utilisée en TA, dans laquelle on calcule une attention entre l'état courant d'un décodeur et tous les états d'un encodeur, Q est le vecteur qui correspond au dernier état du décodeur, et K et V sont tous les deux la même matrice qui correspond à tous les états de l'encodeur.

Dans le modèle Transformer, on calcule non seulement cette attention entre l'encodeur et l'état courant du décodeur, mais aussi une attention au sein même de l'encodeur, comme au sein même du décodeur, ce qui remplace les celulles récurrentes type LSTM ou GRU. Dans ce mécanisme, qu'on appelle « auto-attention », la requête est chaque élément de l'encodeur (ou du décodeur), et les clés et valeurs sont toutes les sorties de ce même encodeur (ou décodeur).

Au niveau du calcul des poids de l'attention, Vaswani et al. (2017) introduisent une nouvelle fonction appelée *scaled dot-product attention*, qui calcule l'attention entre le vecteur de requête Q et deux vecteurs de clé et valeur K et V, avec la formule suivante :

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

avec d_k la dimension de Q, K et V (qui sont les mêmes).

L'ensemble du mécanisme d'attention est finalement appelé attention « multi-tête », parce qu'il repose sur n têtes qui calculent chacune en parallèle un score d'attention sur une portion des valeurs Q, K et V, qui sont ensuite concaténés. Plus précisément, si par exemple la dimension d_k est de 512, et qu'on a huit têtes d'attention (comme le modèle de base des auteurs), alors on calcule simultanément l'attention entre huit portions du vecteur. Chaque portion est de taille 64 ($512/8$), puis elles sont concaténées pour avoir en sortie un vecteur de taille 512.

Ainsi, avec le modèle Transformer, tous les vecteurs qui rentrent et sortent des couches d'attention multi-tête doivent être de taille identique, et cette taille doit être un multiple du nombre de têtes qui exécutent le calcul en parallèle.

2.2.4.2 Vecteurs de position

L'autre nouveauté principale de l'architecture Transformer, qui complète le mécanisme d'attention afin de pouvoir se passer entièrement des réseaux récurrents, se situe au niveau de la manière d'encoder la position des éléments dans une séquence.

Traditionnellement, dans les réseaux de neurones qui traitent des séquences, on assigne un vecteur à chaque élément de la séquence en fonction de sa nature. Ainsi, la séquence $[a, b, b, c, a]$ a pour représentation vectorielle $[v_a, v_b, v_b, v_c, v_a]$ avant de passer dans l'encodeur. C'est ensuite le mécanisme de récurrence qui permet de contextualiser les vecteurs en fonction des mots précédents et suivants.

L'encodage de position, tel qu'il est proposé dans l'architecture Transformer, consiste à calculer les représentations vectorielles des éléments d'une séquence en fonction de leur nature ainsi que de leur position dans la séquence. Les deux informations s'additionnent simplement pour avoir la représentation finale. Ainsi, la séquence $[a, b, b, c, a]$ a pour représentation vectorielle $[v_a + w_0, v_b + w_1, v_b + w_2, v_c + w_3, v_a + w_4]$, les vecteurs v_x et w_n devant être de même dimension.

Les auteurs proposent ensuite deux méthodes pour calculer ces vecteurs de position : soit en apprenant ces vecteurs conjointement avec le modèle, soit en utilisant les fonctions périodiques $sinus$ et $cosinus$ avec plusieurs fréquences. L'idée de cette dernière méthode est ainsi d'assigner à chaque dimension du vecteur w_n une valeur suivant une sinusoïde de fréquence différente. Les auteurs utilisent ainsi les formules suivantes :

$$PE_{(pos,2i)} = sin\left(pos/10000^{2i/d_{model}}\right) \; PE_{(pos,2i+1)} = cos\left(pos/10000^{2i/d_{model}}\right)$$

Grâce à cette méthode, illustrée dans la figure 2.8, plutôt que d'apprendre les vecteurs de position sur un corpus d'entraînement, le modèle est capable d'extrapoler les représentations vectorielles pour des éléments dans des séquences de longueur plus grande que celles rencontrées pendant l'entraînement.

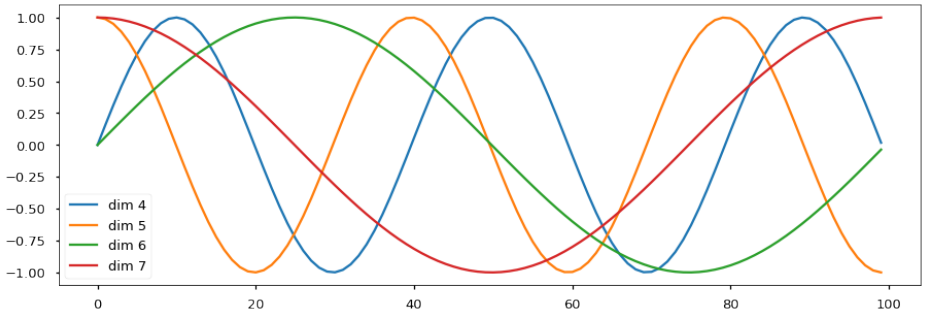


FIGURE 2.8 – Encodage des positions avec les fonctions $sinus$ et $cosinus$. Figure issue de l'article *The Annotated Transformer*. [7]

Avec ces deux changements, les cellules récurrentes ne sont plus nécessaires parce que chaque vecteur de mot contient l'information de sa propre position dans la séquence. L'encodeur et le décodeur peuvent ainsi exploiter cette information depuis n'importe quel autre mot de la séquence grâce au mécanisme d'attention, sans limite de distance.

2.2.4.3 Autres particularités de l'architecture Transformer

En plus de l'attention multi-tête et de l'encodage des positions, d'autres particularités font la robustesse et l'efficacité du modèle Transformer. Nous allons les présenter ici brièvement.

Feed-forward networks

Dans chaque encodeur et dans chaque décodeur Transformer, après le calcul de l'attention se trouve une couche réalisant deux opérations linéaires successives permettant d'ajouter des paramètres au modèle tout en conservant la même dimension en entrée et en sortie de chaque couche. Cette nouvelle couche, appelée *feed forward network* (FFN) réalise le calcul suivant :

$$FFN(x) = max\left(0, xW_1 + b_1\right) W_2 + b_2$$

7. https://nlp.seas.harvard.edu/2018/04/03/attention.html

Soit d_x la dimension du vecteur x donné en entrée, et d_{ff} une dimension arbitraire, W_1 est ainsi une matrice de dimension $d_x \times d_{ff}$, b_1 est un vecteur de dimension d_{ff}, W_2 est de dimension $d_{ff} \times d_x$ et b_2 est de dimension d_x. Ainsi, le résultat de $FFN(x)$ est toujours de dimension d_x.

Connexions résiduelles et normalisation

Pour chaque couche du Transformer (couche d'attention ou couche de FFN), deux techniques permettant d'accélerer l'entraînement sont appliquées : une connexion résiduelle, méthode proposée par He et al. (2016), et une normalisation de couche, méthode proposée par Ba et al. (2016). Le résultat est que pour chaque couche $Couche$, la sortie de la couche est finalement :

$$x = Norm(x + Couche(x))$$

Enfin, une dernière forme de normalisation est appliquée en sortie de chaque couche, le Dropout (Srivastava et al., 2014), avec une probabilité de $0,1$.

2.2.4.4 Améliorations et démocratisation de Transformer

L'architecture Transformer connaît plus tard des améliorations. Par exemple l'architecture Transformer-XL (Dai et al., 2019) ajoute une nouvelle façon d'encoder les positions de façon relative, et des cellules récurrentes supplémentaires permettent de conserver une mémoire d'une séquence à une autre. De même, l'architecture RNMT+ (Chen et al., 2018) reprend des éléments du Transformer avec des éléments de récurrence supplémentaires.

Cependant, l'architecture Transformer « classique » reste largement la plus utilisée, et c'est toujours elle qui est utilisée dans les systèmes qui sont en tête des principales campagnes d'évaluation en TA, par exemple sur les tâches de traduction anglais-français et anglais-allemand des campagnes d'évaluation WMT. [8] Plus globalement, l'architecture devient ainsi la base d'un grand nombre de réseaux de neurones pour traiter des séquences, notamment dans l'apprentissage non supervisé de modèles de langue pré-entraînés tels que BERT (Devlin et al., 2019) et tous ses successeurs : XLM, XLNet, RoBERTa, FlauBERT, CamemBERT, etc. (voir section 1.3.3.2).

8. https://paperswithcode.com/task/machine-translation. Consulté le 25/11/19

Dans nos contributions, nous utilisons aussi l'architecture Transformer dans nos systèmes de DL neuronaux (voir chapitre 5 et chapitre 6) et de TA neuronaux (voir chapitre 7 et chapitre 8).

2.2.5 Autres architectures neuronales notables

D'autres architectures neuronales notables ont été developpées pour la TA en parallèle du Transformer. Nous allons ici en passer quelques-unes en revue.

2.2.5.1 ConvS2S

ConvS2S (abbréviation de *Convolutional Sequence to Sequence*), est un système proposé par Gehring et al. (2017), qui s'appuie sur des convolutions pour remplacer les cellules récurrentes classiques. Comme pour le Transformer, des vecteurs de position sont ajoutés aux vecteurs de mot afin de permettre aux opérations de convolution successives de savoir l'ordre des mots dans la séquence. De plus, le remplacement des cellules récurrentes permet aussi de paralléliser le calcul et donc de gagner en efficacité.

Cependant, comme le montrent Vaswani et al. (2017), à nombre de paramètres équivalent, cette architecture a de moins bonnes performances que le Transformer. De plus, la taille du contexte pouvant être pris en compte est limitée par la taille des convolutions utilisées.

2.2.5.2 Pervasive attention

Pervasive Attention (Elbayad et al., 2018) est un système qui remplace complètement l'architecture encodeur-décodeur par une convolution à deux dimensions (une dimension « source » et une dimension « cible »).

Cette architecture se distingue par ses performances sur des petits modèles, car elle permet d'obtenir des résultats atteignant presque ceux de Transformer pour deux fois moins de paramètres. Cependant, elle n'atteint pas les performances état de l'art avec plus de paramètres.

2.2.5.3 Kermit

Kermit (Chan et al., 2019) est un système qui propose aussi une nouvelle architecture qui remplace complètement l'architecture encodeur-décodeur par un nouveau modèle génératif fondé sur Transformer et qui est capable de traduire dans les deux sens pour une même paire de langues.

L'architecture repose sur celles des modèles de langue type MLM (voir section 1.3.3.2), notamment BERT. La principale différence de Kermit avec ce dernier est qu'il est entraîné à prédire les mots masqués sur des paires de phrases de langues différentes. Ainsi, au décodage, il permet de prédire un mot dans une langue cible à partir des mots dans une langue source. Au niveau des résultats, Kermit obtient des scores globalement similaires au Transformer pour un même nombre de paramètres.

2.3 Gestion du vocabulaire

Dans les premiers systèmes de TA neuronaux de Sutskever et al. (2014) et Bahdanau et al. (2015), les vocabulaires d'entrée et de sortie utilisés sont limités aux n mots les plus fréquents du corpus d'entraînement utilisé. Par exemple, Sutskever et al. (2014) utilisent les 160 000 mots les plus fréquents dans la langue source et les 80 000 mots les plus fréquents dans la langue cible, tandis que Bahdanau et al. (2015) utilisent les 30 000 mots les plus fréquents dans les deux langues. Dans ces systèmes, les mots ne faisant pas partie de ce vocabulaire sont remplacés par un symbole spécial `<unk>`, ce qui permet à l'encodeur et au décodeur de modéliser tous les mots rares ou inconnus sous une même représentation vectorielle. Au moment d'évaluer, ces symboles sont généralement retirés.

Progressivement, d'autres méthodes sont apparues et ont permis d'améliorer cette gestion du vocabulaire. Dans cette section, nous allons décrire quelques-unes de ces méthodes.

2.3.1 Remplacement des mots hors vocabulaire

Luong et al. (2015b) proposent une phase de post-traitement applicable à n'importe quel système de TA neuronal, dans laquelle chaque symbole « <unk> » produit en sortie est remplacé par le mot de la phrase source avec lequel il est aligné, ou bien, si elle existe, une traduction de ce mot dans une table de traduction.

Pour cela, les auteurs utilisent un aligneur externe, celui de Liang et al. (2006), de deux manières : d'abord, pour générer la table de traduction, associant à chaque mot dans la langue source sa traduction la plus probable, et ensuite, pendant la phase de post-traitement du réseau neuronal, afin de trouver le mot dans la phrase source aligné à un symbole « <unk> » généré dans la sortie. Si le mot de la source existe dans la table de traduction, on l'utilise pour remplacer le symbole « <unk> », sinon, on copie le mot source tel quel.

Les auteurs ont ainsi appliqué leur méthode à un système similaire à celui de Sutskever et al. (2014), et ils observent ainsi une amélioration significative dans leurs scores.

En parallèle, Jean et al. (2015) proposent aussi une méthode similaire, mais en utilisant cette fois-ci directement le modèle d'alignement proposé par le mécanisme d'attention du système de Bahdanau et al. (2015) pour remplacer les mots inconnus en sortie, et ils obtiennent des gains similaires.

2.3.2 Découpage des mots en sous-unités

Les méthodes de remplacement des mots hors vocabulaire, avec ou sans dictionnaire, posent toujours certains problèmes, en particulier parce qu'elles présupposent généralement qu'un symbole « <unk> » dans la sortie correspond toujours à un seul mot dans la phrase d'entrée. D'une manière générale, ces méthodes fonctionnent raisonnablement bien sur des noms et des entités nommées présentes dans le corpus d'entraînement, et sur des paires de langues morphologiquement similaires. Elles sont cependant incapables de gérer certains cas, comme par exemple la translittération de nouvelles entités nommées ou la conjugaison de nouveaux mots à la volée.

Sennrich et al. (2016b) partent de ce constat, et proposent une autre méthode pour gérer les mots hors vocabulaire : le découpage des mots en sous-unités. Leur méthode, qui est une adaptation du Byte Pair Encoding (BPE) (Gage, 1994), une technique de compression des données, consiste à remplacer les paires de caractères consécutifs les plus fréquentes dans un texte par un nouveau symbole, puis de

réitérer le processus en considérant ces symboles comme des nouveaux caractères, et ainsi de suite, jusqu'à avoir une taille de vocabulaire souhaitée dans notre texte.

En effet, cette méthode a pour effet d'identifier les mots, et plus généralement les n-grammes de caractères, les plus fréquents dans un texte afin de leur assigner un symbole unique dans le vocabulaire, et de découper en suites de symboles les mots et n-grammes de caractères les moins fréquents. Ainsi, non seulement beaucoup de formes dérivées d'un mot peuvent se construire à partir du radical et de préfixes ou suffixes présents dans le vocabulaire, mais en plus, comme les caractères individuels font partie du vocabulaire, n'importe quel mot peut être découpé en suites de caractères dans le cas extrême où il ne repose sur aucune suite de caractères faisant partie du vocabulaire.

Cette méthode, ou sa variante similaire appelée WordPiece Model (WPM), développée en parallèle pour le système de traduction neuronal de Google (Wu et al., 2016), sera ensuite utilisée dans la plupart des travaux pour traiter les mots hors vocabulaire. On la retrouve notamment dans le système Transformer (Vaswani et al., 2017), avec un vocabulaire de 32 000 sous-unités de mots, partagé entre la langue source et la langue cible, mais aussi dans le système ConvS2S de Gehring et al. (2017), et plus généralement dans la grande majorité des systèmes état de l'art qui ont suivi.

Le découpage en sous-unités est aussi appliqué dans la plupart des modèles de langue pré-entraînés comme BERT et tous ses dérivés (voir section 1.3.3.2), de même que dans nos travaux sur la désambiguïsation lexicale et la traduction automatique (voir contributions).

2.3.3 Traduction factorisée

Une autre méthode permettant à la fois de réduire le vocabulaire de sortie des systèmes de TA neuronaux et de gérer la plupart des mots hors vocabulaire consiste à découper la prédiction des mots en deux : d'un côté on prédit les lemmes des mots et de l'autre on prédit des « facteurs » tels que leur partie du discours, leur genre, leur nombre, etc. Au moment de décoder, on reconstruit la forme de surface des mots grâce à leur lemme et à ces facteurs.

Cette méthode, nommée traduction factorisée, se retrouve dans des travaux bien avant la traduction neuronale, par exemple dans les travaux de Koehn et Hoang (2007). Elle a ensuite été exploitée plus récemment dans des travaux comme ceux de García-Martínez et al. (2017). Dans cet article, le découpage des mots en facteurs permet aux auteurs de générer 172 000 mots différents avec un vocabulaire de seulement 30 000 lemmes et 142 facteurs. De plus, leurs résultats montrent

une amélioration supérieure à un système utilisant un découpage en sous-unités de mots.

Cependant, la limitation principale de cette approche est qu'elle nécessite un outil externe pour extraire les lemmes et les facteurs de la langue cible, contrairement au découpage en sous-unités qui peut être appliqué à n'importe quelle langue sans connaissance préalable. De plus, cette méthode n'élimine pas complètement la présence de mots inconnus, par exemple dans le cas de néologismes ou d'entités nommées rares. Pour ces raisons, la traduction factorisée est en pratique peu utilisée pour réduire la taille des vocabulaires.

On peut noter toutefois qu'elle est utilisée dans certains travaux comme ceux de Sennrich et Haddow (2016) et ceux de Hadj Salah (2018) en plus des techniques vues précédemment, afin d'améliorer les performances des systèmes de TA. En effet, on peut voir dans certains cas que prédire à la fois des sous-unités de mots et des facteurs linguistiques, ou bien utiliser ces facteurs en entrée comme information supplémentaire permet d'obtenir de meilleurs résultats.

2.4 Ressources et leur exploitation

Les systèmes de TA statistiques et neuronaux nécessitent au moins un type de ressource pour leur apprentissage : des corpus parallèles, c'est-à-dire des corpus écrits en plusieurs langues et généralement alignés au niveau de la phrase. En plus des corpus parallèles, certains systèmes s'appuient aussi sur des corpus monolingues, soit pour améliorer la représentation des mots en entrée, soit pour améliorer la qualité des sorties générées. Enfin, certains systèmes utilisent des corpus comparables, c'est-à-dire du texte écrit en plusieurs langues et qui traite d'un même sujet, mais qui n'est pas forcément aligné.

Dans cette section, nous allons faire un tour d'horizon de ces différentes ressources et des quantités nécessaires au fonctionnement de systèmes de TA neuronaux. Nous évoquerons le problème des langues peu dotées, pour lesquelles ces ressources sont rares, ainsi que quelques méthodes utilisées pour contourner ce problème.

2.4.1 Corpus parallèles

Les corpus parallèles, aussi appelés corpus bilingues, sont des ensembles de textes écrits en plusieurs langues et généralement alignés au niveau de la phrase. Ces textes sont habituellement issus d'une traduction manuelle. Ils proviennent par exemple de textes de lois, comme le corpus EuroParl (Koehn, 2005) qui contient les décisions du parlement européen en 21 langues, ou le corpus MultiUN (Ziemski et al., 2016) des Nations unies. Ils peuvent aussi venir par exemple de traductions de sous-titres créés pour des films ou séries, comme dans le corpus OpenSubtitles (Lison et Tiedemann, 2016), ou encore de fichiers de localisation de certains logiciels (Gnome, KDE, OpenOffice...).

Les corpus parallèles sont la matière première des systèmes de TA neuronaux pour leur apprentissage ainsi que pour leur évaluation. Ils sont en effet indispensables pour entraîner le système à passer d'une langue à une autre au moment de l'apprentissage, et pour comparer les sorties d'un système avec une référence au moment de l'évaluation.

Si les premiers systèmes IBM (Brown et al., 1990, 1991) s'appuyaient uniquement sur un corpus parallèle anglais-français du parlement canadien (le Hansard), contenant seulement 40 000 phrases, les systèmes actuels en utilisent généralement bien plus. Par exemple, le système Transformer (Vaswani et al., 2017) est entraîné sur plusieurs corpus anglais-français comprenant plus de 36 millions de phrases parallèles.

En effet, pour les langues dites « bien dotées », c'est-à-dire des langues pour lesquelles il existe une grande quantité de corpus parallèles de bonne qualité (typiquement l'anglais, le français, l'allemand, le chinois, etc.), on peut trouver des corpus parallèles très facilement. À titre d'exemple, la collection OPUS de textes parallèles [9] liste, pour la paire de langues français-anglais, plus de 400 000 documents provenant de multiples sources, comprenant plus de 160 millions de phrases parallèles.

Cependant, pour les autres langues dites « peu dotées », la quantité de ressources disponibles est beaucoup plus limitée. Par exemple, toujours dans la collection OPUS, la paire de langues français-breton contient seulement 1 300 documents, dont la plupart sont des fichiers de localisations de logiciels, totalisant 400 000 phrases. Si on enlève les fichiers de localisations, on tombe à moins de 100 000 phrases. Plus généralement, pour la plupart des paires de langues, la situation est même pire. Pour n'en citer que quelques-unes, on voit très vite que pour des paires de langues comme le français-bouriate, le français-achinais, le français-

9. http://opus.nlpl.eu/

akan, etc. très peu de données sont recensées, alors que ces langues comptent pourtant des centaines de milliers voire plusieurs millions de locuteurs.

2.4.2 Méthodes pour contourner le manque de corpus parallèles

Pour les langues peu dotées, qui constituent pourtant la majorité des langues existantes, diverses techniques doivent être mises en œuvre si l'on veut créer un système de TA neuronal, afin de contourner ce problème de manque de données parallèles. Nous allons ici en décrire quelques-unes.

2.4.2.1 Apprentissage à partir de corpus comparables

Les corpus comparables sont, comme les corpus parallèles, des ensembles de textes multilingues traitant d'un sujet commun, mais à l'inverse de ces derniers, les corpus comparables ne sont pas forcément alignés au niveau de la phrase. Par exemple, beaucoup d'articles Wikipédia sont écrits en plusieurs langues, et ils sont des sources de données riches et peuvent être facilement exploités dans les travaux de recherche sur les langues peu dotées.

Ramesh et Sankaranarayanan (2018) constatent ainsi que pour la paire de langues anglais-ourdou par exemple, à la date d'écriture de leur article, l'ensemble des corpus parallèles disponibles est composé de seulement 35 916 phrases, alors que 124 078 articles Wikipédia bilingues existent.

Comme dans d'autres travaux (par exemple Irvine et Callison-Burch (2013)), ils proposent ainsi une méthode pour extraire des phrases parallèles de ces articles bilingues, afin d'ajouter ces phrases au petit ensemble de données d'apprentissage d'origine pour un système de TA neuronal.

Leur méthode consiste dans un premier temps à entraîner un réseau de neurones à reconnaître si deux phrases sont des traductions l'une de l'autre, à partir des données parallèles disponibles, et dans un deuxième temps, à appliquer ce réseau de neurones sur toutes les phrases d'articles Wikipédia bilingues, afin d'extraire toutes les phrases que le réseau considère comme être des traductions. Ces phrases sont ensuite ajoutées aux données d'entraînement d'un système de TA classique.

2.4.2.2 Apprentissage par transfert

L'apprentissage par transfert est une méthode qui consiste à entraîner un système sur une tâche, puis transférer ses connaissances pour faciliter l'apprentissage sur une autre tâche.

Zoph et al. (2016) proposent ainsi une méthode d'apprentissage par transfert qui consiste à d'abord entraîner un système de TA neuronal sur une paire de langues bien dotées : le français-anglais, puis à poursuivre l'entraînement sur une paire de langues dont l'une des deux est peu dotée : l'ouzbek-anglais.

D'une manière plus générale, l'apprentissage par transfert est une méthode qui gagne de plus en plus en popularité, en attestent les récentes avancées sur les modèles de langue pré-entraînés (voir section 1.3.3.2). En effet, on peut voir ces modèles de langue comme des réseaux de neurones génériques, entraînés sur des tâches générales (prédire un mot), dont les connaissances peuvent être transférées à des tâches plus spécifiques.

Dans nos contributions sur la TA (voir chapitre 7 et chapitre 8) nous tirons ainsi parti indirectement d'une méthode d'apprentissage par transfert en nous appuyant sur le modèle de langue pré-entraîné BERT (Devlin et al., 2019).

2.4.2.3 Apprentissage *zero-shot*

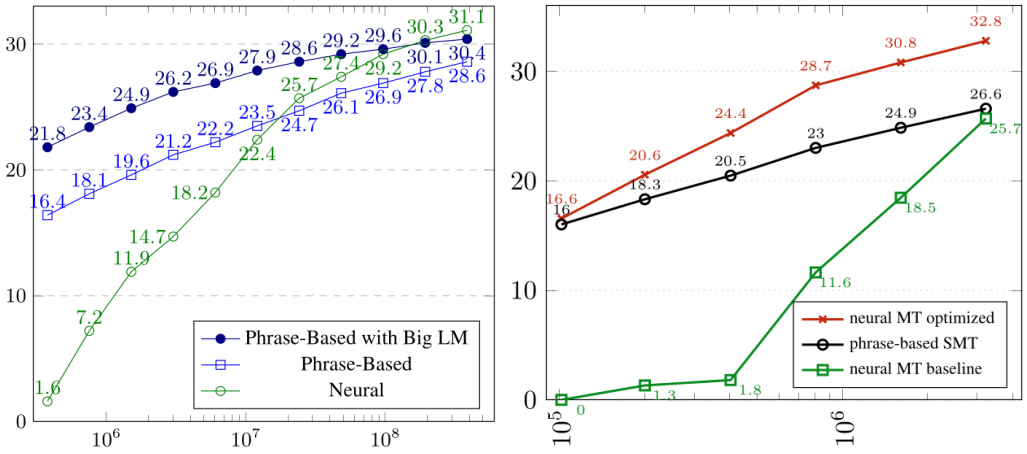
L'apprentissage *zero-shot* est une méthode proche de l'apprentissage par transfert, dont l'idée générale consiste à faire en sorte qu'un modèle puisse être capable d'apprendre à prédire des données qu'il n'a jamais vues pendant l'entraînement. En traduction automatique, on parle ainsi de traduction *zero-shot* lorque le système est capable de traduire du texte d'une langue à une autre, sans avoir observé de données parallèles de ces deux langues.

Dans les travaux de Johnson et al. (2017) portant sur le système de TA neuronal de Google, les auteurs présentent un modèle multilingue unique permettant de traduire depuis une phrase dans plusieurs langues sources, vers une phrase dans plusieurs langues cibles. Concrètement, leur système a la même architecture que leur précédent modèle bilingue (Wu et al., 2016), sauf que pour l'apprentissage, un symbole spécial est donné en entrée du décodeur afin de lui indiquer dans quelle langue traduire un texte. L'encodeur, lui, est inchangé, et il est capable de lire du texte source dans n'importe quelle langue.

Les auteurs font ainsi une expérience d'apprentissage *zero-shot* qui consiste à entraîner un premier système sur du texte portugais-anglais et anglais-espagnol, et

un deuxième système sur du texte anglais-espagnol, anglais-portugais, portugais-anglais et espagnol-anglais. Ils évaluent ensuite leurs systèmes sur une tâche de traduction portugais-espagnol, paire de langues jamais directement vue pendant l'entraînement. Ils constatent des résultats qui sont inférieurs à un système entraîné sur des données portugais-espagnol directement, mais qui sont tout de même intéressants : jusqu'à 24,75 points de BLEU contre 31,50 pour le système entraîné directement sur les données parallèles.

2.4.2.4 Un retour aux systèmes statistiques classiques ?



(a) Figure de Koehn et Knowles (2017)

(b) Figure de Sennrich et Zhang (2019)

FIGURE 2.9 – Comparaison des performances entre systèmes de TA statistiques classiques (*phrase-based SMT*) et systèmes de TA neuronaux (*neural MT*), en fonction de la quantité de données d'apprentissage utilisée (nombre de mots).

Pour finir cette section sur les méthodes permettant de contourner le manque de données parallèles, nous souhaitions mentionner que l'on considère souvent qu'en-dessous d'un certain seuil de données disponibles, les systèmes de TA neuronaux sont moins performants que les systèmes de TA statistiques classiques. Cette idée est en effet démontrée dans l'article de Koehn et Knowles (2017) (voir figure 2.9a), puis elle est reprise dans d'autres travaux, comme par exemple ceux de Lample et al. (2018).

Toutefois, l'article plus récent de Sennrich et Zhang (2019) offre une critique de ces travaux, et les auteurs montrent que le choix des hyperparamètres utilisés par Koehn et Knowles (2017) pour le système neuronal n'est pas pertinent dans ce contexte. Ils montrent ainsi qu'avec un bon choix, les systèmes neuronaux

sont systématiquement supérieurs aux systèmes statistiques, même avec très peu de données. Plus précisément, comme on peut le voir dans la figure 2.9b, même avec seulement 5 000 phrases parallèles (100 000 mots dans la langue source) le réseau neuronal est déjà capable de produire une traduction d'aussi bonne, voire de meilleure qualité que le système statistique.

Cependant, si l'on met de côté les performances de ces systèmes selon une mesure automatique, il est important de garder à l'esprit que les systèmes de TA statistiques classiques offrent toujours dans certains cas une certaine robustesse plus importante que les systèmes de TA neuronaux. En effet, certaines études comme celle de Belinkov et Bisk (2018) ont montré que ces derniers peuvent être extrêmement sensibles au bruit, et produire du texte qui ne reflète absolument pas une phrase donnée en entrée si elle s'éloigne trop des données d'entraînement. Les systèmes de TA statistiques, eux, comme ils s'appuient sur des tables de traductions interprétables plus facilement, ont un comportement beaucoup plus prédictible.

2.4.3 Corpus monolingues

Les corpus monolingues sont un autre type de ressource souvent utilisé en TA. En effet, dès les premiers travaux sur la TA statistique et les modèles IBM (Brown et al., 1990), l'approche proposée consiste à calculer d'un côté la probabilité $p(x|y)$, qui se calcule grâce à un modèle de traduction entraîné sur des corpus parallèles, et de l'autre côté la probabilité $p(y)$ qui se calcule grâce à un modèle de langue pouvant être entraîné séparément, soit sur la partie monolingue des corpus parallèles, soit sur des corpus monolingues externes.

En TA neuronale, un unique réseau neuronal modélise conjointement les deux modèles (de traduction et de langue). Les données monolingues ne sont donc pas nécessaires, mais plusieurs travaux tels que ceux de Gulcehre et al. (2015), Sennrich et al. (2016a) et Edunov et al. (2018a) montrent différentes façons de les exploiter afin d'améliorer les performances d'un système de TA neuronal.

2.4.3.1 Intégration d'un modèle de langue externe

Gulcehre et al. (2015) proposent ainsi d'utiliser des corpus monolingues pour entraîner un modèle de langue neuronal qui prédit un mot en fonction des mots vus précédemment. Ce modèle est ensuite intégré au décodeur d'un système de TA avec attention comme celui de Bahdanau et al. (2015). Deux possibilités sont proposées pour l'intégration :

— Soit « en surface », en moyennant les prédictions du décodeur du modèle de TA et celles du modèle de langue, pour chaque nouveau mot à prédire. Pendant l'entraînement, seul le modèle de TA est entraîné et le modèle de langue est figé.

— Soit « en profondeur », en concaténant les états cachés des deux réseaux. Le modèle résultant est ensuite entraîné entièrement avec le reste du modèle de TA.

Les auteurs expérimentent leur méthode sur quatre paires de langues avec des tailles de données d'apprentissage variables : le chinois-anglais (environ 436 000 phrases parallèles), le turc-anglais (environ 160 000), le tchèque-anglais (12,1 millions) et l'allemand-anglais (4,1 millions). Le modèle de langue anglais, qui est utilisé dans tous les cas, est entraîné sur le corpus English Gigaword (Graff et al., 2003) contenant plus de quatre millions de documents.

Dans quasiment tous les cas, l'ajout de ce modèle de langue entraîné sur des corpus monolingues améliore la qualité de traduction des systèmes de TA. De plus, c'est l'intégration qu'ils appellent « en profondeur » qui fonctionne le mieux.

2.4.3.2 Rétrotraduction

Une autre méthode plus récente d'utilisation des corpus monolingues en TA neuronale consiste à simplement considérer ces données monolingues dans la langue cible comme des données d'entraînement auxquelles il manquerait la source. Cette idée s'inscrit essentiellement dans la continuité de celle de Gulcehre et al. (2015) qu'on a vue précédemment, sauf qu'au lieu de chercher à améliorer seulement la partie décodeur du modèle neuronal, on cherche à améliorer l'ensemble du réseau.

Sennrich et al. (2016a) proposent ainsi une méthode appelée rétrotraduction (*back-translation*), dans laquelle on va utiliser un système de TA pour traduire ces données monolingues vers la langue source, et ainsi utiliser ces nouvelles données parallèles « synthétiques » exactement comme les données parallèles originales.

Les auteurs évaluent sur les paires de langues allemand<->anglais et turc->anglais, et constatent une amélioration par rapport à la méthode de Gulcehre et al. (2015).

La rétrotraduction est aujourd'hui une méthode utilisée dans toutes les campagnes d'évaluation pour obtenir le meilleur système possible. On peut s'en rendre compte notamment en consultant ce site [10] qui recense les scores des meilleurs systèmes de TA, pour les tâches anglais-français et anglais-allemand de la campagne d'évaluation WMT 2014, deux tâches utilisées pour évaluer la plupart des modèles les plus importants qu'on a vus (Wu et al., 2016; Vaswani et al., 2017; Chan et al., 2019; Chen et al., 2018; Gehring et al., 2017), ce sont les systèmes de Edunov et al. (2018a) qui arrivent en tête, en ajoutant plus de 30 millions de phrases synthétiques à leur corpus d'entraînement composé de 35 millions de phrases.

Il est à noter cependant avec ce dernier exemple que, pour entraîner un système sur autant de données, les auteurs ont utilisé 128 GPU NVidia V100 pendant plus de 27 heures. À titre de comparaison, un GPU Nvidia V100 est environ 1,7 à 2 fois plus performant qu'un GPU NVidia 1080 Ti [11], et notre équipe de recherche se partage, à l'heure d'écriture de cette thèse, une quinzaine de ces derniers. Si nous voulions reproduire ces expériences, il nous faudrait, dans le meilleur des cas, monopoliser tous les 15 GPU pendant 392 jours.

2.5 Évaluation

Comme on a pu le voir, l'objectif des systèmes de TA est de produire des traductions les plus proches possibles de celles que pourraient produire un traducteur humain. L'évaluation des systèmes de TA est ainsi une tâche complexe, parce qu'une traduction est forcément subjective : deux traducteurs peuvent traduire une même phrase de deux façons différentes, en fonction de leurs intentions, leur culture, leur style, etc. sans qu'une des deux traductions soit meilleure que l'autre.

Pour illustrer la difficulté d'évaluer un système de TA, prenons la phrase anglaise suivante : « *It is scary.* », et deux traductions françaises possibles : « *C'est effrayant.* » ou bien « *Ça fait peur.* ». Les deux traductions sont tout à fait acceptables, et pourtant, elles n'ont aucun mot en commun. Comment alors donner un score permettant d'évaluer le système qui aura produit l'une ou l'autre des traductions ?

Dans les campagnes d'évaluation telles que WMT et IWSLT, l'évaluation des

10. https://paperswithcode.com/task/machine-translation
11. https://lambdalabs.com/blog/best-gpu-tensorflow-2080-ti-vs-v100-vs-titan-v-vs-1080-ti-benchmark/

systèmes concurrents se fait en deux temps : d'abord avec des métriques d'évaluation automatiques, qui comparent les sorties des systèmes avec une ou plusieurs références produites manuellement, et ensuite avec des méthodes d'évaluation manuelles, pour lesquelles un groupe de traducteurs professionnels va soit évaluer directement la qualité d'une traduction en lui assignant un score, soit compter le nombre de modifications à apporter à la traduction jusqu'à obtenir une traduction acceptable.

Dans les articles présentant de nouveaux systèmes de TA par contre, l'évaluation de ces systèmes se fait généralement avec des métriques automatiques telles que BLEU et TER uniquement. On utilise en principe les mêmes références que celles utilisées dans les campagnes d'évaluation afin de comparer les systèmes entre eux.

Dans cette section, nous allons décrire les principales métriques d'évaluation automatiques ainsi que les méthodes d'évaluation manuelles utilisées dans les campagnes d'évaluation.

Il est à noter cependant que, bien que le terme *métrique* soit systématiquement employé en TA, il ne correspond pas strictement à son sens mathématique. En particulier, les métriques traditionnelles en TA ne satisfont pas la propriété de l'inégalité triangulaire des métriques au sens strict du terme.

2.5.1 Métriques d'évaluation humaines

Il existe de nombreux standards pour évaluer manuellement la qualité de traduction des systèmes de TA. Par exemple, dans l'article de White et al. (1994) qui décrit l'évolution des méthodes d'évaluation utilisées dans le programme de TA de l'ARPA [12], la dernière méthode utilisée consiste à demander à des évaluateurs de noter des traductions selon trois critères :

1. L'adéquation, une note comprise entre 1 et 5 qui évalue la quantité d'informations effectivement transmise entre la phrase originale et sa traduction.
2. La fluidité, une note comprise entre 1 et 5 qui évalue la qualité de la phrase traduite indépendamment de la phrase originale.
3. L'informativité, sous la forme d'un questionnaire à choix multiples, qui permet d'évaluer si la traduction contient suffisamment d'informations pour permettre à quelqu'un de déduire toutes les informations nécessaires à sa compréhension.

12. L'ARPA est l'agence de recherche du département de la Défense des États-Unis.

Dans les campagnes d'évaluation pour la TA, par exemple dans la campagne d'évaluation WMT 2007 (Callison-Burch et al., 2007), les mesures d'adéquation et de fluidité sont aussi utilisées, ainsi qu'une troisième mesure : le classement, qui évalue relativement entre eux les systèmes soumis à la tâche de TA. Les accords intra-annotateurs (cohérence d'un même annotateur) et inter-annotateurs (cohérence entre les annotateurs) sont aussi calculés.

Pour finir, on peut aussi citer la mesure semi-automatique HTER (Snover et al., 2006), qui consiste à d'abord éditer les sorties d'un système de TA jusqu'à obtenir des traductions estimées correctes, puis à calculer une métrique automatique (ici le TER) sur cette nouvelle référence. Nous détaillerons plus la mesure TER dans la section suivante.

2.5.2 Métriques d'évaluation automatiques

Les métriques d'évaluation automatiques permettent de mesurer automatiquement la performance des systèmes de TA en comparant leurs sorties avec des références produites par des humains. Nous allons ainsi détailler ici trois métriques couramment utilisées : la mesure BLEU (Papineni et al., 2002), Meteor (Banerjee et Lavie, 2005) et TER (Snover et al., 2006).

2.5.2.1 BLEU

La métrique BLEU, pour *BiLingual Evaluation Understudy*, est une métrique présentée par Papineni et al. (2002) qui consiste à mesurer une moyenne des précisions des n-grammes de mots, pour n compris entre 1 et 4, entre une hypothèse H et une ou plusieurs références R_i, et de la combiner à un facteur qui pénalise les phrases trop courtes. Sa formule est la suivante :

$$\text{BLEU} = \text{BP} \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4}\log(p_n)\right)$$

où BP est la pénalité pour les phrases trop courtes, calculée de la manière suivante :

$$\text{BP} = \begin{cases} 1 & \text{si } |H| > |R| \\ e^{\left(1-\frac{|R|}{|H|}\right)} & \text{si } |H| \leq |R| \end{cases}$$

Et p_n est la précision des n-grammes de mots, c'est-à-dire le nombre de n-grammes de mots présents à la fois dans H et dans une des références R_i, divisé par le

nombre total de n-grammes dans H. Lorsque plusieurs références R sont données, la mesure de BP prend en compte la longueur de phrase la plus proche de H.

Cette métrique est devenue très populaire en particulier grâce à sa facilité de calcul, mais aussi parce que les auteurs montrent qu'appliquée à l'échelle d'un document entier, la métrique obtient une bonne corrélation avec le jugement humain. Aujourd'hui, même si d'autres mesures ont depuis une meilleure corrélation avec le jugement humain, la métrique BLEU est utilisée dans la quasi-totalité des travaux sur la TA, comme on peut le voir par exemple dans les soumissions de la dernière campagne d'évaluation WMT (Barrault et al., 2019).

