Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Stratégies multi-tâches pour la détection des relations lexico-sémantiques.

Présentée et soutenue par
HOUSSAM AKHMOUCH

Thèse soutenue le 28 Novembre 2022
devant le jury composé de

Mme Agata Savary, Professeur, Université Paris Saclay (Rapporteur)
Mme Lynda Tamine-Lechany, Professeur, Université Paul Sabatier (Rapporteur)
M Massih-Reza Amini, Professeur, Université Grenoble Alpes
M Youssef Diouane, Professeur, Institut Supérieur de l'Aéronautique et de l'Espace
Mme Haïfa Zargayouna, Maître de conférences, Université Paris 13
M Gaël Dias, Professeur, Université Caen Normandie (Directeur de thèse)

Thèse dirigée par GAEL DIAS (Groupe de recherche en informatique, image, automatique et instrumentation)

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Résumé

L'identification de la relation lexico-sémantique qui existe entre deux mots est d'une importance cruciale pour le traitement du langage naturel, comme c'est essentiel pour des tâches telles que l'expansion des requêtes ou la réponse aux questions dans la recherche d'information. À cette fin, il existe différentes méthodologies qui se concentrent sur une seule relation lexicale (classification binaire), identifient plusieurs relations sémantiques (classification multi-classes) ou apprennent simultanément des relations lexico-sémantiques liées cognitivement (classification multi-tâches). Différentes méthodologies ont été proposées qui soit (1) s'attaquent à l'ingénierie des caractéristiques, (2) affinent les espaces sémantiques latents, soit (3) tirent parti des liens cognitifs entre les relations sémantiques dans des contextes multi-tâches. Nous proposons et étudions des architectures multi-tâches partagées et privées qui combinent des représentations d'entrée continues distributionnelles et basées sur des patrons lexicaux et qui se concentrent spécifiquement sur les caractéristiques des mots uniques. Puis, nous étudions comment l'ingénierie des caractéristiques et les architectures multi-tâches peuvent être améliorées et donc combinées pour identifier les relations lexico-sémantiques. Enfin, nous définissons des architectures multi-tâches de type *one-vs-rest* qui combinent des ensembles de classifieurs appris dans une configuration multi-tâche en utilisant des réseaux de neurones pour résoudre des problèmes multi-classe pour la classification des sentiments, des émotions, des thématiques et des relations lexico-sémantiques.

Sommaire

Résumé **1**

Dedication **6**

Remerciements **5**

0 Introduction **6**

1 Background **16**
- 1.1 Introduction à l'apprentissage artificiel . . . 16
  - 1.1.1 Apprentissage supervisé . . . 18
  - 1.1.2 Classification binaire . . . 19
  - 1.1.3 Classification multi-classes . . . 20
  - 1.1.4 Evaluation des modèles . . . 21
    - 1.1.4.1 Métriques pour la classification binaire . . . 21
    - 1.1.4.2 Métriques pour la classification multi-classes . . . 22
- 1.2 Les plongements lexicaux . . . 23
  - 1.2.1 Le modèle général word2vec . . . 24
    - 1.2.1.1 Le modèle skip-gram . . . 26
    - 1.2.1.2 Le modèle CBOW . . . 27
  - 1.2.2 Le modèle GloVe . . . 28
- 1.3 Les réseaux de neurones . . . 30
  - 1.3.1 Perceptron multi-couche . . . 30
  - 1.3.2 Encodage des séquences . . . 31
  - 1.3.3 Les modèles du langage contextualisés . . . 33

2 Architectures multi-tâches pour l'identification des relations lexico-sématiques **35**
- 2.1 Motivation . . . 36
- 2.2 Un aperçu de la littérature . . . 37

2.3 Architecture d'apprentissage . . . 40
2.3.1 Représentations continues . . . 41
2.3.2 Représentation des patrons lexicaux . . . 41
2.3.3 L'architecture *shared-private* . . . 43
2.3.3.1 L'architecture générale . . . 44
2.3.3.2 Architéctures spécifiques . . . 47
2.4 Expériences . . . 48
2.4.1 Jeux de données . . . 48
2.4.2 Extraction des patrons lexicaux . . . 50
2.4.3 Configurations d'apprentissage . . . 50
2.5 Evaluation . . . 51
2.5.1 L'apprentissage concurrent entre deux tâches . . . 52
2.5.1.1 Interprétation des résultats . . . 52
2.5.1.2 Jeux de données équilibrés . . . 54
2.5.1.3 Analyse d'ablation . . . 55
2.5.1.4 BiLSTM privé . . . 57
2.5.2 Apprentissage concurrent de trois tâches . . . 60
2.5.3 Conclusion . . . 62