2.5.2.2 Meteor

Meteor, pour *Metric for Evaluation of Translation with Explicit ORdering* est une métrique plus récente présentée par Banerjee et Lavie (2005) qui vise à améliorer BLEU en intégrant plusieurs modules permettant de comparer les mots au-delà de leur forme de surface, en comparant aussi leur racine lexicale et leur sens.

Plus précisément, Meteor intègre plusieurs modules qui interviennent lors de différentes phases. Le module *Exact* compare ainsi d'abord les mots suivant leur forme de surface, puis le module *Stem* compare les mots suivant leur racine calculée avec l'algorithme de Porter (Porter, 1980), enfin le module *Synonym* compare les sens des mots à partir de WordNet (Miller et al., 1990), en considérant que deux mots sont synonymes s'ils ont un sens en commun dans l'inventaire de sens.

Les auteurs ont ainsi pu montrer que Meteor obtenait une bien plus grande corrélation avec le jugement humain que BLEU. Cependant, la mesure est tout de même limitée du fait qu'elle nécessite des ressources spécifiques qui ne sont pas disponibles dans toutes les langues, notamment WordNet.

Meteor permet, grâce à sa décomposition en plusieurs modules, une grande flexibilité. En effet, plusieurs travaux ont cherché à améliorer la métrique en intégrant des nouveaux modules ou en améliorant les modules existants. Ainsi, Servan et al. (2016) proposent deux modifications au Meteor standard : d'abord, l'utilisation d'une base de données lexicale multilingue, DBNary, à la place de WordNet, afin d'étendre le module *Synonym* à d'autres langues que l'anglais, et ensuite, l'utilisation de la mesure de distance entre vecteurs de mot en remplacement de la comparaison des modules *Stem* et *Synonym*.

Apidianaki et Marie (2015) remplacent le module *Synonym* par un module *WSD* qui désambiguïse d'abord les mots de l'hypothèse et des références, afin de comparer plus finement les sens WordNet. À noter que nous avons aussi contribué à

intégrer de la désambiguïsation lexicale dans Meteor, durant mes travaux de master (Vial, 2016).

2.5.2.3 TER

La métrique TER, pour *Translation Edit Rate* [13] qu'on peut traduire comme taux d'édition de traduction, est une métrique proposée par Snover et al. (2006) qui se veut plus intuitive et simple à calculer que BLEU, et ayant une meilleure corrélation avec le jugement humain, sans utiliser de ressources langagières comme Meteor.

On peut décrire cette métrique comme étant le nombre d'opérations d'éditions à appliquer à une hypothèse afin d'arriver à la référence. Les quatre opérations possibles d'édition sont l'insertion, la suppression et la subtitution d'un unique mot, ainsi que le décalage d'une séquence de mots. Cette mesure est ainsi intuitive, car son résultat est interprétable à l'échelle humaine. En effet, le TER peut être vu comme la quantité d'éditions qu'un humain appliquerait pour changer une hypothèse en une référence. On peut donc déduire le temps que prendrait une telle édition dans le cadre de traductions professionnelles par exemple.

En plus de la mesure TER, les auteurs Snover et al. (2006) proposent le HTER, pour *Human-targeted TER*, une mesure semi-automatique qui nécessite que des traducteurs bilingues post-éditent les phrases à évaluer jusqu'à obtenir des phrases qu'ils jugent correctes, et où l'on calcule le TER entre les hypothèses de départ et ces nouvelles références post-éditées.

Les auteurs mesurent finalement les corrélations avec le jugement humain avec les mesures TER, BLEU et Meteor, ainsi que le HTER et le HBLEU et HMeteor (en appliquant la mesure, à chaque fois, entre l'hypothèse et une version post-éditée de l'hypothèse). Ils remarquent que, pour les métriques automatiques, la mesure TER se situe entre BLEU et Meteor. Cependant, entre HBLEU, HMeteor et HTER, c'est cette dernière qui obtient la plus forte corrélation avec le jugement humain.

13. La métrique est aussi parfois appelée *Translation **Error** Rate*, en référence au *Word Error Rate* employé en reconaissance de la parole, mais les auteurs préfèrent le terme ***Edit** Rate*.

2.5.2.4 Évaluation et performances des métriques

Comme on l'a vu, les métriques peuvent être elles-mêmes évaluées en calculant leur corrélation avec le jugement humain. Dans la plupart des campagnes d'évaluation WMT, en plus des tâches d'évaluation de la TA, des tâches proposent ainsi de soumettre des métriques d'évaluation, et ces métriques sont utilisées pour attribuer des scores à des sorties de systèmes de TA. On compare ensuite ces scores avec des scores attribués par des humains, et on mesure ainsi leur corrélation avec le jugement humain en utilisant généralement la formule de corrélation de Pearson ou celle de Spearman.

Ainsi, sur la tâche d'évaluation des métriques de la campagne d'évaluation WMT 2019 (Ma et al., 2019), sur quatres paires de langues, les corrélations de Pearson des trois métriques que nous avons décrites se trouvent dans le tableau 2.1.

Paire de langues	BLEU	TER	Meteor
allemand-anglais	84,9	87,4	**89,6**
chinois-anglais	89,9	84,0	**95,2**
anglais-allemand	92,1	**96,9**	-
anglais-chinois	**90,1**	85,6	-

TABLE 2.1 – Corrélation de Pearson entre le jugement humain et les métriques d'évaluation automatiques BLEU, TER et Meteor sur quatre paires de langues.

Il est à noter que ces corrélations sont calculées en appliquant les métriques à l'échelle du document entier à traduire (environ 2 000 phrases), et pas au niveau de la phrase. En effet, si l'on applique ces métriques à l'échelle des phrases, la corrélation dépasse rarement les 10%. Une autre chose à noter est que les métriques ont été appliquées sur des phrases avec une unique référence. Cependant, toutes les métriques que l'on a vues permettent de comparer une hypothèse avec un ensemble de références, en considérant généralement la référence la plus proche de l'hypothèse. Dans le cas où plusieurs références sont fournies, les métriques obtiennent de bien meilleures performances.

Pour récapituler cette partie sur les performances des métriques d'évaluation de TA, même si aucune métrique automatique n'obtient de corrélation parfaite avec le jugement humain, et même si leurs performances sont médiocres lorsqu'elles sont appliquées au niveau de phrases individuelles, elles sont tout à fait acceptables lorsqu'elles sont utilisées au niveau d'un document constitué de quelques milliers de phrases.

Pour cette raison, les métriques automatiques comme BLEU, Meteor et TER

sont largement utilisées dans les travaux sur la TA ainsi que dans les campagnes d'évaluation et elles permettent quand même de conclure à des améliorations de performances de systèmes de TA, en particulier si plusieurs références sont utilisées. Même si Meteor a généralement de meilleures performances que BLEU et TER, la métrique n'est utilisable, dans sa version standard, que sur l'anglais, ce qui fait qu'elle est au final très peu utilisée dans les travaux de recherche.

Concernant BLEU et TER, même si cette dernière montre de meilleures performances sur la plupart des paires de langues (voir tous les résultats de Ma et al. (2019)) et qu'elle a l'avantage d'être facilement interprétable, BLEU est tout de même plus utilisée dans les travaux sur la TA.

2.5.2.5 Normalisation des données

Pour finir cette partie sur les métriques d'évaluation automatique, il est important de noter qu'au moment de comparer les sorties d'un système de TA avec des références, la méthode de segmentation de la ponctuation, la normalisation et la casse des caractères doivent être respectées sous peine d'influer grandement sur le score final (BLEU, TER ou Meteor). En effet, ces trois paramètres peuvent varier énormément selon les paires de langues et les corpus d'évaluation.

Lors des campagnes d'évaluation, il est souvent ainsi précisé si un type de normalisation a été appliqué aux données de référence. En particulier, il est parfois précisé quel symbole est utilisé pour représenter les guillemets (" ", ‘ ’ ou « », etc.). Certaines tâches précisent aussi que l'évaluation sera insensible à la casse. Enfin, certaines tâches utilisent des références dans lesquelles la ponctuation est segmentée alors que d'autres non.

2.5.3 Campagnes d'évaluation

Afin d'évaluer les systèmes de TA et de les comparer entre eux, des campagnes d'évaluation sont organisées régulièrement. Parmi les campagnes les plus connues on peut notamment citer WMT et IWSLT.

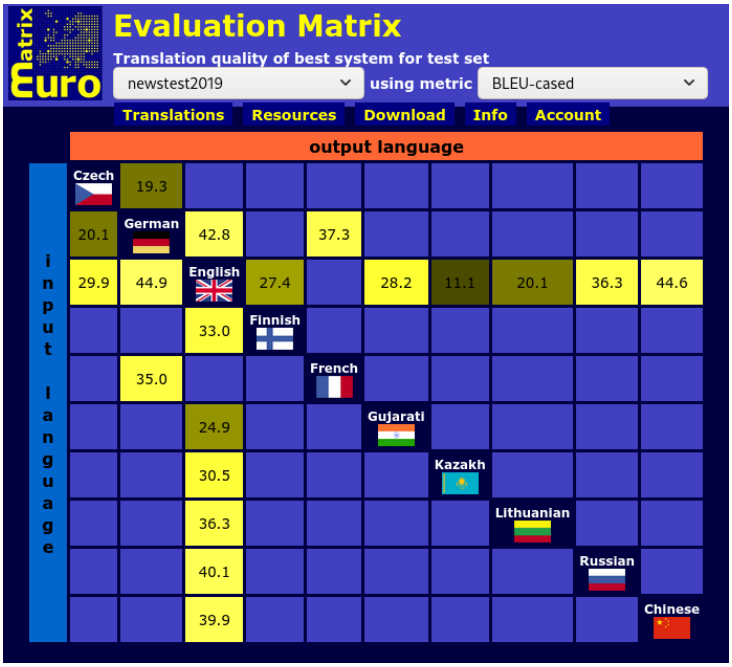


FIGURE 2.10 – Matrice des langues. Capture du site http://matrix.statmt.org pour la campagne d'évaluation WMT 2019 et la métrique BLEU.

2.5.3.1 WMT

WMT, abbréviation de *Workshop on Statistical Machine Translation* est sans doute la campagne d'évaluation la plus importante en termes de nombre de participants. Elle est organisée tous les ans depuis 2006 au sein des conférences ACL, NAACL, EACL ou EMNLP, et elle a été un témoin important de la soudaine popularité des systèmes neuronaux. En effet, lors de WMT 2015 (Bojar et al., 2015), un seul système ayant participé était fondé sur un réseau de neurones, alors que l'année suivante, quasiment tous les gagnants l'étaient (Bojar et al., 2016).

Certaines tâches de traduction de WMT sont aujourd'hui incontournables pour l'évaluation de nouveaux systèmes de TA à grande échelle, en particulier les tâches anglais-français et anglais-allemand de WMT 2014 (Bojar et al., 2014), qui sont systématiquement utilisées dans les travaux majeurs (Bahdanau et al., 2015; Cho et al., 2014; Gehring et al., 2017; Jean et al., 2015; Johnson et al., 2017; Sutskever et al., 2014; Vaswani et al., 2017).

Cependant, les systèmes évalués sur ces tâches utilisent généralement un très grand nombre de phrases parallèles (autour de 40 millions pour la tâche anglais-français et de 4 millions pour la tâche anglais-allemand), ce qui peut rendre l'en-

traînement de systèmes pour ces tâches assez long pour l'expérimentation de nouvelles architectures.

À l'issue des campagnes d'évaluation WMT, on peut retrouver sur le site Web `http://matrix.statmt.org` une matrice résumant les scores de tous les participants aux tâches, pour toutes les paires de langues évaluées. Cette matrice, illustrée dans la figure 2.10, met en avant les meilleurs scores suivant une métrique choisie (BLEU, TER, etc.) et permet de rapidement voir les langues qui sont les plus exploitées.

2.5.3.2 IWSLT

Une autre campagne d'évaluation, l'*International Workshop on Spoken Language Translation* (IWSLT), qui se focalise sur la transcription et la traduction de la parole plutôt que sur le texte, contient elle aussi des tâches de traduction automatique, à partir de transcriptions de textes oraux. Dans certains travaux récents (Elbayad et al., 2018; Deng et al., 2018; Edunov et al., 2018b), ces tâches sont ainsi utilisées afin d'évaluer plus rapidement et sur un ensemble restreint de données (environ 100 000 phrases), les performances de nouvelles architectures neuronales pour la TA.

Nous avons ainsi participé à la campagne d'évaluation IWSLT 2019 (Niehues et al., 2019), sur la paire de langues anglais-tchèque (plus de détails dans le chapitre 7).

2.6 Traduction automatique et désambiguïsation lexicale

Les liens entre la traduction automatique et la désambiguïsation lexicale semblent, au premier abord, évidents. En effet, on peut imaginer que le décodeur d'un système de TA fera un meilleur choix lexical, au moment de produire un mot, s'il connait le sens exact des mots de la source. En pratique pourtant, très peu de travaux tentent d'intégrer de la DL à des systèmes de TA, et le bénéfice apporté n'est pas toujours évident.

Dans cette section, nous allons dans un premier temps passer en revue les différentes méthodes d'intégration de la DL pour la TA, puis dans un second temps parler de la capacité de désambiguïsation propre aux systèmes de TA, sans le besoin d'un système de DL externe.

2.6.1 Enrichissement des systèmes de traduction avec des sens

Une manière évidente pour enrichir un système de TA avec des sens donnés par un système de DL consiste à désambiguïser le texte source en amont, puis à se servir de ces prédictions pour influencer le comportement du système de TA.

Dans cette section, nous allons voir différentes méthodes pour atteindre cet objectif et que nous allons séparer en deux groupes : d'un côté les méthodes dites pré-neuronales, c'est-à-dire dans lesquelles le système de TA utilise une architecture statistique classique telle qu'abordée dans la section 2.1.2, et de l'autre côté les méthodes neuronales, qui s'appuient donc sur un système de TA neuronal comme évoqué dans la section 2.2.

2.6.1.1 Enrichissement sémantique des systèmes pré-neuronaux

On peut trouver les premières méthodes d'intégration d'un système de DL au sein d'un système de TA statistique dans la thèse de Carpuat (2008) et dans ses travaux associés (Carpuat et Wu, 2005, 2007), ainsi que dans les travaux de Cabezas et Resnik (2005) et de Chan et al. (2007a).

Dans la thèse de Carpuat (2008), un système de DL est construit spécialement pour une tâche « échantillon lexical » (voir figure 1.5.1.1) du chinois, utilisant l'inventaire de sens HowNet [14], puis un système de TA statistique classique est entraîné sur des corpus parallèles chinois-anglais.

En s'appuyant sur le fait que les étiquettes de sens de HowNet ont à la fois une description textuelle en chinois et en anglais, le système de TA est modifié pour prendre en compte les prédictions du système de DL appliqué sur le texte source chinois. Pour cela, les auteurs comparent plusieurs méthodes :

1. En forçant le modèle de traduction chinois-anglais à produire uniquement des mots anglais présents dans la description du sens prédit par le système de DL.
2. En laissant le système de TA produire une traduction anglaise complète et en remplaçant, à postériori, les mots anglais qui ont un sens différent de celui prédit par le système de DL.
3. En prenant en compte les prédictions du système de DL dans le calcul des probabilités du système de TA, au même titre que le modèle de langue et le modèle de traduction.

14. http://www.keenage.com/zhiwang/e_zhiwang.html

Les deux premières méthodes dégradent ainsi la performance du système de TA, tandis que la dernière a tendance à l'améliorer. Dans les travaux de Cabezas et Resnik (2005) et de Chan et al. (2007a), ce sont aussi des méthodes similaires à cette troisième méthode qui sont utilisées, avec différentes paires de langues, et qui permettent aussi d'améliorer les résultats du système de traduction.

Dans mes travaux de master (Vial, 2016), nous avions aussi intégré les prédictions d'un système de DL au sein d'un système de TA statistique anglais-français, en utilisant un système de DL plus récent et en prenant en compte les prédictions de sens lors du calcul des probabilités, et nous avions aussi constaté une nette amélioration des résultats.

2.6.1.2 Enrichissement sémantique des systèmes neuronaux

Comme vu précédemment, avec les systèmes statistiques classiques, plusieurs travaux ont montré qu'on pouvait améliorer les performances d'un système de TA grâce aux prédictions d'un système de DL.

Cependant, avec le basculement de paradigme en faveur des architectures neuronales, et les améliorations importantes de leurs performances qui en découlent, les méthodes d'intégration de sens dans les systèmes de TA neuronaux sont depuis bien différentes, et nous allons ici les décrire.

Les sens en tant qu'information discrète

Sennrich et Haddow (2016) proposent une méthode permettant d'ajouter n'importe quelle donnée discrète, qu'on appelera trait, en entrée d'un système de TA neuronal. La méthode consiste à simplement concaténer les vecteurs de mot et de chaque trait utilisé, et d'apprendre conjointement ces vecteurs avec le reste du modèle neuronal.

Ainsi, les mots de la langue source sont eux-mêmes considérés comme des traits, et les auteurs ajoutent d'autres traits linguistiques tels que les lemmes et les parties du discours. Ils montrent ainsi une amélioration des performances sur deux tâches de traduction, anglais-allemand et allemand-anglais.

Bien que les auteurs n'aient pas essayé d'utiliser des traits de sens, leur méthode peut facilement s'appliquer à des traits produits par un système de DL. Dans la thèse de Hadj Salah (2018), l'autrice a d'ailleurs mené l'expérience sur un système de TA anglais-arabe et arabe-anglais, en désambiguïsant le texte source dans chacun des cas, à l'aide d'un système de DL supervisé neuronal qui assigne aux

mots un sens provenant de WordNet, et elle a constaté une amélioration supplémentaire par rapport au simple ajout des lemmes et des parties du discours.

Pu et al. (2018) mènent une expérience similaire, mais ils utilisent cette fois un système de désambiguïsation non supervisé, dans lequel des vecteurs de sens sont créés à partir de leur définition dans WordNet, et où des groupes sont ensuite formés à partir de leur proximité dans l'espace vectoriel. Enfin, ces vecteurs de sens sont intégrés à plusieurs systèmes de TA neuronaux et là encore ils constatent une amélioration systématique des résultats.

Enfin, Vanmassenhove et Way (2018) utilisent une méthode de DL qui assigne aux mots une des 41 catégories de *supersenses* de WordNet (des catégories de sens générales telles que « social », « cognition », etc.), et ils intègrent ces traits à un système de TA avec la méthode de Sennrich et Haddow (2016). Cette fois, les auteurs ne concluent pas à des améliorations significatives, mais tout de même à une vitesse de convergence plus élevée.

Les sens en tant qu'information continue

Liu et al. (2018) proposent une méthode pour intégrer indirectement les informations d'un système de DL à un système de TA neuronal. Ils s'appuient sur l'architecture de DL neuronale de Kågebäck et Salomonsson (2016) (voir section 1.4.2.2) et celle de Yuan et al. (2016) (voir section 1.4.2.3) afin d'apprendre des vecteurs de contexte, puis ils concatènent ces vecteurs de contexte aux vecteurs de mot en entrée du système de TA.

Plus précisément, les auteurs entraînent un premier modèle qu'ils appellent « modèle de contexte » et qui a pour objectif de prédire un mot x_t à l'indice t en fonction des mots le précédant $(x_0, \ldots, x_{t-1})$ et des mots le suivant $(x_{t+1}, \ldots, x_n)$. Le modèle fondé sur l'architecture de Kågebäck et Salomonsson (2016) utilise pour cela deux couches LSTM (avant et arrière), tandis que le modèle fondé sur l'architecture de Yuan et al. (2016) utilise une seule couche LSTM (avant) et un symbole spécial pour masquer le mot à prédire.

En concaténant ensuite les vecteurs de contexte avec les vecteurs de mot en entrée du système de TA, les auteurs constatent ainsi une amélioration notable de ses performances.

Ce qu'on peut remarquer, en plus de ces gains de performances, c'est que cet article propose finalement, à travers son modèle de contexte, la même chose que les modèles de langue comme ELMo et BERT cités précédemment (section 1.3.3.2), à la différence que ce modèle est appris sur les données monolingues des corpus

parallèles utilisés pour le système de traduction uniquement.

Dans nos contributions (chapitre 7), nous comparerons les méthodes qui consistent à ajouter des sens en tant qu'information discrète à un système de TA, et les méthodes qui considèrent indirectement les sens à partir d'un modèle de langue.

2.6.2 Capacité de désambiguïsation des systèmes de traduction

Si les méthodes dont on a parlé précédemment visent à améliorer les systèmes de traduction grâce aux informations extraites d'un système de désambiguïsation, d'autres travaux étudient directement les capacités intrinsèques qu'ont les systèmes de traduction à désambiguïser. En effet, si un modèle neuronal de TA est entraîné sur suffisament de données parallèles, il est déjà capable de traduire correctement un mot ambigu ayant plusieurs traductions possibles, par sa bonne traduction.

Rios et al. (2017) proposent ainsi une nouvelle tâche d'évaluation pour la TA intitulée ContraWSD (contra étant l'abbréviation de *contrastive*), dans laquelle on donne un ensemble de phrases dans une langue source, avec pour chaque phrase un mot ambigu. Pour chaque phrase, on fournit une traduction de référence dans une langue cible, ainsi que plusieurs autres traductions contrastives, dans lesquelles le mot ambigu est remplacé par un autre (une mauvaise traduction). L'objectif des systèmes de TA évalués est finalement de donner un score plus élevé pour la traduction de référence que pour les traductions contrastives, en se servant des prédictions du décodeur.

La tâche propose deux paires de langues : allemand-anglais et allemand-français, et elle sera ensuite reprise dans la campagne d'évaluation WMT 2018 (Rios et al., 2018). Elle permettra notamment de prendre directement en compte une sortie de traduction d'un modèle de TA neuronal plutôt qu'un seul score.

Tang et al. (2018, 2019) étudient ainsi séparément les capacités de chaque couche d'un encodeur et de plusieurs mécanismes d'attention pour désambiguïser les mots, et ils évaluent leurs systèmes notamment sur ContraWSD.

Pour finir, on peut citer les travaux de Marvin et Koehn (2018), dans lesquels les auteurs évaluent la capacité de désambiguïsation d'un système de TA neuronal sur quelques exemples sélectionnés manuellement, afin de montrer que les systèmes récents ont encore de grandes lacunes pour déterminer le sens de certains mots très polysémiques.

On notera que tous ces travaux permettent seulement d'estimer la capacité d'un système de TA à distinguer différents sens d'un mot dans le cadre d'une traduction

entre deux langues spécifiques, mais pas de manière absolue. Par exemple, le terme « souris » se traduit en anglais par « mouse », qu'il soit employé dans son sens de rongeur ou celui de périphérique informatique, alors il est impossible de savoir si un système de TA français-anglais est bien capable de reconnaître le bon sens du mot souris en regardant uniquement sa sortie.

2.7 Conclusion

Comme on a pu le voir dans ce chapitre, la traduction automatique est une des toutes premières tâches du TAL à avoir vu le jour, et elle reste une des tâches fondamentales aujourd'hui. En effet, elle est au cœur d'enjeux importants comme l'apprentissage ou la compréhension de langues étrangères.

La traduction automatique est, en pratique, complexe à mettre en œuvre comme à évaluer, entre autres, car elle fait appel à de nombreux critères subjectifs (le style d'écriture, la culture, etc.).

Bien que plusieurs approches pour la construction de systèmes de TA existent, dont les systèmes à base de règles et les systèmes statistiques, la grande majorité des travaux portent aujourd'hui sur les systèmes qui s'appuient sur un réseau de neurones, ou systèmes de TA neuronaux.

La recherche en TA neuronale a grandement influencé la recherche dans les autres domaines du TAL. En effet, les systèmes séquence à séquence, les modèles d'attention et l'architecture Transformer sont au moins trois avancées majeures imaginées d'abord pour la TA et qui ont ensuite été reprises dans de nombreux autres modèles neuronaux pour d'autres tâches.

Les systèmes de TA neuronaux, à l'instar des systèmes statistiques, s'appuient presque uniquement sur des corpus parallèles pour leur apprentissage, ainsi que pour leur évaluation. Avoir ces ressources pour une paire de langues visée, et en avoir en grande quantité, est donc primordial afin de construire des systèmes de traduction de bonne qualité. Cependant, on a recensé dans la section 2.4 plusieurs méthodes employées dans le cas des langues moins dotées.

Finalement, nous avons réalisé un tour d'horizon des travaux alliant la désambiguïsation lexicale à la traduction automatique dans la section 2.6. Certaines méthodes consistent à ajouter aux mots des étiquettes de sens issues d'un inventaire de sens (WordNet ou *supersenses* plus généraux), tandis que d'autres méthodes intègrent directement un modèle de langue, qui n'est donc pas rattaché à un inventaire de sens particulier.

Dans nos contributions aux chapitre 7 et chapitre 8, nous allons d'une part étudier et comparer ces deux approches d'intégration de sens dans un système de TA neuronal (avec et sans inventaire de sens) ainsi que plusieurs méthodes (trait supplémentaire en entrée ou apprentissage guidé par la sortie), et d'autre part étudier les capacités et l'intérêt d'un modèle joint de désambiguïsation lexicale et de traduction automatique.

Deuxième partie

Contributions

Chapitre 3

Vecteurs de sens pour la désambiguïsation à base de similarité sémantique

3.1 Introduction

Parmi les différentes approches pour la désambiguïsation lexicale que nous avons décrites précédemment (voir section 1.4), celles qui obtiennent en pratique des résultats état de l'art sont le plus souvent les approches supervisées, dans lesquelles un classifieur est entraîné sur une grande quantité de textes annotés en sens. Les approches à base de connaissances, à l'inverse, obtiennent généralement de moins bons résultats, mais nécessitent aussi beaucoup moins de ressources pour fonctionner.

En effet, parce que ces dernières s'appuient uniquement sur des bases de connaissances, il en découle plusieurs avantages, notamment :

— une meilleure couverture des systèmes sur les données d'évaluation, car ils ne sont pas tributaires des sens observés lors de l'apprentissage ;

— une meilleure généralisation des méthodes à d'autres langues, car on peut trouver des bases de connaissances dans de nombreuses langues (à minima des dictionnaires), au contraire des corpus annotés en sens, qui ne sont disponibles que pour des langues très dotées telles que l'anglais.

Ce deuxième point est important à relever, car l'annotation de corpus en sens est extrêmement lourde, et peu de langues disposent de ce type de ressource. [1]

1. Moins de dix langues possèdent au moins un corpus annoté en sens via WordNet, listées sur

Pour ces raisons, nous présentons dans ce chapitre des travaux qui s'articulent autour des approches à base de connaissances et nous présentons plus particulièrement une extension d'un système fondé sur l'algorithme de Lesk (voir section 1.4.1.1), à l'aide de vecteurs de sens créés à partir de leur définition dans WordNet, et d'un modèle de vecteurs de mot pré-entraînés.

En effet, les modèles pré-entraînés de vecteurs de mot, tels que Word2Vec (Mikolov et al., 2013), GloVe, (Pennington et al., 2014) ou encore ceux de Levy et Goldberg (2014) qui reposent sur des dépendances syntaxiques, ont montré des gains intéressants dans de nombreuses tâches du TAL. Ils sont notamment utilisés dans plusieurs méthodes de DL telles que Iacobacci et al. (2016), Yuan et al. (2016) ou encore Luo et al. (2018a,b).

Cependant, leur intégration concerne, dans les cas évoqués précédemment, des système supervisés. Rares sont les utilisations de vecteurs de mot au sein d'un système à base de connaissances. On peut tout de même citer les travaux de Chen et al. (2014) dans lesquels les auteurs entraînent un modèle de vecteurs de mot similaire à Word2Vec, puis créent des vecteurs de sens à partir de leur définition dans WordNet, et proposent un nouvel algorithme de DL utilisant uniquement ces vecteurs de sens.

Dans nos travaux, nous allons explorer la création de vecteurs de sens similaires à ceux de Chen et al. (2014), ainsi que leur application dans une nouvelle méthode pour l'extension de la mesure de Lesk, à la manière du Lesk étendu (Banerjee et Pedersen, 2002) mais sans utiliser de réseau lexical construit manuellement.

Les travaux présentés dans ce chapitre sont issus de deux de nos articles : Vial et al. (2017c) (en français) et Vial et al. (2017a) (en anglais).

3.2 Création de vecteurs de sens

Notre modèle permettant de représenter les sens d'un dictionnaire sous forme vectorielle repose sur leur définition, ainsi qu'un modèle pré-entraîné de vecteurs de mot. Dans la pratique, la méthode est relativement similaire à celle présentée dans l'article de Ferrero et al. (2017) dans lequel les auteurs créent des vecteurs de phrases pour la détection de plagiat inter-lingue. Nos vecteurs de sens sont ainsi calculés comme la somme normée des vecteurs des termes présents dans la définition du sens donné. Ces vecteurs sont pondérés en fonction de leur partie du discours : nom, verbe, adjectif ou adverbe, et également pondérés par leur *Inverse*

`http://globalwordnet.org/wordnet-annotated-corpora/`, en date du 2 août 2019.

Document Frequency (IDF), c'est-à-dire l'inverse de leur nombre d'occurrence dans tout le dictionnaire. Plus formellement nous avons :

— $D(S) = \{w_0, w_1, w_2, \ldots, w_n\}$ la définition du sens S dans le dictionnaire ;
— $pos(w_n) = \{n, v, a, r\}$ la partie du discours du terme w_n (nom, verbe, adjectif, ou adverbe) ;
— $weight(pos)$ le poids associé à une partie du discours ;
— $idf(w_n)$ la valeur IDF de w_n.

La définition du vecteur du sens S, notée $\phi(S)$ correspond à :

$$\phi(S) = \sum_{i=0}^{n} (\phi(w_n) \times weight(pos(w_n)) \times idf(w_n))$$

$\phi(S)$ est ensuite normé afin d'avoir la même taille que les vecteurs de mot (qui sont eux aussi normés). Les poids attribués aux POS sont les mêmes que ceux utilisés par Ferrero et al. (2017).

Nous avons créé cinq modèles de vecteurs de sens. Tous s'appuient sur le vocabulaire et les définitions de WordNet 3.0, mais exploitent des modèles de vecteurs de mot différents. Les cinq modèles utilisés sont :

1. Un modèle Word2Vec pré-entraîné et proposé par Mikolov et al. (2013) [2]. Ce modèle a été entraîné sur environ 100 milliards de mots issus du corpus de nouvelles de Google. La taille du vocabulaire est d'environ de trois millions de mots et les vecteurs ont une dimension de 300.
2. Un modèle GloVe provenant de Pennington et al. (2014) [3], entraîné sur 42 milliards de mots issus de *Common Crawl*. Le vocabulaire est de deux millions de mots et les vecteurs ont une taille de 300.
3. Un modèle de vecteurs de mot fondé sur les interdépendances de Levy et Goldberg (2014) [4]. L'apprentissage a été effectué sur Wikipedia, le vocabulaire est de 175 000 mots et la taille des vecteurs 300.
4. Le meilleur des modèles de prédiction de Baroni et al. (2014) [5]. La taille du vocabulaire est de 300 000 et la taille des vecteurs est de 400.

2. https://code.google.com/archive/p/word2vec/
3. https://nlp.stanford.edu/projects/glove/
4. https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/
5. http://clic.cimec.unitn.it/composes/semantic-vectors.html

5. Et finalement, le meilleur modèle à base de comptage également créé par Baroni et al. (2014)[5] de dimension 500 et une taille de vocabulaire identique au précédent modèle.

Dans la section expérimentale, nous comparons les performances de chacun de ces modèles dans une tâche de désambiguïsation lexicale en anglais. WordNet 3.0 est composé de 206 941 sens, et nous avons donc créé un vecteur pour chacun d'eux. Tous nos modèles de vecteurs de sens sont rendus publics [6], afin de reproduire les expériences.

3.3 Évaluation des vecteurs de sens

Les vecteurs de sens ainsi générés sont évalués sur une tâche de désambiguïsation lexicale. L'idée est d'exploiter le modèle en lieu et place d'un réseau lexical. Pour cela, nous comparons les sens entre eux via la distance cosinus, et selon un seuil δ appris sur une tâche de développement, les sens les plus proches sont sélectionnés. Les sens sélectionnés vont alors contribuer à l'algorithme de DL au niveau de sa mesure de similarité locale, dans un mode de fonctionnement similaire à celui présenté par Banerjee et Pedersen (2002) à la différence qu'eux exploitent le réseau lexical de WordNet, créé manuellement.

3.3.1 Système de désambiguisation

Le système de DL que nous proposons est un système à base de similarité sémantique (voir section 1.4.1.1) composé de deux éléments : un algorithme local qui calcule un score de similarité pour une paire de sens, et un algorithme global qui va chercher la meilleure combinaison de sens à l'échelle du document, en utilisant l'algorithme local.

6. https://persyval-platform.univ-grenoble-alpes.fr/DS117/detaildataset

3.3.1.1 Algorithme global

L'algorithme global repose sur une heuristique permettant de ne pas explorer exhaustivement l'ensemble des combinaisons de sens possibles, ce qui engendrerait des coûts calculatoires trop élevés.

En effet, le nombre de combinaisons de sens possibles pour un document de N mots est égal au nombre de sens moyen de chaque mot, à la puissance N. Le nombre moyen de sens pour les mots polysémiques de WordNet étant de 3, il suffit de 20 mots polysémiques dans le document que l'on cherche à désambiguïser pour atteindre plus de trois milliards de combinaisons, et pour un document contenant 100 mots polysémiques, ce chiffre est de l'ordre de 10^{47}.

Pour bien comprendre l'impossibilité de calculer la mesure de similarité pour toutes ces combinaisons de sens, même si l'appel à la mesure prenait seulement 1 milliseconde, il faudrait toujours plus de 10^{36} années pour calculer toutes les combinaisons. C'est pourquoi nous avons besoin d'une heuristique qui ne parcourt pas l'espace de recherche complet.

L'heuristique utilisée dans notre système exploite un algorithme de recherche à base de coucous, comme celui présenté dans Vial et al. (2017d) [7]. En effet, nous avions montré dans cet article que cet algorithme global est actuellement un des plus efficaces connus pour les systèmes de DL à base de similarité.

Notre implémentation réutilise les meilleurs paramètres présentés dans Vial et al. (2017d), c'est-à-dire un seul coucou, la variable *Levy location* à 5 et *Levy scale* à 0,5.

Dans l'ensemble de nos expériences nous avons utilisé un grand nombre d'itérations (300 000) afin de réduire l'effet aléatoire d'une expérience à une autre. De plus, nous réalisons en moyenne une dizaine d'exécutions pour en extraire un score moyen. De cette manière nous assurons un écart-type inférieur à 0,1 : les résultats sont ainsi stables et reproductibles.

7. Article issu de mes travaux de master (Vial, 2016).

3.3.1.2 Algorithme local

L'algorithme local est l'élément central de notre système et c'est pour l'amélioration de ce dernier qu'est utilisé le réseau lexical. Comme système de référence, nous utilisons une mesure de Lesk standard. Cet algorithme retourne, comme mesure de similarité, le nombre de mots communs à deux définitions de sens. Formellement, si nous notons $D(S) = \{w_1, w_2, \ldots, w_n\}$ la définition de S, alors la mesure de Lesk originale entre deux sens S_1 et S_2 notée $Lesk(S_1, S_2)$ est la suivante :

$$Lesk(S_1, S_2) = |D(S_1) \cap D(S_2)|$$

Comme second système de référence, et dans l'objectif de mesurer la qualité de notre réseau lexical généré automatiquement, nous utilisons aussi la mesure de Lesk étendue de Banerjee et Pedersen (2002).

Cette mesure considère à la fois la définition du sens cible mais également les définitions de tous les sens qui ont une relation avec le sens cible (hyponyme, etc.). Si l'on note $rel(S)$ l'ensemble des sens reliés à S à travers un lien explicite dans *WordNet*, alors la mesure de Lesk étendue entre les sens S_1 et S_2 notée $ExtLesk(S_1, S_2)$ est la suivante :

$$ExtLesk(S_1, S_2) = \left|(D(S_1) \cup_{r \in rel(S_1)} D(r)) \cap (D(S_2) \cup_{r \in rel(S_2)} D(r))\right|$$

3.3.2 Extension de définitions grâce aux vecteurs de sens

Nous proposons d'étendre l'algorithme de Lesk avec nos vecteurs de sens de la même manière que Banerjee et Pedersen (2002), en augmentant les définitions de WordNet à partir des sens proches. Seulement, nous allons utiliser la similarité cosinus entre deux sens pour considérer qu'ils sont liés, au lieu d'utiliser les liens explicites de WordNet.

Étant donné que nos vecteurs de sens sont créés à partir de leur définition, les termes utilisés dans ces définitions sont très importants. Pour cette raison, nous considérons que deux sens sont reliés si, d'une part, la similarité cosinus entre les deux est élevée, et si, en même temps, la similarité cosinus entre le vecteur du lemme d'un sens et le vecteur de l'autre sens est aussi élevée. Nous filtrons ainsi les sens qui ont une similarité cosinus élevée simplement parce qu'ils utilisent des mots semblables dans leur définition.

Nous pouvons effectivement calculer une similarité cosinus entre vecteurs de sens et vecteurs de mot, parce que les vecteurs de sens sont une somme de vecteurs de mot, et donc sont de même dimension.

On note $rel(S, \delta_1, \delta_2)$ l'ensemble des sens liés à S, avec δ_1 le seuil de sélection sur la similarité cosinus avec le vecteur du lemme et δ_2 le seuil de sélection sur la similarité avec le vecteur du sens. La fonction $rel(S, \delta_1, \delta_2)$ est calculée de la manière suivante :

$$rel(S_1, \delta_1, \delta_2) = \{S_2 \mid cos(\phi(lemma(S_1)), \phi(S_2)) > \delta_1 \wedge cos(\phi(S_1), \phi(S_2)) > \delta_2\}$$

Finalement, le calcul pour notre nouvel algorithme local de DL que nous appellons $VecLesk(S_1, S_2, \delta_1, \delta_2)$, peut être décrit de la manière suivante :

$$VecLesk(S_1, S_2, \delta_1, \delta_2) = \\ \left|(D(S_1) \cup_{r \in rel(S_1, \delta_1, \delta_2)} D(r)) \cap (D(S_2) \cup_{r \in rel(S_2, \delta_2, \delta_2)} D(r))\right|$$

3.3.3 Résultats

Notre extension considère les sens proches d'un autre en fonction d'un seuil de similarité δ_1 sur la distance cosinus avec le lemme du sens cible, et un filtrage supplémentaire sur le vecteur de sens cible, avec un seuil δ_2.

Ces deux seuils ont été paramétrés sur un corpus de développement. Ici, nous avons utilisé la tâche 7 de SemEval 2007 (Navigli et al., 2007) pour fixer le meilleur ensemble de paramètres puis testé sur la tâche 13 de SemEval 2015 (Moro et Navigli, 2015). Nous avons échangé les corpus dans un second temps. Ce choix a été motivé pour présenter des résultats sur deux corpus de même nature.

Les cinq modèles présentés dans la section 3.2 sont évalués séparément. Les paramètres δ_1 et δ_2 ont été estimés en les faisant varier dans les intervalles $[0.5, 0.9]$ avec un pas de 0.1.

Les résultats des meilleurs jeux de paramètres sur SemEval 2007 et SemEval 2015 sont présentés dans le tableau tableau 3.1. La comparaison de notre méthode par rapport aux deux systèmes de référence est présentée dans le tableau tableau 3.2.

Les résultats montrent que notre extension améliore très significativement les résultats obtenus par la mesure de Lesk. L'amélioration, sur SemEval 2007, est d'environ +5 points de pourcentage pour la pire combinaison et de +7 points pour la meilleure. Sur SemEval 2015 les améliorations oscillent entre +3 et +9 points. Cependant, notre extension n'atteint pas les résultats du Lesk étendu ; ceci montre que notre réseau lexical est un peu moins performant qu'un réseau créé manuellement (tel que WordNet).

Modèle	Meilleurs paramètres sur la tâche			
	SemEval 2007		SemEval 2015	
	δ_1	δ_2	δ_1	δ_2
Word2Vec	0,5	0,6	0,5	0,6
GloVe	0,5	0,6	0,5	0,7
Dépendances	0,6	0,8	0,6	0,8
Baroni (préd.)	0,5	0,5	0,5	0,6
Baroni (compt.)	0,6	0,6	0,5	0,8

TABLE 3.1 – Estimation des paramètres δ_1 et δ_2 sur SemEval 2007 et SemEval 2015.