3 Etude approfondie des caractéristiques pour l'identification des relations lexico-sémantiques **64**
3.1 Analyse des caractéristiques . . . 68
3.1.1 Caractéristiques symétriques . . . 68
3.1.2 Caractéristiques distributionnelles asymétriques . . . 69
3.1.3 Caractéristiques paradigmatiques basées sur les patrons . . . 72
3.2 Paramètres multi-tâches . . . 73
3.2.1 Architectures multi-tâches . . . 74
3.2.2 Jeux de données . . . 75
3.2.3 Les configurations d'apprentissage . . . 75
3.2.4 Le fractionnement lexical . . . 76
3.3 Evaluation . . . 76
3.3.1 Modèles privés . . . 76
3.3.2 Modèles multi-tâches . . . 79
3.3.3 *Embeddings* sphériques . . . 83
3.4 Conclusion . . . 84

4 L'apprentissage multi-classes **85**
4.1 Stratégie *one-vs-rest* multi-tâches . . . 87

4.1.1 Architecture générale 87
4.1.2 Architectures inter-classes 90
4.2 Jeux de données 91
4.3 Configurations expérimentales 93
4.4 Résultats 95
4.4.1 Modèles multi-tâches 95
4.4.2 Analyse des boxplots pour le cas des relations lexico-sémantiques 98

5 Conclusion **101**

Appendix A Annexes **104**
A.1 Rétropropagation du gradient 104
A.2 Boxplots des architectures multi-tâches pour des problèmes multi-classes 112

Liste des figures

1.1 Modèles skip-gram et CBOW. 24
1.2 Architecture LSTM. 34

2.1 L'architecture générique partagée privée *shared-private*. Différentes configurations de $V(X)$ permettent de proposer des architectures tout partagé *fully-shared* pour M tâches. 44

3.1 Les architectures entièrement partagées et partagées-privées avec de multiples combinaisons de caractéristiques d'entrée. Le réseau entièrement partagé ne comprend que la couche bleue, c'est-à-dire $S(X_b)$. 73
3.2 Résultats du score F_1 pour toutes les combinaisons de caractéristiques. Le cadre 1 représente toute caractéristique individuelle seule, par exemple (*cosG*,UF) signifie uniquement *cosG*. Le cadre 2 représente toute combinaison 2 par 2 de caractéristiques, par exemple (*cosBr*,F2) correspond à (*cosBr*,*cosG*). Le cadre 3 désigne la suppression d'une caractéristique de l'ensemble de toutes les caractéristiques, par exemple (Cadre bleu 3,F2) désigne (*cos*,*cosBr*,*cosG*1*D*). Le cadre 4 représente la combinaison de toutes les caractéristiques pour une famille donnée (AF). *BcosF*, *BKullF* et *BpatF* représentent la meilleure combinaison de caractéristiques dans leur famille respective. 80

4.1 Architectures *fully-shared* et *shared-private* de la stratégie *one-vs-rest* multitâches. Les connexions en pointillés ne sont présentes que dans l'architecture *shared-private*. 89
4.2 Architectures *fully-shared* et *shared-private* de la stratégie K par K pour le cas spécifique 3 tâches et $K = 2$. 91
4.3 Représentation T-SNE des donnée. 92

4.4 Évolution du F1 Micro pour toutes les architectures avec des paramètres fixes pour tous les jeux de données. Afin de faciliter la lecture et du fait de l'impossibilité de garantir exactement le même nombre de paramètres, 5 niveaux ont été définis pour regrouper la taille des architectures explorées : *Très faible* entre 10k et 30k paramètres, *Faible* entre 30k et 100k, *Moyen* entre 100k et 200k, *Fort* entre 200k et 500k, et finalement, *Très Fort* entre 500k et 1M. 99
4.5 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée RRW. (1) MULTI : Modèle MLP multi-classes; (2) OR: Modèle *one-vs-rest*; (3) HIER : Modèle *Hierarchical*; (4) ORSPK : Modèle *one-vs-rest shared-private* K par K; (5) ORASK : Modèle *one-vs-rest all-shared* K par K. 100

A.1 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée Tweet2016 112
A.2 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée Rotten Toma. 113
A.3 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée BBC 113
A.4 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée BBC Sport 114
A.5 Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée EmoContext 114

A mon fils Illan, ma femme Doha, mes Parents et mes Soeurs.

Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse, M. Gaël Dias, Professeur des Universités au sein de l'université Caen Normadie. Je lui suis également reconnaissant pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

Je voudrais remercier les rapporteurs de cette thèse pour l'intérêt apporté à mon travail.

Je désire grandement remercier Mme Claire Feldman qui m'a encadré au Crédit Agricole Brie Picardie pour m'avoir fait confiance tout au long de ces trois années. Ses remarques, son ouverture d'esprit, sa franchise, sa gentillesse sont autant d'éléments qui m'ont permis d'atteindre les objectifs de l'entreprise dans le cadre du doctorat. Je la remercie pour tout cela.

Je remercie également tous mes collègues du crédit Agricole Brie Picardie avec qui j'ai pu collaborer et spécialement : Pierre Thorel, Lydie Laurent, Vincent Decker et Cyril Pasquet. Bien sûr, atteindre mes objectifs n'aurait pas été possible sans leur aide préciseuse.

Merci à Georgios Balikas, Nesrine Bennour et Karan Praharaj avec qui j'ai pu collaborer très étroitement sur différents sujets de recherche.

Un grand merci à mes parents pour leur soutien et leur éducation tout au long de mon parcours scolaire.

0

Introduction

La capacité d'identifier automatiquement les relations lexico-sémantiques est un enjeu important pour les applications de recherche d'information et de traitement du langage naturel telles que les systèmes de question-réponse (Dong et al. 2017), l'expansion de requêtes (Kathuria et al. 2017), ou le résumé de texte (Gambhir and Gupta 2017). Les relations sémantiques incarnent des phénomènes linguistiques symétriques et asymétriques

tels que la synonymie (par exemple, vélo $\leftrightarrow$ bicyclette), la co-hyponymie (par exemple, vélo $\leftrightarrow$ voiture), l'hyperonymie (par exemple, bâtiment $\rightarrow$ maison) ou la méronymie (par exemple, vélo $\rightarrow$ roue), mais on peut en énumérer davantage (Vylomova et al. 2016).

La plupart des approches se concentrent sur la modélisation d'une seule relation sémantique et consistent à décider si une relation r donnée existe entre une paire de mots (x, y) ou non (c'est-à-dire une classification binaire). Dans ce contexte, la grande majorité des efforts (Snow et al. 2004, Roller et al. 2014, Shwartz et al. 2016, Nguyen et al. 2017a, Vulić and Mrkšić 2018, Wang and He 2020) se concentrent sur l'hyperonymie qui est le principe d'organisation clé de la mémoire sémantique, mais des études existent sur l'antonymie (Nguyen et al. 2017b), la méronymie (Glavaš and Ponzetto 2017) et la co-hyponymie (Weeds et al. 2014, Jana et al. 2020).

Plus récemment, des stratégies multi-tâches ont été proposées, qui consistent à apprendre simultanément des relations sémantiques uniques. Formellement, un ensemble de classificateurs binaires $\{c^l | l = 1...p\}$ sont appris conjointement, où chaque classificateur c^l apprend si une relation donnée r^l existe entre une paire de mots (x, y) ou non, de sorte que si l'apprentissage d'une relation r^k est informatif pour le processus d'apprentissage d'une autre relation r^i, une performance de classification accrue peut être attendue pour les deux relations.[1] L'idée sous-jacente du paradigme multi-tâches est que si les tâches sont corrélées ou cognitivement liées, l'architecture d'apprentissage devrait améliorer sa capacité de généralisation en prenant en compte les informations partagées. Dans ce cadre, (Attia et al. 2016, Balikas et al. 2019) ont montré que des résul-

[1] Cette explication ne s'adapte pas exactement à toutes les configurations multi-tâches. Mais pour des raisons de compréhension, nous nous concentrons sur cette définition simple.

tats améliorés peuvent être obtenus. Cependant, les deux approches abordent les problèmes en question avec des architectures d'apprentissage de base entièrement partagées, et ne tirent pas profit des avancées récentes des architectures d'apprentissage multi-tâches, à savoir les modèles partagés-privés (Liu et al. 2017). De plus, ils ignorent l'impact des caractéristiques des mots uniques dans le processus d'apprentissage. Nous supposons que cela peut être particulièrement intéressant lors de l'apprentissage de relations asymétriques. Par exemple, plus un mot unique est général (par exemple, équipement vs. téléphone), plus il est probable qu'il s'agisse d'un hyperonyme plutôt que d'un hyponyme. Enfin, les deux méthodes s'appuient sur des représentations simples de paires de mots, en particulier la concaténation de plongements lexicaux de mots, bien que (Shwartz et al. 2016, Nguyen et al. 2017b) montrent l'intérêt de combiner l'approche distributionnelle avec l'approche basée sur les patrons lexicaux.