Système	Tâche	
	SemEval 2007	SemEval 2015
Chen et al. (2014)	*75.80%*†	-
Lesk	68.70%	50.65%
Lesk étendu	78.01%	61.42%
VecLesk (Word2Vec)	73.30%	57.00%
VecLesk (GloVe)	73.00%	**59.01**%
VecLesk (Dépendances)	73.02%	56.40%
VecLesk (Baroni (préd.))	73.52%	53.46%
VecLesk (Baroni (compt.))	**75.29**%	58.02%

TABLE 3.2 – Score F1 sur les tâches SemEval 2007 et SemEval 2015 pour chacun des modèles de vecteurs de mot par rapport aux références Lesk et Lesk étendu. Les paramètres δ_1 et δ_2 utilisés sont ceux estimés dans le précédent tableau sur **l'autre tâche**, pas celle qui est testée. † Un biais est présent dans l'article référencé étant donné que les auteurs estiment leur paramètre $\delta \in [-0.1, 0.3]$ comparable au nôtre, mais directement sur le corpus de test.

Un point intéressant à noter est la différence de score obtenue entre les différents modèles de vecteurs de mot. Les meilleurs résultats sur SemEval 2007 utilisent la méthode de Baroni et al. (2014) qui repose sur le comptage. Ceci tend à montrer que les méthodes de création de vecteurs de mot moins récentes à base de comptage sont parfois meilleures que les modèles prédictifs. Cependant, sur SemEval 2015, c'est la méthode GloVe qui obtient les meilleurs résultats. Ces fluctuations dans les résultats sont peut-être dues aux différentes natures des méthodes de création des vecteurs de mot, ou bien dues aussi aux différents corpus sur lesquels ils ont été entrainés. En tout cas, nous montrons que quel que soit le modèle sous-jacent utilisé, notre méthode améliore systématiquement le score de Lesk.

Par ailleurs, si l'on compare notre système à la meilleure méthode état de l'art à notre connaissance (Chen et al., 2014) exploitant des ressources similaires aux nôtres (un dictionnaire et des corpus non annotés), nous observons que nos résultats sont les meilleurs sur les systèmes de DL à base de connaissances. En effet, les auteurs estiment un paramètre δ similaire à nos δ_1 et δ_2 directement sur la tâche d'évaluation. Leurs scores varient entre 72.10% et 75.80% en fonction de δ. À noter que nous obtenons un score de 77.08% sur SemEval 2007 avec le meilleur jeu de paramètres appris sur cette même tâche.

3.4 Conclusion

Dans ce chapitre nous avons présenté un modèle de vecteurs de sens, représentant tous les sens issus d'un dictionnaire. Ce modèle est construit en s'appuyant sur les définitions du dictionnaire et sur un modèle de vecteurs de mot existant. Notre méthode consiste à sommer les vecteurs de mot présents dans la définition, pondérés en fonction de leur partie du discours ainsi que de leur fréquence.

Nous avons créé cinq modèles de vecteurs de sens, distribués librement [8], chacun reposant sur un modèle de vecteurs de mot différents.

Ces modèles sont ensuite utilisés comme un réseau lexical pour améliorer un système de DL fondé sur l'algorithme de Lesk. Nous utilisons une méthode similaire à Banerjee et Pedersen (2002) avec comme différence principale l'utilisation d'un réseau lexical construit totalement automatiquement.

Les résultats que nous obtenons sur deux tâches de DL montrent une amélioration systématique du score par rapport à la mesure de Lesk classique (d'environ

8. https://persyval-platform.univ-grenoble-alpes.fr/DS117/detaildataset

+3 points de pourcentage à +9 points selon le modèle).

Nous avons appliqué notre méthode uniquement sur la base lexicale WordNet pour évaluer nos modèles sur des tâches très connues en DL sur la langue anglaise. Cependant, notre méthode peut être appliquée très facilement à d'autres langues où les ressources sont moindres : un dictionnaire classique ainsi que des corpus non annotés sont les ressources suffisantes pour créer ainsi un système de désambiguïsation lexicale performant.

Chapitre 4

UFSAC : Unification de corpus annotés en sens

4.1 Introduction

Les corpus annotés en sens sont une ressource centrale pour la DL. En effet, ils sont indispensables d'une part pour l'apprentissage des systèmes supervisés, qui sont généralement les plus performants, mais aussi pour l'évaluation de tous les systèmes de DL. L'évaluation des systèmes *in vivo*, c'est-à-dire en intégrant la DL à une tâche plus large, étant encore peu exploitée.

Depuis sa création, WordNet (Miller et al., 1990) est sans aucun doute l'inventaire de sens et la base de données lexicale la plus connue et la plus utilisée pour la désambiguïsation de l'anglais. C'est aujourd'hui un standard *de facto* utilisé pour l'annotation de la vaste majorité des corpus annotés en sens, et même les autres inventaires de sens qui sont parfois utilisés (tels que BabelNet, ou l'inventaire de sens de l'Ontonote) sont aussi reliés à WordNet.

Malgré cette prédominance de WordNet, deux problèmes majeurs freinent l'utilisation des corpus annotés en sens. Premièrement, les corpus n'utilisent pas tous la même version de l'inventaire de sens, et les étiquettes de sens entre différentes versions sont incompatibles entre elles. Deuxièmement, chaque corpus est distribué dans un format qui lui est propre et qui est parfois très différent des autres corpus. La conséquence de ces deux problèmes est que très peu de travaux dans la littérature de la DL entraînent ou évaluent un système sur plus d'un ou deux corpus annotés en sens.

De plus, nous faisons le constat que les corpus initialement créés pour l'éva-

luation (par exemple les corpus issus des campagnes SensEval/SemEval) et ceux créés dans d'autres buts, comme l'apprentissage ou l'estimation des distributions de sens, sont très rarement utilisés d'une façon différente de celle initialement prévue.

Dans ce chapitre, nous présentons un travail d'unification des corpus anglais annotés en sens WordNet, dans un format unique, simple à comprendre et simple d'utilisation. Nous mettons sur le même plan les corpus créés pour l'évaluation ou pour l'apprentissage, afin de faciliter la création de nouveaux systèmes de DL qui pourraient par exemple exploiter l'ensemble des corpus disponibles pour l'apprentissage à l'exception d'un pour l'évaluation, puis répéter l'expérience en changeant le corpus d'évaluation.

Dans ce but d'unification, nous avons ainsi créé le format UFSAC (Unified Format for Sense Annotated Corpora), et nous fournissons à la communauté tous les corpus annotés en sens que nous connaissons, dans ce format, avec toutes les étiquettes de sens converties à la dernière version de WordNet (la 3.0). Nous fournissons aussi une bibliothèque Java permettant de facilement lire, écrire et modifier des corpus dans ce format, ainsi qu'un ensemble de scripts permettant de convertir les corpus d'un format à un autre.

Les travaux présentés dans ce chapitre sont issus d'un de nos articles de conférence (Vial et al., 2018b) (en anglais) lui-même étant la continuité d'un autre de nos articles (Vial et al., 2017b) (en français).

4.2 Corpus annotés en sens

Les corpus annotés en sens sont des ressources rares et coûteuses. En effet, la Global WordNet Association recense une liste de 26 corpus annotés avec l'inventaire de sens de WordNet [1]. Ces corpus concernent 17 langues, mais seulement trois d'entre elles possèdent au moins 100 000 annotations. L'anglais, avec plus de deux millions de mots annotés en sens arrive en tête, devant le néerlandais avec environ 300 000 annotations et le bulgare avec 100 000 annotations. Cela est d'ailleurs une des raisons qui explique pourquoi la plupart des recherches scientifiques en DL se focalisent sur l'anglais.

Comme abordé dans la section 1.3.2, l'anglais dispose ainsi de nombreux corpus annotés en sens, créés avec différents objectifs et s'appuyant sur différents inventaires de sens.

1. http://globalwordnet.org/wordnet-annotated-corpora/

Après leur distribution, il n'existe aucune raison scientifique de ne pas utiliser indifféremment ces corpus soit pour exploiter leurs exemples pour la construction d'un système de désambiguïsation lexicale, soit pour évaluer de tels systèmes, soit pour estimer une distribution des sens. D'ailleurs, le SemCor est utilisé depuis longtemps dans la construction de systèmes de désambiguïsation lexicale (Chan et al., 2007b; Navigli et al., 2007) ou même dans le cadre de l'évaluation par Yuan et al. (2016). Cette dernière utilisation est tout de même assez rare puisque c'est l'une des seules expériences que nous avons pu trouver dans la littérature avec Màrquez et al. (2002).

Cependant, le format des ressources distribuées diffère fortement suivant leur utilisation d'origine, ce qui peut être un frein à leur utilisation. Pour l'estimation de la distribution des sens ou la construction d'un système de désambiguïsation lexicale par exemple, un seul fichier regroupe toutes les informations tandis que dans le cas de l'évaluation, il y a deux fichiers, l'un qui contient les données non-annotées en sens et l'autre qui ne contient que les annotations. Enfin pour certains corpus, comme le DSO et l'OMSTI, il y a un fichier pour chaque lemme du dictionnaire, et chaque fichier contient des exemples de phrases dans lesquelles ce lemme est le seul mot annoté en sens.

La liste de tous les corpus annotés en sens portés à notre connaissance, utilisant un inventaire de sens de WordNet, ou relié à WordNet se trouve dans la section 1.3.2. Dans le tableau 1.1, nous faisons un récapitulatif des différents inventaire de sens utilisés dans ces corpus. Comme nous pouvons le voir, ces inventaires sont très variables, et même si la majorité utilise WordNet 3.0 ou y est relié, il n'y a pas de comptatibilité directe des étiquettes de sens avec les anciennes version de WordNet, BabelNet ou Ontonotes.

4.3 Format unique pour les corpus annotés en sens

L'hypothèse principale de ce travail est que posséder tous les corpus annotés en sens disponibles sous un même format, avec le même inventaire de sens, permettra la création de systèmes de DL supervisés plus robustes et leur évaluation à plus grande échelle. De plus, la distribution d'un outil simple d'utilisation pour leur lecture, création ou modification, peut faciliter leur diffusion.

En effet, concernant les performances, dans la plupart des articles décrivant un système supervisé, avoir plus de ressources annotées en sens mène généralement à de meilleures performances. Ainsi, pour obtenir leurs meilleurs résultats, Chan et al. (2007b) exploitent le SemCor et le DSO, Iacobacci et al. (2016) exploitent le

SemCor et l'OMSTI, Yuan et al. (2016) le SemCor, l'OMSTI et le MASC, Pasini et Navigli (2017) le Train-O-Matic, etc. Cependant, personne n'exploite la totalité des corpus disponibles, sans vraiment en justifier la raison.

Concernant l'évaluation, un problème important en DL est que la plupart des travaux se focalisent seulement sur une ou deux tâches (Chen et al., 2014; Iacobacci et al., 2016; Kågebäck et Salomonsson, 2016). Cela limite fortement l'analyse des résultats et interroge la robustesse des méthodes mises en œuvre.

Dans des langues moins dotées, l'arabe par exemple, il n'existe pas de standard d'évaluation et il n'est pas rare de lire dans un article que tel système fonctionne mieux que tel autre car il a obtenu un score supérieur alors qu'il ne s'agit ni du même inventaire de sens ni du même corpus d'évaluation.

L'uniformisation des formats des corpus annotés peut permettre ainsi l'évaluation à bien plus large échelle en procédant, par exemple, à une validation croisée par rotation dans laquelle on utilise tour à tour chacun des corpus pour l'évaluation d'un système et l'ensemble des autres pour sa construction.

Récemment, dans le travail de Raganato et al. (2017a), les auteurs proposent un ensemble de cinq tâches de DL de l'anglais (SensEval 2 et 3, Semeval 2007, 2013 et 2015) afin d'unifier l'évaluation des systèmes de DL. Depuis, les systèmes plus récents sont évalués sur toutes ces tâches. À noter que ce travail a été conduit en parallèle de celui qui est présenté ici, et qu'il se concentre uniquement sur l'évaluation des systèmes de DL.

4.4 La ressource UFSAC

Notre travail consiste à réunir tous les corpus anglais annotés en sens, que nous avons listés dans le tableau 1.1, et à les convertir dans un format unique capable de contenir toutes les informations présentes dans leur format original. Pour cela, notre ressource contient des outils qui permettent de convertir les corpus d'un format à un autre, nettoyer les corpus et convertir les annotations de sens depuis leur inventaire de sens original vers la version 3.0 de WordNet.

Les corpus résultants sont distribués au sein de notre ressource quand les droits le permettent, et dans tous les cas nous fournissons tous nos outils permettant de convertir les ressources originales. Ainsi, pour les corpus dont nous n'avons pas le droit de distribution, les possesseurs de la ressource originale peuvent toujours créer une version UFSAC.

Enfin, nous fournissons en plus des corpus et des outils, une bibliothèque Java

permettant de lire, créer et éditer des corpus dans notre format. La ressource est accessible à l'adresse suivante : https://github.com/getalp/UFSAC.

4.4.1 Corpus inclus dans la ressource

Notre ressource UFSAC est continuellement mise à jour en fonction de la publication de nouveaux corpus annotés en sens. En date du 01/10/2018, UFSAC a ainsi connu quatre versions.

UFSAC 1.0.0

Dans la première version de la ressource, sortie en mai 2018, nous avons recensé les corpus suivants pour former notre ressource :

— le SemCor, converti depuis la version fournie par Rada Mihalcea sur son site Web [2] ;

— le DSO, mais uniquement le code pour la conversion du corpus propriétaire achetable sur le site du Linguistic Data Consortium (LDC) [3] ;

— le WordNet Gloss Corpus, converti depuis la version fournie par WordNet [4] ;

— l'OMSTI, converti depuis la version fournie par les auteurs [5] ;

— le MASC, converti depuis la version fournie par Google [6] pour leur article (Yuan et al., 2016) ;

— l'Ontonotes, mais uniquement le code pour la conversion du corpus téléchargeable gratuitement sur le site du LDC [7] ;

— tous les corpus des tâches d'évaluation « tous mots » des campagnes d'évaluation SensEval/SemEval, convertis depuis leur format d'origine, c'est-à-dire SensEval 2 [8], SensEval 3 [9], SemEval 2007 [10], SemEval 2013 [11] et SemEval 2015 [12].

2. http://web.eecs.umich.edu/~mihalcea/
3. https://catalog.ldc.upenn.edu/LDC97T12
4. http://wordnetcode.princeton.edu/glosstag-files/
5. https://www.comp.nus.edu.sg/~nlp/corpora.html
6. https://github.com/google-research-datasets/word_sense_disambigation_corpora
7. https://catalog.ldc.upenn.edu/LDC2013T19
8. http://www.hipposmond.com/senseval2/
9. http://web.eecs.umich.edu/~mihalcea/senseval/senseval3/
10. http://nlp.cs.swarthmore.edu/semeval/tasks/task07/data.shtml
11. https://www.cs.york.ac.uk/semeval-2013/task12/index.php
12. http://alt.qcri.org/semeval2015/task13/index.php

UFSAC 1.1.0

La deuxième version d'UFSAC, sortie en juin 2018, apporte des correctifs mineurs aux corpus existants, en unifiant les parties du discours selon la convention du Penn TreeBank (Taylor et al., 2003).

UFSAC 2.0.0

La troisième version d'UFSAC, sortie en juillet 2018, ajoute plusieurs nouveaux corpus à la collection :

— le Train-O-Matic, converti depuis la version fournie par les auteurs [13] ;

— les tâches « échantillon lexical » de SensEval 2 [14] et SensEval 3 [15] ;

— et enfin les versions des tâches d'évaluations de SensEval/SemEval fournies par Raganato et al. (2017a).

UFSAC 2.1

Enfin, la dernière version d'UFSAC, sortie en octobre 2018, apporte des correctifs mineurs aux outils de conversions de formats.

4.4.2 Format de fichier UFSAC

Notre approche pour l'unification des différents corpus annotés en sens commence par un format qui est descriptif, facilement compréhensible et lisible par un humain, et en même temps simple et efficace à lire et à écrire pour un programme. Enfin, il doit être capable de contenir toutes les informations des ressources originales. De ces informations, nous avons extrait ainsi les concepts suivants :

— Une **entité lexicale (EL)** est une entité qui peut contenir un ensemble d'annotations.

— Une **annotation** est un couple clé-valeur (avec potentiellement plusieurs valeurs).

— Un **corpus** est une EL qui contient un ensemble de documents.

13. http://trainomatic.org/trainomatic
14. http://www.hipposmond.com/senseval2/
15. http://web.eecs.umich.edu/~mihalcea/senseval/senseval3/

— Un **document** est une EL qui contient un ensemble de paragraphes.

— Un **paragraphe** est une EL qui contient un ensemble de phrases.

— Une **phrase** est une EL qui contient un ensemble de mots.

— Un **mot** est une EL qui possède une annotation spéciale obligatoire qui est sa **forme de surface**.

Afin de représenter ces concepts, UFSAC repose sur une syntaxe XML simple avec quelques conventions : les entités lexicales sont représentées par des nœuds XML (`corpus`, `document`, `paragraph`, `sentence` and `word`), et les annotations sont des attributs de ces nœuds (`key="value"`).

Les annotations suivent aussi des conventions. Nous utilisons les suivantes pour l'annotation d'entités lexicales :

— L'identifiant (`id`) d'une entité lexicale, particulièrement utile pour les corpus d'évaluation qui donnent généralement un identifiant aux mots à annoter, suivant la forme `d001.s002.t003`.

— La forme de surface (`surface_form`) d'un mot.

— Le lemme (`lemma`) d'un mot.

— La partie du discours (`pos`) d'un mot.

— Le sens d'un mot dans un inventaire lexical spécifique. Par exemple WordNet 3.0 (`wn30_key`), WordNet 1.7.1 (`wn171_key`)... Si plusieurs sens sont spécifiés par l'annotateur (par exemple dans le cas d'une annotation gros-grain comme pour la tâche 7 de SemEval 2007), ils sont séparés par un point-virgule (`;`).

L'information du sens est, pour le cas de la DL, la plus utile, et nous avons choisi de la rendre spécifique à chaque inventaire de sens, plutôt que d'avoir une annotation « sens » unique comme nous pouvons le voir dans la plupart des autres formats. De cette manière, nous permettons plusieurs annotations en sens depuis plusieurs inventaires de sens à la fois.

Par exemple, pour le DSO qui est originellement annoté avec WordNet 1.5, la conversion des sens vers WordNet 3.0 est parfois impossible du fait de sens qui ont par exemple été abandonnés entre les deux versions. Cette convention nous permet ainsi de garder l'annotation originale, tout en convertissant les annotations vers la dernière version de WordNet, ou tout autre inventaire de sens à la fois.

Voici un exemple de petit corpus au format UFSAC, avec quelques annotations :

```
<corpus id="exemple">
  <document id="d001" >
    <paragraph>
      <sentence>
        <word surface_form="A" pos="DT" />
        <word surface_form="precise" wn30_key="precise%3:00:00::" />
        <word surface_form="example" pos="NN" lemma="example" />
        <word surface_form="." />
      </sentence>
    </paragraph>
  </document>
</corpus>
```

4.4.3 Processus de conversion

La conversion d'un corpus depuis son format original vers UFSAC suit un processus en plusieurs étapes :

4.4.3.1 Conversion du format

Cette étape se charge de convertir le format d'origine d'un corpus vers le nouveau format UFSAC, sans apporter aucune autre modification (pas de conversion des étiquettes de sens, pas de nettoyage des caractères, etc.). L'objectif est en effet de garder toutes les informations contenues dans le corpus d'origine sans l'altérer.

Chaque corpus possède son propre format d'origine, et donc chaque corpus a son propre code de conversion. Par exemple, l'OMSTI original se présente sous la forme de plus de 40 000 fichiers, soient deux fichiers par lemmes annotés : un fichier XML contenant les phrases et les identifiants des mots annotés, et un fichier contenant le sens WordNet pour chaque identifiant. Cette étape fusionne donc tous ces fichiers dans un fichier UFSAC unique.

4.4.3.2 Fusion des phrases identiques

Certains corpus annotés en sens sont construits autour d'ensembles de phrases non consécutives plutôt que sur un ensemble de documents. C'est ce qui différencie par exemple le DSO du SemCor : le premier se présente comme un ensemble de phrases prises hors contexte dans lesquelles un seul mot cible est annoté en sens, tandis que le second est construit autour d'un ensemble de documents dans lesquels tous les mots sont annotés en sens.

Pour cette première catégorie de corpus, nous appliquons cette étape de fusion des phrases identiques, qui va identifier des phrases qui sont réparties à plusieurs endroits du corpus, mais dans lesquelles des mots différents sont annotés en sens, puis nous les fusionnons.

En plus de réduire le nombre total de phrases du corpus, cette étape ajoute une information cruciale par rapport au corpus d'origine, qui peut être utile pour certains systèmes de DL. En effet, un système fondé sur des similarités sémantiques peut par exemple se servir de l'information de la co-occurence d'un sens avec un autre.

En pratique, cette étape permet de fusionner près de 300 000 phrases de l'OMSTI, qui passe de 1 151 748 phrases à 863 648.

4.4.3.3 Conversion des étiquettes de sens

Les étiquettes de sens sont converties à cette étape, lorsque c'est nécessaire, depuis leur inventaire de sens original vers la dernière version de WordNet (3.0). Pour les anciennes versions de WordNet, nous utilisons les tables de conversion de Daudé et al. (2000), et pour les autres inventaires de sens tels que BabelNet et Ontonotes, nous utilisons les tables de correspondances incluses dans l'inventaire. En effet, BabelNet 1.1.1, BabelNet 2.5 et Ontonotes offrent tous les trois une correspondance avec WordNet 3.0.

Cependant, du fait que certains sens sont supprimés depuis les anciennes versions de WordNet ou les autres inventaires de sens, certaines annotations ne sont pas converties. Dans tous les cas, l'étiquette de sens originale est conservée avec la nouvelle.

Les tables de conversion de Daudé et al. (2000) fournissent, pour un sens donné dans la version de WordNet X, une probabilité de correspondance avec un sens de WordNet Y. Quand un sens a une correspondance avec plusieurs sens à probabilités égales, nous faisons la correspondance avec tous les sens, qui sont séparés par

un point-virgule.

4.4.3.4 Annotation en lemmes et parties du discours

À cette étape, nous annotons tous les mots possibles avec leur lemme grâce à l'outil *morphy* inclus dans WordNet, si cette information n'était pas déjà présente dans le corpus.

De même, nous annotons en parties du discours les mots qui n'ont pas déjà cette information grâce à l'annotateur de Stanford (Toutanova et al., 2003), qui produit des étiquettes provenant de l'ensemble des parties du discours du Penn Treebank (Taylor et al., 2003).

4.4.3.5 Nettoyage des caractères, mots et phrases

Enfin, cette étape consiste d'abord à supprimer les caractères invisibles, normaliser les ponctuations, et supprimer les tags XML vides.

Ensuite, elle s'occupe de supprimer ou corriger les annotations incohérentes. Par exemple, si un mot a été annoté en sens avec le sens d'un verbe, et qu'il est annoté en partie du discours avec une étiquette JJ (adjectif), on corrige la partie du discours en considérant celle donnée par le sens (un verbe).

4.4.4 Résultat

Nous avons appliqué le processus de conversion décrit précédemment à tous les corpus anglais annotés en sens que nous avons recensés afin de produire la première version de notre ressource en mai 2018 : UFSAC 1.0.0. Cette première version était constituée du SemCor, du DSO, du WordNet Gloss Corpus, du MASC, de l'OMSTI, de l'Ontonotes, et des corpus des campagnes d'évaluation SensEval/SemEval de 2001 à 2015.

Par la suite, de nouveaux corpus se sont ajoutés, notamment le Train-O-Matic, et les corpus des tâches « échantillon lexical » de SensEval 2 et SensEval 3, pour aboutir à la version 2.1 d'UFSAC en octobre 2018.

Dans le tableau 1.1, nous donnons ainsi les statistiques du nombre de phrases et de mots (annotés et non annotés) de tous ces corpus, que nous avons pu calculer grâce aux scripts fournis avec notre ressource.

4.4.5 Outils et bibliothèque UFSAC

En plus des corpus, nous fournissons une bibliothèque Java permettant de lire, écrire et modifier des corpus dans le format UFSAC. Elle permet deux styles de programmation : soit charger le corpus entier en mémoire, faire les opérations souhaitées et sauvegarder le corpus dans un fichier, soit séquentiellement charger, éditer et écrire le corpus, à la manière d'un flux.

Cette dernière façon de faire est particulièrement utile pour travailler sur des gros corpus qui ne rentrent pas en mémoire. Par exemple, pour le plus gros corpus de la collection, l'OMSTI, le fichier au format UFSAC fait 2,1 Go.

Pour finir, nous fournissons un ensemble d'outils permettant quelques opérations générales sur les corpus, en plus de la conversion d'un corpus depuis son format original :

— conversion de notre format vers le format de Raganato et al. (2017a) pour favoriser les collaborations avec d'autres travaux de l'état de l'art qui s'appuient sur ce dernier ;

— calcul de statistiques générales sur un corpus UFSAC (nombre de phrases, mots, mots annotés, etc.) ;

— évaluation d'un système de DL, en comparant les annotations d'un corpus UFSAC à celles produites par un système de DL ;

— annotation en lemme et partie du discours d'un corpus UFSAC.

Plus de détails sur les outils et la bibliothèque Java que nous fournissons dans notre dépôt Github [16] se trouve en annexe C.

4.5 Conclusion

Au travers des travaux que nous avons présentés dans ce chapitre, nous cherchons à rendre plus simple la distribution et l'utilisation de corpus annotés en sens, grâce à un nouveau format de corpus permettant leur uniformisation nommée UFSAC, qui repose sur la syntaxe du XML.

Cette uniformisaton peut ainsi faciliter la création et l'évaluation de nouveaux systèmes de DL. En effet, les corpus annotés en sens sont, historiquement, non seulement séparés entre corpus d'apprentissage et corpus d'évaluation, mais le format de chacun d'eux est souvent radicalement différent. La syntaxe change et

16. https://github.com/getalp/UFSAC

ne suit parfois aucun standard. Certains corpus sont présentés comme un ensemble de milliers de fichiers et d'autres rassemblent tout en un seul. Enfin, les inventaires de sens utilisés sont aussi différents. En conséquence, beaucoup de systèmes de DL sont entraînés ou évalués sur peu de corpus par rapport à l'ensemble de ceux qui existent.

L'unification de tous les corpus annotés en sens permet ainsi de rapidement étendre un système de DL entraîné initialement sur un ensemble de corpus, avec de nouvelles données, sans avoir à écrire un nouvel analyseur syntaxique. De plus, un système peut maintenant facilement inclure dans sa phase d'apprentissage des corpus initialement destinés à l'évaluation, ou bien évaluer sa performance sur des corpus initialement créés pour l'entraînement. Cela peut permettre une bien meilleure couverture et une meilleure analyse des performances d'un système.

Nous avons rassemblé l'ensemble des corpus anglais annotés en sens WordNet que nous connaissons, et nous les avons convertis au format UFSAC et à l'inventaire de sens de WordNet 3.0, en plus d'avoir effectué des traitements pour nettoyer les textes et les annoter en lemmes et parties du discours. Les corpus ne sont disponibles que lorsque les droits le permettent. Dans le cas contraire, nous fournissons quand même les scripts permettant à ceux qui possèdent ces corpus de les convertir.

De plus, nous fournissons une bibliothèque Java permettant la lecture, l'écriture et la modification de corpus dans notre format unifié, ainsi que des exemples de codes et des outils pour de nombreuses applications telles que la lemmatisation, l'étiquetage en parties du discours, le calcul de la distribution des sens, etc. La ressource est disponible à cette URL : https://github.com/getalp/UFSAC.

Chapitre 5

Architectures neuronales pour la désambiguïsation lexicale supervisée

5.1 Introduction

Comme nous l'avons vu dans la section 1.4 et la figure 1.5, les systèmes de désambiguïsation fondés sur des réseaux de neurones sont classés en deux grandes catégories : d'un côté, ceux qui sont entraînés à directement assigner un sens aux mots donnés en entrée du réseau (Kågebäck et Salomonsson, 2016; Raganato et al., 2017b; Luo et al., 2018a,b) et de l'autre ceux qui reposent sur un modèle de langue permettant de créer des vecteurs de sens à partir des contextes, et qui assignent le sens le plus proche du vecteur de sens ainsi créé (Yuan et al., 2016; Peters et al., 2018; Loureiro et Jorge, 2019).

Dans ce chapitre, nous nous intéressons à la première catégorie de systèmes, où nous voyons une marge de progression importante. En effet, d'une part les représentations vectorielles contextuelles de mots telles que ELMo et BERT n'ont, à l'écriture de cette thèse, jamais été exploitées pour ces systèmes, et d'autre part presque aucun autre corpus que le SemCor n'a été utilisé pour leur entraînement. Enfin, nous pensons que les architectures peuvent être largement simplifiées et tirer parti des récentes avancées comme le modèle Transformer (Vaswani et al., 2017) utilisé largement en traduction automatique pour remplacer les cellules récurrentes type LSTM ou GRU.

Afin d'exploiter notre nouvelle ressource UFSAC, qui regroupe l'intégralité des corpus annotés en sens disponibles pour la DL (voir chapitre 4), nous allons dans un premier temps fournir une analyse détaillée de ces corpus pour l'entraîne-

ment des systèmes supervisés, puis nous allons présenter notre nouvelle architecture. Nous allons ensuite évaluer nos systèmes sur l'ensemble des tâches d'évaluation SensEval/SemEval, puis présenter différentes analyses des résultats en fonction des corpus et des hyper-paramètres, avant de conclure.

Nous utilisons ainsi pour la première fois tous les corpus annotés en sens WordNet à notre connaissance pour l'apprentissage d'un système plus robuste. Par souci de comparaison avec les systèmes état de l'art, nous avons évalué notre approche à la fois en utilisant tous les corpus UFSAC disponibles, mais aussi en nous restreignant uniquement au SemCor.

Le code permettant d'entraîner, évaluer ou exploiter nos modèles ainsi que nos meilleurs modèles pré-entraînés sont tous librement accessibles à l'adresse suivante : `https://github.com/getalp/disambiguate`.

Les travaux présentés dans ce chapitre sont issus d'un de nos article de conférence et de sa version étendue dans un journal (Vial et al., 2018c, 2019b).

5.2 Corpus annotés en sens pour l'entraînement de systèmes supervisés

Les corpus annotés en sens sont la matière première des approches supervisées pour la DL, et leur utilisation ou non en tant que données d'entraînement est ainsi un choix crucial.

Le SemCor (Miller et al., 1993) est indéniablement le corpus le plus utilisé en tant que données d'entraînement dans les systèmes supervisés. Il est en effet la source principale de données annotées des meilleurs systèmes depuis la création des premières campagnes d'évaluation (Hoste et al., 2001; Decadt et al., 2004) jusqu'aux systèmes les plus récents (Yuan et al., 2016; Raganato et al., 2017b; Luo et al., 2018a,b; Loureiro et Jorge, 2019).

D'autres corpus sont aussi utilisés dans un plus faible nombre de travaux. C'est le cas, par exemple, du DSO (Ng et Lee, 1997) qui est utilisé en plus du SemCor dans le système NUS-PT (Chan et al., 2007b), le meilleur système de la campagne d'évaluation SemEval 2007 (Navigli et al., 2007) ou de l'OMSTI (Taghipour et Ng, 2015b), un corpus d'un million de mots annotés en sens, utilisé notamment par Iacobacci et al. (2016) en remplacement du SemCor et qui permet aux auteurs d'obtenir leurs meilleurs résultats. On peut aussi citer le MASC (Ide et al., 2008), un corpus utilisé dans le récent travail de Yuan et al. (2016) pour l'évaluation de leur système.

On remarque ainsi qu'aucun travail n'utilise l'ensemble des corpus annotés en sens disponibles, et rares sont ceux qui justifient l'emploi d'un corpus plutôt qu'un autre. Jusqu'à récemment, ces ressources pouvaient être difficiles d'accès et d'utilisation, notamment à cause des différents inventaires de sens utilisés (différentes versions de WordNet, ou même d'autres bases lexicales), ou bien des différents formats de fichiers. Cependant, avec nos travaux sur la ressource UFSAC (voir chapitre 4) contenant l'ensemble des corpus annotés en sens cités précédemment, dans un format et un inventaire de sens unifiés, leur utilisation est facilitée. Ainsi, nous proposons d'entraîner pour la première fois un système supervisé à partir de toutes ces données.

Dans le tableau 5.1, nous présentons une synthèse des différences notables entre les corpus que nous utiliserons pour l'apprentissage. Nous avons considéré tous les corpus d'UFSAC à l'exception des corpus d'évaluation de SensEval et SemEval afin de pouvoir évaluer nos systèmes sur l'ensemble de ces tâches. De plus, dans notre analyse, nous excluons certaines annotations en sens :

— D'abord, on exclut les annotations à gros grains, c'est-à-dire les annotations pour lesquelles, soit l'annotateur n'a pas pu choisir un seul sens, soit la conversion d'un inventaire lexical à un autre (par ex. d'une ancienne version de WordNet à WordNet 3.0) a laissé une ambiguïté entre plusieurs sens.

— Ensuite, on exclut les annotations sur les mots monosémiques. En effet, ces annotations apportent moins d'informations dans le cas de notre analyse que les annotations sur les mots polysémiques, du fait qu'elles peuvent être créées de façon triviale.

Ces chiffres font ainsi ressortir les caractéristiques propres de chaque corpus.

Comme nous pouvons le voir, l'OMSTI et le Train-O-Matic sont les plus grands corpus en termes de nombre de mots annotés et non annotés, mais ils sont ceux qui ont le plus faible ratio entre ces deux nombres : moins d'un mot sur trente est effectivement annoté en sens, là où pour le SemCor, près d'un mot sur quatre est annoté, et pour le WNGC, près d'un mot sur cinq. Enfin, l'OntoNotes, le DSO et le MASC ont aussi cette caractéristique d'avoir un faible ratio de mots annotés (moins d'un mot sur douze).

Pour ce qui est de la diversité des lemmes, le WNGC se démarque le plus des autres corpus avec 19 149 lemmes différents annotés en sens, mais le SemCor et l'OMSTI en ont aussi beaucoup avec respectivement 11 646 et 11 125. À l'extrême opposé, le DSO ne contient que 191 lemmes différents. À noter que le nombre de lemmes polysémiques dans WordNet 3.0 s'élève à 26 896 [1].

1. https://wordnet.princeton.edu/documentation/wnstats7wn. Consulté le 28/04/20.

Corpus	SemCor	DSO	WNGC	MASC	OMSTI	OntoNotes	Train-O-Matic	Tous
Nombre de mots, en milliers (w_{total})	779	2 705	1 635	585	36 637	2 476	31 708	76 525
Nombre de mots annotés en sens, en milliers (w_{annot})	189	176	329	45	1 074	75	719	2 608
Ratio w_{annot}/w_{total}	0,24	0,07	0,20	0,08	0,03	0,03	0,02	0,03
Nombre de lemmes distincts annotés en sens	11 646	191	19 149	3 766	11 125	2 124	1 100	20 346
Nombre d'exemples par lemme en moyenne	16	922	17	12	97	35	654	128
Nombre d'exemples par sens en moyenne	8	125	8	9	48	21	138	52
Polysémie moyenne dans WordNet des lemmes présents dans le corpus ($p_{wordnet}$)	3,66	10,62	3,18	4,41	3,71	5,35	4,95	3,14
Polysémie moyenne dans le corpus des lemmes présents dans le corpus (p_{corpus})	1,94	7,35	2,27	1,25	2,00	1,69	4,75	2,43
Ratio $p_{corpus}/p_{wordnet}$	0,53	0,69	0,71	0,28	0,54	0,31	0,96	0,78

TABLE 5.1 – Statistiques des corpus d'entraînement de la ressource UFSAC 2.1. Le corpus « Tous » correspond à un corpus constitué de la concaténation de tous les autres.

Dans les cinquième, sixième et septième lignes du tableau, on fait émerger une des caractéristiques principales de ces corpus : le DSO et le Train-O-Matic sont en effet construits autour d'un nombre restreint de lemmes très fortement polysémiques, mais avec un très grand nombre d'exemples pour chaque lemme et chaque sens de ces lemmes. À l'opposé, le MASC, l'OntoNotes, l'OMSTI, le SemCor et le WNGC sont construits plutôt autour d'un certain type de documents (actualités, définitions d'un dictionnaire...) et donc les lemmes sont plus variés, avec un nombre de sens plus variable et un nombre d'exemples par lemme plus réduit.

Enfin les deux dernières lignes font ressortir une dernière caractéristique de ces corpus : à quel point tous les sens de chaque lemme sont représentés au sein du corpus, par rapport à ceux présents dans WordNet. En effet, un ratio inférieur à 1 dans la dernière ligne indique que pour chaque lemme présent dans le corpus, tous ses différents sens existants dans le dictionnaire ne sont pas représentés dans le corpus. On peut même dire que pour le MASC et l'OntoNotes, par exemple, en moyenne moins de la moitié des sens possibles d'un lemme sont représentés dans le corpus, ce qui pourrait poser problème à un système supervisé pour généraliser à de nouvelles données ce qu'il a appris sur ces corpus. Le DSO et le WNGC se distinguent ainsi en ayant plus de deux tiers des sens représentés pour les lemmes présents dans leurs données. Enfin, dans le Train-O-Matic, quasiment tous les sens des lemmes présents sont représentés.

Outre le fait de mieux comprendre quelles sont les différences marquées entre les corpus d'entraînement de la ressource UFSAC, ces chiffres nous permettent de voir l'intérêt de tous les combiner pour l'apprentissage d'un système supervisé de désambiguïsation lexicale. En effet, la colonne « Tous » montre ces statistiques appliquées à cet ensemble constitué de tous les corpus et on voit bien, par exemple sur la quatrième ligne, que cette combinaison constitue un ensemble riche de données annotées, avec plus de 20 000 lemmes uniques annotés en sens.

De plus, on voit à travers cette analyse que le SemCor, corpus le plus largement utilisé dans tous les systèmes supervisés, souffre de fortes faiblesses, en particulier moins de la moitié des lemmes polysémiques de WordNet y sont représentés (voir ligne 4), et pour beaucoup de ces lemmes, tous leurs sens ne sont pas représentés (voir dernière ligne).

Utiliser la combinaison de corpus « Tous » peut toutefois poser d'autres problèmes, notamment parce que le ratio de mots annotés en sens est très faible. Les phrases quasi entièrement annotées du SemCor et du WNGC se retrouvent donc un peu noyées dans les phrases faiblement annotées des autres corpus.

Afin de voir l'impact du choix du corpus d'apprentissage sur les performances du système que nous proposons, nous allons utiliser comme données d'entraîne-

ment cet ensemble de tous les corpus UFSAC d'une part, mais nous allons aussi d'autre part nous restreindre aux données du SemCor uniquement, afin de nous comparer aux systèmes de l'état de l'art entraînés uniquement sur ce corpus. Enfin, nous allons expérimenter avec le couple SemCor+WNGC, ce dernier n'étant exploité dans aucun autre travail alors qu'il nous semble être, sur certains aspects, encore plus complet que le SemCor (en particulier plus de lemmes y sont présents).

5.3 Architecture

Notre approche est, comme celle de Raganato et al. (2017b), de considérer la désambiguïsation lexicale comme un problème de classification dans lequel on assigne une étiquette de sens à chaque mot (voir section 1.4.2.2).

Nous généralisons et simplifions cependant leurs modèles pour ne garder que l'essentiel, c'est-à-dire un système construit en trois couches, représenté dans la figure 5.1 :

— Une **couche d'entrée**, qui transforme les mots en vecteurs.

— Un ensemble de **couches cachées**, qui calculent une nouvelle représentation pour les mots en tenant compte des mots autour.

— Une **couche de sortie**, qui attribue une probabilité de sens à chacun des mots.

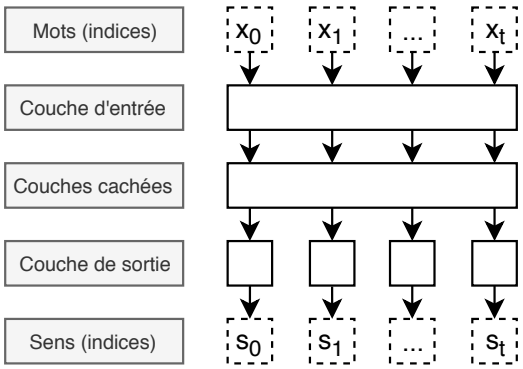


FIGURE 5.1 – Architecture neuronale proposée pour la désambiguïsation lexicale.

Dans la suite de cette section, nous allons détailler chacune de ces trois couches.

5.3.1 Couche d'entrée

À la différence des travaux cités précédemment, notre architecture va permettre trois configurations pour la couche d'entrée. On pourra ainsi choisir :

— d'apprendre des vecteurs à l'aide d'une table de correspondance apprise conjointement avec le modèle ;

— d'utiliser des vecteurs de mot pré-entraînés fixes (type Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), etc.) ;

— d'utiliser des vecteurs de mot contextuels pré-entraînés (type ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), etc.).

Dans le cas où l'on utilise des vecteurs de mot avec un vocabulaire fixe, on associe un vecteur nul aux mots absents du vocabulaire.

Dans le cas de BERT, qui segmente certains mots en plusieurs sous-unités (voir section 2.3.2), par exemple `[rodent]` est découpé en deux sous-unités `[rode]` et `[##nt]`, on aligne le sens à prédire avec la première sous-unité du mot, comme les auteurs préconisent de le faire [2].

5.3.2 Couches cachées

Pour les couches cachées, les travaux précédents utilisaient des couches LSTM bidirectionnelles (Hochreiter et Schmidhuber, 1997), notre système permet en plus d'utiliser des couches d'encodeurs Transformer (Vaswani et al., 2017).

Nous utilisons l'implémentation d'une cellule LSTM incluse dans l'outil *PyTorch* [3] dont nous avons détaillé les formules dans la section 2.2.1.1. Pour obtenir une sortie de LSTM *bidirectionnelles*, nous appliquons ces formules pour chacune des deux directions, et nous concaténons leur sortie.

Pour l'encodeur Transformer, nous utilisons l'implémentation d'OpenNMT [4], dont les formules sont les mêmes que dans l'article d'origine (voir section 2.2.4). À noter cependant que dans le cas où nous utilisons des vecteurs de mot contextualisés en entrée, comme la position des mots est déjà encodée dans leur représentation vectorielle, nous n'ajoutons pas les vecteurs de position comme Vaswani et al. (2017).

2. https://github.com/google-research/bert. Consulté le 28/04/20.
3. https://pytorch.org
4. https://github.com/OpenNMT/OpenNMT-py

5.3.3 Couche de sortie et fonction objectif

Enfin, la dernière couche de notre architecture, la couche de sortie, applique une transformation linéaire et une fonction *softmax* à la sortie des couches cachées, afin de générer pour chacun des mots en entrée, une distribution de probabilité sur tous les sens observés. Nous suivons ainsi la formule $y = softmax(Ax + b)$, avec x la sortie des couches cachées, A la matrice de poids et b le vecteur de biais, A et b étant des paramètres de notre modèle. La fonction softmax étant définie comme suit :

$$y_i = \frac{\mathrm{e}^{x_i}}{\sum_{s=1}^{S} \mathrm{e}^{x_s}}, \forall i \in \{1, \ldots, S\}$$

avec S le nombre de sens possibles dans le dictionnaire.

La fonction de coût à minimiser pendant la phase d'apprentissage est l'entropie croisée entre la couche de sortie et un vecteur de type *one-hot*, pour lequel toutes les composantes sont à 0 sauf à l'index du sens cible où elle est à 1. On cherche ainsi à minimiser la fonction $H(p, q) = -\sum_x p(x) \log q(x)$, où x est une composante du vecteur de la couche de sortie, p est la distribution de probabilité réelle et q la sortie de notre réseau de neurones. Comme toutes les valeurs de la distribution réelle sont à 0 sauf à l'index du sens correct, pour un exemple donné, on cherche ainsi à minimiser la formule $-\log q(s)$, où s est l'index du sens à prédire.

Notre système prédit un sens pour chaque entrée, mais nous effectuons la rétro-propagation du gradient uniquement sur les mots annotés en sens. On garde ainsi un vocabulaire contenant seulement les sens observés pendant l'entraînement, et on ignore les sorties du système sur les mots qui ne sont pas annotés en sens.

Nous pouvons ainsi entraîner notre modèle non seulement sur des données entièrement annotées, comme c'est le cas avec le SemCor (Miller et al., 1993) par exemple, mais également sur des données partiellement annotées, comme le DSO (Ng et Lee, 1997), dans lequel un seul mot est annoté par phrase. Il est en effet capable d'apprendre à prédire les sens de tous les mots annotés dans une séquence en même temps, et à la fois d'en ignorer les éléments non annotés.

5.4 Protocole expérimental

Dans cette section, nous détaillons le processus que nous avons suivi concernant l'apprentissage et l'évaluation de notre système de DL neuronal.

5.4.1 Corpus d'entraînement et de développement

Nous avons tiré parti de notre travail sur UFSAC (voir chapitre 4) et décidé d'utiliser pour la première fois l'ensemble des sept corpus d'entraînement de la ressource, à savoir : le SemCor, le DSO, le WordNet Gloss Corpus (WNGC), l'OMSTI, le MASC, l'Ontonotes et le Train-O-Matic.

Afin de comparer l'architecture que nous proposons avec les travaux de l'état de l'art tels que Raganato et al. (2017b); Yuan et al. (2016); Luo et al. (2018a,b) qui utilisent uniquement le SemCor comme corpus d'apprentissage, nous avons aussi évalué notre approche en limitant l'apprentissage du modèle à ce corpus.

Enfin, nous entraînons un système sur le couple de corpus SemCor+WNGC, car, après notre précédente analyse sur les corpus (voir section 5.2) nous pensons que ces deux corpus allient un bon équilibre entre un taux important des mots annotés, une grande richesse du nombre de lemmes distincts représentés, et une taille très réduite (seulement 3,5 millions de mots contre 76,5 millions de mots pour l'ensemble UFSAC).

Pour le corpus de développement, qui est utilisé pendant l'apprentissage pour évaluer périodiquement notre modèle, afin d'éviter le surapprentissage, nous avons trois cas différents : dans le cas de l'entraînement sur l'ensemble UFSAC, on extrait 10 000 phrases prises aléatoirement dans cet ensemble, dans le cas de l'entraînement sur le SemCor+WNGC, on extrait 4 000 phrases aléatoires, et dans le cas de l'entraînement sur le SemCor seul, nous avons pris les 4 000 premières phrases du WNGC.

Dans certains corpus, les mots sont parfois annotés avec plusieurs sens, soit parce que l'annotateur a trouvé qu'ils étaient tous applicables, ou bien parce qu'ils ont été initialement annotés avec un autre inventaire de sens converti ensuite dans celui de WordNet. Dans ce cas nous supprimons entièrement l'annotation du mot pour ne garder au final que ceux qui ont été annotés avec un seul sens dans notre corpus d'apprentissage. Pour le SemCor, le DSO, le WordNet Gloss Corpus, l'OMSTI et le Train-O-Matic, ce cas ne concerne respectivement que 0,38%, 0,02%, 0%, 0,02% et 0% des annotations. Cependant pour le MASC et l'Ontonotes, qui ont été initialement annotés avec un dictionnaire distinguant moins finement les

sens que WordNet, ce sont respectivement 55,98% et 65,93% des annotations qui sont enlevées.

5.4.2 Hyperparamètres du modèle neuronal

En entrée de notre réseau, nous avons exploité les vecteurs contextuels de BERT (Devlin et al., 2019). Nous avons utilisé le modèle pour l'anglais « bert-large-cased » qui est pré-entraîné sur BookCorpus (Zhu et al., 2015) et Wikipedia, et qui produit des vecteurs de dimension 1024.

Pour les couches cachées, nous avons appliqué six couches d'encodeurs Transformer (Vaswani et al., 2017), avec les même paramètres que le modèle « base » de l'article original (huit têtes d'attention, dimension cachée de 2 048, et régularisation dropout à 0,1). Les couches Transformer s'appuient sur le mécanisme d'auto-attention, et nous les avons utilisées à la place des cellules récurrentes plus classiques comme des LSTM ou des GRU, parce que plusieurs travaux récents ont montré leur plus grande efficacité dans une multitude de tâches (voir section 2.2.4).

À la suite de nos résultats principaux, nous évaluerons l'impact du choix de ces hyperparamètres en comparaison à d'autres, notamment : le remplacement de BERT par des vecteurs fixes (GloVe) et le remplacement de Transformer par des couches récurrentes (LSTM).

5.4.3 Entraînement du modèle neuronal

Les paramètres utilisés pour l'apprentissage sont les suivants :

— La méthode d'optimisation est Adam (Kingma et Ba, 2015), avec les mêmes paramètres par défaut tels que décrits dans leur article.

— Chaque mini-lot (*batch*) contient 100 phrases ;

— les phrases sont tronquées à 80 mots, pour faciliter l'entraînement tout en minimisant la perte d'informations (moins de 5% des mots annotés dans nos données d'entraînement sont perdus) ;

— les séquences sont remplies de vecteurs nuls depuis la fin de façon à ce qu'elles aient toutes la même taille au sein d'un mini-lot.

Nous avons effectué l'apprentissage pendant 20 cycles. Un cycle correspondant à une passe complète sur nos données d'entraînement. Nous avons évalué périodiquement (à la fin de chaque cycle) notre modèle sur le corpus de développement, et conservé uniquement celui ayant obtenu le plus grand score F1.

Cette configuration permet de reproduire aisément nos résultats. En effet, en plus du modèle de vecteurs de mot pré-entraîné, tous les corpus utilisés sont libres d'accès et dans un format unifié [5]. La seule exception est le corpus DSO qui est payant, mais il ne contient qu'approximativement 5% des mots annotés de l'ensemble UFSAC, avec seulement 121 noms et 70 verbes différents. De plus, nous fournissons tout de même le code pour la conversion du DSO dans le format UFSAC pour ceux qui le possèdent. Enfin, nous fournissons aussi notre implémentation du système de DL au sein de l'outil Disambiguate [6] que nous avons développé.

5.4.4 Processus de désambiguïsation

Pour réaliser la désambiguïsation d'une séquence de mots en utilisant le réseau entraîné, la méthode suivante est utilisée :

1. Chaque mot est d'abord transformé en vecteur à l'aide de la couche d'entrée, puis donné en entrée au réseau.
2. En sortie, une distribution de probabilité sur tous les *synsets* observés pendant l'apprentissage est retournée pour chaque élément de la séquence. Nous assignons aux mots leur *synset* le plus probable en suivant cette distribution, parmi les sens possibles du mot cible dans WordNet, en fonction de son lemme et de sa partie du discours. Ces deux informations étant systématiquement données pendant les campagnes d'évaluation de la DL.
3. Si aucun sens n'est assigné, parce qu'aucun des sens du mot cible n'a été observé pendant l'apprentissage, une stratégie de repli est effectuée. La plus courante et celle que nous utilisons est d'assigner au mot son sens le plus fréquent dans WordNet.

Le processus d'apprentissage est forcément stochastique, en effet non seulement les poids du modèle sont initialisés aléatoirement par la bibliothèque sous-jacente, mais le corpus d'apprentissage est également mélangé à chaque début de cycle. Nous avons entraîné ainsi huit modèles séparément pour chacune de nos configurations, puis pour chacune de nos évaluations, nous donnons le score d'un système « ensemble », c'est-à-dire un système qui moyenne les prédictions faites par ces huit modèles afin d'obtenir une nouvelle distribution de sens plus stable et généralement de meilleure qualité.

Pour ce système « ensemble », nous avons utilisé une moyenne géométrique sur les prédictions faites par les modèles. C'est une pratique couramment utilisée (par exemple par Sutskever et al. (2014)) car elle permet non seulement d'avoir

5. https://github.com/getalp/UFSAC
6. https://github.com/getalp/disambiguate

un système moins sensible au bruit et donc plus robuste, mais aussi un système de meilleure qualité. En effet, un modèle peut être individuellement bloqué dans un minimum local pendant l'entraînement et avoir un très bon score sur le corpus de développement, mais être incapable de généraliser, alors qu'il est moins probable que ce problème arrive à l'ensemble de modèles.

Enfin, à la suite de nos résultats principaux, nous analyserons, en plus des scores des ensembles, la moyenne des scores obtenus par chacun des huit modèles, ainsi que leur écart-type. Nous pourrons ainsi nous comparer avec les autres travaux de l'état de l'art qui n'entraînent qu'un seul modèle tout en ayant un moyen d'estimer la variabilité et la significativité des résultats.

5.5 Résultats

Nous avons évalué nos modèles sur tous les corpus d'évaluation des tâches de DL des campagnes d'évaluation SensEval/SemEval. Les scores obtenus par nos systèmes, avec et sans stratégie de repli, sont comparés à ceux des systèmes semblables de l'état de l'art à base de réseaux de neurones (Yuan et al., 2016; Raganato et al., 2017b; Luo et al., 2018b; Loureiro et Jorge, 2019; Huang et al., 2019), ainsi que l'étalon du sens le plus fréquent dans le tableau 5.2.

On remarque d'abord qu'en termes de scores F1, avec la stratégie de repli, notre système entraîné sur l'ensemble des corpus UFSAC obtient globalement de moins bonnes performances que ceux entraînés sur le SemCor et le couple SemCor+WNGC. Sur ces tâches, et avec notre architecture, avoir plus de données annotées en sens n'est donc pas forcément synonyme de meilleures performances.

Sans la stratégie de repli, l'ajout de nouveaux corpus au SemCor montre plus d'intérêts. En effet, sur la tâche « ALL », le SemCor a une couverture de 93,2%, le SemCor+WNGC a une couverture de 98,2% et l'ensemble UFSAC a une couverture de 99,2%. Cependant, même si cette meilleure couverture permet d'avoir de meilleurs résultats avec l'ensemble UFSAC qu'avec le SemCor seul, le couple SemCor+WNGC obtient encore, dans cette configuration, de meilleurs résultats.

Ces premiers résultats montrent, en plus de la très grande qualité du SemCor, que le WNGC offre une très bonne complémentarité, en permettant d'avoir une bien meilleure couverture, sans apporter de bruit qui nuise à l'apprentissage. On notera au passage que le SemCor seul permet d'obtenir les meilleurs résultats sur la tâche de SensEval 3, tâche pour laquelle sa couverture est la plus grande (96,8%).

Afin de comparer nos systèmes à ceux de l'état de l'art, il est d'abord néces-

Système	SE2	SE3	SE07 17	SE13	SE15	ALL (concat. tâches préc.)					SE07 07
						noms	verbes	adj.	adv.	total	
Sens le plus fréquent	65,6	66,0	54,5	63,8	67,1	67,7	49,8	73,1	80,5	65,5	78,9
Yuan et al. (2016)	73,8	71,8	63,5	69,5	72,6	†73,9	-	-	-	†71,5	83,6
Raganato et al. (2017b)	72.0	69.1	⋆64.8	66.9	71.5	71.5	57.5	75.0	83.8	69.9	83.1
Luo et al. (2018b)	72,8	70,3	†⋆85,4	68,5	72,8	72,7	58,2	77,4	84,1	71,1	-
Loureiro et Jorge (2019)	76,3	75,6	68,1	75,1	77,0	-	-	-	-	75,4	-
Huang et al. (2019)	77,7	75,2	⋆72,5	76,1	80,4	79,3	66,9	78,2	86,4	77,0	-
Nos systèmes - Sans stratégie de repli											
SemCor	71,9	74,8	68,7	70,4	73,6	-	-	-	-	72,4	82,9
SemCor+WNGC	78,4	75,4	73,9	77,3	79,7	-	-	-	-	77,3	89,7
UFSAC	75,6	73,0	64,0	71,6	74,0	74,3	62,6	81,5	85,3	73,1	87,0
Nos systèmes - Avec stratégie de repli											
SemCor	77,2	**76,5**	70,1	74,7	77,4	78,7	65,2	79,1	85,5	76,0	87,7
SemCor+WNGC	**79,7**	76,1	**74,1**	**78,6**	**80,4**	80,6	68,1	82,4	86,1	**78,3**	**90,4**
UFSAC	75,9	73,5	64,2	72,4	74,5	75,0	62,7	81,9	85,3	73,6	87,7

TABLE 5.2 – Scores F1 (%) sur les tâches de DL de l'anglais des campagnes d'évaluation SensEval/SemEval. La tâche « ALL » est la concaténation de SE2, SE3, SE07 17, SE13 et SE15. Les scores remplacés par des tirets (-) ne sont pas fournis par les auteurs. Les scores prefixés par une obélisque (†) ne sont pas fournis par les auteurs mais sont déduits de leurs autres scores. Les scores préfixés par une étoile (⋆) concernent un corpus utilisé en tant que corpus de développement.

saire de préciser les ressources exploitées par chacun d'entre eux. En effet, en plus d'utiliser le SemCor comme données annotées en sens, ces données sont très variables : Raganato et al. (2017b) utilisent Word2Vec (Mikolov et al., 2013) comme vecteurs pré-entraînés, Luo et al. (2018b) utilisent GloVe (Pennington et al., 2014), Yuan et al. (2016) utilisent des vecteurs calculés sur un corpus privé de 100 milliards de mots, Loureiro et Jorge (2019) et Huang et al. (2019) utilisent, comme nous, BERT. Enfin, en plus des vecteurs de mot, Luo et al. (2018b) et Huang et al. (2019) utilisent les définitions des sens dans WordNet.

Système	Données annotées en sens	Vecteurs en entrée	Données supplémentaires	Score F1 (%)
Yuan et al. (2016)	SemCor	privé	corpus privé	71,5
Raganato et al. (2017b)	SemCor	Word2Vec	-	69,9
Luo et al. (2018b)	SemCor	GloVe	WordNet (définitions)	71,1
Loureiro et Jorge (2019)	SemCor	BERT	-	75,4
Huang et al. (2019)	SemCor	BERT	WordNet (définitions)	77,0
Notre système	SemCor	BERT	-	76,0
Notre système	SemCor + WNGC	BERT	-	78,3

TABLE 5.3 – Score sur la tâche « ALL » de nos systèmes et de ceux de l'état de l'art en fonction des ressources exploitées.

Nous récapitulons les scores sur la tâche « ALL » en fonction de ces ressources dans le tableau 5.3. Ainsi, à ressources équivalentes, notre architecture nous permet d'obtenir de meilleurs résultats que Loureiro et Jorge (2019) (+0,6 points sur la tâche « ALL »). Avec l'ajout du WNGC comme corpus d'apprentissage, nous surpassons les scores des meilleurs systèmes (Huang et al., 2019) de 1,3 points de pourcentage. Les ressources utilisées ne sont pas tout à fait équivalentes, mais rappelons que le WNGC est un corpus constitué des définitions de WordNet annotées en sens par l'équipe de WordNet, et qui est fourni avec la base lexicale.

Concernant les architectures utilisées par ces autres systèmes, celle de Raganato et al. (2017b) consiste en un réseau neuronal en trois couches comme la nôtre, mais avec des composants supplémentaires. Ils effectuent ainsi un apprentissage multi-tâches, dans lequel leur réseau prédit un label de mot ou de sens, ainsi que la partie du discours (POS) du mot, et son label sémantique dans WordNet (LEX). Dans une section suivante, nous mènerons une analyse des performances de notre système en utilisant les mêmes ressources qu'eux (notamment en

utilisant Word2Vec en remplacement de BERT), afin de comparer nos systèmes à ressources vraiment égales.

Le système de Luo et al. (2018b) quant à lui, repose sur une architecture neuronale avec un réseau par lemme (comme décrit dans la section 1.4.2.2), comme Kågebäck et Salomonsson (2016), avec en plus un mécanisme d'attention entre le contexte d'un mot à désambiguïser et les définitions de ses sens dans WordNet.

Le système de Yuan et al. (2016) a une architecture différente de la nôtre (similaire à celle de Loureiro et Jorge (2019), voir section 1.4.2.3), mais les auteurs exploitent, en plus du SemCor, une grande quantité de phrases issues d'un corpus privé brut (100 milliards de mots). On peut alors difficilement comparer notre système à ressource équivalente. Le système de Loureiro et Jorge (2019), lui, utilise comme nous BERT, en plus du SemCor, et obtient des résultats inférieurs.

5.5.1 Analyse des hyperparamètres

Afin de mieux comprendre l'influence de nos principaux hyperparamètres sur les performances de nos systèmes, et pour pouvoir comparer nos résultats avec les autres systèmes de l'état de l'art ayant une même architecture, et à ressource équivalente, nous proposons de faire varier trois hyperparamètres : les vecteurs pré-entraînés en entrée, les couches cachées, et la méthode d'ensemble.

Pour les vecteurs en entrée, nous allons comparer BERT avec d'autres vecteurs contextualisés : ELMo, ainsi que des vecteurs fixes : Word2Vec (le même modèle utilisé par Raganato et al. (2017b)) et GloVe (le même modèle utilisé par Kågebäck et Salomonsson (2016) et Luo et al. (2018a,b)). ELMo produit, comme BERT, des vecteurs de dimension 1 024, les vecteurs de Word2Vec sont de dimension 400 et les vecteurs de GloVe sont de dimension 300.

Pour les couches cachées, nous allons comparer l'utilisation de Transformer par rapport à des cellules LSTM (utilisées dans tous les travaux cités dans le paragraphe précédent). Pour Transformer, nous allons utiliser les paramètres du modèle *base* de Vaswani et al. (2017). Pour les LSTM, nous utilisons une seule couche de cellules LSTM bidirectionnelles de taille 1 000. C'est environ deux fois moins que Raganato et al. (2017b), qui utilisent deux couches de tailles 1 024. Nous avons en effet montré dans un de nos articles (Vial et al., 2019b) qu'avoir deux couches au lieu d'une n'apportait aucun gain à notre système.

Avec Transformer, le nombre de paramètres dépend fortement de la taille des vecteurs utilisés en entrée (car ils sont propagés à chaque couche), c'est pourquoi

afin de garder un nombre de paramètres comparable d'un modèle de vecteurs à un autre, nous allons redimensionner les vecteurs à 512 dans chacun des cas.

Enfin, nous allons comparer les performances des modèles individuels (avec moyenne et écarts-types) face à la méthode d'ensemble que nous utilisons (qui moyenne les prédictions des modèles).

Vecteurs en entrée	Couches cachées	#Paramètres	Scores		
			Moyenne	Écart-type	Ensemble
Raganato et al. (2017b) BLSTM (sans attention ni multi-tâche)					
Word2Vec	LSTM	>90M⋆	68,9	-	-
Word2Vec	LSTM	63,44M	67,95	±0,31	68,05
Word2Vec	Transformer	32,51M	66,45	±0,29	67,65
GloVe	LSTM	62,64M	70,51	±0,16	70,77
GloVe	Transformer	32,46M	68,90	±0,48	70,07
ELMo	LSTM	68,44M	72,28	±0,17	72,71
ELMo	Transformer	32,83M	72,07	±0,16	73,83
BERT	LSTM	68,44M	74,58	±0,15	74,77
BERT	Transformer	32,83M	75,00	±0,28	76,02

TABLE 5.4 – Scores F1 (%) sur la tâche « ALL » et nombre de paramètres de nos systèmes et de celui de Raganato et al. (2017b), entraînés sur le SemCor, en fonction des hyperparamètres du modèle neuronal. L'étoile (⋆) indique que le nombre de paramètres n'est pas donné par les auteurs mais est estimé. Les tirets (-) remplacent l'écart-type et le score avec un ensemble des auteurs qui n'évaluent qu'un seul modèle.

Dans le tableau 5.4, on peut ainsi voir qu'à ressources strictement égales, notre architecture obtient des résultats similaires à ceux de Raganato et al. (2017b) pour leur système de base, avec un nombre de paramètres bien moins important. Avec GloVe à la place de Word2Vec, nous améliorons nos résultats de plus de 2 points. Mais ce sont surtout les modèles de langue pré-entraînés (ELMo et BERT) qui nous font gagner encore 2 à 5 points supplémentaires.

Si on compare maintenant le choix des couches cachées, on remarque d'abord qu'utiliser Transformer avec les vecteurs fixes a tendance à dégrader les résultats. Avec ELMo, les résultats avec Transformer sont meilleurs seulement en utilisant la méthode des ensembles. Enfin, avec BERT, Transformer apporte clairement de meilleures performances que des LSTM. On notera qu'avec les paramètres que nous avons choisis, Transformer offre ces bonnes performances avec deux fois moins de paramètres que les LSTM.

5.5.2 Robustesse face aux taux d'annotation

Comme nous l'avons vu dans nos résultats principaux (voir tableau 5.2), l'ensemble des corpus UFSAC utilisé comme données d'entraînement de notre système offre de moins bons résultats que le SemCor seul, ou bien que le couple SemCor+WNGC.

Les corpus inclus dans la ressource UFSAC ont une méthode de construction qui peut varier énormément d'un corpus à un autre. On a notamment vu dans la section 5.2 qu'un critère les distinguait particulièrement : le taux d'annotation par mot. En effet, les corpus comme le DSO et l'OMSTI sont construits autour d'un ensemble de lemmes, et seuls ces lemmes sont annotés au sein d'un très grand nombre de phrases, tandis que les corpus comme le SemCor ou le WNGC sont construits autour de documents pour lesquels tous les mots de toutes les phrases sont annotés en sens.

Le mélange de phrases avec des taux d'annotation variables au sein des données d'entraînement peut nuire à l'apprentissage de notre réseau de neurones. C'est pourquoi, afin de mesurer la robustesse de notre architecture face à des données d'entraînement très hétérogènes comme l'ensemble UFSAC, nous avons mené l'expérience qui consiste à répartir les annotations du SemCor en plusieurs phrases, et de mesurer les répercussions sur notre système.

Nous avons créé deux nouvelles versions du SemCor. Une version « éclatée » à 100% dans laquelle pour chaque phrase contenant n annotations nous dupliquons cette phrase n fois et annotons un seul mot différent dans chacune d'elles. Puis une version « éclatée » à 50% dans laquelle nous appliquons cette opération sur seulement la moitié des phrases du SemCor, puis mélangeons toutes les phrases. Nous avons ainsi entraîné huit modèles sur chacun de ces corpus, et les résultats de nos systèmes sur le corpus de développement se trouvent dans le tableau 5.5.

Données d'entraînement	Nombre de phrases	Score F1 (%)	Écart-type
SemCor original	36 321	72,24	±0,64
SemCor « éclaté » à 50%	132 945	71,50	±0,62
SemCor « éclaté » à 100%	227 993	71,72	±0,61

TABLE 5.5 – Résultats obtenus par notre système sur notre corpus de développement, selon la proportion de mots annotés dans le SemCor.

Comme nous pouvons le voir, bien qu'une tendance se dégage en faveur de la version originale du SemCor, même en se plaçant dans le cas extrême d'un seul mot annoté par phrase, l'impact sur le score final est négligeable. Il semble ainsi

que notre architecture soit bien capable d'apprendre sur des phrases entièrement comme partiellement annotées, et soit assez insensible à ces variations.

5.5.3 Analyse des erreurs

Afin de mieux comprendre l'impact du choix du corpus d'entraînement sur la qualité des annotations produites par notre système, nous proposons dans cette section une analyse plus fine des sorties, en nous appuyant sur les statistiques du tableau 5.6. Ces chiffres sont calculés avec les sorties de nos systèmes sur l'ensemble des corpus d'évaluation (tâche « ALL ») et sans la stratégie de repli.

Tout d'abord la première chose que l'on peut remarquer, dans les deux premières sections du tableau, est que sur les 7 253 mots à annoter, l'ajout du WNGC aux données d'entraînement permet d'annoter 364 mots de plus qu'en utilisant le SemCor seul, soit une couverture supplémentaire d'environ 5% des mots à annoter. En ajoutant l'intégralité d'UFSAC, c'est 67 mots supplémentaires qui sont couverts, soit environ 1% du total des mots à annoter. Concernant les autres métriques, la précision du système entraîné sur SemCor+WNGC est améliorée de plus de 2 points par rapport à celui entraîné sur le SemCor, mais elle est dégradée d'un peu moins de 2 points en utilisant l'ensemble UFSAC. Au niveau du rappel, l'ensemble UFSAC est meilleur que le SemCor seul, mais c'est toujours le couple SemCor+WNGC qui est en tête, de même que pour la mesure F1.

La troisième section du tableau, « Moyenne du nombre de sens dans WordNet », montre une tendance visible sur les trois systèmes : les mots mal annotés sont généralement aussi les plus polysémiques. Cette statistique permet de rappeler que même pour les systèmes de désambiguïsation les plus performants, la distinction extrêmement « fine » entre les sens de WordNet reste un des challenges principaux de cette tâche. Pour rappel, le verbe « make » a, par exemple, 49 sens différents dans WordNet 3.0.

Dans la quatrième section, « Nombre moyen d'exemples du lemme cible », on peut voir un indice supplémentaire pour montrer la qualité du SemCor et du WNGC par rapport aux autres corpus annotés en sens. En effet, on peut voir que le rapport entre le nombre moyen d'exemples des lemmes des mots bien annotés sur ceux des mots mal annotés est supérieur pour le SemCor et le WNGC par rapport à l'ensemble UFSAC. Cela tend à montrer que, sur les corpus de bonne qualité, plus il y a d'exemples d'un lemme et plus il y a de chances de bien désambiguïser ce lemme. On peut aussi émettre l'hypothèse que parfois, un trop grand nombre d'exemples dans le corpus d'entraînement va nuire à l'apprentissage des systèmes supervisés en ajoutant plus de bruit que d'informations pertinentes.

Corpus d'entraînement	SemCor	SC+WNGC	UFSAC
Nombre de mots			
total des mots à annoter	7 253	7 253	7 253
mots non annotés	491	127	60
mots bien annotés	5 009	5 474	5 225
mots mal annotés	1 753	1 652	1 968
Métriques			
Couverture (C)	93,23%	98,25%	99,17%
Précision (P)	74,08%	76,82%	72,64%
Rappel (R)	69,06%	75,47%	72,04%
F-mesure (F1)	71,48%	76,14%	72,34%
Moyenne du nombre de sens dans WordNet			
mots bien annotés	5,42	5,19	4,89
mots mal annotés	8,26	8,28	8,49
Nombre moyen d'exemples du lemme cible dans le corpus d'entraînement			
(a) mots bien annotés	125	223	1 318
(b) mots mal annotés	174	308	2 001
rapport a/b	0,72	0,72	0,66
Nombre moyen d'exemples du sens attendu dans le corpus d'entraînement			
(a) mots bien annotés	77	135	675
(b) mots mal annotés	22	42	397
rapport a/b	3,50	3,21	1,70
Représentation des différents sens du lemme cible dans le corpus d'entraînement			
mots bien annotés	1,77	2,23	2,20
mots mal annotés	0,72	1,17	1,41
Nombre de mots mal annotés dont le sens attendu n'est jamais représenté dans le corpus d'entraînement	494	113	40

TABLE 5.6 – Analyse des erreurs commises sur l'ensemble des corpus d'évaluation (tâche « ALL ») par nos systèmes en fonction du corpus d'entraînement.

Ensuite, le nombre d'exemples du sens attendu dans le corpus d'entraînement montre que les sens les plus vus dans les données d'apprentissage sont généralement plus souvent choisis par le système. Autrement dit, si un sens est souvent représenté, que ce soit dans un contexte pertinent ou bien dans un contexte bruité, le système aura plus souvent tendance à le sélectionner. Là encore, on voit une nette différence entre les corpus SemCor et WNGC par rapport à l'ensemble UFSAC : pour ce dernier, le rapport du nombre de sens des mots bien annotés sur les mots mal annotés est bien en dessous des deux autres corpus, ce qui tend à montrer que, même si un sens est souvent représenté dans ce corpus, ces exemples n'aident pas forcément notre système à correctement le désambiguïser.

L'avant-dernière section « Représentation des différents sens » met en avant une statistique particulière : pour les mots bien et mal annotés, est-ce que les différents sens du lemme cible étaient représentés de manière équilibrée dans le corpus d'entraînement ? Ces chiffres s'appuient sur la formule suivante :

$$\frac{nombre\ d'exemples\ du\ sens\ cible}{nombre\ d'exemples\ du\ lemme\ cible} * nombre\ de\ sens\ du\ lemme\ cible$$

Plus le chiffre est proche de 1, et plus la représentation du sens est « équilibrée », c'est-à-dire qu'il n'est pas sur-représenté ni sous-représenté par rapport aux autres sens possibles du lemme. Sur nos trois systèmes, les chiffres montrent qu'en général, les mots bien annotés ont en moyenne un sens qui est quasiment deux fois plus représenté que les autres sens, et les sens mal assignés sont moins représentés dans le cas du SemCor, et assez équilibrés dans le cas de l'ensemble de corpus.

Enfin, les derniers chiffres du tableau mettent en avant le fait que, dans de nombreux cas, sur ces corpus d'évaluation, même si le système est capable d'annoter un mot en sens, il ne pourra jamais sélectionner le sens attendu, tout simplement parce qu'il ne l'a jamais observé pendant l'apprentissage. C'est le cas pour 494 mots, pour le système appris sur le SemCor, ce qui correspond à plus de 6,8% du total des mots à annoter, qu'aucun système supervisé appris sur ces données ne serait capable d'annoter correctement. Pour le système SC+WNGC ce chiffre est de 1,5%, et pour l'ensemble UFSAC, il tombe à 0,5% du total des mots à annoter, cela montre ainsi l'intérêt d'utiliser plus de données annotées pour l'entraînement de systèmes supervisés robustes.

Pour conclure, ces statistiques permettent de mieux se rendre compte de l'impact des corpus choisis pour l'entraînement sur les performances de notre système, mais aussi de l'importance de la quantité et de la qualité des données annotées en sens pour l'entraînement de systèmes supervisés en général.

En effet, si la majorité des systèmes supervisés sont entraînés sur le SemCor seulement, très peu sont les travaux qui justifient son utilisation plutôt qu'un autre

corpus. Avec l'ensemble des corpus annotés en sens WordNet maintenant facilement accessibles et dans un format unifié, il est désormais plus facile d'identifier et de sélectionner des corpus ou même seulement des parties de corpus qui seront bénéfiques aux systèmes de désambiguïsation supervisés.

5.6 Conclusion

Nous présentons dans ce chapitre une nouvelle architecture de réseau neuronal pour la désambiguïsation lexicale fondée sur trois couches : tout d'abord une couche d'entrée, qui permet de convertir des mots sous forme vectorielle, soit à l'aide de vecteurs statiques pré-entraînés, entraînés conjointement avec le modèle, ou encore avec des vecteurs pré-entraînés contextualisés ; ensuite, un ensemble de couches cachées, permettant d'utiliser des cellules récurrentes (LSTM) ou un encodeur Transformer ; et enfin, une couche de sortie qui permet d'assigner une probabilité sur tous les *synsets* vus pendant l'entraînement.

Après avoir mené une analyse des corpus UFSAC pour l'entraînement d'un système comme le nôtre, nous avons décidé d'entraîner notre modèle neuronal sur trois ensembles : le SemCor, parce qu'il est utilisé dans la grande majorité des travaux de l'état de l'art, le couple SemCor+WNGC, parce que ce dernier nous semble être d'aussi bonne qualité que le SemCor, et tous les corpus d'entraînement d'UFSAC, à savoir le SemCor, le DSO, le WNGC, l'OMSTI, le MASC, l'Ontonotes et le Train-O-Matic. Nous avons évalué notre système sur toutes les tâches de DL des campagnes d'évaluation SensEval/SemEval.

Notre système se distingue par sa simplicité, tout en obtenant des résultats surpassant l'état de l'art sur toutes les tâches de désambiguïsation lexicale. Nous avons ainsi montré qu'en utilisant aussi le SemCor comme seules données d'entraînement, et BERT comme modèle de langue pré-entraîné, nous obtenons des résultats supérieurs à ceux de Loureiro et Jorge (2019). En ajoutant le WordNet Gloss Corpus aux données d'entraînement, notre système utilise des ressources comparables à Huang et al. (2019) (qui utilisent les définitions de WordNet non annotées), et nous obtenons encore des résultats bien supérieurs.

Cependant, nous avons aussi montré dans nos résultats que l'utilisation de tous les corpus de la ressource UFSAC pouvait nuire à l'apprentissage, du fait d'une trop grande quantité de bruit apporté par les corpus non annotés manuellement.

Afin d'aller plus loin, nous avons présenté une analyse fine des résultats de nos systèmes afin de comprendre l'impact du choix des corpus d'entraînement pour l'apprentissage d'un système supervisé tel que le nôtre. Nous montrons ainsi les

effets positifs d'utiliser davantage de données annotées en sens, mais aussi les problèmes que peuvent apporter des données de moins bonne qualité. Nous espérons ainsi voir se développer des approches supervisées utilisant d'autres corpus que le SemCor uniquement, notamment le couple SemCor+WNGC.

Finalement, nous avons mis en avant l'importance d'entraîner plusieurs réseaux de neurones séparément et de moyenner leurs prédictions au sein d'un système « ensemble » qui permet de maximiser les performances de nos systèmes.

Les études sur les réseaux de neurones pour la désambiguïsation lexicale sont encore très récentes comme en atteste le faible nombre de systèmes existants pour le moment. C'est cependant une direction prometteuse, tant les résultats obtenus par ces nouveaux systèmes ont montré leur qualité dans les campagnes d'évaluation.

Chapitre 6

Compression de vocabulaire de sens

6.1 Introduction

Comme on l'a vu au chapitre 1, en désambiguïsation lexicale, les méthodes supervisées sont de loin les plus représentées car elles offrent généralement les meilleurs résultats dans les campagnes d'évaluation (par exemple (Navigli et al., 2007)). Les classifieurs état de l'art combinaient jusqu'à récemment des caractéristiques précises telles que les parties du discours et les lemmes des mots voisins, (Zhong et Ng, 2010), mais ils sont maintenant remplacés par des réseaux de neurones récurrents qui apprennent leur propre représentation des mots (voir chapitre 5).

Une des limitations majeures des systèmes supervisés est la quantité limitée de corpus manuellement annotés en sens. En effet, le SemCor (Miller et al., 1993), qui est le plus grand corpus manuellement annoté en sens disponible, contient 33 760 labels de sens différents, ce qui correspond à seulement 16% environ de l'inventaire de sens de WordNet [1] (Miller et al., 1990).

De nombreux travaux tentent de résoudre ce problème via la création de nouveaux corpus annotés en sens, générés soit automatiquement (Pasini et Navigli, 2017), soit semi-automatiquement (Taghipour et Ng, 2015b), ou bien par *crowdsourcing* (Yuan et al., 2016). Ces corpus ainsi que d'autres sont ainsi tous regroupés au sein d'une même ressource grâce à nos travaux (voir chapitre 4), mais on a vu au chapitre précédent qu'ils ne sont pas tous d'aussi bonne qualité que le SemCor et apportent ainsi du bruit qui peut nuire à l'apprentissage.

1. https://wordnet.princeton.edu/documentation/wnstats7wn

Dans ce chapitre, nous allons explorer une nouvelle solution à ce problème en tirant parti des relations sémantiques présentes entre les sens de WordNet comme l'hyperonymie, l'hyponymie, l'antonymie, la méronymie, etc. afin d'étendre la couverture d'un même corpus, sans perdre en qualité.

Notre méthode est fondée sur les observations suivantes :

1. Un sens et ses sens voisins dans le graphe des relations sémantiques de WordNet véhiculent tous une même idée ou concept, à des niveaux d'abstraction différents.
2. Dans certains cas, un mot peut être désambiguïsé en utilisant seulement les sens voisins de ses sens, et pas nécessairement ses sens propres.
3. Par conséquent, nous n'avons pas besoin de connaître tous les sens de WordNet pour désambiguïser tous les mots de WordNet.

Par exemple, considérons le mot « souris » et deux de ses sens : la souris *d'ordinateur* et la souris *l'animal*. Les notions plus générales comme « être vivant » (hyperonyme de souris/animal) et « appareil électronique » (hyperonyme de souris/ordinateur), permettent déjà de distinguer les deux sens, et toutes les notions plus spécialisées telles que « rongeur » ou « mammifère » sont, elles, superflues. En regroupant ces étiquettes de sens ensemble, on peut bénéficier de tous les autres exemples mentionnant un appareil électronique ou un être vivant dans un corpus d'entraînement, même si le mot « souris » n'est pas mentionné spécifiquement, pour désambiguïser le mot « souris ».

Dans ce chapitre, nous émettons ainsi l'hypothèse que seul un sous-ensemble des sens de WordNet peut être considéré pour pouvoir désambiguïser tous les mots de la base lexicale. Par conséquent, nous proposons deux méthodes différentes pour construire ce sous-ensemble que nous appelons méthodes de compression de vocabulaire de sens. En utilisant ces techniques, nous sommes en mesure d'améliorer considérablement la couverture des systèmes de DL supervisés, en éliminant quasiment le besoin d'une stratégie de repli habituellement employée pour les mots jamais observés pendant l'entraînement. Nous présentons des résultats qui surpassent l'état de l'art de façon significative sur toutes les tâches d'évaluation de la DL, et nous fournissons à la communauté notre outil ainsi que nos meilleurs modèles pré-entraînés, sur notre dépôt GitHub dédié [2].

Les travaux présentés dans ce chapitre sont issus de deux de nos articles de conférence : Vial et al. (2019c) (en français) et Vial et al. (2019a) (en anglais).

2. https://github.com/getalp/disambiguate

6.2 Travaux connexes

Plusieurs travaux exploitent déjà l'idée de grouper ensemble plusieurs sens de WordNet afin de créer un inventaire de sens avec une granularité moins fine, pouvant être plus utile aux tâches utilisant la désambiguïsation lexicale.

Dans cette section, nous allons décrire deux de ces méthodes : les *Supersenses* (Ciaramita et Altun, 2006) et les *Basic Level Concepts* (Izquierdo et al., 2007), et expliquer en quoi elles diffèrent des nôtres.

Dans les travaux de Ciaramita et Altun (2006), les auteurs proposent un système supervisé qui est entraîné à prédire des étiquettes de « supersens », qui sont des grandes catégories de sens, déjà utilisées dans WordNet afin d'organiser tous les sens de l'inventaire. Cet ensemble consiste ainsi en 26 catégories pour les noms (telles que « nourriture », « personne » ou encore « objet ») et 15 catégories pour les verbes (telles que « émotion » ou encore « météo »).

En prédisant des supersens à la place des habituels sens à granularité fine de WordNet, le vocabulaire de sortie de leur système est réduit à seulement 41 différentes classes, ce qui les mène à un petit modèle facile à entraîner, et capable de réaliser de la DL partielle, qui peut être suffisante à des tâches de TAL pour lesquelles la distinction fine des sens n'est pas nécessaire.

Dans les travaux de Izquierdo et al. (2007), les auteurs proposent plusieurs méthodes pour construire un ensemble de *Basic Level Concepts* (BLC). Les BLC sont des groupes de sens proches, dont la taille, qui est généralement plus petite que les supersens, peut être contrôlée par un seuil. Leur méthode s'appuie sur les relations sémantiques de WordNet, et, comme pour Ciaramita et Altun (2006), ils évaluent leur méthode sur une tâche de DL modifiée, dont l'objectif est de prédire le bon groupe du sens cible plutôt que le sens WordNet original.

La principale différence entre ces méthodes et celles décrites dans ce chapitre réside dans cet objectif : en effet, nous voulons construire et exploiter des groupes de sens pour l'amélioration des systèmes de DL évalués sur l'inventaire de sens original de WordNet, à granularité fine, et non sur des tâches modifiées. Ainsi, si les méthodes que nous proposons permettent aussi de générer des groupes de sens, ces groupes sont construits de telle sorte que deux sens différents d'un même lemme résident toujours dans deux groupes différents. De cette manière, au moment de désambiguïser un mot, nos systèmes produisent dans un premier temps des étiquettes de groupes de sens, mais nous retrouvons ensuite le sens original à partir de ce groupe et du lemme du mot cible, en gardant une trace des groupes auxquels appartient chaque sens de WordNet.

6.3 Compression de vocabulaire de sens

Les systèmes supervisés neuronaux de l'état de l'art tels que Yuan et al. (2016); Raganato et al. (2017b); Le et al. (2018); Luo et al. (2018b,a); Loureiro et Jorge (2019) sont tous confrontés aux mêmes limitations :

1. La quantité de données annotées manuellement en sens étant très restreinte, il se peut qu'un mot cible ne soit jamais observé pendant l'entraînement. Dans ce cas, le système ne peut pas être en mesure de l'annoter, et une stratégie de repli est généralement effectuée (par exemple, utiliser le premier sens du mot dans WordNet).
2. Pour la même raison, un mot peut être observé, mais pas tous ses sens. Dans ce cas, le système va être capable d'annoter ce mot, mais si le sens attendu n'a jamais été observé, le résultat sera faux, quelle que soit l'architecture sous-jacente du système supervisé.
3. L'empreinte mémoire des modèles neuronaux ainsi que leurs temps d'entraînement et d'exécution augmentent avec la quantité de données d'apprentissage et le nombre d'étiquettes de sens différentes prises en compte, nombre qui monte jusqu'à 206 941 si l'on considère toutes les étiquettes de sens de WordNet.

Afin de résoudre ces problèmes, nous proposons deux nouvelles méthodes permettant de regrouper ensemble des étiquettes de sens qui se réfèrent à des concepts similaires, tout en nous assurant que ces groupes de sens permettent toujours de discriminer les différents sens de tous les mots du lexique, afin de retrouver l'étiquette de sens originale pour un mot au moment de le désambiguïser. En conséquence, le vocabulaire de sens, c'est-à-dire le nombre total d'étiquettes de sens dans notre inventaire de sens diminue, le système est capable de mieux généraliser, et sa couverture augmente.

Plus précisément, en construisant des groupes de sens similaires, la couverture d'un même corpus d'apprentissage va augmenter car chaque exemple d'un sens va aussi fournir un exemple pour tous les sens de son groupe. De plus, le vocabulaire de sortie et la taille des modèles d'apprentissage vont être réduits car les modèles vont prédire des étiquettes de groupes de sens, en nombre inférieur aux sens eux-mêmes. Enfin, au moment de désambiguïser, parce que le lemme des mots à désambiguïser est connu et que tous les sens d'un même lemme résident dans un groupe différent, nous serons capables de retrouver le sens original à partir du groupe prédit.

Nous appelons ainsi ces méthodes « méthodes de compression de vocabulaire de sens », parce qu'elles permettent de réduire la taille d'un vocabulaire tout en conservant son information initiale.

6.3.1 Des sens aux *synsets* : une première compression de vocabulaire de sens à travers la synonymie

Dans la base de données lexicale WordNet (Miller et al., 1990), les sens sont organisés en ensembles de synonymes appelés *synsets*. Un *synset* est concrètement un groupe d'un ou plusieurs sens qui ont la même définition et donc la même signification. Par exemple, les premiers sens des mots *eye*, *optic* et *oculus* appartiennent tous au même *synset* dont la définition est « l'organe de la vue ».

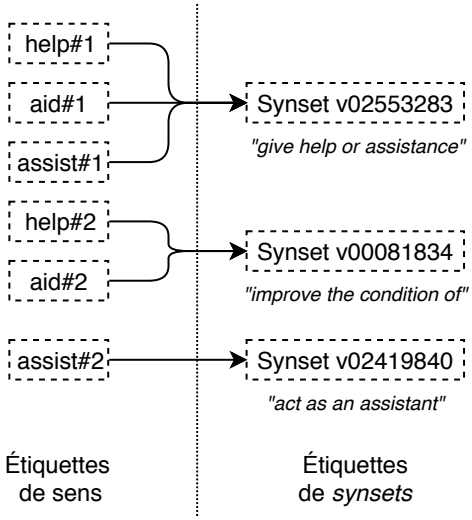


FIGURE 6.1 – Conversion des étiquettes de sens vers des étiquettes de *synsets*, appliqué aux deux premiers sens des mots *help*, *aid* et *assist*. Le nombre de sens différents dans notre vocabulaire passe ainsi de six à trois.

La conversion des étiquettes de sens (« Xème sens du mot N ») aux étiquettes de *synsets* (« *synset* numéro Y »), illustrée dans la figure 6.1, est ainsi une façon de compresser le vocabulaire qui est déjà appliquée dans plusieurs travaux (Yuan et al., 2016; Le et al., 2018; Vial et al., 2019b) sans être toujours explicitement précisée. Cette méthode contribue pourtant clairement à améliorer la couverture des systèmes supervisés. En effet, si le verbe « *aid* » annoté avec son premier sens est observé dans les données d'apprentissage, le contexte autour du mot cible peut être aussi utile pour annoter ultérieurement les verbes *assist* ou *help* avec la même étiquette de *synset*.

En allant plus loin, on peut trouver d'autres informations dans WordNet qui peuvent aider à mieux généraliser. La première nouvelle méthode que nous proposons repose ainsi sur ce même principe de regroupement de sens, mais en exploitant les relations d'hyperonymie et d'hyponymie entre les sens.

6.3.2 Compression de vocabulaire de sens à travers les relations d'hyperonymie, d'hyponymie et d'instance

Selon Polguère (2003), l'hyperonymie et l'hyponymie sont deux relations sémantiques qui correspondent à un cas particulier d'inclusion de sens : l'hyponyme d'un terme est une spécialisation de ce terme, alors que son hyperonyme est une généralisation. Par exemple, une « souris » est un type de « rongeur » qui est à son tour un type d'« animal ». Dans WordNet, ces relations lient presque tous les noms ensemble allant de la racine générique, le nœud « entité » aux feuilles les plus spécifiques, par exemple « souris à pattes blanches ». Si l'on prend aussi en compte la relation d'instance, qui fonctionne de la même manière mais qui lie les entités nommées aux noms courants (par exemple, « Einstein » est une instance de « physicien »), tous les noms de WordNet font partie de cette même hiérarchie.

Ces relations sont également présentes pour plusieurs verbes : par exemple, « additionner » est une manière de « calculer », qui est à son tour une manière de « raisonner ».

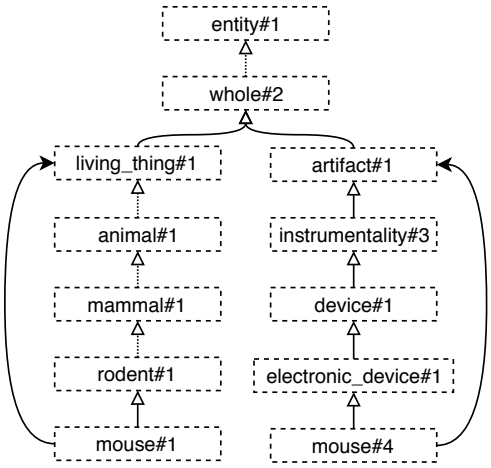


FIGURE 6.2 – Compression de vocabulaire utilisant la hiérarchie d'hyperonymie, appliquée au premier et quatrième sens du mot « *mouse* ». Les lignes en pointillés indiquent que des nœuds ont été omis par clarté.

Pour la DL, tout comme le regroupement des synonymes en *synsets* aide à mieux généraliser, nous faisons l'hypothèse que le regroupement des sens faisant partie d'une même hiérarchie d'hyperonymie va aussi aider à mieux généraliser, et que les concepts les plus spécialisés de WordNet sont souvent superflus. En effet, si l'on considère un sous-ensemble de WordNet qui ne comprend que le mot « souris », avec son premier sens (le petit rongeur), son quatrième sens (le dispositif électronique), et tous leurs hyperonymes, tel qu'illustré dans la figure 6.2, on voit que les concepts « artefact » et « être vivant » suffisent à différencier les deux sens, et toutes les étiquettes plus spécialisées pourraient être ramenées à ces

deux concepts. Ainsi, non seulement le vocabulaire de sens, c'est-à-dire le nombre d'étiquettes de sens dans notre inventaire, sera réduit, mais en plus tous les autres « êtres vivants » donneront des exemples qui pourront ensuite permettre de différencier les deux sens de souris.

En considérant maintenant tout le vocabulaire de WordNet, l'objectif de notre méthode est ainsi de faire correspondre chaque sens à son ancêtre le plus haut dans sa hiérarchie d'hyperonymie, avec les contraintes suivantes : Premièrement, cet ancêtre doit permettre de discriminer tous les différents sens du mot cible. Deuxièmement, nous devons conserver les hyperonymes qui sont indispensables pour discriminer les sens des autres mots du dictionnaire. Par exemple, en prenant tout WordNet en considération, nous ne pouvons pas faire correspondre « souris#1 » à « être vivant#1 », parce qu'une étiquette plus spécifique, « animal#1 » est nécessaire pour distinguer les deux sens du mot « proie » (un sens décrit une personne et l'autre un animal).

Notre méthode fonctionne donc en deux étapes :

1. Nous marquons comme « nécessaires » les enfants du premier ancêtre commun de chaque paire de sens de chaque mot de WordNet.
2. Nous faisons correspondre chaque sens à son ancêtre le plus bas dans sa hiérarchie d'hyperonymie ayant été précédemment marqué comme « nécessaire ».

En conséquence, les sens les plus spécifiques de l'arbre qui ne sont pas indispensables pour distinguer un mot de l'inventaire lexical seront automatiquement supprimés du vocabulaire. En d'autres termes, l'ensemble de sens qui reste dans le vocabulaire est le plus petit sous-ensemble de tous les *synsets* qui sont nécessaires pour distinguer chaque sens de chaque mot de WordNet, en considérant seulement les liens d'hyperonymie, d'hyponymie et d'instance.

6.3.3 Compression de vocabulaire de sens à travers l'ensemble des relations sémantiques de WordNet

En plus de l'hyperonymie, de l'hyponymie et de la relation d'instance, WordNet contient plusieurs autres relations entre *synsets*, telles que la méronymie (X fait partie de Y, ou X est un membre de Y) et son opposé l'holonymie, l'antonymie (X est le contraire de Y) et son opposé la similarité, etc.

Nous proposons une deuxième méthode de compression du vocabulaire de sens, qui prend en compte toutes les relations sémantiques offertes par WordNet, afin de former des groupes de *synsets* proches.

Par exemple, en utilisant toutes les relations sémantiques disponibles, nous pourrions former un groupe contenant « physicien », « physique » (domaine), « Einstein » (instance), « astronome » (hyponyme), mais aussi d'autres sens connexes tels que « photon », car c'est un méronyme de « rayonnement », qui est un hyponyme d'« énergie », qui appartient au même domaine de « physique », etc.

Notre méthode fonctionne en construisant ces groupes de manière itérative. Soit S l'ensemble des *synsets* de WordNet et C l'ensemble des groupes de *synsets* que l'on cherche à construire, on initialise d'abord C comme des singletons contenant chacun un *synset* différent.

$$S = \{s_0, s_1, ..., s_n\} \qquad C = \{c_0, c_1, ..., c_n\} \qquad C = \{\{s_0\}, \{s_1\}, ..., \{s_n\}\}$$

Ensuite, à chaque étape, on trie C par taille de groupes, et on sélectionne le plus petit groupe c_x ainsi que le plus petit groupe relié à c_x, c_y. On considère qu'un groupe c_a est relié à un groupe c_b si un *synset* $s_a \in c_a$ est relié à un *synset* $s_b \in c_b$ par n'importe quel lien sémantique. On fusionne c_x et c_y ensemble, si et seulement si l'opération permet toujours de discriminer les différents sens de tous les mots de la base lexicale. Si c'est le cas, on valide la fusion et on passe à l'étape suivante. Si ce n'est pas le cas, on annule la fusion et on essaye avec un autre groupe relié à c_x. S'il est impossible de fusionner un groupe avec c_x, alors on essaye avec le plus petit groupe suivant, et si aucune fusion n'est possible pour aucun des groupes, l'algorithme s'arrête.

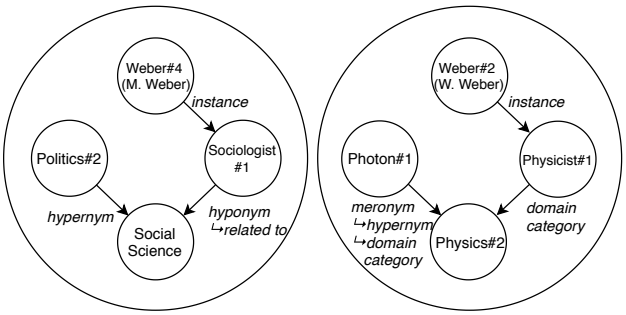


FIGURE 6.3 – Exemple de groupes de sens pouvant résulter de notre méthode, si on ne considère que deux sens du nom « Weber » et seulement certaines relations.

Dans la figure 6.3, nous montrons un ensemble possible de groupes qui pourraient résulter de notre méthode.

Cette méthode produit des groupes significativement plus grands que celle s'appuyant sur les hyperonymes. En effet, en moyenne, un groupe contient 5 *synsets* avec cette dernière, alors qu'il en contient 17 avec cette méthode. De plus, cette méthode, contrairement à la précédente, est également stochastique, parce qu'à chaque fois qu'on ordonne les groupes par taille, l'algorithme de tri place les groupes de même taille dans un ordre aléatoire. Cependant, comme nous réordonnons les groupes après chaque fusion, les groupes sont de taille assez équilibrée,

et nous avons observé que la taille finale du vocabulaire (c'est-à-dire le nombre de groupes) se situe toujours entre 11 000 et 13 000.

Dans la suite, on considère un ensemble C généré après que l'algorithme se soit arrêté après 105 775 étapes de fusion (générant ainsi 11 885 groupes de sens).

Méthode de compression	Taille du vocabulaire	Taux de compression	Couverture du SemCor
Référence	206 941	référence	16%
Synonymes	117 659	43%	22%
Hyperonymes	39 147	81%	32%
Toutes relations	11 885	94%	39%

TABLE 6.1 – Résultats de nos deux méthodes de compression de vocabulaire sur la taille du vocabulaire et la couverture du SemCor. La couverture correspond au ratio du nombre d'étiquettes de sens différentes observables dans le corpus sur le nombre total d'étiquettes (taille du vocabulaire).

La tableau 6.1 montre l'effet de la compression de vocabulaire via les synonymes (sens vers *synsets*), de notre première méthode utilisant les hyperonymes, ainsi que de notre deuxième méthode utilisant toutes les relations de WordNet, sur la taille du vocabulaire de sens de WordNet, et sur la couverture du SemCor. Comme nous pouvons le constater, la taille du vocabulaire diminue considérablement grâce à nos méthodes, et la couverture d'un même corpus est nettement améliorée.

6.4 Protocole expérimental

Afin d'évaluer nos méthodes de compression de vocabulaire de sens, nous les avons appliquées à un système neuronal de DL état de l'art similaire à celui que nous avons présenté au chapitre 5 (voir aussi Vial et al. (2019b)). Notre réseau de neurones prend en entrée directement les mots sous forme vectorielle, à partir d'un modèle de vecteurs de mot pré-entraîné, il repose ensuite sur une ou plusieurs couches cachées, puis sur une couche de sortie, qui associe à chaque mot une distribution de probabilité sur tous les sens du vocabulaire utilisé, à l'aide de la fonction *softmax*.

6.4.1 Détails de l'architecture

Comme pour notre système présenté au chapitre 5, nous avons utilisé en entrée de notre réseau les vecteurs contextualisés BERT (Devlin et al., 2019). Nous avons utilisé le modèle pour l'anglais « bert-large-cased » qui est pré-entraîné sur BookCorpus (Zhu et al., 2015) et Wikipedia, et qui produit des vecteurs de dimension 1 024.

Pour les couches cachées, nous avons appliqué six couches d'encodeurs *Transformer* (Vaswani et al., 2017), avec les même paramètres que le modèle « base » de l'article original (huit têtes d'attention, dimension cachée de 2 048, et régularisation *dropout* à 0,1).

Étant donné que les vecteurs retournés par BERT encodent directement les positions des mots, il n'est pas nécessaire d'avoir une récurrence au niveau des couches cachées. Ainsi, nous n'ajoutons pas de vecteurs de position supplémentaires en entrée de notre encodeur.

Pour tous les autres paramètres du modèle, comme le nombre de phrases par mini-lot, et la méthode d'optimisation, nous avons utilisé les mêmes paramètres que Vial et al. (2019b).

6.4.2 Entraînement du modèle

Nous avons comparé nos méthodes sur deux ensembles de corpus d'entraînement : le SemCor (Miller et al., 1993), le plus grand corpus annoté en sens utilisé pour l'apprentissage de la plupart des systèmes supervisés de DL, et la concaténation du SemCor et du WordNet Gloss Corpus [3], comme pour nos expériences du chapitre 5. Nous avons utilisé les versions de ces corpus fournies avec la ressource UFSAC 2.1 [4] (Vial et al., 2018b).

Nous avons ainsi cherché à utiliser seulement des données de la meilleure qualité possible, pour éviter d'ajouter du bruit et/ou de rallonger le temps d'entraînement de nos modèles.

Nous avons entraîné chaque modèle sur 20 passes de nos données d'entraînement. Au début de chaque passe, nous avons mélangé toutes les phrases aléatoirement, et à la fin de chaque passe, nous avons évalué notre modèle sur un jeu de développement, et nous avons conservé celui qui a obtenu le meilleur score F1 de

3. http://wordnetcode.princeton.edu/glosstag-files/glosstag.shtml
4. https://github.com/getalp/UFSAC

DL. Le corpus de développement est constitué de 4 000 phrases prises aléatoirement du WNGC pour le système entraîné sur le SemCor seul, et de 4 000 phrases extraites aléatoirement de nos données d'entraînement pour les autres.

Nous avons ainsi entraîné trois systèmes :

1. Un système « référence » dont le vocabulaire de sens est celui de tous les *synsets* vus pendant l'entraînement (utilisant ainsi la compression classique via les synonymes) ;
2. Un système « hyperonymes » entraîné dans les mêmes conditions, mais avec notre première méthode de compression du vocabulaire via les hyperonymes, les hyponymes et les instances appliquée au corpus d'entraînement ;
3. Un système « toutes relations » qui applique cette fois-ci au corpus d'entraînement notre deuxième méthode de compression de vocabulaire via toutes les relations sémantiques de WordNet.

	Nombre de paramètres	
Système	SemCor	SemCor+WNGC
Référence	77,15M	120,85M
Hyperonymes	63,44M	79,85M
Toutes relations	55,16M	60,27M

TABLE 6.2 – Nombre de paramètres d'un modèle en fonction du corpus d'apprentissage et de notre méthode de compression de vocabulaire.

Tous les entraînements ont été effectués sur un seul GPU Titan X de Nvidia. Dans la tableau 6.2, nous montrons le nombre de paramètres des différents modèles, en fonction du corpus d'entraînement et de notre méthode de compression du vocabulaire. Comme nous pouvons le voir, ce nombre est réduit par un facteur de 1,2 à 2 grâce à nos méthodes de compression.

6.4.3 Résultats

Nous avons évalué nos modèles sur tous les corpus d'évaluation de la DL de l'anglais des campagnes d'évaluation SensEval/SemEval, c'est-à-dire les corpus d'évaluation « grain fin » de SensEval 2 (Edmonds et Cotton, 2001), SensEval 3 (Snyder et Palmer, 2004), SemEval 2007 (tâche 17) (Pradhan et al., 2007), SemEval 2013 (Navigli et al., 2013) et SemEval 2015 (Moro et Navigli, 2015), ainsi que le corpus « ALL » constitué de leur concaténation. Nous avons également comparé nos résultats sur la tâche « gros grain » de SemEval 2007 (tâche 7) (Navigli et al., 2007).

Pour chaque évaluation, nous avons entraîné huit modèles indépendants, et nous donnons le score obtenu par un système « ensemble » qui moyenne leurs prédictions à l'aide d'une moyenne géométrique.

Système	SE2	SE3	SE07 17	SE13	SE15	ALL	SE07 07
SemCor, référence	91,15	96,76	97,58	91,06	94,78	93,23	92,84
SemCor, hyperonymes	98,03	99,19	99,78	99,15	98,39	98,75	98,85
SemCor, toutes relations	99,56	99,84	100	100	98,92	99,67	99,69
SemCor+WNGC, référence	97,81	98,92	99,34	97,63	99,34	98,26	98,45
SemCor+WNGC, hyperonymes	99,74	99,95	100	99,76	99,91	99,83	99,91
SemCor+WNGC, toutes relations	100	100	100	100	99,91	99,99	100
Nombre de mots à annoter	2282	1850	455	1644	1022	7253	2261

TABLE 6.3 – Couverture (en %) des corpus d'évaluation en fonction du corpus d'apprentissage et de l'utilisation de notre méthode de compression de vocabulaire.

Les scores obtenus par nos systèmes en comparaison avec les meilleurs systèmes de l'état de l'art et l'étalon du premier sens sont présents dans le tableau 6.4, et le tableau 6.3 montre la couverture de nos systèmes sur les tâches d'évaluation.

Nos méthodes de compression améliorent cependant grandement la couverture de nos systèmes. En effet, comme nous pouvons le voir dans le tableau 6.3, sur un total de 7 253 mots à annoter pour le corpus « ALL », le système de référence entraîné sur le SemCor n'est pas capable d'annoter 491 d'entre eux, alors qu'avec la compression du vocabulaire à travers les hyperonymes, ce nombre descend à 91, et 24 avec la compression à travers toutes les relations.

Lors de l'ajout du WordNet Gloss Corpus aux données d'entraînement, seulement 12 mots ne peuvent pas être annotés avec le système « hyperonymes », et avec le système « toutes relations », plus qu'un seul mot (l'adjectif monosémique « cytotoxic ») ne peut pas être annoté parce que son sens n'a pas été vu pendant l'entraînement. Si nous prenons en compte uniquement les mots polysémiques, le système qui applique la compression à travers toutes les relations et qui entraîné sur le SemCor n'est pas capable d'annoter seulement un seul mot (l'adverbe « eloquently »). Avec le WNGC en plus, il a une couverture de 100%.

Concernant les résultats présentés dans le tableau 6.4, nous observons que nos systèmes qui utilisent nos méthodes de compression de vocabulaire, que ce soit à

travers les relations d'hyperonymie ou à travers toutes les relations obtiennent des scores qui sont globalement équivalents ou légèrement supérieurs aux systèmes « référence » qui n'utilisent pas nos méthodes.

Par rapport aux autres travaux, nous obtenons, comme avec notre système de référence (identique à celui présenté au chapitre 5), des résultats surpassant significativement l'état de l'art dans toutes les tâches. Nous rappelons que Loureiro et Jorge (2019) exploitent uniquement le SemCor comme données d'apprentissage, avec BERT, et que Huang et al. (2019) exploitent en plus les définitions de WordNet.

Système	SE2	SE3	SE07 17	SE13	SE15	ALL (concat. tâches préc.)					SE07 07
						noms	verbes	adj.	adv.	total	
Sens le plus fréquent	65,6	66,0	54,5	63,8	67,1	67,7	49,8	73,1	80,5	65,5	78,9
Loureiro et Jorge (2019)	76,3	75,6	68,1	75,1	77,0	-	-	-	-	75,4	-
Huang et al. (2019)	77,7	75,2	⋆72,5	76,1	80,4	79,3	66,9	78,2	86,4	77,0	-
SemCor, référence	77,2	76,5	70,1	74,7	77,4	78,7	65,2	79,1	85,5	76,0	87,7
SemCor, hyperonymes	77,5	77,4	69,5	76,0	78,3	79,6	65,9	79,5	85,5	76,7	87,6
SemCor, toutes relations	76,6	76,9	69,0	73,8	75,4	77,2	66,0	80,1	85,0	75,4	86,7
SemCor+WNGC, référence	**79,7**	76,1	**74,1**	78,6	80,4	80,6	68,1	82,4	86,1	78,3	90,4
SemCor+WNGC, hyperonymes	**79,7**	77,8	73,4	**78,7**	**82,6**	81,4	68,7	83,7	85,5	**79,0**	90,4
SemCor+WNGC, toutes relations	79,4	**78,1**	71,4	77,8	81,4	80,7	68,6	82,8	85,5	78,5	**90,6**

TABLE 6.4 – Scores F1 (%) sur les tâches de DL de l'anglais des campagnes d'évaluation SensEval/SemEval. La tâche « ALL » est la concaténation de SE2, SE3, SE07 17, SE13 et SE15. La stratégie de repli est appliquée sur les mots dont aucun sens n'a été observé pendant l'entraînement. Les scores en **gras** sont à notre connaissance les meilleurs résultats obtenus sur la tâche. Les scores prefixés par une étoile (⋆) concernent un corpus de développement.

6.4.4 Étude des hyperparamètres

Afin de mieux comprendre l'impact de nos principaux choix d'hyperparamètres sur les résultats, et de voir si nos méthodes de compression de vocabulaire sont efficaces dans d'autres configurations, nous proposons ici de faire varier, en plus du corpus d'entraînement et de la méthode de compression du vocabulaire, deux autres paramètres : le modèle de vecteurs de mot pré-entraînés, et la méthode d'ensemble.

Corpus d'entraînement	vecteurs de mot pré-entraînés	Ensemble	Scores F1 sur la tâche "ALL" (%)					
			Référence		Hyperon.		Toutes rel.	
			$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
SemCor+WNGC	BERT	Oui	78,27	-	79,00	-	78,48	-
SemCor+WNGC	BERT	Non	76,97	±0,38	77,08	±0,17	76,52	±0,36
SemCor+WNGC	ELMo	Oui	75,16	-	74,65	-	70,58	-
SemCor+WNGC	ELMo	Non	74,56	±0,27	74.36	±0,27	68,77	±0.30
SemCor+WNGC	GloVe	Oui	72,23	-	72,74	-	71,42	-
SemCor+WNGC	GloVe	Non	71,93	±0,35	71,79	±0,29	69,60	±0,32
SemCor	BERT	Oui	76,02	-	76,73	-	75,40	-
SemCor	BERT	Non	75,06	±0,26	75,59	±0,16	73,91	±0,33
SemCor	ELMo	Oui	72,55	-	73,09	-	69,43	-
SemCor	ELMo	Non	72,21	±0,13	72,83	±0,24	68,74	±0,29
SemCor	GloVe	Oui	70,77	-	71,18	-	68,44	-
SemCor	GloVe	Non	70,51	±0,16	70,77	±0,21	67,48	±0,55
HCAN (Luo et al., 2018b)								
SemCor+WN déf.	GloVe	-	71,1					
LMMS (Loureiro et Jorge, 2019)								
SemCor	BERT	-	75,4					
SemCor	ELMo	-	66,2					
GlossBERT (Huang et al., 2019)								
SemCor+WN déf.	BERT	-	77,0					

TABLE 6.5 – Étude des hyperparamètres sur la tâche « ALL ». Pour les systèmes qui n'utilisent pas l'ensemble, nous montrons la moyenne des scores ($\bar{x}$) de huit modèles entraînés séparément, avec l'écart type (σ).

Pour le modèle de vecteurs de mot, nous avons expérimenté avec BERT (Devlin et al., 2019) comme pour nos résultats principaux, mais aussi avec ELMo (Peters et al., 2018) et GloVe (Pennington et al., 2014). Pour ELMo, nous avons utilisé le modèle entraîné sur Wikipedia et les données monolingues de WMT 2008-2012. [5] Pour GloVe, nous avons utilisé le même modèle que Luo et al. (2018b) et Vial et al. (2019b) entraîné sur Wikipedia 2014 et Gigaword 5. [6]

5. https://allennlp.org/elmo
6. https://nlp.stanford.edu/projects/glove/

Comme les représentations vectorielles de GloVe n'encodent pas la position des mots, c'est-à-dire qu'un mot a la même représentation quelle que soit sa position ou son contexte, et que nous avions montré au chapitre précédent (voir section 5.5.1) que Transformer n'était pas bien adapté à ces vecteurs, nous avons réutilisé dans ce cas une couche de cellules *LSTM* bidirectionnelles de taille 1 000 par direction pour les couches cachées (comme Vial et al. (2019b)).

Pour la méthode d'ensemble, nous avons expérimenté soit en l'utilisant, comme dans nos résultats principaux, c'est-à-dire en moyennant les prédictions de huit modèles entraînés séparément, soit en donnant la moyenne et l'écart-type des scores des huit modèles évalués individuellement.

Comme nous pouvons le voir dans le tableau 6.5, le corpus d'entraînement supplémentaire (WNGC) et encore plus l'utilisation de BERT en tant que vecteurs de mot ont tous les deux un impact majeur sur nos résultats et conduisent à des scores supérieurs à l'état de l'art. L'utilisation de BERT au lieu de ELMo ou GloVe améliore respectivement le score d'environ 3 et 5 points dans chaque expérience, et l'ajout du WNGC aux données d'entraînement l'améliore encore d'environ 2 points. Enfin, l'utilisation d'ensembles ajoute environ 1 point au score F1 final.

Enfin, à travers les scores obtenus par les modèles individuels (sans ensemble), nous pouvons observer sur les écarts-types que la méthode de compression du vocabulaire par les hyperonymes n'a jamais d'impact significatif sur le score final. Cependant, la méthode de compression via toutes les relations semble avoir un impact négatif sur les résultats dans certains cas (en utilisant GloVe et ELMo particulièrement, et en utilisant le SemCor seul comme corpus d'entraînement).

6.5 Conclusion

Dans ce chapitre, nous avons présenté deux nouvelles méthodes qui améliorent la couverture et la capacité de généralisation des systèmes de DL supervisés, en réduisant le nombre d'étiquettes de sens différentes dans WordNet afin de ne conserver que celles qui sont essentielles pour différencier les sens de tous les mots présents dans la base lexicale. À l'échelle de l'ensemble de la base de données lexicale, nous avons montré que ces méthodes permettaient de réduire le nombre total d'étiquettes de sens différentes dans WordNet à seulement 6% de sa taille originale, et que la couverture d'un même corpus d'entraînement est ensuite plus que doublée.

Nous avons entraîné un système de DL neuronal état de l'art et nous avons montré que nos méthodes permettaient de réduire la taille des modèles par un facteur de 1,2 à 2 et d'augmenter largement leur couverture, sans dégrader leurs perfor-

mances. Au final, nous obtenons une couverture de 99,99% sur l'ensemble des tâches d'évaluation (soit un seul mot manquant sur les 7 253) lorsque l'on entraîne notre système sur le SemCor uniquement, et 100% lorsque l'on ajoute le WordNet Gloss Corpus aux données d'entraînement. On élimine ainsi quasiment le besoin d'une méthode de repli pour désambiguïser n'importe quel mot du vocabulaire de WordNet.

Notre méthode combinée avec notre architecture neuronale présentée au chapitre 5 permet à notre système à la fois de surpasser nettement l'état de l'art dans toutes les tâches d'évaluation de la DL de l'anglais, mais en plus avec un modèle très réduit et avec une bien meilleure couverture.

Pour finir, bien que nous ayons appliqué nos méthodes uniquement sur l'anglais dans ce chapitre, elles peuvent facilement s'appliquer à d'autres langues en utilisant toujours WordNet comme inventaire de sens. Par exemple, elles peuvent s'appliquer à la désambiguïsation d'une langue moins bien dotée en utilisant la méthode de Hadj Salah et al. (2018). Cependant, du fait que les autres langues ont très peu de ressources annotées manuellement en sens (que ce soit avec l'inventaire de sens WordNet ou un autre), l'évaluation *in vivo* des systèmes de DL pour d'autres langues que l'anglais est limitée.

Chapitre 7

Amélioration de la traduction automatique grâce à la désambiguïsation lexicale

7.1 Introduction

Comme nous l'avons vu au chapitre 2, aujourd'hui les systèmes état de l'art en traduction automatique sont, à l'instar de la désambiguïsation lexicale, tous fondés sur des réseaux de neurones. Plus précisément, depuis les travaux de Vaswani et al. (2017), les meilleurs systèmes s'appuient tous sur l'architecture Transformer ou sur l'une de ses améliorations comme le Transformer-XL (Dai et al., 2019). De même, au niveau des vocabulaires d'entrée et de sortie utilisés, la plupart des systèmes depuis Sennrich et al. (2016b) s'appuient sur des méthodes de découpage des mots en sous-unités qu'on a décrites dans la section 2.3.2.

Dans la section 2.6.1, nous avons vu que plusieurs travaux ont visé l'amélioration de systèmes de TA neuronaux grâce à des informations supplémentaires, ou traits, donnés en entrée. Parmi ces travaux, on trouve notamment Sennrich et Haddow (2016), qui présentent une méthode générale qui consiste à concaténer des vecteurs de traits aux vecteurs de mot utilisés en entrée du modèle neuronal, puis Hadj Salah (2018), qui a appliqué cette méthode avec des traits de sens WordNet (Miller et al., 1990) issus d'un système de désambiguïsation lexicale.

D'autres travaux, comme Pu et al. (2018) intègrent des vecteurs de sens appris de manière non supervisée, à partir des définitions de WordNet. Enfin les travaux de Liu et al. (2018) intègrent indirectement des sens à partir de vecteurs contex-

tualisés appris sur des corpus monolingues, à la manière de ELMo (Peters et al., 2018) ou BERT (Devlin et al., 2019).

Dans tous ces travaux précédents, que les sens soient directement reliés à l'inventaire de WordNet, ou qu'ils soient issus d'un modèle de vecteurs contextualisés, leur ajout à un système de TA neuronal améliore significativement ses résultats. Cependant, on remarque que, d'une part, ces différentes méthodes ne sont jamais comparées entre elles, et d'autre part, elles ne s'appuient pas sur les derniers travaux et les derniers modèles de l'état de l'art, tant au niveau des systèmes de TA que des systèmes de DL.

Plus particulièrement, aucun de ces travaux ne s'appuie sur l'architecture Transformer (Vaswani et al., 2017), qui est aujourd'hui omniprésente dans la plupart des modèles neuronaux traitant du texte. Ils utilisent de plus des vocabulaires fixes limités aux mots les plus fréquents du vocabulaire, à la place des méthodes à base de sous-unités de mots qui sont aujourd'hui aussi dominantes. Enfin, les systèmes de DL utilisés dans les expériences ont des scores qui sont bien en dessous des systèmes état de l'art actuels (notamment grâce aux modèles qui s'appuient sur BERT, voir section 1.5.1.3).

Dans ce chapitre, nous allons ainsi proposer plusieurs méthodes pour intégrer de la désambiguïsation lexicale à un système de TA état de l'art, en nous appuyant sur nos derniers systèmes de DL (voir chapitre 5 et chapitre 6), ainsi que le modèle de langue pré-entraîné BERT (Devlin et al., 2019), en considérant que les vecteurs contextualisés de BERT représentent déjà indirectement des vecteurs de sens.

Nous mènerons d'une part des expériences qui consistent à ajouter des sens WordNet en entrée d'un système de TA état de l'art, comme un trait supplémentaire aux mots pouvant apporter de l'information, mais nous évaluerons aussi les vecteurs de BERT utilisés directement en entrée du système, et nous comparerons leur apport respectif. Ensuite, nous expérimenterons une forme de traduction multi-tâches, qui consiste à traduire du texte en mots et en sens à la fois, afin que l'apprentissage se retrouve guidé par les deux objectifs. Finalement, nous évaluerons la combinaison de nos deux premières méthodes : celle qui exploite les vecteurs de BERT, et celle qui exploite les sens de WordNet en entrée du système.

Dans un premier temps, nous allons décrire les différentes méthodes que nous proposons, ainsi que le système de TA sur lequel nous allons les appliquer, et le système de DL qui sera utilisé. Nous discuterons ensuite des résultats des expériences avant de conclure.

À noter que les travaux présentés dans ce chapitre ont fait l'objet de notre participation à la campagne d'évaluation IWSLT 2019 (Niehues et al., 2019) sur la tâche de traduction automatique de l'anglais vers le tchèque (Vial et al., 2019d).

7.2 Méthode

Comme nous l'avons dit précédemment, nous proposons trois méthodes différentes afin d'intégrer des sens dans un système de TA neuronal :

1. En ajoutant les sens aux mots sources, en entrée du système de TA.
2. En prédisant les sens en plus des mots cibles, en sortie du système de TA.
3. En remplaçant les mots sources par des vecteurs de mot contextualisés.

Nous allons détailler chacune de ces trois méthodes dans cette section.

7.2.1 Les sens comme information discrète supplémentaire

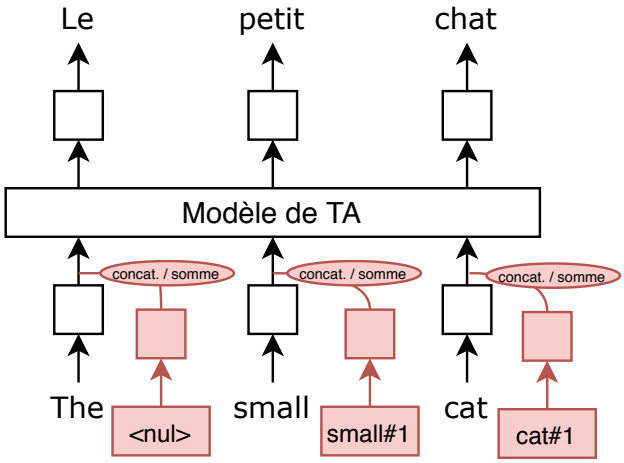


FIGURE 7.1 – Ajout des sens en tant qu'information discrète supplémentaire pour un système de TA neuronal.

La première méthode que nous proposons consiste à ajouter l'information du sens des mots, en plus des mots eux-mêmes, en entrée du système de TA. Ces sens sont issus d'un système de DL externe qui s'appuie sur l'inventaire de sens de WordNet, et on part du principe que ce système annote tous les mots présents dans WordNet avec un seul sens le plus probable, et tous les autres mots avec un symbole spécial `<nul>`.

Ensuite, comme pour les mots, les sens doivent être convertis en vecteurs pour être utilisés en entrée du réseau neuronal. On apprendra ainsi conjointement les vecteurs de sens avec le reste des paramètres du modèle pendant l'entraînement.

Enfin, deux méthodes sont ensuite possibles pour intégrer ces sens :

1. Soit, comme Sennrich et Haddow (2016) et Hadj Salah (2018), on concatène les deux informations (vecteurs de sens et vecteurs de mot), puis on donne le vecteur résultant en entrée du système de TA. Il faudra ensuite redimensionner le vecteur résultant si on veut garder une taille de vecteur uniforme tout au long du modèle.
2. Soit, comme Vaswani et al. (2017) et Lample et Conneau (2019), qui ajoutent des vecteurs de position et des vecteurs marqueurs de langue aux vecteurs de mot, on additionne ces deux vecteurs. Il faudra cette fois faire attention à ce que les vecteurs soient de même taille.

La figure 7.1 illustre cette première méthode. Comme on peut le voir, avec cette méthode, les sens viennent s'ajouter aux informations contenues dans les mots, et donc on peut penser que le réseau sera capable d'identifier et de prendre en compte les marqueurs de sens utiles dans certaines situations ambigües tout en étant capable d'ignorer les marqueurs inutiles (marqueur `nul`, mots monosémiques, etc.).

Il est à noter qu'avec cette méthode, il faut forcément associer un vecteur pour le sens `<nul>`, nous avons choisi d'apprendre ce vecteur conjointement avec le reste du modèle comme pour les autres sens, à la manière d'un symbole `<unk>` utilisé pour représenter tous les mots inconnus dans un système de TA classique à vocabulaire fixe (voir section 2.3).

7.2.2 Apprentissage guidé par les sens

Dans la deuxième méthode que nous proposons, les sens ne sont plus donnés en entrée du système de TA, mais ils sont plutôt prédits en sortie, en même temps que les mots cibles. L'idée derrière cette méthode est cette fois-ci de guider l'apprentissage du réseau neuronal afin de le faciliter. En effet, en donnant plusieurs critères de prédiction à notre réseau, on réalise un apprentissage multi-tâches. Lorsque ces différentes tâches sont proches, il a été montré dans plusieurs travaux que cela pouvait être bénéfique pour la tâche principale (par exemple dans Raganato et al. (2017b)).

Un des avantages de cette méthode, illustrée dans la figure 7.2, est que, par rapport à la précédente, nous n'avons pas besoin d'apprendre un vecteur pour le symbole `<nul>` associé aux mots qui n'ont pas été annotés en sens. En effet, nous pouvons simplement ignorer la rétro-propagation du gradient sur ces cas-là.

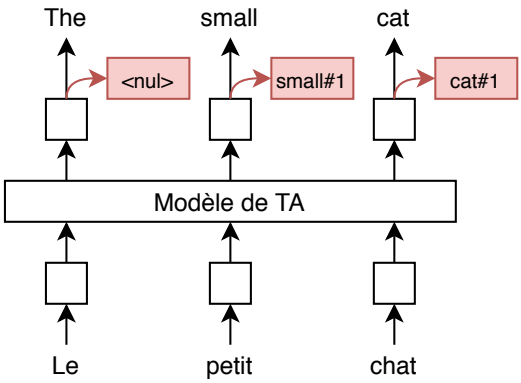


FIGURE 7.2 – Apprentissage d'un système de TA neuronal guidé par les sens.

7.2.3 Les modèles de langue comme information sémantique continue supplémentaire

Finalement, la dernière méthode que nous proposons est, dans l'idée, assez similaire à la première, c'est-à-dire que nous allons ajouter des sens en entrée du système de TA. Cependant, cette fois, nous faisons l'hypothèse que l'ambiguïté sémantique des mots est une information qui est déjà capturée par les modèles de langue pré-entraînés, type ELMo (Peters et al., 2018) ou BERT (Devlin et al., 2019), et donc nous proposons d'utiliser directement ces représentations-là pour représenter les sens.

En effet, comme nous l'avons vu dans la section 1.3.3.2, on peut voir ces nouveaux modèles de langue, ou vecteurs contextualisés, comme des vecteurs implicitement désambiguïsés, car, comme ont pu le montrer des travaux comme ceux de Wiedemann et al. (2019) par exemple, le vecteur pour un même mot dans BERT ou ELMo est très différent lorsque ce mot est utilisé dans un sens plutôt que dans un autre.

La figure 7.3 illustre ainsi cette troisième méthode. Cette fois, comme le modèle de langue offre à la fois une représentation vectorielle des mots et du contexte, nous n'avons pas besoin d'apprendre nos propres vecteurs de mot ni de sens. Nous utilisons directement ces représentations et nous ne modifions pas les paramètres du modèle de langue pendant l'apprentissage.

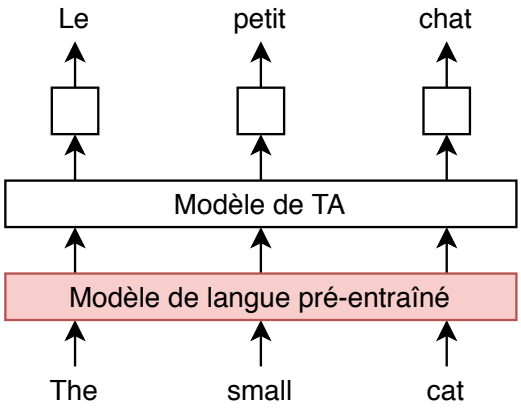


FIGURE 7.3 – Intégration d'un modèle de langue pré-entraîné à un système de TA neuronal.

7.3 Protocole expérimental

Afin d'évaluer nos méthodes, nous avons implémenté notre propre système de TA état de l'art nous permettant d'intégrer nos différentes extensions. Le système de TA, les données d'apprentissage et le système de DL utilisé pour apporter les informations sémantiques nécessaires sont décrits dans cette section. Tous nos systèmes sont aussi diffusés librement. [1]

7.3.1 Système de traduction automatique

Le système de TA que nous avons implémenté s'appuie sur une architecture encodeur-décodeur classique et plus particulièrement l'architecture Transformer (Vaswani et al., 2017) que nous avons décrite dans la section 2.2.4.

Notre modèle prend ainsi en entrée les mots et leur position dans la phrase sous forme vectorielle, puis il est constitué de plusieurs couches d'encodeur Transformer qui s'appuient sur le mécanisme d'auto-attention multi-tête. Ensuite, un décodeur constitué de plusieurs couches de décodeur Transformer va produire des mots en prenant en compte le contexte local (les mots produits jusqu'au mot courant) et le contexte source, toujours grâce au mécanisme d'attention multi-tête, et ce jusqu'à l'apparition du symbole de fin de séquence `<eos>`.

Nous apprenons les vecteurs de mot conjointement avec le modèle et nous par-

1. https://github.com/getalp/disambiguate-translate

tageons ces vecteurs, ainsi que le vocabulaire, entre l'entrée et la sortie, à l'exception du cas où l'on utilise un modèle de langue pré-entraîné. Dans ce cas, on apprend seulement des vecteurs pour la langue cible. Pour l'encodage des positions, nous utilisons la méthode de représentation vectorielle sinusoïdale telle que décrite dans la section 2.2.4.2, et nous additionnons les vecteurs de mot avec les vecteurs de position.

Pour le reste des paramètres, nous avons conservé ceux du modèle « base » de Transformer, c'est-à-dire six couches d'encodeur/décodeur, une dimension cachée de 2048 et huit têtes d'attention. La taille des vecteurs, quant à elle, est de 1024, car nous utilisons le modèle de langue *bert-large-cased* dont les vecteurs ont cette dimension. Tous ces paramètres correspondent aussi à ceux utilisés dans nos systèmes de DL (voir chapitre 5). Seule la probabilité de *dropout* a été augmentée, passant de 0,1 à 0,3, comme Elbayad et al. (2018).

Pour notre première méthode, lorsque nous concaténons les vecteurs de mot et ceux de sens, nous devons ensuite ajouter une couche linéaire pour retrouver une dimension de 1024. Afin de garder un nombre de paramètres comparable dans toutes nos expériences, nous avons ainsi conservé cette couche linéaire dans nos autres méthodes.

7.3.2 Données d'apprentissage

Afin de voir l'impact de nos méthodes comparativement à un système de référence classique, nous proposons d'utiliser la tâche de traduction anglais-allemand de IWSLT 2014 (Cettolo et al., 2014). Cette tâche sert en effet de référence dans plusieurs travaux récents comme par exemple Edunov et al. (2018b) et Elbayad et al. (2018). Dans ce dernier article, les auteurs ont de plus réimplémenté et comparé d'autres systèmes importants comme celui de Bahdanau et al. (2015) (RNN-search), de Gehring et al. (2017) (ConvS2S) et de Vaswani et al. (2017) (Transformer). Ces résultats servent aujourd'hui de référence dans les tableaux de comparaisons des systèmes de TA. [2]

Dans cette tâche, les données d'entraînement consistent en 171 721 phrases parallèles anglais-allemand, qui sont des transcriptions de présentations à des conférences TED traitant de divers sujets, et manuellement traduites. Le corpus de test, lui, est constitué de 6 750 phrases issues aussi de conférences TED.

Pour l'apprentissage, nous avons extrait aléatoirement 6 750 phrases du corpus

2. Notamment celui-ci : https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english

d'entraînement en tant que corpus de développement. Nous évaluons notre modèle périodiquement sur ce corpus (à la fin de chaque *epoch*), et nous gardons seulement celui qui obtient le meilleur score BLEU (Papineni et al., 2002).

Grâce à cette tâche standard qui restreint l'usage des données d'entraînement à un ensemble de petite taille [3], nous pensons que l'impact de nos méthodes ressortira d'autant plus sur une tâche non contrainte. De plus, nous avons l'avantage de pouvoir entraîner rapidement nos modèles tout en nous comparant à des systèmes état de l'art reconnus. Enfin, avec peu de données d'entraînement, on peut aussi voir l'intérêt de nos méthodes dans un contexte de TA avec peu de données disponibles, comme pour une paire de langues moins bien dotées.

Au niveau des directions des langues, nous allons d'abord expérimenter sur la traduction anglais vers allemand afin d'évaluer nos méthodes 1 et 3, car elles consistent à ajouter des informations de sens sur la langue source, et nos travaux sur la désambiguïsation lexicale portent principalement sur l'anglais. Pour la méthode 2 dans laquelle nous désambiguïsons la langue cible, nous inverserons les langues et effectuerons de la traduction allemand vers anglais.

7.3.3 Pré-traitement des données

Pour traiter nos données d'entraînement, nous avons suivi le même processus que Edunov et al. (2018b) et Elbayad et al. (2018), c'est-à-dire que nous avons segmenté les mots des ponctuations, et mis tous les caractères en minuscules grâce aux scripts de l'outil Moses (Koehn et al., 2007). Ensuite nous avons filtré les phrases de plus de 175 mots et dont le ratio de mots entre la source et la cible dépasse 1,5. Enfin nous avons appris un modèle de *Byte Pair Encoding* (BPE) (Sennrich et al., 2016b) constitué de 14 000 sous-unités de mots, vocabulaire joint entre la source et la cible et entraîné sur la concaténation des phrases anglaises et allemandes du corpus d'entraînement.

Cependant lorsque l'on utilise BERT en entrée, nous gardons la casse originale des mots, et nous apprenons un vocabulaire de 14 000 sous-unités de mots uniquement sur la langue cible (l'allemand).

3. En effet, l'ensemble des corpus parallèles anglais-allemand disponible est en réalité de plusieurs dizaines de millions de phrases (`http://opus.nlpl.eu/`).

7.3.4 Système de désambiguïsation lexicale

Pour l'annotation en sens nécessaire à nos expérimentations, nous avons exploité notre système de DL état de l'art dont l'architecture est présentée au chapitre 5, avec la méthode de compression de vocabulaire avec les hyperonymes telle qu'on l'a présentée au chapitre 6, qui annote l'anglais avec des sens issus de l'inventaire de sens WordNet 3.0.

Nous avons sélectionné le meilleur de nos systèmes, selon ses résultats sur l'ensemble des tâches d'évaluation de la DL, c'est-à-dire un système entraîné sur le couple de corpus SemCor+WNGC, qui exploite le modèle de langue pré-entraîné BERT en entrée, et qui réalise un repli sur le premier sens lorsqu'il doit annoter un mot qu'il n'a jamais observé pendant l'apprentissage. De cette manière, nous nous appuyons sur un système avec les meilleures performances possibles, avec une couverture maximale, et nous pourrons précisément comparer l'apport de la DL « explicite », en annotant avec des étiquettes WordNet, et la DL « implicite » apportée par le modèle BERT seul.

Dans nos expériences, nous nous limiterons à l'annotation en sens de la partie anglaise de nos corpus, car c'est, en DL, une des seules langues pour lesquelles suffisamment de ressources sont disponibles pour la construction de systèmes de qualité, et c'est donc la langue dont les systèmes sont les plus performants.

7.3.5 Processus d'apprentissage

Notre architecture neuronale pour la TA, présentée précédemment, étant très similaire à notre architecture neuronale pour la DL, présentée aux chapitres précédents, le processus d'apprentissage est lui aussi très semblable. Nous avons ainsi utilisé les paramètres suivants pour l'apprentissage :

1. La fonction de coût à optimiser est l'entropie croisée entre les mots produits par le décodeur et la référence. Comme pour Vaswani et al. (2017) ou Elbayad et al. (2018), nous utilisons l'entropie croisée « lissée sur les étiquettes » (*label smoothed*) (Szegedy et al., 2016) avec $\epsilon = 0,1$.
2. Nous utilisons Adam (Kingma et Ba, 2015) comme méthode d'optimisation, avec les mêmes paramètres par défaut tels que décrits dans leur article.

Nous avons entraîné nos modèles pendant 80 passes (*epoch*) sur nos données d'entraînement, avec une taille de lot (*batch*) de 100, et des phrases tronquées à 80 mots comme pour nos modèles de DL (voir section 5.4.3). À chaque fin d'une passe, nous avons mélangé aléatoirement les données, et nous avons évalué notre

modèle sur le corpus de développement, en termes de score BLEU. Au final, nous avons conservé le modèle qui a obtenu le meilleur score BLEU sur ces données.

7.4 Résultats

Nous avons évalué nos systèmes sur les données d'évaluation de la tâche de traduction anglais-allemand d'IWSLT 2014. Pour le décodage, nous réalisons une recherche en faisceau (*beam search*) avec un faisceau de taille 5, et une pénalité sur la longueur des phrases similaire à celle utilisée dans le système de Wu et al. (2016).

Nos résultats ainsi que ceux des systèmes de référence sont présents dans le tableau 7.1. Nous donnons les scores en termes de BLEU, TER et Meteor insensibles à la casse et avec les ponctuations segmentées.

	Système	BLEU (%) ↑	TER (%) ↓	Meteor (%) ↑
anglais → allemand	⋆Bahdanau et al. (2015)	25,0	-	-
	⋆Gehring et al. (2017)	26,7	-	-
	⋆Vaswani et al. (2017)	**28,1**	-	-
	Elbayad et al. (2018)	27,2	-	-
	Mot → mot (référence)	26,8	53,43	47,10
	Mot + sens (somme) → mot	27,1	53,04	47,40
	Mot + sens (concat.) → mot	27,4	52,40	47,58
	BERT → mot	**29,5**	**49,64**	**49,04**
allemand → anglais	⋆Bahdanau et al. (2015)	29,9	-	-
	⋆Gehring et al. (2017)	32,3	-	-
	⋆Vaswani et al. (2017)	**34,4**	-	-
	Elbayad et al. (2018)	33,9	-	-
	Mot → mot (référence)	**30,3**	**47,25**	**34,27**
	Mot → mot + sens	30,1	47,58	34,18

TABLE 7.1 – Résultats de nos méthodes d'intégration de la DL dans un système de TA sur la tâche de traduction allemand - anglais de IWSLT 2014. Les systèmes préfixés par une étoile (⋆) sont des réimplémentations issues de l'article d'Elbayad et al. (2018) des articles cités. Les meilleurs résultats par tâche sont en gras pour les systèmes de référence ainsi que pour nos systèmes. Les flèches (↑↓) indiquent le sens d'amélioration des scores. Les tirets (-) remplacent les scores non fournis par les auteurs.

Comme nous pouvons le voir, nos méthodes d'intégration des sens en entrée du système de TA améliorent systématiquement les résultats sur les trois mesures

utilisées, mais la méthode qui consiste à produire des étiquettes de sens en plus des mots dans la traduction semble, elle, dégrader les résultats.

Parmi les méthodes qui améliorent les résultats, celle qui se démarque particulièrement est celle qui consiste à utiliser le modèle de langue pré-entraîné BERT en entrée du système, car elle améliore le score BLEU de 2,7 points. L'utilisation des étiquettes de sens WordNet en sommant les vecteurs de mot et les vecteurs de sens, ainsi qu'en les concaténant, améliore respectivement les scores BLEU de 0,3 et 0,6 points.

Notre méthode qui consiste à produire les sens en sortie du système de TA dégrade légèrement les résultats (de 0,2 point de score BLEU), ce qui va dans le sens des travaux de Hadj Salah (2018). Cela semble montrer que le décodeur, lors du choix des mots à produire, n'est pas guidé par le sens de ces mots. Au contraire, le décodeur a plus de difficultés à produire les mots et les sens pour un même nombre de paramètres. On peut en déduire que ces deux informations sont suffisament disjointes pour qu'un même décodeur ait plus de difficultés à produire les deux informations en même temps, plutôt que le mot seul.

De plus, on remarque que, dans cette configuration, au contraire de celles qui consistent à donner les sens en entrée, le réseau ne peut pas ici apprendre à ignorer certains sens. Lorsqu'un mot est annoté dans le corpus d'entraînement, le réseau est forcé à produire une étiquette de sens. Comme ces étiquettes sont issues d'un système de DL qui a une précision d'environ 80%, cette configuration est donc potentiellement plus sensible au bruit.

Concernant les autres systèmes neuronaux de l'état de l'art, celui de Bahdanau et al. (2015) est un système de TA neuronal à base de cellules récurrentes (LSTM) avec un mécanisme d'attention classique, celui de Gehring et al. (2017) et d'Elbayad et al. (2018) remplacent la récurrence par des convolutions, et enfin le système de Vaswani et al. (2017) est fondé, comme le nôtre, sur l'architecture Transformer.

Sur les métriques utilisées, seuls les scores BLEU sont donnés dans ces autres systèmes, mais on peut voir sur les nôtres qu'une amélioration de score BLEU entraîne systématiquement une amélioration des scores TER et Meteor, donc nous utiliserons principalement cette mesure seule pour comparer les performances.

On remarque, dans un premier temps, que notre système référence n'atteint pas les performances du système Transformer de l'article de Elbayad et al. (2018), mais il obtient un score BLEU proche pour la traduction de l'anglais vers l'allemand (-1,3 points de BLEU). En revanche pour la traduction de l'allemand vers l'anglais, notre système de référence obtient un score BLEU 4,1 points inférieur.

Nous pensons que ces différences peuvent être dues notamment à l'outil utilisé et à certains choix d'implémentation de ces outils. Elbayad et al. (2018) utilisent en effet l'outil Fairseq [4], et ne précisent pas la méthode utilisée pour la création des *batchs*, ni leur taille. Nous utilisons de plus certains hyper-paramètres différents (taille de vecteurs, taille des couches cachées du Transformer...) pour garder une taille de vecteurs identique à celle de BERT, et des paramètres pour les encodeurs et décodeurs Transformer identiques à ceux utilisés dans notre système de DL présenté aux chapitres précédents.

Finalement, pour en revenir aux résultats de nos systèmes exploitant les sens en entrée, on peut voir que nous obtenons des résultats qui surpassent l'état de l'art grâce à BERT, et que l'utilisation de ce dernier, à la place d'étiquettes de sens discrètes, est largement plus efficace. Dans la section suivante, nous allons voir si la combinaison de BERT et des sens en entrée apporte des performances supplémentaires.

7.4.1 Interactions entre BERT et les sens WordNet

Comme on l'a vu dans nos résultats principaux, l'utilisation de BERT en entrée du réseau permet d'apporter des informations supplémentaires grandement utiles à notre système de TA, bien plus utiles que les étiquettes de sens WordNet. On pouvait cependant s'attendre à un tel résultat, car BERT encode de nombreuses informations lexicales et sémantiques qui vont au delà des simples sens WordNet.

D'un autre côté, concernant les méthodes d'intégration des sens WordNet en entrée du réseau, on a vu que la méthode de concaténation des vecteurs de sens aux vecteurs de mot fonctionne mieux que leur somme. Cependant, la méthode de concaténation ajoute un nouveau paramètre : la taille des vecteurs de sens.

Dans nos résultats principaux, nous avons utilisé une taille de vecteurs de sens de 128. Dans cette section, nous allons étudier l'impact de la variation de cette taille sur les résultats, ainsi que la combinaison de BERT et des sens WordNet, pour voir si ces derniers apportent une informations complémentaire, qui n'est pas directement encodée dans le modèle de langue.

Les résultats, en faisant varier la taille des vecteurs de sens, et en combinant les vecteurs de sens et BERT, se trouvent ainsi dans le tableau 7.2. Comme on peut le voir, l'utilisation combinée de BERT et des étiquettes de sens WordNet améliore nos résultats, allant de 0,2 à 0,4 point de score BLEU supplémentaire, ce qui est comparable à l'amélioration constatée sans BERT, qui va de 0,3 à 0,6 point.

4. https://github.com/pytorch/fairseq

Vecteurs de mot	Vecteurs de sens	BLEU (%) ↑	TER (%) ↓	Meteor (%) ↑
TdC	-	26,8	53,43	47,10
TdC	128	**27,4**	**52,40**	47,58
TdC	256	27,3	53,28	47,53
TdC	512	27,3	52,90	**47,59**
TdC	1024	27,1	53,37	47,35
BERT	-	29,5	49,64	49,04
BERT	128	29,8	49,60	49,38
BERT	256	29,7	49,46	49,33
BERT	512	**29,9**	**49,40**	**49,49**
BERT	1024	29,7	49,44	49,17

TABLE 7.2 – Résultats de nos méthodes combinées sur la tâche de traduction de l'anglais vers l'allemand, en fonction de la taille des vecteurs de mot et des vecteurs de sens. TdC est l'acronyme de « Table de Correspondance ». Les meilleurs résultats sont en gras, dans le cas avec BERT, et dans le cas sans BERT. Les flèches (↑↓) indiquent le sens d'amélioration des scores. Les tirets (-) indiquent l'absence de vecteurs de sens.

Cette expérience nous permet de montrer qu'à priori, les sens WordNet donnés par un système de DL externe permettent toujours, malgré le fort potentiel de BERT, de donner des informations supplémentaires utiles qui ne sont pas directement encodées dans le modèle de langue.

Concernant la taille des vecteurs de sens, il semble qu'une amélioration des performances soit notable quelle qu'elle soit, mais 128 et 512 sont les tailles qui donnent les meilleures performances dans les deux cas (avec et sans BERT).

Avec BERT combiné aux vecteurs de sens concaténés en entrée, nous améliorons les résultats de notre système de référence de 3,1 points de BLEU.

7.5 Conclusion

Dans ce chapitre, nous avons experimenté plusieurs méthodes d'intégration de sens au sein d'un système de traduction automatique neuronal, dans le but d'améliorer ce dernier.

Nous avons proposé pour cela trois méthodes : la première consiste à ajouter des sens WordNet issus de notre propre système de désambiguïsation lexicale état de l'art (voir chapitre 6), la deuxième consiste à forcer le modèle à produire des mots et les sens WordNet de ces mots afin de guider son apprentissage, et enfin la

troisième méthode consiste à remplacer les vecteurs de mot utilisés en entrée par le modèle de langue pré-entraîné BERT, dont nous faisons l'hypothèse qu'il encode indirectement les informations de sens dans ses représentations.

Afin d'évaluer nos méthodes, nous avons implémenté un système de TA fondé sur l'architecture état de l'art Transformer, et nous avons comparé l'impact des méthodes sur la tâche de traduction anglais-allemand de IWSLT 2014, déjà utilisée dans plusieurs travaux récents de l'état de l'art, sur trois mesures d'évaluation : BLEU, TER et Meteor.

On observe que, de nos trois méthodes, la première et la troisième améliorent les résultats de notre système de référence, en particulier pour la troisième. La deuxième méthode cependant les dégrade légèrement. Nous en déduisons que, d'une part, la prédiction des sens des mots en plus des mots eux-mêmes ne guide pas l'apprentissage mais au contraire le complexifie, d'autre part, ces mêmes sens WordNet peuvent être utiles s'ils sont utilisés comme des traits supplémentaires, ajoutés aux mots sources en entrée, et enfin, les modèles de langue pré-entraînés tels que BERT en remplacement des vecteurs de mot apportent des informations lexicales et sémantiques bien plus bénéfiques à la TA.

En allant plus loin, et afin de mesurer si l'information des sens WordNet est contenue dans BERT, ou bien si elle est complémentaire, nous avons évalué un système de TA avec BERT en entrée, et les sens WordNet comme trait additionnel. Les résultats ont montré une amélioration supplémentaire grâce à ces deux méthodes combinées, démontrant que BERT n'encode pas directement toutes les informations apportées par les sens d'un système de désambiguïsation externe.

Finalement, nous pouvons conclure de l'utilité certaine des informations sémantiques (désambiguïsation lexicale explicite et modèles de langue) pour la traduction automatique dans le contexte de nos expériences, c'est-à-dire avec un ensemble de données parallèles limitées. Bien qu'il serait intéressant de mener ces expériences sur des systèmes de TA entraînés sur une plus grande quantité de données, nos méthodes peuvent déjà être particulièrement utiles pour des langues peu dotées, pour lesquelles il existe peu de données parallèles, mais pour lesquelles des méthodes de création de modèles de langue et de systèmes de désambiguïsation lexicale sont toujours possibles (voir par exemple les travaux de Hadj Salah (2018)).

Chapitre 8

Apprentissage joint de traduction automatique et de désambiguïsation lexicale

8.1 Introduction

Comme nous avons pu le voir tout au long de cette thèse, la désambiguïsation lexicale et la traduction automatique sont deux tâches dont l'histoire est riche et complexe, et qui ont évolué largement indépendamment l'une de l'autre. Dans le chapitre 2, nous avons passé en revue plusieurs travaux qui mêlent ces deux tâches, que ce soit via l'étude des capacités des systèmes de TA à désambiguïser un texte en traduisant correctement un mot ambigü (voir section 2.6.2), ou en mesurant l'apport de la DL pour améliorer globalement les systèmes de TA (voir section 2.6.1).

Dans le chapitre 7, nous avons nous-mêmes mené des expériences afin d'étudier l'apport de la DL pour la TA, à travers l'ajout d'étiquettes de sens WordNet (Miller et al., 1990) aux mots, ou en remplaçant ces mots directement par des vecteurs contextualisés comme BERT (Devlin et al., 2019). Nous avons ainsi pu comparer plusieurs méthodes pour ajouter ces informations et les appliquer sur un système fondé sur l'architecture neuronale Transformer (Vaswani et al., 2017).

Il ressort de ces études que les sens apportés par une ressource externe peuvent clairement améliorer les performances d'un système de TA entraîné sur une quantité de données restreinte, et que les étiquettes de sens WordNet semblent apporter des informations complémentaires à un modèle de langue entraîné de façon non-

supervisée.

Dans ce chapitre, nous allons plus loin en proposant une nouvelle méthode d'apprentissage joint mêlant les modèles neuronaux utilisés pour la TA et ceux utilisés pour la DL. En effet, nous partons du constat que nos meilleurs systèmes de DL et de TA s'appuient à la fois sur la même architecture Transformer (Vaswani et al., 2017), et sur le même modèle de langue pré-entraîné BERT (Devlin et al., 2019). Nous pouvons finalement aujourd'hui constuire un modèle unique, qui repose sur une même architecture, qui puisse résoudre les deux tâches à la fois.

L'intérêt d'un tel modèle est multiple. En plus d'offrir une nouvelle méthode qui peut améliorer à la fois les performances des systèmes de TA et celles des systèmes de DL, cette nouvelle configuration nous permet d'évaluer la capacité d'un même modèle neuronal à résoudre deux tâches simultanément, et ainsi de mesurer, indirectement, la similitude entre ces deux tâches.

Pour cela, nous nous appuierons sur un ensemble de données d'entraînement à la fois bilingues et annotées en sens WordNet. À notre connaissance, il n'existe pas de tels corpus créés manuellement, c'est pourquoi nous nous servirons de nos meilleurs systèmes de DL, présentés aux chapitres précédents, afin de désambiguïser automatiquement les ressources parallèles, et, à l'opposé, nous nous servirons des meilleurs systèmes de TA de l'état de l'art afin de traduire automatiquement les corpus manuellement annotés en sens que nous utiliserons.

Nous allons décrire notre nouvelle architecture dans la section suivante, puis le protocole expérimental que nous avons suivi afin de l'évaluer sur les tâches de TA et de DL, enfin nous détaillerons les résultats et nous conclurons.

8.2 Architecture

Nous allons ici décrire notre nouvelle architecture pour l'apprentissage d'un modèle joint de TA et de DL, ainsi que les fonctions objectif associées. La principale difficulté dans la conception d'une telle architecture est d'identifier une base commune qui soit la plus proche possible des modèles état de l'art dans les deux tâches, afin d'obtenir les meilleurs résultats possibles.

En TA, c'est l'architecture Transformer (Vaswani et al., 2017) qui est aujourd'hui prédominante dans tous les systèmes état de l'art[1] et qui s'est exportée à d'autres tâches du TAL, comme pour les modèles de langue type BERT (Devlin

1. Voir par exemple les résultats de la dernière campagne d'évaluation WMT (Barrault et al., 2019)

et al., 2019). En DL, les architectures état de l'art s'appuient aussi sur Transformer, mais seulement depuis nos travaux présentés au chapitre 5.

En plus de Transformer, nos modèles de DL reposent fortement sur le modèle de langue BERT. De la même manière, grâce à nos travaux présentés au chapitre 7, nous disposons aussi d'un système de TA état de l'art qui exploite BERT.

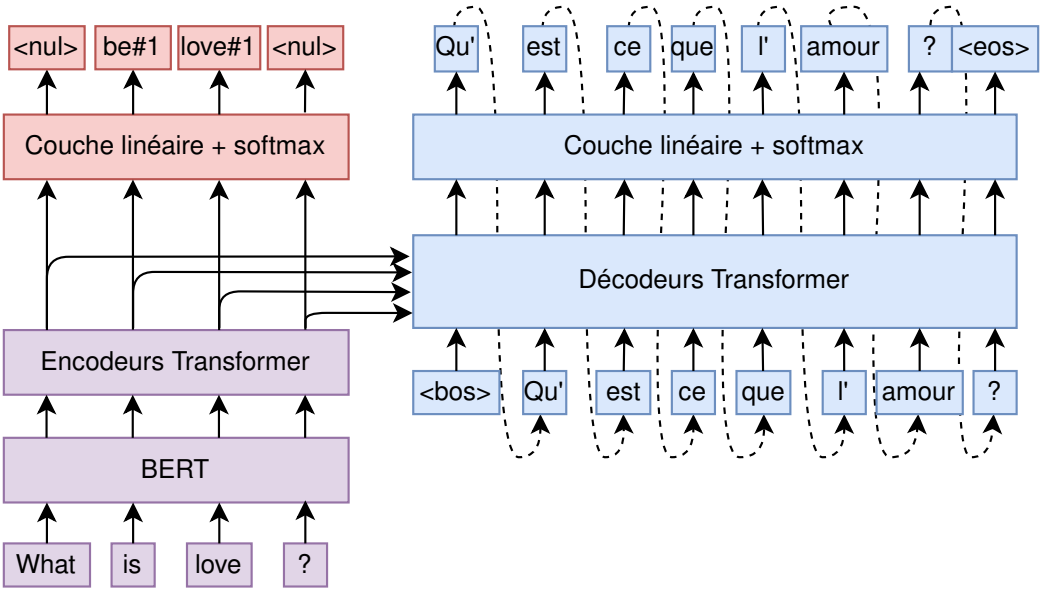


FIGURE 8.1 – Architecture neuronale pour un modèle joint de désambiguïsation lexicale et de traduction automatique.

Notre architecture pour un modèle joint de TA et de DL, illustré dans la figure 8.1, repose ainsi principalement sur ces deux composants : BERT et Transformer. Comme on peut le voir, l'architecture est découpée en trois parties distinctes qu'on appelle l'encodeur, le décodeur et le classifieur. L'encodeur (en bas à gauche) contient les paramètres qui sont communs aux deux tâches, le décodeur (en haut à droite) permet la traduction, et le classifieur (en haut à gauche) permet la désambiguïsation de la source.

Il est à noter que cette architecture est fondamentalement différente de celle présentée au chapitre 7 (section 7.2.2). En effet, plutôt qu'une architecture « encodeur-décodeur » classique, avec un décodeur qui produit plusieurs traits, la nouvelle architecture « encodeur-décodeur-classifieur » que nous proposons ici n'a jamais été évaluée à notre connaissance.

Une des principales différences entre cette méthode et celle présentée en section 7.2.2, est qu'en entraînant notre système à prédire les sens des mots de la

langue source, plutôt que ceux de la langue cible, nous avons deux sorties du réseau de longueurs différentes : une sortie de la longueur de la phrase source, et une sortie de la longueur de la phrase cible. De plus, cette fois, seul l'encodeur est commun aux deux objectifs (produire des mots et produire des sens), ce qui devrait offrir une meilleure représentation des mots dans l'encodeur, sans nécessairement forcer le décodeur à prendre ces sens en compte.

Enfin, avec cette nouvelle architecture, nous avons maintenant deux objectifs à combiner et à optimiser pendant l'apprentissage : la fonction de coût liée à la classification des sens dans le classifieur, et celle liée à la classification des mots dans le décodeur.

Les deux fonctions objectif à minimiser sont l'entropie croisée, avec deux différences : d'abord, la fonction objectif du classifieur ignore les étiquettes de sens qui correspondent à l'absence de sens, et ensuite, la fonction objectif du décodeur est une version « lissée sur les étiquettes » (*label smoothed*), déjà utilisée dans notre système de TA (voir section 5.3.3 et section 7.3.5 pour plus de détails).

Nous combinons ainsi ces deux fonctions objectif pendant l'apprentissage en les sommant, lors de la phase de rétro-propagation du gradient.

8.3 Protocole expérimental

Dans cette section, nous allons détailler le protocole expérimental que nous avons suivi afin d'entraîner et d'évaluer les systèmes qui s'appuient sur notre nouvelle architecture. Nous évaluerons à la fois leur capacité à désambiguïser ainsi que leur capacité à traduire.

8.3.1 Données d'apprentissage

Pour entraîner le plus efficacement possible notre modèle, nous avons besoin de corpus qui soient en même temps bilingues et annotés en sens WordNet. De tels corpus constitués manuellement n'existent pas à notre connaissance, [2] alors nous proposons d'exploiter les meilleurs systèmes de TA et de DL de l'état de l'art afin de traduire automatiquement les données annotées en sens, et d'annoter automatiquement en sens les données parallèles que nous utiliserons.

2. On peut tout de même citer les corpus d'évaluation de la tâche 12 de SemEval 2013 (Navigli et al., 2013), disponibles en cinq langues et annotés en sens BabelNet. Il n'est constitué cependant que d'environ 300 phrases.

Nous évaluerons ensuite les performances de nos modèles entraînés sur l'ensemble de nos données, qui seront ainsi partiellement générées automatiquement, sur les tâches de DL et sur les tâches de TA. De plus, nous mènerons des expériences dans lesquelles nous poursuivrons l'apprentissage en ajustement fin (*fine tuning*), en exploitant uniquement les parties manuelles des corpus pour chacune des deux tâches.

Nous allons maintenant décrire nos données d'entraînement, ainsi que la méthode pour leur traduction ou leur désambiguïsation.

8.3.1.1 Corpus annotés en sens et leur traduction

Pour entraîner et évaluer nos modèles sur la désambiguïsation lexicale, nous utilisons le SemCor (Miller et al., 1993) et le WordNet Gloss Corpus (fourni au sein de WordNet (Miller et al., 1990) depuis la version 3.0), car nous avons montré au chapitre 5 que cet ensemble nous permettait d'obtenir nos meilleurs résultats.

Nous exploitons nos versions UFSAC des corpus (voir chapitre 4), et nous utilisons la méthode de compression de vocabulaire vue au chapitre 6, pour avoir les meilleures performances possibles. L'ensemble des données d'entraînement consiste ainsi en 154 835 phrases annotées en sens WordNet.

Afin d'obtenir les versions traduites de ces corpus, nous nous sommes appuyés sur le meilleur système de traduction anglais-allemand fourni avec l'outil Fairseq [3]. Le modèle de traduction ainsi utilisé, décrit dans Ng et al. (2019), est fondé sur l'architecture Transformer (Vaswani et al., 2017) et exploite environ 27 millions de phrases parallèles ainsi que 521 millions de phrases parallèles synthétiques supplémentaires générées par *back-translation* (voir section 2.4.3.2).

Ce système de traduction a obtenu un score BLEU de 42,7 sur la tâche de traduction anglais-allemand de la campagne d'évaluation WMT 2019 (Barrault et al., 2019) en ayant été entraîné sur 128 GPU NVidia Volta, et il a ainsi remporté la compétition.

3. https://github.com/pytorch/fairseq

8.3.1.2 Corpus parallèles et leur désambiguïsation

Concernant la traduction automatique, nous avons repris la tâche de traduction anglais-allemand d'IWSLT 2014 (Cettolo et al., 2014), comme dans le chapitre 7, avec les mêmes pré-traitements (voir section 7.3.3).

Une fois pré-traitées, les données d'entraînement consistent en 164 179 phrases parallèles, ce qui est du même ordre de taille que nos données d'entraînement pour la DL.

Pour la désambiguïsation automatique de ces corpus parallèles, nous avons suivi le même processus que pour les expériences du chapitre précédent (voir section 7.3.4), c'est-à-dire que nous avons désambiguïsé la partie anglaise avec notre meilleur système qui obtient des performances état de l'art et qui est présenté au chapitre 6.

8.3.1.3 Récapitulatif

Corpus	#phrases	#mots anglais	#mots allemands	#annotations en sens
IWSLT	164 179	3 267 185	$3\,090\,267^{+}$	$1\,420\,173^{-}$
SemCor+WNGC	154 835	2 413 278	$2\,489\,865^{-}$	$726\,309^{+}$
Total	319 014	5 680 463	$5\,580\,132^{\pm}$	$2\,146\,482^{\pm}$

TABLE 8.1 – Taille de nos données d'apprentissage par corpus. Les symboles $+$, $-$ et $\pm$ indiquent respectivement les données originales, générées automatiquement ou un mélange des deux.

Dans le tableau 8.1, nous fournissons un récapitulatif du nombre de phrases, de mots et de mots annotés en sens dans nos corpus d'entraînement. Nous pouvons ainsi voir qu'en choisissant ces données, nous avons un nombre de phrases assez équilibré entre les deux tâches.

Concernant la proportion du nombre de mots annotés sur le nombre de mots total, elle est supérieure dans le corpus d'IWSLT par rapport à l'ensemble SemCor+WNGC, car tous les mots ne sont pas annotés dans le SemCor.

Dans le tableau 8.2, nous montrons la taille des vocabulaires de mots et de sens utilisés en entrée et en sortie de notre système. Le vocabulaire de mots de la langue source correspond ainsi aux sous-unités de mots présents dans le modèle BERT `bert-large-cased` que nous utilisons, et le vocabulaire de mots de la langue

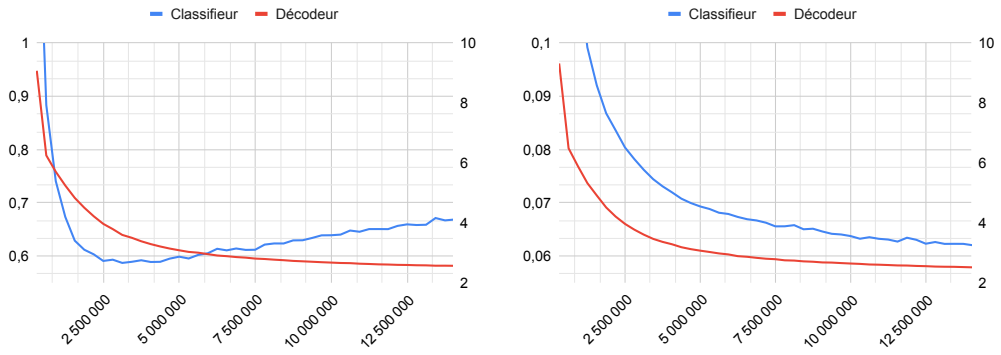
Vocabulaire	Taille
Mots (langue source)	28 996
Mots (langue cible)	14 121
Sens	29 877

TABLE 8.2 – Taille des vocabulaires de mots et de sens utilisés en entrée et en sortie de notre système.

cible correpond aux mêmes sous-unités de mots que celles utilisées au chapitre 7. Enfin, le vocabulaire de sens correspond au nombre de sens distincts présents sur l'ensemble de nos données d'entraînement, après avoir appliqué la compression de vocabulaire à travers les hyperonymes décrite au chapitre 6.

8.3.2 Entraînement

L'entraînement de nos modèles joints vise à maximiser deux critères à la fois : un score de DL et un score de TA. Pour cela, en plus de combiner les fonctions de coût pour la rétro-propagation du gradient, nous évaluons aussi périodiquement nos modèles pendant l'apprentissage sur un corpus de développement.



(a) Avant pondération (b) Après pondération

FIGURE 8.2 – Évolution des fonctions de coût du décodeur et du classifeur de notre modèle sur le corpus de développement au cours de l'apprentissage, avant et après leur pondération. L'axe des abscisses indique le nombre d'échantillons traités.

Le corpus de développement consiste en 6 750 phrases extraites aléatoirement du corpus d'entraînement. Comme les quantités de phrases issues d'IWSLT et celles issues du couple SemCor+WNGC sont à peu près équivalentes (rapport de

1,06:1), le corpus de développement alterne ainsi les phrases annotées originellement en sens et celles traduites manuellement. La proportion de mots annotés en sens, elle, est supérieure sur le corpus IWSLT (rapport de 1,44:1) mais nous ne pensons pas que cela aura un impact négatif sur l'entraînement.

Pendant l'apprentissage, nous évaluons donc le score BLEU (Papineni et al., 2002) du décodeur et le score F1 du classifieur, sur ce corpus de développement, à la fin de chaque passe (*epoch*) sur nos données d'entraînement. Nous gardons à la fin de l'apprentissage le modèle qui maximise les deux critères, selon les fonctions de coût associées.

Comme nous pouvons le voir dans la figure 8.2a, qui montre les courbes d'évaluation de notre modèle sur le corpus de développement pendant l'entraînement, la valeur de la fonction de coût du classifieur diminue bien plus rapidement que celle du décodeur, et on observe rapidement du surapprentissage sur ce critère. Pour pallier cela, nous ajustons la fonction de coût du classifieur de façon à ce qu'elle soit dix fois moins importante que sa valeur originale. Après cette pondération, les deux fonctions de coût convergent à peu près à la même vitesse, comme on peut le voir sur la figure 8.2b.

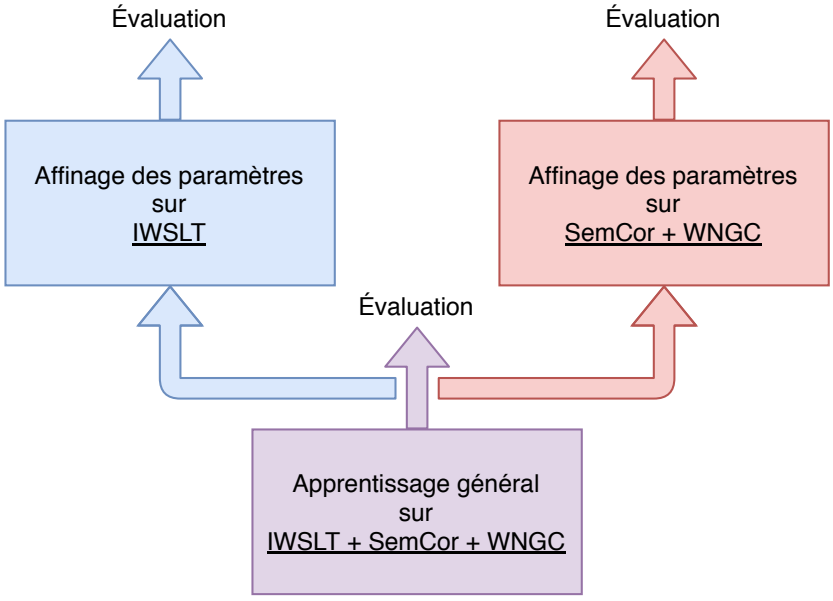


FIGURE 8.3 – Processus d'apprentissage de notre modèle neuronal joint.

Comme pour toutes nos expériences effectuées dans les chapitres précédents, nous utilisons l'optimiseur Adam (Kingma et Ba, 2015) avec les paramètres par défaut. Cependant, parce que nos données sont partiellement générées automatiquement, nous effectuons plusieurs phases d'apprentissage. Une première phase

consiste en 80 passes sur toutes nos données d'entraînement mélangées, avec l'optimiseur par défaut, puis une deuxième phase consiste en 20 passes d'affinage des paramètres (*fine-tuning*) sur les données originales seulement de chacune des tâches, et en divisant la vitesse d'apprentissage (plus précisément le *learning rate*) de l'optimiseur par dix.

Avec cette procédure, que nous illustrons dans la figure 8.3, nous pouvons ainsi maximiser les performances de la tâche sur laquelle le modèle est évalué, tout en mesurant l'apport de l'autre tâche dans les résultats.

8.4 Résultats

Données d'entraînement	Scores (%)			
	Traduction automatique			DL
	BLEU ↑	TER ↓	Meteor ↑	F1 ↑
Systèmes de référence - Modèles individuels				
IWSLT (chapitre 7)	29,5	49,64	49,04	-
SemCor+WNGC (chapitre 6)	-	-	-	77,08
Nos systèmes - Modèles joints				
IWSLT	29,0	49,75	48,53	76,42
SemCor+WNGC	25,3	54,06	45,61	77,40
IWSLT+SemCor+WNGC	**31,0**	48,44	50,35	77,84
↪ affinage sur IWSLT	30,9	**48,19**	50,26	78,00
↪ affinage sur SemCor+WNGC	30,9	48,53	**50,38**	**78,36**

TABLE 8.3 – Résultats de nos systèmes avec apprentissage joint de désambiguïsation lexicale et de traduction automatique. Les flèches (↑↓) indiquent le sens d'amélioration des scores. Les tirets (-) remplacent les scores qui ne peuvent pas être donnés par les modèles individuels car ils ne réalisent qu'une des deux tâches.

Nous présentons les résultats principaux de nos systèmes face aux systèmes de référence dans le tableau tableau 8.3, et nous présentons des résultats focalisés sur les tâches de TA et de DL dans les tableaux 8.4 et 8.5.

L'architecture du système de référence entraîné sur IWSLT correspond à la partie encodeur-décodeur de notre architecture jointe, et elle est donc identique à celle de notre système de TA présenté au chapitre 7. L'architecture du système de référence entraîné sur le couple SemCor+WNGC correspond à la partie encodeur-classifieur de notre architecture jointe, et elle est donc identique à celle de notre système de DL présenté au chapitre 6.

Pour la traduction, nous donnons les scores BLEU (Papineni et al., 2002), TER (Snover et al., 2006) et Meteor (Banerjee et Lavie, 2005) obtenus par nos systèmes sur le corpus d'évaluation de la tâche de traduction anglais-allemand de IWSLT 2014 (Cettolo et al., 2014). Pour la désambiguïsation, nous donnons le score F1 obtenu par nos systèmes sur la tâche standard « ALL » qui est la concaténation des tâches de DL issues des campagnes d'évaluation SensEval/SemEval (Edmonds et Cotton, 2001; Snyder et Palmer, 2004; Pradhan et al., 2007; Navigli et al., 2013; Moro et Navigli, 2015).

Nous donnons les résultats obtenus par nos modèles joints sur les données de traduction (IWSLT, désambiguïsées) ainsi que sur les données de désambiguïsation (SemCor+WNGC, traduites) seules, en plus des données mélangées.

On observe d'abord, dans le tableau 8.3, que sur la traduction automatique, notre modèle joint entrainé sur les mêmes données que le système de référence (IWSLT) obtient des résultats légèrement inférieurs (-0,5 BLEU). Cependant, avec les données automatiques ajoutées aux données d'entraînement, notre système obtient des résultats supérieurs (+1,5 BLEU). Enfin, la phase d'affinage sur le corpus original ne semble pas apporter de gain supplémentaire : les scores BLEU et Meteor sont détériorés d'environ 0,1 point mais le score TER est amélioré de 0,25 point.

Sur la tâche de désambiguïsation lexicale, le modèle joint entraîné sur le couple de corpus SemCor+WNGC obtient de meilleurs résultats que le système de référence (+0,32). Avec les données annotées automatiquement ajoutées, il obtient un gain encore supérieur (+0,76). Enfin, après la phase d'affinage sur les données originales, nous obtenons nos meilleurs résultats, avec un gain de +1,36 en comparaison avec le système de référence.

On remarque ainsi qu'à données d'entraînement équivalentes, notre méthode d'apprentissage joint ne semble être bénéfique qu'à la tâche de DL. En effet, sur la TA, l'apprentissage joint des sens ne semble ni guider l'apprentissage, ni apporter des informations supplémentaires, comme pour nos expériences d'apprentissage multi-tâches du section 2.6.1.

En revanche, les deux tâches bénéficient de l'ajout des données d'entraînement issues de l'autre tâche, et la méthode d'affinage sur les données originales améliore largement les résultats sur la DL. Sur la TA, la phase d'affinage sur les données originales n'apporte rien à notre système, mais nous pensons que cela peut être dû au fait que le système de TA utilisé pour traduire les corpus de DL a déjà des performances qui sont bien supérieures à notre système de base. Ainsi les données issues de ce système, même si elles sont automatiquement générées, apportent déjà beaucoup d'informations.

Enfin, on peut constater, grâce à ces expériences, qu'un même modèle neuronal est bien capable d'obtenir des résultats état de l'art sur les deux tâches à la fois, avant comme après la phase d'affinage sur l'une des tâches.

8.4.1 Résultats sur les tâches prises indivduellement

Afin d'aller plus loin, dans le tableau 8.4 et dans le tableau 8.5, nous comparons plus en détail les résultats obtenus par nos systèmes face à des systèmes avec un modèle individuel. En effet, nous avons ré-entraîné les systèmes de référence sur les mêmes données originales et automatiquement générées, et avec le même processus d'apprentissage que nos modèles joints (apprentissage général suivi d'un affinage sur les données originales). De cette manière, nous pouvons mesurer si les gains de performance que nous avons constatés sont vraiment issus de notre architecture jointe, ou bien des données d'entraînement supplémentaires.

De plus, pour la désambiguïsation lexicale, nous comparons les résultats de nos modèles joints face aux modèles individuels sur chaque partie du discours (noms, verbes, adjectifs et adverbes) avec un ensemble de 8 modèles, comme pour le système de référence du chapitre 6, en plus d'une moyenne des scores obtenus par ces 8 modèles.

Données d'entraînement	Scores de traduction automatique (%)		
	BLEU ↑	TER ↓	Meteor ↑
Modèles individuels - Architecture encodeur-décodeur			
IWSLT (chapitre 7)	29,5	49,64	49,04
IWSLT+SemCor+WNGC	31,7	47,89	50,93
↪ affinage sur IWSLT	**31,9**	**47,69**	**51,16**
Modèles joints - Architecture encodeur-décodeur-classifieur			
IWSLT	29,0	49,75	48,53
IWSLT+SemCor+WNGC	**31,0**	48,44	**50,35**
↪ affinage sur IWSLT	30,9	**48,19**	50,26

TABLE 8.4 – Résultats sur la tâche de traduction automatique anglais-allemand de IWSLT 2014. Nos systèmes sont comparés à des systèmes avec architecture encodeur-décodeur classique, sur les mêmes données d'entraînement. Les flèches (↑↓) indiquent le sens d'amélioration des scores. Les résultats en gras sont les meilleurs des modèles individuels et des modèles joints.

Dans le tableau 8.4, on observe qu'à ressources équivalentes, les modèles individuels obtiennent de meilleures performances que les modèles joints. De plus

le processus d'affinage semble améliorer significativement les performances du modèle individuel, mais pas celles du modèle joint.

Données d'entraînement	Scores F1 (%)				
	noms	verbes	adj.	adv.	total
Modèles individuels - Architecture encodeur-classifieur					
SemCor+WNGC (chapitre 6)	79,60	66,16	81,65	85,26	77,08
↪ ensemble (chapitre 6)	**81,40**	**68,70**	**83,66**	85,55	**79,00**
IWSLT+SemCor+WNGC	80,09	66,83	82,41	87,28	77,72
↪ affinage sur SemCor+WNGC	80,46	67,96	82,68	87,54	78,25
↪ ensemble	80,60	67,92	82,72	**87,57**	78,33
Modèles joints - Architecture encodeur-décodeur-classifieur					
SemCor+WNGC	79,73	66,89	81,83	86,42	77,40
↪ ensemble	80,60	**68,04**	82,30	**86,99**	78,27
IWSLT+SemCor+WNGC	80,00	67,55	82,83	86,42	77,84
↪ affinage sur SemCor+WNGC	80,79	67,62	82,91	86,81	78,36
↪ ensemble	**80,88**	67,49	**82,93**	86,71	**78,38**

TABLE 8.5 – Résultats obtenus par nos modèles joints en comparaison à des modèles entraînés individuellement, sur l'ensemble des tâches d'évaluation pour la désambiguïsation lexicale, en fonction des parties du discours. Les résultats en gras sont les meilleurs des modèles individuels et des modèles joints.

Enfin, dans le tableau 8.5, on peut observer que pour un même ensemble de données d'apprentissage, nos modèles joints obtiennent des scores systématiquement supérieurs aux modèles individuels, lorsque l'on évalue les modèles sans ensemble. Cependant, avec ensemble, les modèles individuels obtiennent des scores similaires et même supérieurs à nos modèles joints.

On peut aussi remarquer, sur les résultats en fonction des parties du discours, que l'ajout de l'IWSLT aux données d'entraînement n'améliore pas toujours les résultats. Nous pensons que cela est dû à la stratégie de repli qui est effectuée lorsqu'un mot n'a jamais été observé pendant l'apprentissage. Cette stratégie est en effet plus souvent appliquée lorsque moins de données sont utilisées pour l'entraînement, et elle obtient parfois des résultats compétitifs par rapport aux autres systèmes de DL (voir section 1.5.1.3).

8.5 Conclusion

Au cours de ce chapitre, nous avons mis en place une nouvelle architecture neuronale ainsi qu'une méthode pour l'apprentissage joint de deux tâches : la traduction automatique et la désambiguïsation lexicale.

Nous sommes partis du postulat que les systèmes état de l'art dans ces deux tâches sont fondés sur une architecture commune : l'architecture Transformer (Vaswani et al., 2017), et qu'ils peuvent bénéficier tous les deux d'un modèle de langue comme BERT (Devlin et al., 2019). L'architecture jointe que nous proposons repose ainsi sur une base commune qui repose sur ces deux composants, et que nous appelons l'encodeur, suivi de deux composants spécifiques aux deux tâches : le décodeur pour la TA, et le classifieur pour la DL.

Ensuite, nous avons mis au point un processus d'apprentissage pour notre modèle joint visant à optimiser à la fois la fonction de coût du décodeur et celle du classifieur, puis nous avons comparé notre architecture encodeur-décodeur-classifieur à des architectures encodeur-décodeur et encodeur-classifieur de référence, sur la tâche d'évaluation de la traduction de l'anglais vers l'allemand d'IWSLT 2014 (Cettolo et al., 2014), et sur la tâche d'évaluation « ALL » issue des campagnes d'évaluation de la désambiguïsation lexicale SensEval/SemEval (Edmonds et Cotton, 2001; Snyder et Palmer, 2004; Pradhan et al., 2007; Navigli et al., 2013; Moro et Navigli, 2015).

Avec nos résultats, nous avons d'abord montré que les modèles neuronaux bénéficiaient tous des données d'entraînement supplémentaires apportées par l'une ou l'autre tâche, même si ces données étaient partiellement générées automatiquement.

À données d'entraînement équivalentes, nos modèles joints obtiennent, sur la tâche de traduction automatique, des performances qui sont légèrement inférieures à celles obtenues par des modèles spécialisés. Cependant, sur la désambiguïsation lexicale, nos modèles joints obtiennent des performances légèrement supérieures.

Nous déduisons ainsi de ces expériences que l'apprentissage d'un modèle pour la TA n'est pas guidé, et l'apprentissage est même complexifié par l'ajout de la fonction objectif de la DL, de la même manière que dans nos expériences précédentes dans lesquelles on prédisait les étiquettes de sens en plus des mots dans la langue cible. Pour la DL, à l'inverse, l'ajout de la fonction objectif de la TA semble aider l'apprentissage et apporter une meilleure représentation de la phrase source au sein de l'encodeur du modèle.

D'une manière plus générale, nous constatons que nos modèles joints, même

s'ils obtiennent dans notre configuration des scores inférieurs sur la tâche de TA, arrivent globalement aussi bien à résoudre les deux tâches à la fois que deux modèles spécialisés, ce qui permet de factoriser grandement le nombre de paramètres des modèles neuronaux.

Pour finir, nous pensons que les études sur les systèmes neuronaux multi-tâches et les modèles joints comme celui que nous avons créé sont encore à leur début, en témoigne par exemple la convergence très récente des outils et architectures utilisés pour la création de systèmes état de l'art en TA et en DL. Cependant, nous pensons que ces méthodes vont se développer de plus en plus à l'avenir. En effet, on observe au travers des modèles de langue récents comme BERT, XLM, Albert, etc. (Devlin et al., 2019; Lample et Conneau, 2019; Lan et al., 2020) et des tâches d'évaluation multi-tâches comme GLUE (Wang et al., 2018) une volonté d'unifier les modèles neuronaux. De plus, les bibliothèques comme PyTorch [4] et transformers [5] apportent justement les moyens pour implémenter facilement de tels systèmes unifiés.

Pour aller plus loin, en partant de notre architecture jointe encodeur-décodeur-classifieur, de nombreuses études sont réalisables, notamment sur la quantité de paramètres des modèles neuronaux et sur la méthode pour combiner les fonctions de coût du classifieur et du décodeur, mais aussi sur la traduction vers d'autres langues plus lointaines que l'anglais, et aussi la désambiguïsation d'une autre langue.

4. https://pytorch.org/
5. https://github.com/huggingface/transformers

Conclusion générale

Au cours de cette thèse, nous avons étudié de manière jointe la désambiguïsation lexicale et la traduction automatique. Nous avons vu l'histoire et les méthodes appliquées dans chacune de ces deux tâches, puis présenté nos contributions portant sur l'amélioration de la DL, l'intégration d'informations sémantiques à des systèmes de TA neuronaux et la modélisation d'une architecture neuronale jointe de TA et de DL.

En TA, après les premières approches fondées sur des méthodes à base de règles et de dictionnaires statiques, les systèmes état de l'art s'appuient maintenant sur des méthodes statistiques qui exploitent des corpus parallèles et des réseaux de neurones pour leur apprentissage (voir chapitre 2). La DL, malgré un lien originellement fort avec la TA, s'en est rapidement détachée, et les méthodes à base de connaissances continuent d'évoluer en même temps que les méthodes supervisées qui s'appuient sur des corpus annotés en sens (voir chapitre 1).

Sur ces deux tâches, les approches à base de réseaux de neurones ont eu énormément d'influence. En TA, les modèles neuronaux ont rapidement remplacé les modèles statistiques « classiques » à partir des premiers travaux de Sutskever et al. (2014) et son architecture « séquence à séquence ». En DL, depuis le premier modèle neuronal de Kågebäck et Salomonsson (2016), tous les systèmes état de l'art reposent sur des réseaux de neurones (voir section 1.4.2 et section 2.2).

Plus généralement, nous avons constaté que les modèles de langue comme BERT (Devlin et al., 2019), les tâches d'évaluation unifiées comme GLUE (Wang et al., 2018), l'architecture neuronale Transformer (Vaswani et al., 2017) et les outils génériques pour faire des réseaux de neurones comme PyTorch [6], mènent tous à une certaine logique d'unification des tâches, des outils et des modèles neuronaux pour le TAL.

Dans ce contexte, nos contributions ont porté sur trois axes : d'abord, l'amélioration des méthodes de DL (chapitres 3, 4, 5 et 6), puis l'amélioration des sys-

6. https://pytorch.org/

tèmes de TA par l'apport d'informations sémantiques (chapitre 7), pour aboutir à la conception d'un modèle neuronal joint pour la TA et la DL (chapitre 8).

En effet, il nous a semblé important de confronter d'abord certaines limitations propres à la DL afin de pouvoir étudier ses interactions avec la TA. Nos travaux au chapitre 3 ont ainsi porté sur l'amélioration et la modernisation des méthodes à base de connaissances. Notre ressource UFSAC présentée au chapitre 4 répond au manque d'unification dans les corpus annotés en sens, ressource pourtant essentielle aux systèmes de DL état de l'art. Notre architecture neuronale présentée au chapitre 5 est la première à exploiter les modèles de langue comme BERT et l'architecture neuronale Transformer, et obtient ainsi des performances significativement supérieures aux précédents systèmes état de l'art. Enfin, nos méthodes de compression de vocabulaire présentées au chapitre 6 permettent de plus que doubler la couverture des systèmes de DL supervisés et d'améliorer leurs performances, en réduisant de plus de 80% le nombre de sens de WordNet.

Grâce à toutes ces contributions en DL, nous avons ainsi été les premiers à obtenir une précision dépassant les 90% sur une tâche reconnue de DL « gros grain » (voir section 1.5.1.3), soit une amélioration de plus de 6 points par rapport aux précédents systèmes état de l'art. Sur l'ensemble des tâches de DL « grain fin », nous avons amélioré l'état de l'art de plus de 7 points par rapport aux précédents systèmes état de l'art de Yuan et al. (2016) (voir chapitre 5 et chapitre 6).

Suite à nos travaux sur la DL, nous avons mené plusieurs expériences sur l'intégration de sens dans des systèmes de TA neuronaux qui vont plus loin que les approches déjà existantes (Hadj Salah, 2018; Pu et al., 2018), concernant les performances du système de DL sous-jacent, mais aussi concernant l'approche utilisée. Dans le chapitre 7, nous nous sommes ainsi appuyés sur l'architecture état de l'art Transformer pour nos systèmes de TA, et nous avons comparé l'apport des informations de sens WordNet avec des informations de sens directement issues de BERT, de manières jointe et séparée. Nous avons ainsi émis la conclusion que, d'une part, les informations contenues dans BERT permettent d'améliorer largement les performances des systèmes de TA qui ont accès à relativement peu de données, et d'autre part, les étiquettes de sens WordNet apportent une information supplémentaire, qui n'est pas encodée directement dans les vecteurs de BERT.

Enfin, à la suite de ces contributions, grâce à nos systèmes de DL et de TA, tous les deux fondés sur la même architecture Transformer, et exploitant le modèle de langue BERT, nous avons ainsi développé dans le chapitre 8 une nouvelle architecture jointe encodeur-décodeur-classifieur, capable tout à la fois de traduire et de désambiguïser du texte, en combinant les deux fonctions objectif. À travers ces expériences, nous avons constaté une meilleure stabilité des résultats sur la DL grâce au guidage de l'apprentissage par la fonction objectif de la TA. Cependant, notre

système de TA ne semblait pas bénéficier de l'apprentissage guidé par la fonction objectif de DL. Notre modèle neuronal joint permettait tout de même d'obtenir des résultats presque au niveau de l'état de l'art sur les deux tâches, tout en factorisant grandement les paramètres qui leur sont nécessaires.

Tous nos travaux ont été menés dans une logique d'unification des méthodes à la fois pour la recherche en DL mais aussi pour son étude jointe avec la TA. Dans un souci de reproductibilité et pour faciliter les travaux futurs, nous diffusons librement tous nos outils (vecteurs de sens [7], UFSAC [8], Disambiguate [9] et Disambiguate-Translate [10]), permettant d'implémenter, d'entraîner et d'évaluer facilement des systèmes de DL et de TA, conjointement et séparément.

Perspectives

Les travaux développés au cours de cette thèse ouvrent sur de nombreuses possibilités d'études sur la DL en tant que telle, sur les interactions entre la DL et la TA, et sur les modèles neuronaux joints pour le TAL.

Tout d'abord, nous pensons qu'il reste une marge d'amélioration potentiellement importante à explorer dans les nouveaux modèles de langue, architectures et méthodes d'apprentissage neuronales pour la DL supervisée. Par exemple, d'autres modèles de langue que BERT comme XLM (Lample et Conneau, 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) ou encore Albert (Lan et al., 2020) peuvent apporter une meilleure représentation ou une représentation complémentaire des mots pour nos systèmes. Des variantes de l'architecture Transformer peuvent être explorées, par exemple le Transformer XL (Dai et al., 2019), l'architecture RNMT+ (Chen et al., 2018) ou encore l'architecture proposée par Nguyen et Salazar (2019). Enfin, des méthodes d'apprentissage différentes peuvent être évaluées, par exemple le *fine tuning* des poids du modèle de langue pendant l'apprentissage ou encore la prise en compte du contexte au niveau du document et pas seulement au niveau de la phrase.

Ensuite, toujours concernant la DL spécifiquement, nous pensons que davantage de mixité entre les approches à base de connaissances et les approches supervisées est nécessaire si l'on souhaite obtenir de meilleures performances. On pourrait par exemple pousser plus loin nos méthodes de compression de vocabulaire

7. https://github.com/getalp/WSD-IWCS2017-Vialetal
8. https://github.com/getalp/UFSAC
9. https://github.com/getalp/disambiguate
10. https://github.com/getalp/disambiguate-translate

de sens, se servir à nouveau des liens sémantiques d'une base lexicale, et exploiter des synonymes, hyponymes ou hyperonymes monosémiques de sens afin d'inférer, dans des corpus bruts, des exemples pouvant servir d'échantillon d'apprentissage pour des mots polysémiques.

Enfin, si nos contributions spécifiques à la DL présentées dans cette thèse ont porté exclusivement sur l'anglais, il nous semble maintenant essentiel de mener des travaux sur la DL d'autres langues. De nombreuses ressources sont en effet disponibles pour de telles études. On trouve par exemple des modèles de langue français comme CamemBERT (Martin et al., 2019) et FlauBERT (Le et al., 2020b) qui offrent des représentations vectorielles contextualisées de qualité, et des inventaires de sens multilingues comme le Wiktionaire [11], BabelNet (Navigli et Ponzetto, 2010) ou monolingues comme celui de JeuxDeMots (Lafourcade, 2007), ainsi que des corpus annotés en sens comme celui issu du projet Ambiguss (Lafourcade et Le Brun, 2017) ou FrenchSemEval (Segonne et al., 2019) qui peuvent servir de base de travail solide pour la DL du français.

Cependant, nous pensons qu'il est nécessaire dans un premier temps de viser encore plus d'unification concernant les données d'entraînement et d'évaluation de la DL, toutes langues confondues. En effet, si la grande majorité des travaux sur la DL ciblent l'anglais, c'est notamment car les autres langues manquent d'inventaire de sens standard et surtout de ressources pour l'apprentissage supervisé et pour l'évaluation des systèmes de DL.

Nous pensons ainsi que la DL de nouvelles langues et même la DL en général profiteraient de disposer de plus de données unifiées, au travers par exemple d'une ressource similaire à UFSAC, mais multilingue, et non limitée à WordNet. Cette ressource, qui serait composée des données d'entraînement et d'évaluation de la DL dans divers inventaires de sens pourrait favoriser grandement le développement de méthodes de DL plus génériques, moins dépendantes de l'écosystème de WordNet et moins centrées sur l'anglais.

À plus long terme, le développement de la DL multilingue permettrait d'imaginer de nouvelles représentations des sens indépendantes des langues (voir les travaux de Tchechmedjiev (2016)). On pourrait aussi imaginer des systèmes de DL s'appuyant sur des représentations visuelles des sens, grâce à des images ou à des pictogrammes, ce qui aiderait par exemple l'acquisition du langage, ou la communication par le regard (voir par exemple Schwab et al. (2018)).

Au travers de nos expériences sur l'amélioration de la TA par les sens, nous avons constaté que les étiquettes de sens WordNet permettaient d'améliorer les résultats d'un système de TA anglais-français, entraîné sur un peu moins de 200 000

11. https://fr.wiktionary.org

phrases parallèles. On pourrait aller plus loin en variant la langue cible de la désambiguïsation par exemple. Des travaux semblables existent déjà, par exemple pour l'arabe (Hadj Salah, 2018), mais nous pensons qu'un travail préalable sur l'évaluation des systèmes de DL dans d'autres langues que l'anglais est d'abord nécessaire, comme nous l'avons évoqué précédemment.

Ensuite, nos méthodes pourraient être appliquées à des sytèmes de TA de plus grande ampleur, par exemple, sur le système de TA anglais-français état de l'art de Edunov et al. (2018a), entraîné sur plus de 60 millions de phrases parallèles. Cependant, de telles expériences devraient être menées en prenant en compte à la fois des considérations scientifiques, mais aussi économiques et écologiques. En effet, comme nous l'avions déjà évoqué dans la section 2.4.3.2 et comme le montre l'étude de Strubell et al. (2019), la consommation en temps et en énergie pour l'apprentissage de tels systèmes est conséquente. Nous pensons ainsi qu'il est toujours pertinent de travailler sur la TA dans un contexte avec des ressources restreintes, en plus de favoriser les travaux sur des langues moins bien dotées, qui sont, eux, essentiels.

Il nous paraît aussi capital de mêler des études plus précises sur la capacité de désambiguïsation des systèmes de TA (par exemple Rios et al. (2017) ou et Marvin et Koehn (2018)) à nos travaux. On pourrait étudier plus précisément et qualitativement l'influence réelle des systèmes de DL sur le choix lexical des systèmes de TA, ou faire varier les performances du système de DL sous-jacent pour voir les repercussions sur le système de TA.

Un autre axe de recherche pourrait aussi consister en l'apport de la TA pour la DL. À l'inverse de nos expériences qui visaient à annoter les mots en sens afin d'améliorer la TA, on pourrait imaginer une nouvelle architecture neuronale dans laquelle la traduction d'une phrase aiderait à sa désambiguïsation. On pourrait traduire automatiquement les données d'entraînement pour la DL, et ainsi transférer les connaissances du système de TA vers le système de DL.

Enfin, concernant les modèles joints et les interactions entre la TA et la DL, il est à ce jour encore difficile de mesurer réellement l'apport de tels modèles pour l'une ou l'autre tâche de manière globale, ainsi que de mesurer l'information partagée entre elles. Il nous paraît cependant essentiel de poursuivre ces travaux, car l'apprentissage multi-tâche et l'apprentissage par transfert ont prouvé une grande utilité dans de nombreux champs du TAL (notamment les modèles de langue comme BERT). Nous pensons que la création de modèles joints de TAL d'encore plus grande ampleur, réalisant à la fois une traduction dans plusieurs langues, une désambiguïsation, et d'autres tâches en même temps (prédiction de phrases, de mots, analyse de sentiment, etc.) seront un moyen essentiel afin d'atteindre de nouvelles performances état de l'art dans plusieurs domaines du TAL.

Annexe A

Publications

Au cours de cette thèse, j'ai écrit ou co-écrit huit articles de conférences internationales francophones, six articles de conférences internationales anglophones, deux articles de journaux francophones, un article de campagne d'évaluation et deux articles en prépublication (arXiv). En voici la liste :

- Loïc Vial, Benjamin Lecouteux, Didier Schwab, Hang Le et Laurent Besacier. The LIG system for the English-Czech Text Translation Task of IWSLT 2019. In Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT 2019). 2019.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Compression de vocabulaire de sens grâce aux relations sémantiques pour la désambiguïsation lexicale. In 26ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2019). 2019.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation. In Proceedings of the 10th Global Wordnet Conference (GWC 2019). 2019.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale. In Traitement Automatique des Langues. 2019.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Improving the coverage and the generalization ability of neural word sense disambiguation through hypernymy and hyponymy relationships. 2018.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale. In 25e

conférence sur le Traitement Automatique des Langues Naturelles (TALN 2018). 2018.

- Loïc Vial, Benjamin Lecouteux et Didier Schwab. UFSAC : Unification of Sense Annotated Corpora and Tools. In Language Resources and Evaluation Conference (LREC 2018). 2018
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Représentation vectorielle de sens pour la désambiguïsation lexicale à base de connaissances. In 24ème conférence sur le Traitement Automatique des Langues Naturelles (TALN 2017). 2017.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Uniformisation de corpus anglais annotés en sens. In 24ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2017). 2017.
- Loïc Vial, Benjamin Lecouteux et Didier Schwab. Sense Embeddings in Knowledge-Based Word Sense Disambiguation. In 12th International Conference on Computational Semantics (IWCS 2017). 2017.
- Loïc Vial, Andon Tchechmedjiev et Didier Schwab. Extension lexicale de définitions grâce à des corpus annotés en sens. In 23ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2016). 2016.
- Loïc Vial. Word sense disambiguation : improvement and integration in machine translation. Master's thesis, Master of Science in Informatics at Grenoble (MoSIG). 2016.
- Loïc Vial, Andon Tchechmedjiev et Didier Schwab. Comparison of global algorithms in word sense disambiguation. 2016.

- Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier et Didier Schwab. FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français. In 27ème conférence sur le Traitement Automatique des Langues Naturelles (TALN 2020). 2020
- Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier et Didier Schwab. FlauBERT : Unsupervised Language Model Pre-training for French. In Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020). 2020
- Didier Schwab, Pauline Trial, Céline Vaschalde, Loïc Vial et Benjamin Lecouteux. Providing semantic knowledge to a set of pictograms for people with disabilities : a set of links between WordNet and Arasaac : Arasaac-WN. In Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020). 2020

- Didier Schwab, Sébastien Riou, Amela Fejza, Loïc Vial, Johana Marku, Wafaa El Husseini, Sannara Ek, Miles Bardon et Yann Robert. Le projet GazePlay : des jeux ouverts, gratuits et une communauté pour les personnes en situation de polyhandicap. Dans le bulletin de la Société informatique de France. 2020.
- Didier Schwab, Pauline Trial, Céline Vaschalde, Loïc Vial et Benjamin Lecouteux. Apporter des connaissances sémantiques à un jeu depictogrammes destiné à des personnes en situation dehandicap : Un ensemble de liens entre Wordnet etArasaac, Arasaac-WN. In 26ème conférence sur le Traitement Automatique des Langues Naturelles (TALN 2019). 2019.
- Didier Schwab, Amela Fejza, Loïc Vial et Yann Robert. The GazePlay Project : Open and Free Eye-trackers Games and a Community for People with Multiple Disabilities. In International Conference on Computers Helping People with Special Needs (ICCHP). 2018.
- Marwa Hadj Salah, Loïc Vial, Hervé Blanchon, Mounir Zrigui, Benjamin Lecouteux et Didier Schwab. Traduction automatique de corpus en anglais annotés en sens pour la désambiguïsation lexicale d'une langue moins bien dotée, l'exemple de l'arabe. In 25e conférence sur le Traitement Automatique des Langues Naturelles (TALN 2018). 2018.

Annexe B

Contributions aux projets *open source*

J'ai créé ou participé à plusieurs projets *open source* au cours de cette thèse. En voici la liste :

— UFSAC (https://github.com/getalp/UFSAC), l'ensemble des corpus annotés en sens WordNet portés à notre connaissance dans notre format unifié ainsi que les outils pour manipuler ces corpus que nous avons présentés au chapitre 4.

— Disambiguate (https://github.com/getalp/disambiguate), notre outil pour la création, l'utilisation et l'évaluation de systèmes neuronaux de désambiguïsation lexicale (voir chapitre 5 et chapitre 6).

— Disambiguate-Translate (https://github.com/getalp/disambiguate-translate), notre extension de l'outil Disambiguate permettant aux systèmes neuronaux d'effectuer de la traduction automatique en plus de la DL, de façon jointe ou séparée (voir chapitre 7 et chapitre 8).

— https://github.com/getalp/WSD-IWCS2017-Vialetal, les vecteurs de sens issues de notre méthode présentée au chapitre 3 .

— FlauBERT (https://github.com/getalp/Flaubert), un modèle de langue dérivé de BERT pour le français pour lequel nous avons participé à la création (voir Le et al. (2020a,b)).

— GazePlay (https://github.com/GazePlay/GazePlay), un logiciel qui rassemble des mini-jeux destinés à aider les enfants en situation de handicap grâce à une interaction avec le regard, dont nous avons aidé au développement, notamment à travers la bibliothèque TobiiStreamEngineForJava (https://github.com/GazePlay/TobiiStreamEngineForJava), qui permet l'utilisation d'un occulomètre (*eye-tracker*) Tobii en Java.

Annexe C

Bibliothèque et outils UFSAC

Comme nous l'avons vu dans le chapitre 4, nous fournissons une ressource nommée UFSAC qui réunit tous les corpus annotés en sens connus à ce jour. De plus, nous avons implémenté une bibliothèque Java et un ensemble d'outils pour la manipulation de corpus dans notre format. Dans cette annexe, nous détaillons un peu plus cette bibliothèque ainsi que ces outils.

C.1 Bibliothèque Java

Notre bibliothèque Java permet deux styles de programmation : soit charger le corpus entier en mémoire, faire les opérations souhaitées et sauvegarder le corpus dans un fichier, soit séquentiellement charger, éditer et écrire le corpus, à la manière d'un flux.

C.1.1 Paquetage `core`

Le paquetage **`core`** contient les classes de base pour manipuler les corpus UFSAC. Par mesure de simplicité, les noms des classes reflètent exactement les noms des concepts décrits dans la section 4.4.2.

— La classe **`Annotation`** décrit ainsi une annotation attachée à une entité lexicale. Concrètement, c'est une paire clé-valeur de chaînes de caractères, ainsi qu'un pointeur vers l'entité lexicale annotée.

— La classe **`LexicalEntity`** décrit une entité lexicale qui possède un ensemble d'annotations, avec des méthodes publiques pour leur accès et modification.

— La classe **`Word`** hérite de `LexicalEntity` et a une annotation spéciale et obligatoire `surface_form`, qui est la valeur du mot, ainsi qu'un pointeur vers sa phrase parent.

— La classe **`Sentence`** hérite de `LexicalEntity`, contient une liste de mots et un pointeur vers son paragraphe parent.

— La classe **`Paragraph`** hérite de `LexicalEntity`, contient une liste de phrases et un pointeur vers son document parent.

— La classe **`Document`** hérite de `LexicalEntity`, contient une liste de paragraphes et un pointeur vers son corpus parent.

— Enfin, la classe **`Corpus`** hérite de `LexicalEntity` et contient une liste de documents.

Ces quelques classes, couplées à deux fonctions, `Corpus.saveToXML` et `Corpus.loadFromXML` permettent ainsi de créer, sauvegarder, charger et modifier des corpus facilement.

C.1.2 Paquetage `streaming`

Le paquetage **`streaming`** contient les classes nécessaires à la lecture et écriture séquentielle de corpus à la manière de flux, sans besoin de charger entièrement le corpus en mémoire. C'est un système similaire à la bibliothèque Java *SAX*[1] : des évènements surviennent au moment de la lecture d'un mot, d'une phrase, etc., et l'utilisateur choisit d'y répondre ou non.

— La classe **`StreamingCorpusReader`** permet de répondre aux évènements `readBeginCorpus`, `readBeginDocument`, `readWord`, etc.

— La classe **`StreamingCorpusModifier`** permet de modifier un corpus sur place, en le lisant séquentiellement, et en répondant aux évènements `modifyWord`, `modifySentence`, etc. qui vont écrire un nouveau corpus séquentiellement puis remplacer l'ancien.

— La classe **`StreamingCorpusWriter`** permet d'écrire un corpus séquentiellement via les méthodes `writeBeginSentence`, `writeWord`, etc.

1. http://www.saxproject.org/

C.2 Scripts

Pour finir, nous fournissons un ensemble de scripts permettant quelques opérations générales sur les corpus, en plus de la conversion d'un corpus depuis son format original.

— Le script **`convert_to_ufsac`** permet de convertir un corpus depuis son format original vers le format UFSAC, en déroulant toutes les opérations décrites dans la section 4.4.3

— Les scripts **`convert_from_raganato`** et **`convert_to_raganato`** permettent de convertir un corpus depuis le format décrit par Raganato et al. (2017a) vers UFSAC et vice-versa.

— Le script **`compute_mfs`** permet de calculer, pour tous les lemmes de WordNet, le sens le plus fréquent (MFS = *Most Frequent Sense*) à partir des corpus UFSAC listés en paramètre. Cela peut être très utile du fait que, d'une part, dans les campagnes d'évaluation, on donne systématiquement les scores d'un système référence appliquant la stratégie d'assigner à chaque mot son sens le plus fréquent, et d'autre part les sens de WordNet sont toujours classés par ordre de fréquence dans le SemCor.

— Les scripts **`add_corpus_lemma`** et **`add_corpus_pos`** utilisent respectivement l'outil *morphy* de WordNet et l'annotateur de Stanford pour annoter en lemmes et parties du discours tous les mots d'un corpus.

— Le script **`evaluate_wsd`** compare les annotations en sens produites par un système de DL avec des annotations de référence, afin de calculer la précision, le rappel, la couverture et le score F1 pour un ensemble de corpus.

— Enfin, le script **`generate_corpus_statistics`** permet de produire des statistiques générales (nombre de mots, de phrases, etc.) pour un ensemble de corpus donné.

Bibliographie

Eneko Agirre et Aitor Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1609067.1609070.

Eneko Agirre, Lluís Màrquez, et Richard Wicentowski, editors. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Association for Computational Linguistics, Prague, Czech Republic, June 2007. URL http://www.aclweb.org/anthology/S/S07/S07-1.

Eneko Agirre, Oier López de Lacalle, et Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Comput. Linguist.*, 40(1) :57–84, March 2014. ISSN 0891-2017. doi : 10.1162/COLI_a_00164. URL http://dx.doi.org/10.1162/COLI_a_00164.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A Smith, et David Yarowsky. Statistical machine translation. In *Final Report, JHU Summer Workshop*, volume 30, 1999.

Marianna Apidianaki et Benjamin Marie. METEOR-WSD : Improved Sense Matching in MT Evaluation. In *SSST-9*, Denver, US, 2015.

Jimmy Lei Ba, Jamie Ryan Kiros, et Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv :1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, et Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Satanjee Banerjee et Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing 2002*, Mexico City, February 2002.

Satanjeev Banerjee et Alon Lavie. Meteor : An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

Marco Baroni, Georgiana Dinu, et Kruszewski. Don't count, predict ! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1023.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, et Marcos Zampieri. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2 : Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, 2019. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W19-5301.

Pierpaolo Basile, Annalina Caputo, et Giovanni Semeraro. An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, pages 1591–1600, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C14-1151.

Yonatan Belinkov et Yonatan Bisk. Synthetic and Natural Noise Both Break Neural Machine Translation. In *ICLR 2018*, 2018.

W. Black, S. Elkateb, H. Rodriguez, M. Alkhalifa, A. Vossen, P.and Pease, M. Bertran, et C. Fellbaum. The Arabic WordNet Project. In *Proceedings of LREC 2006*, 2006.

Piotr Bojanowski, Edouard Grave, Armand Joulin, et Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5 :135–146, 2017. URL http://aclweb.org/anthology/Q17-1010.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, et Lucia Specia, editors. *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, June 2014. URL http://www.aclweb.org/anthology/W/W14/W14-33.

Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, et Pavel Pecina, editors. *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal, September 2015. URL http://aclweb.org/anthology/W15-30.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Liane Guillou, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Aurélie Névéol, Mariana Neves, Pavel Pecina, Martin Popel, Philipp Koehn, Christof Monz, Matteo Negri, Matt Post, Lucia Specia, Karin Verspoor, Jörg Tiedemann, et Marco Turchi, editors. *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, August 2016. URL http://www.aclweb.org/anthology/W/W16/W16-2200.

Sergey Brin et Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V. URL http://dl.acm.org/citation.cfm?id=297805.297827.

Denny Britz, Anna Goldie, Minh-Thang Luong, et Quoc Le. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi : 10.18653/v1/D17-1151. URL https://www.aclweb.org/anthology/D17-1151.

Samuel Brody et Mirella Lapata. Good neighbors make good senses : Exploiting distributional similarity for unsupervised WSD. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 65–72, Manchester, UK, 2008.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, et Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2) :79–85, 1990. URL https://www.aclweb.org/anthology/J90-2002.

Peter F. Brown, Jennifer C. Lai, et Robert L. Mercer. Aligning sentences in parallel corpora. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 169–176, Berkeley, California, USA, June 1991. Association for Computational Linguistics. doi : 10.3115/981344.981366. URL https://www.aclweb.org/anthology/P91-1022.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, et Robert L. Mercer. The mathematics of statistical machine translation : Parameter estimation. *Computational Linguistics*, 19(2) :263–311, 1993. URL https://www.aclweb.org/anthology/J93-2003.

Lou Burnard. The british national corpus, 1998.

Clara Cabezas et Philip Resnik. Using WSD Techniques for Lexical Selection in Statistical Machine Translation. Technical Report LAMP-TR-124,CS-TR-4736,UMIACS-TR-2005-42, University of Maryland, College Park, July 2005.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, et Josh Schroeder. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158. Association for Computational Linguistics, 2007.

Marine Carpuat et Dekai Wu. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 387–394. Association for Computational Linguistics, 2005.

Marine Carpuat et Dekai Wu. Improving statistical machine translation using word sense disambiguation. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 61–72, 2007.

Marine Jacinthe Carpuat. *Word Sense Disambiguation for Statistical Machine Translation*. Hong Kong University of Science and Technology, 2008.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, et Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57, 2014.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, et Jakob Uszkoreit. Kermit : Generative insertion-based modeling for sequences, 2019.

Yee Seng Chan, Hwee Tou Ng, et David Chiang. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June 2007a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1005.

Yee Seng Chan, Hwee Tou Ng, et Zhi Zhong. Nus-pt : Exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 253–256, Stroudsburg, PA, USA, 2007b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1621474.1621528.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, et Macduff Hughes. The best of both worlds : Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1008.

Xinxiong Chen, Zhiyuan Liu, et Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1110.

Monique Chevalier, Jules Dansereau, et Guy Poulin. *TAUM-METEO : description du système*. TAUM/Université de Montréal, 1978.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi : 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Massimiliano Ciaramita et Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language*

Processing, EMNLP '06, pages 594–602, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-73-6. URL http://dl.acm.org/citation.cfm?id=1610075.1610158.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, et Martin Wattenberg. Visualizing and Measuring the Geometry of BERT. *arXiv preprint arXiv :1906.02715*, 2019.

Jim Cowie, Joe Guthrie, et Louise Guthrie. Lexical disambiguation using simulated annealing. In *Proceedings of the workshop on Speech and Natural Language (HLT '91)*, 1992.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, et Ruslan Salakhutdinov. Transformer-xl : Attentive language models beyond a fixed-length context. In *ACL 2019*, 2019.

J. Daudé, L. Padró, et G. Rigau. Mapping WordNets Using Structural Information. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 504–511, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi : 10.3115/1075218.1075282. URL http://dx.doi.org/10.3115/1075218.1075282.

Bart Decadt, Véronique Hoste, Walter Daelemans, et Antal Van den Bosch. Gambl, genetic algorithm optimization of memory-based wsd. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004. URL http://www.aclweb.org/anthology/W04-0827.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, et Alexander Rush. Latent alignment and variational attention. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, et R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9712–9724. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8179-latent-alignment-and-variational-attention.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, et Kristina Toutanova. BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Philip Edmonds et Scott Cotton. Senseval-2 : Overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2387364.2387365.

Sergey Edunov, Myle Ott, Michael Auli, et David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics. doi : 10.18653/v1/D18-1045. URL https://www.aclweb.org/anthology/D18-1045.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, et Marc'Aurelio Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana, June 2018b. Association for Computational Linguistics. doi : 10.18653/v1/N18-1033. URL https://www.aclweb.org/anthology/N18-1033.

Andreas Eisele et Yu Chen. Multiun : A multilingual corpus from united nation documents. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, et Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Maha Elbayad, Laurent Besacier, et Jakob Verbeek. Pervasive attention : 2D convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 97–107, Brussels, Belgium, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/K18-1010.

Luis Espinosa-Anke, Thierry Declerck, Dagmar Gromann, Jose Camacho-Collados, et Mohammad Taher Pilehvar. Proceedings of the 5th workshop on semantic deep learning (semdeep-5). In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, Macau, China, 12 August 2019. Association for Computational Linguistics.

Cécile Fabre, Nabil Hathout, Lydia-Mai Ho-Dac, François Morlane-Hondère, Philippe Muller, Franck Sajous, Ludovic Tanguy, et Tim Van de Cruys. Présentation de l'atelier semdis 2014 : sémantique distributionnelle pour la substitution

lexicale et l'exploration de corpus spécialisés. In *Traitement Automatique du Langage Naturel - TALN 2014*, pages 196–205, Marseille, FR, 2014. Laboratoire Parole et Langage. URL http://oatao.univ-toulouse.fr/13215/.

Jérémy Ferrero, Laurent Besacier, Didier Schwab, et Frédéric Agnès. CompiLIG at SemEval-2017 Task 1 : Cross-Language Plagiarism Detection Methods for Semantic Textual Similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, Vancouver, Canada, August 2017.

W. N. Francis et H. Kučera. A Standard Corpus of Present-Day Edited American English, for use with Digital Computers (Brown). Technical report, Brown University, Providence, Rhode Island, 1964.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2) : 23–38, 1994.

William Gale, Kenneth Church, et David Yarowsky. One sense per discourse. In *Fifth DARPA Speech and Natural Language Workshop*, pages 233–237, Harriman, New-York, États-Unis, February 1992.

Mercedes García-Martínez, Loïc Barrault, et Fethi Bougares. Neural machine translation by generating multiple linguistic factors. In Nathalie Camelin, Yannick Estève, et Carlos Martín-Vide, editors, *Statistical Language and Speech Processing*, pages 21–31, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68456-7.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, et Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup et Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/gehring17a.html.

Alexander Gelbukh, Grigori Sidorov, et Sang Yong Han. Evolutionary approach to natural language wsd through global coherence optimization. *WSEAS Transactions on Communications*, 2(1) :11–19, 2003.

David Graff, Junbo Kong, Ke Chen, et Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1) :34, 2003.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, et Yoshua Bengio. On using monolingual corpora in neural machine translation, 2015.

Benoit Habert, Cécile Fabre, et Fabrice Issac. *De l'écrit au numérique. Constituer, normaliser et exploiter les corpus électroniques*. Elsevier Masson, 1998.

Marwa Hadj Salah. *Arabic word sense disambiguation for and by machine translation*. Université Grenoble Alpes ; Université de Sfax (Tunisie). Faculté des Sciences économiques et de gestion, December 2018. URL https://tel.archives-ouvertes.fr/tel-02139438.

Marwa Hadj Salah, Loïc Vial, Hervé Blanchon, Mounir Zrigui, Benjamin Lecouteux, et Didier Schwab. Traduction automatique de corpus en anglais annotés en sens pour la désambiguïsation lexicale d'une langue moins bien dotée, l'exemple de l'arabe. In *25e conférence sur le Traitement Automatique des Langues Naturelles*, Rennes, France, May 2018. URL https://hal.archives-ouvertes.fr/hal-01781185.

Sanda M Harabagiu, George A Miller, et Dan I Moldovan. Wordnet 2 - a morphologically and semantically enhanced resource. *SIGLEX99 : Standardizing Lexical Resources*, 1999.

Zellig S. Harris, Michael Gottfried, Thomas Ryckman, Paul Mattick Jr., Anne Daladier, T.N. Harris, et S. Harris. *The Form of Information in Science*, volume 104 of *Boston Studies in the Philosophy and History of Science*. Springer Netherlands, 1989.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, et Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

G. Hirst et David D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet : An electronic Lexical Database. C. Fellbaum. Ed. MIT Press. Cambridge. MA*, pages 305–332, 1998. Ed. MIT Press.

Sepp Hochreiter et Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8) :1735–1780, 1997. doi : 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Véronique Hoste, Anne Kool, et Walter Daelemans. Classifier optimization and combination in the english all words task. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, pages 83–86, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2387364.2387384.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, et Ralph Weischedel. Ontonotes : The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1614049.1614064.

Luyao Huang, Chi Sun, Xipeng Qiu, et Xuanjing Huang. GlossBERT : BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3507–3512, Hong Kong, China, November 2019. Association for Computational Linguistics. doi : 10.18653/v1/D19-1355. URL https://www.aclweb.org/anthology/D19-1355.

HuggingFace. Smaller, faster, cheaper, lighter : Introducing distilbert, a distilled version of bert. *Blog de HuggingFace*, 2019.

W John Hutchins. The georgetown-ibm experiment demonstrated in january 1954. In *Conference of the Association for Machine Translation in the Americas*, pages 102–114. Springer, 2004.

William John Hutchins. *Machine translation : past, present, future*. Ellis Horwood Chichester, 1986.

Ignacio Iacobacci, Mohammad Taher Pilehvar, et Roberto Navigli. 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, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1085.

Nancy Ide et Catherine Macleod. The american national corpus : A standardized resource of american english. In *Proceedings of Corpus Linguistics 2001*, volume 3, 2001.

Nancy Ide et Jean Véronis. Wsd : the state of the art. *Computational Linguistics*, 28(1) :1–41, 1998.

Nancy Ide, Collin Baker, Christiane Fellbaum, Charles Fillmore, et Rebecca Passonneau. Masc : the manually annotated sub-corpus of american english. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language

Resources Association (ELRA). ISBN 2-9517408-4-0. URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/617_paper.pdf. http ://www.lrec-conf.org/proceedings/lrec2008/.

Ann Irvine et Chris Callison-Burch. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 262–270, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-2233.

Rubén Izquierdo, Armando Suárez, et German Rigau. Exploring the automatic selection of basic level concepts. In *Proceedings of RANLP*, volume 7. Citeseer, 2007.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, et Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics. doi : 10.3115/v1/P15-1001. URL https://www.aclweb.org/anthology/P15-1001.

J. J. Jiang et D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *ROCLING X*, 1997.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, et Jeffrey Dean. Google's multilingual neural machine translation system : Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5 :339–351, 2017. doi : 10.1162/tacl_a_00065. URL https://www.aclweb.org/anthology/Q17-1024.

Mikael Kågebäck et Hans Salomonsson. Word sense disambiguation using a bidirectional lstm. In *5th Workshop on Cognitive Aspects of the Lexicon (CogALex)*. Association for Computational Linguistics, 2016.

A. Kilgarriff et J. Rosenzweig. Framework and results for english senseval. *Special Issue on SENSEVAL. Computers and the Humanties*, pages 15–48, 2000.

Adam Kilgarriff. Senseval : An exercise in evaluating word sense disambiguation programs. In *First international conference on language resources and evaluation (LREC)*, 1998.

Yoon Kim, Carl Denton, Luong Hoang, et Alexander M. Rush. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Diederik P. Kingma et Jimmy Ba. Adam : A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980.

Philipp Koehn. Europarl : A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings : the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL http://mt-archive.info/MTS-2005-Koehn.pdf.

Philipp Koehn et Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D07-1091.

Philipp Koehn et Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August 2017. Association for Computational Linguistics. doi : 10.18653/v1/W17-3204. URL https://www.aclweb.org/anthology/W17-3204.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses : Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180, 2007.

Sawan Kumar, Sharmistha Jat, Karan Saxena, et Partha Talukdar. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1568.

Mathieu Lafourcade. Making people play for lexical acquisition with the jeuxdemots prototype. In *SNLP'07 : 7th international symposium on natural language processing*, page 7, 2007.

Mathieu Lafourcade et Nathalie Le Brun. Ambiguss, a game for building a sense annotated corpus for french. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*, 2017.

Guillaume Lample et Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv :1901.07291*, 2019.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, et Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi : 10.18653/v1/D18-1549. URL https://www.aclweb.org/anthology/D18-1549.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, et Radu Soricut. Albert : A lite bert for self-supervised learning of language representations. In *Proceedings of ICLR 2020*, 2020.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, et Didier Schwab. Flaubert : des modèles de langue contextualisés pré-entraînés pour le français. In *27ème conférence sur le Traitement Automatique des Langues Naturelles (TALN 2020)*, 2020a.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, et Didier Schwab. Flaubert : Unsupervised language model pre-training for french. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, 2020b.

Minh Le, Marten Postma, Jacopo Urbani, et Piek Vossen. A deep dive into word sense disambiguation with lstm. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 354–365. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/C18-1030.

C. Leacock et M. Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet : An Electronic Lexical Database. C. Fellbaum. Ed. MIT Press. Cambridge. MA*, 1998.

Claudia Leacock, Geoffrey Towell, et Ellen Voorhees. Corpus-based statistical sense resolution. In *HUMAN LANGUAGE TECHNOLOGY : Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*, 1993. URL https://www.aclweb.org/anthology/H93-1051.

Yoong Keok Lee et Hwee Tou Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 41–48, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi : 10.3115/1118693.1118699. URL http://dx.doi.org/10.3115/1118693.1118699.

Claire Lemaire. *Traductologie et traduction outillée : du traducteur spécialisé professionnel à l'expert métier en entreprise*. Université Grenoble Alpes, 2017. URL http://www.theses.fr/2017GREAL005. Thèse de doctorat dirigée par Lavault-Olléon, Élisabeth Sciences du langage et Informatique Grenoble Alpes 2017.

Michael Lesk. Automatic sense disambiguation using mrd : how to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC '86*, pages 24–26, New York, NY, USA, 1986. ACM. ISBN 0-89791-224-1.

Omer Levy et Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2 : Short Papers*, pages 302–308, 2014. URL http://aclweb.org/anthology/P/P14/P14-2050.pdf.

Percy Liang, Ben Taskar, et Dan Klein. Alignment by agreement. In *Proceedings of HLT-NAACL*, pages 104–111, New York City, USA, June 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N06/N06-1014.

Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657297.

Pierre Lison et Jörg Tiedemann. Opensubtitles2016 : Extracting large parallel corpora from movie and tv subtitles. In *the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, 2016.

Frederick Liu, Han Lu, et Graham Neubig. Handling homographs in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language*

Technologies, Volume 1 (Long Papers), pages 1336–1345. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1121.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, et Veselin Stoyanov. Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*, 2019.

Daniel Loureiro et Alipio Jorge. Language Modelling Makes Sense : Propagating Representations through WordNet for Full-Coverage Word Sense Disambiguation. In *ACL 2019*, 2019.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, et Baobao Chang. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411. Association for Computational Linguistics, 2018a. URL http://aclweb.org/anthology/D18-1170.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, et Zhifang Sui. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 2473–2482. Association for Computational Linguistics, 2018b. URL http://aclweb.org/anthology/P18-1230.

Thang Luong, Hieu Pham, et Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015a. Association for Computational Linguistics. doi : 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, et Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 11–19, Beijing, China, July 2015b. Association for Computational Linguistics. doi : 10.3115/v1/P15-1002. URL https://www.aclweb.org/anthology/P15-1002.

Qingsong Ma, Johnny Wei, Ondřej Bojar, et Yvette Graham. Results of the WMT19 metrics shared task : Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation*

(Volume 2 : Shared Task Papers, Day 1), pages 62–90, Florence, Italy, August 2019. Association for Computational Linguistics. doi : 10.18653/v1/W19-5302. URL https://www.aclweb.org/anthology/W19-5302.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, et Benoît Sagot. CamemBERT : a Tasty French Language Model. *arXiv e-prints*, art. arXiv :1911.03894, Nov 2019.

Rebecca Marvin et Philipp Koehn. Exploring word sense disambiguation abilities of neural machine translation systems (non-archival extended abstract). In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1 : Research Papers)*, pages 125–131, Boston, MA, March 2018. Association for Machine Translation in the Americas. URL https://www.aclweb.org/anthology/W18-1812.

Bryan McCann, James Bradbury, Caiming Xiong, et Richard Socher. Learned in translation : Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

Diana McCarthy. Lexical substitution as a task for WSD evaluation. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation : Recent Successes and Future Directions*, pages 089–115. Association for Computational Linguistics, July 2002. doi : 10.3115/1118675.1118691. URL https://www.aclweb.org/anthology/W02-0816.

Diana McCarthy et Roberto Navigli. Semeval-2007 task 10 : English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 48–53, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1621474.1621483.

Rada Mihalcea et Dan Moldovan. extended wordnet : progress report. In *NAACL 2001 - Workshop on WordNet and Other Lexical Resources*, Pittsburgh, USA, 2001.

Rada Mihalcea, Paul Tarau, et Elizabeth Figa. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi : 10.3115/1220355.1220517. URL http://dx.doi.org/10.3115/1220355.1220517.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, et Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, et K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, et Katherine Miller. Wordnet : An on-line lexical database. *International Journal of Lexicography*, 3 :235–244, 1990.

George A. Miller, Claudia Leacock, Randee Tengi, et Ross T. Bunker. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, HLT '93, pages 303–308, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics. ISBN 1-55860-324-7. doi : 10.3115/1075671.1075742. URL http://dx.doi.org/10.3115/1075671.1075742.

Dan I. Moldovan et Adrian Novischi. Word sense disambiguation of wordnet glosses. *Computer Speech & Language*, 18 :301–317, 2004.

Andrea Moro et Roberto Navigli. Semeval-2015 task 13 : Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado, June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S15-2049.

Andrea Moro, Alessandro Raganato, et Roberto Navigli. Entity linking meets word sense disambiguation : a unified approach. *TACL*, 2 :231–244, 2014.

Lluś Màrquez, Javier Raya, John Carroll, Diana McCarthy, Eneko Agirre, David Martínez, Carlo Strapparava, et Alfio Gliozzo. Experiment a : Several all-words wsd systems for english. Technical report, Meaning, Developing multilingual Web-scale Language Technologies, 2002. URL http%3A%2F%2Fwww.lsi.upc.es%2F~nlp%2Fmeaning%2Fdocumentation%2FWP6.2a.ps.gz&usg=AFQjCNF4IErCDO1wt-ZOHzQvc9sMKvJspQ&sig2=Gbk0qcAJ03cyMRjYuzjyow.

Roberto Navigli. Word sense disambiguation : A survey. *ACM Computing Surveys*, 41(2) :10 :1–10 :69, feb. 2009. ISSN 0360-0300. doi : 10.1145/1459352.1459355. URL http://doi.acm.org/10.1145/1459352.1459355.

Roberto Navigli et Simone Paolo Ponzetto. Babelnet : Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the*

association for computational linguistics, pages 216–225. Association for Computational Linguistics, 2010.

Roberto Navigli, Kenneth C. Litkowski, et Orin Hargraves. Semeval-2007 task 07 : Coarse-grained english all-words task. In *SemEval-2007*, pages 30–35, Prague, Czech Republic, June 2007.

Roberto Navigli, David Jurgens, et Daniele Vannella. SemEval-2013 Task 12 : Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2 : Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, 2013. URL http://www.aclweb.org/anthology/S13-2040.

Hwee Tou Ng. Exemplar-based word sense disambiguation" some recent improvements. In *Second Conference on Empirical Methods in Natural Language Processing*, 1997. URL https://www.aclweb.org/anthology/W97-0323.

Hwee Tou Ng et Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense : an exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 40–47, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. doi : 10.3115/981863.981869. URL http://dx.doi.org/10.3115/981863.981869.

Hwee Tou Ng et Hian Beng Lee. DSO Corpus of Sense-Tagged English, 1997.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, et Sergey Edunov. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2 : Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy, August 2019. Association for Computational Linguistics. doi : 10.18653/v1/W19-5333. URL https://www.aclweb.org/anthology/W19-5333.

Toan Q. Nguyen et Julian Salazar. Transformers without Tears : Improving the Normalization of Self-Attention. In *Proceedings of the 16th International Workshop on Spoken Language Translation 2019*, 2019.

J. Niehues, R. Cattoni, S. Stüker, M. Negri, M. Turchi, T. Ha, E. Salesky, R. Sanabria, L. Barrault, L. Specia, et M. Federico. The IWSLT 2019 Evaluation Campaign. In *16th International Workshop on Spoken Language Translation 2019*, 2019. doi : 10.5281/zenodo.3532822. URL https://doi.org/10.5281/zenodo.3532822.

Franz Josef Och et Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1) :19–51, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, et Wei-Jing Zhu. Bleu : a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Tommaso Pasini et Roberto Navigli. Train-o-matic : Large-scale supervised word sense disambiguation in multiple languages without manual training data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 78–88. Association for Computational Linguistics, 2017. doi : 10.18653/v1/D17-1008. URL http://aclweb.org/anthology/D17-1008.

Siddharth Patwardhan, Satanjeev Banerjee, et Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'03, pages 241–257, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-00532-3. URL http://dl.acm.org/citation.cfm?id=1791562.1791592.

Jeffrey Pennington, Richard Socher, et Christopher D. Manning. Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, et Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi : 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Mohammad Taher Pilehvar et Jose Camacho-Collados. WiC : the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi : 10.18653/v1/N19-1128.

Alain Polguère. *Lexicologie et sémantique lexicale*. Les Presses de l'Université de Montréal, 2003.

Martin F Porter. An algorithm for suffix stripping. *program*, 14(3) :130–137, 1980.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, et Martha Palmer. Semeval-2007 task 17 : English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 87–92, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org.gate6.inist.fr/citation.cfm?id=1621474.1621490.

Xiao Pu, Nikolaos Pappas, James Henderson, et Andrei Popescu-Belis. Integrating weakly supervised word sense disambiguation into neural machine translation. *Transactions of the Association for Computational Linguistics*, 6 :635–649, 2018. doi : 10.1162/tacl_a_00242. URL https://www.aclweb.org/anthology/Q18-1044.

Alec Radford, Karthik Narasimhan, Tim Salimans, et Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, et Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1 (8), 2019.

Alessandro Raganato, Jose Camacho-Collados, et Roberto Navigli. Word sense disambiguation : A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, pages 99–110, Valencia, Spain, April 2017a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-1010.

Alessandro Raganato, Claudio Delli Bovi, et Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1167–1178. Association for Computational Linguistics, 2017b. URL http://www.aclweb.org/anthology/D17-1121.

Sree Harsha Ramesh et Krishna Prasad Sankaranarayanan. Neural machine translation for low resource languages using bilingual lexicon induced from comparable corpora. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Student Research Workshop*, pages 112–119, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi : 10.18653/v1/N18-4016. URL https://www.aclweb.org/anthology/N18-4016.

Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8, 978-1-558-60363-9.

Philip Resnik. WSD in NLP Applications. In Eneko Agirre et Philip Edmonds, editors, *Word Sense Disambiguation : Algorithms and Applications*, chapter 11, pages 299–337. Springer Netherlands, Dordrecht, 2006. ISBN 978-1-4020-4809-8. URL https://doi.org/10.1007/978-1-4020-4809-8_11.

Annette Rios, Laura Mascarell, et Rico Sennrich. Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings. In *Proceedings of the Second Conference on Machine Translation, Volume 1 : Research Papers*, Copenhagen, Denmark, 2017. URL http://www.statmt.org/wmt17/pdf/WMT02.pdf.

Annette Rios, Mathias Müller, et Rico Sennrich. The Word Sense Disambiguation Test Suite at WMT18. In *Proceedings of the Third Conference on Machine Translation*, pages 594–602, Brussels, Belgium, October 2018. Association for Computational Linguistics. URL http://www.statmt.org/wmt18/pdf/WMT064.pdf.

Gerard. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968. ISBN 0070544859.

Mark Sanderson. Word sense disambiguation and information retrieval. In *SIGIR'94*, pages 142–151. Springer, 1994.

Didier Schwab. *Approche hybride pour la modélisation, la détection et l'exploitation des fonctions lexicales en vue de l'analyse sémantique de texte.* U. Montpellier 2, 2005.

Didier Schwab, Jérôme Goulian, et Nathan Guillaume. Désambiguïsation lexicale par propagation de mesures sémantiques locales par algorithmes à colonies de fourmis. In *Traitement Automatique des Langues Naturelles (TALN)*, Montpellier, France, 2011.

Didier Schwab, Jérôme Goulian, et Andon Tchechmedjiev. Désambiguïsation lexicale de textes : efficacité qualitative et temporelle d'un algorithme à colonies de fourmis. *TAL*, 54(1) :99–138, 2013.

Didier Schwab, Amela Fejza, Loïc Vial, et Yann Robert. The gazeplay project : Open and free eye-trackers games and a community for people with multiple

disabilities. In *International Conference on Computers Helping People with Special Needs*, 2018.

Vincent Segonne, Marie Candito, et Benoît Crabbé. Using Wiktionary as a resource for WSD : the case of French verbs. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 259–270, Gothenburg, Sweden, May 2019. Association for Computational Linguistics. doi : 10.18653/v1/W19-0422. URL https://www.aclweb.org/anthology/W19-0422.

Rico Sennrich et Barry Haddow. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-2209.pdf.

Rico Sennrich et Biao Zhang. Revisiting low-resource neural machine translation : A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1021. URL https://www.aclweb.org/anthology/P19-1021.

Rico Sennrich, Barry Haddow, et Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 86–96, Berlin, Germany, August 2016a. Association for Computational Linguistics. doi : 10.18653/v1/P16-1009. URL https://www.aclweb.org/anthology/P16-1009.

Rico Sennrich, Barry Haddow, et Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi : 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Christophe Servan, Alexandre Bérard, Zied Elloumi, Hervé Blanchon, et Laurent Besacier. Word2Vec vs DBnary : Augmenting METEOR using vector representations or lexical resources ? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, pages 1159–1168, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1110.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, et John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231, 2006.

Benjamin Snyder et Martha Palmer. The english all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004. URL http://www.aclweb.org/anthology/W04-0811.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, et Ruslan Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1) :1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

Emma Strubell, Ananya Ganesh, et Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1355. URL https://www.aclweb.org/anthology/P19-1355.

Ilya Sutskever, Oriol Vinyals, et Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, et Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Kaveh Taghipour et Hwee Tou Ng. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 314–323, Denver, Colorado, May–June 2015a. Association for Computational Linguistics. doi : 10.3115/v1/N15-1035. URL https://www.aclweb.org/anthology/N15-1035.

Kaveh Taghipour et Hwee Tou Ng. One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction. In *Proceedings of the Nineteenth*

Conference on Computational Natural Language Learning, pages 338–344, Beijing, China, July 2015b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K15-1037.

Gongbo Tang, Rico Sennrich, et Joakim Nivre. An Analysis of Attention Mechanisms : The Case of Word Sense Disambiguation in Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation*, pages 26–35, Brussels, Belgium, October 2018. Association for Computational Linguistics. URL http://www.statmt.org/wmt18/pdf/WMT004.pdf.

Gongbo Tang, Rico Sennrich, et Joakim Nivre. Encoders help you disambiguate word senses in neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1429–1435, Hong Kong, China, November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D19-1149.

Ann Taylor, Mitchell Marcus, et Beatrice Santorini. The penn treebank : An overview, 2003.

Andon Tchechmedjiev. état de l'art : Mesures de similarité sémantique et algorithmes globaux pour la désambiguïsation lexicale a base de connaissances. In *RECITAL 2012*, Grenoble, June 2012. ATALA.

Andon Tchechmedjiev. *Interopérabilité Sémantique Multi-lingue des Ressources Lexicales en Données Liées Ouvertes*. Université Grenoble Alpes, 2016. URL http://www.theses.fr/2016GREAM067. Thèse de doctorat dirigée par Sérasset, Gilles Informatique Grenoble Alpes 2016.

Kristina Toutanova, Dan Klein, Christopher D. Manning, et Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi : 10.3115/1073445.1073478. URL https://doi.org/10.3115/1073445.1073478.

Eva Vanmassenhove et Andy Way. SuperNMT : Neural machine translation with semantic supersenses and syntactic supertags. In *Proceedings of ACL 2018, Student Research Workshop*, pages 67–73, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-3010.

Florentina Vasilescu, Philippe Langlais, et Guy Lapalme. Evaluating variants of the lesk approach for disambiguating words. In *Proceedings of LREC 2004*, pages 633–636, Lisbon, Portugal, May 2004.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, et Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Loïc Vial, Andon Tchechmedjiev, et Didier Schwab. Extension lexicale de définitions grâce à des corpus annotés en sens. In *23ème Conférence sur le Traitement Automatique des Langues Naturelles*, Paris, France, July 2016. URL https://hal.archives-ouvertes.fr/hal-01332850.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Sense Embeddings in Knowledge-Based Word Sense Disambiguation. In *12th International Conference on Computational Semantics*, Montpellier, France, September 2017a. URL https://hal.archives-ouvertes.fr/hal-01599685.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Uniformisation de corpus anglais annotés en sens. In *24ème Conférence sur le Traitement Automatique des Langues Naturelles*, Orléans, France, June 2017b. URL https://hal.archives-ouvertes.fr/hal-01599578.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Représentation vectorielle de sens pour la désambiguïsation lexicale à base de connaissances. In *24ème Conférence sur le Traitement Automatique des Langues Naturelles*, Orléans, France, June 2017c. URL https://hal.archives-ouvertes.fr/hal-01599572.

Loïc Vial, Andon Tchechmedjiev, et Didier Schwab. Comparison of global algorithms in word sense disambiguation. *CoRR*, abs/1704.02293 :1–22, 2017d. URL http://arxiv.org/abs/1704.02293.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Improving the coverage and the generalization ability of neural word sense disambiguation through hypernymy and hyponymy relationships, 2018a. URL http://arxiv.org/abs/1811.00960.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. UFSAC : Unification of Sense Annotated Corpora and Tools. In *Language Resources and Evaluation Confe-*

rence (LREC), Miyazaki, Japan, May 2018b. URL https://hal.archives-ouvertes.fr/hal-01718237.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale. In *25e conférence sur le Traitement Automatique des Langues Naturelles*, Rennes, France, May 2018c. URL https://hal.archives-ouvertes.fr/hal-01781183.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation. In *Proceedings of the 10th Global Wordnet Conference*, Wroclaw, Poland, 2019a. URL https://hal.archives-ouvertes.fr/hal-02131872.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale. *Traitement Automatique des Langues*, 2019b. URL https://hal.archives-ouvertes.fr/hal-02010901.

Loïc Vial, Benjamin Lecouteux, et Didier Schwab. Compression de vocabulaire de sens grâce aux relations sémantiques pour la désambiguïsation lexicale. In *Conférence sur le Traitement Automatique des Langues Naturelles (TALN-RECITAL)*, Toulouse, France, 2019c. URL https://hal.archives-ouvertes.fr/hal-02092559.

Loïc Vial. Word Sense Disambiguation : improvement and integration in Machine Translation. Master's thesis, Université Grenoble Alpes, 2016.

Loïc Vial, Benjamin Lecouteux, Didier Schwab, Hang Le, et Laurent Besacier. The LIG system for the English-Czech Text Translation Task of IWSLT 2019. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT 2019)*. Zenodo, November 2019d. doi : 10.5281/zenodo.3525529. URL https://doi.org/10.5281/zenodo.3525529.

David Vickrey, Luke Biewald, Marc Teyssier, et Daphne Koller. Word-sense disambiguation for machine translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 771–778, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi : 10.3115/1220575.1220672. URL http://dx.doi.org/10.3115/1220575.1220672.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, et Samuel Bowman. GLUE : A multi-task benchmark and analysis platform for natural

language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi : 10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446.

Warren Weaver. Translation. *Machine translation of languages*, 14 :15–23, 1955.

Gail Weiss, Yoav Goldberg, et Eran Yahav. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 740–745, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi : 10.18653/v1/P18-2117. URL https://www.aclweb.org/anthology/P18-2117.

John White, Theresa O'Connell, et Francis O'Mara. The arpa mt evaluation methodologies : evolution, lessons, and future approaches. In *Proceedings of the 1994 Conference, Association for Machine Translation in the Americas*, pages 193–205, 1994.

Gregor Wiedemann, Steffen Remus, Avi Chawla, et Chris Biemann. Does BERT Make Any Sense ? Interpretable Word Sense Disambiguation with Contextualized Embeddings. *arXiv preprint arXiv :1909.10430*, 2019.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, et Jeffrey Dean. Google's neural machine translation system : Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, et Quoc V. Le. Xlnet : Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv :1906.08237*, 2019.

David Yarowsky. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 454–460, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. doi :

10.3115/992133.992140. URL https://doi.org/10.3115/992133.992140.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. doi : 10.3115/981658.981684. URL http://dx.doi.org/10.3115/981658.981684.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, et Eric Altendorf. Semi-supervised word sense disambiguation with neural models. In *COLING 2016*, 2016.

Zhi Zhong et Hwee Tou Ng. It makes sense : A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858933.1858947.

Zhi Zhong et Hwee Tou Ng. Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 273–282. Association for Computational Linguistics, 2012. URL http://aclweb.org/anthology/P12-1029.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, et Sanja Fidler. Aligning books and movies : Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. doi : 10.1109/iccv.2015.11. URL http://dx.doi.org/10.1109/ICCV.2015.11.

Michał Ziemski, Marcin Junczys-Dowmunt, et Bruno Pouliquen. The united nations parallel corpus v1. 0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3530–3534, 2016.

Barret Zoph, Deniz Yuret, Jonathan May, et Kevin Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi : 10.18653/v1/D16-1163. URL https://www.aclweb.org/anthology/D16-1163.

Thèse de doctorat de Loïc Vial

Modèles neuronaux joints de désambiguïsation lexicale et de traduction automatique

Résumé

La désambiguïsation lexicale (DL) et la traduction automatique (TA) sont deux tâches centrales parmi les plus anciennes du traitement automatique des langues (TAL). Bien qu'ayant une origine commune, la DL ayant été conçue initialement comme un problème fondamental à résoudre pour la TA, les deux tâches ont par la suite évolué très indépendamment. En effet, d'un côté, la TA a su s'affranchir d'une désambiguïsation explicite des termes grâce à des modèles statistiques et neuronaux entraînés sur de grandes quantités de corpus parallèles, et de l'autre, la DL, qui est confrontée à certaines limitations comme le manque de ressources unifiées et un champ d'application encore restreint, reste un défi majeur pour permettre une meilleure compréhension de la langue en général.

Aujourd'hui, dans un contexte où les méthodes à base de réseaux de neurones et les représentations vectorielles des mots prennent de plus en plus d'ampleur dans la recherche en TAL, les nouvelles architectures neuronales et les nouveaux modèles de langue pré-entraînés offrent non seulement de nouvelles possibilités pour développer des systèmes de DL et de TA plus performants, mais aussi une opportunité de réunir les deux tâches à travers des modèles neuronaux joints, permettant de faciliter l'étude de leurs interactions.

Dans cette thèse, nos contributions porteront dans un premier temps sur l'amélioration des systèmes de DL, par l'unification des données nécessaires à leur mise en oeuvre, la conception de nouvelles architectures neuronales et le développement d'approches originales pour l'amélioration de la couverture et des performances de ces systèmes. Ensuite, nous développerons et comparerons différentes approches pour l'intégration de nos systèmes de DL état de l'art et des modèles de langue dans des systèmes de TA, pour l'amélioration générale de leur performance. Enfin, nous présenterons une nouvelle architecture pour l'apprentissage d'un modèle neuronal joint pour la DL et la TA, s'appuyant sur nos meilleurs systèmes neuronaux pour l'une et l'autre tâche.