Dans cette thèse, nous proposons d'améliorer l'approche multi-tâches à grain fin très récente de (Balikas et al. 2019), qui vise à déterminer si le processus d'apprentissage d'une relation sémantique donnée peut être amélioré par l'apprentissage simultané d'une autre relation, où les relations sont la synonymie, la co-hyponymie, l'hyperonymie et la méronymie.[2] Nous concentrons particulièrement notre étude sur les directions de recherche suivantes:

- La définition de différentes représentations continues distributionnelles pour les paires de mots;

- La définition d'encodages basés sur des modèles BiLSTM;

[2]Nous étudions spécifiquement ce travail car le code et les ensembles de données sont disponibles pour la reproductibilité.

- La définition d'architectures multi-tâches partagées-privées comprenant des représentations partagées de chaque mot unique.

Dans ce contexte, les résultats de l'évaluation sur quatre jeux de données de référence (RUMEN (Balikas et al. 2019), ROOT9 (Santus et al. 2016b), Weeds (Weeds et al. 2004), et Bless (Baroni and Lenci 2011)) montrent que des améliorations significatives sont obtenues de manière cohérente en combinant des architectures partagées-privées conscientes du mot unique avec des représentations continues distributionnelles et basées sur la paire de mots en question.

Bien que les stratégies multi-tâches existantes aient montré des résultats prometteurs, elles ne tirent pas avantage des caractéristiques spécialisées. Cela pourrait être dû au fait que la combinaison des caractéristiques dans les architectures multi-tâches partagées et privées n'est pas simple et nécessite un réglage spécifique. Nous supposons qu'une étude plus approfondie sur des caractéristiques spécialisées pourrait améliorer les performances. Dans ce cadre, différentes stratégies ont été proposées qui définissent de nouvelles caractéristiques (Santus et al. 2017, Vu and Shwartz 2018) ou construisent des espaces sémantiques latents spécifiques (Nguyen et al. 2017a, Rei et al. 2018, Wang and He 2020). D'une part, Vu and Shwartz (2018) montrent que l'introduction du cosinus généralisé améliore drastiquement les résultats par rapport à la concaténation unique des plongements lexicaux de mots, mettant ainsi clairement en évidence les limites des espaces latents à usage général. Cependant, une étude complète des caractéristiques symétriques et asymétriques ne semble pas être réalisée, à l'exception de (Santus et al. 2017), l'un des travaux les plus complets dans le domaine.

Nous proposons d'étudier comment l'ingénierie des caractéristiques peut être cou-

plée à des stratégies multi-tâches pour l'identification de relations lexico-sémantiques. Nous définissons des nouvelles familles de caractéristiques symétriques et asymétriques afin de mieux discriminer les relations lexico-sémantiques.

Les résultats de l'évaluation sur l'ensemble de jeux de données Rumen, ROOT9, Weeds et Bless d'une architecture couplant des ensembles de caractéristiques optimisés et des modèles partagés-privés montrent que:

- La combinaison de caractéristiques au sein d'un ensemble de familles améliore les performances par rapport à l'utilisation d'un membre unique de la famille ;
- Les caractéristiques asymétriques sont un indice fort de discrimination pour les relations lexico-sémantiques asymétriques ;
- Les modèles partagés-privés améliorent les performances par rapport aux classificateurs binaires et entièrement partagés (Balikas et al. 2019, Bannour et al. 2020), aussi bien qu'ils équilibrent correctement la distribution des caractéristiques entre les couches privées et partagées ;
- Les caractéristiques asymétriques jouent un rôle important dans les architectures multi-tâches, étant une source d'information importante pour combiner les tâches symétriques et asymétriques.

Une autre direction de recherche consiste à traiter plusieurs relations sémantiques à la fois et peut être définie comme la décision de la relation sémantique r_i (s'il y en a une) entre une paire de mots (x, y). Ce problème multi-classes est un défi car il est connu que la distinction entre différentes relations sémantiques (par exemple la synonymie et

l'hyperonymie) est difficile (Shwartz et al. 2016). Dans ce cadre, nous avons créé le jeu de données multi-classes RRW qui combine Rumen, ROOT9 , Weeds et vise à aborder la synonymie, l'hyperonymie, la méronymie et la co-hyponymie comme un problème multi-classes. La classification multi-classe implique l'attribution d'une classe parmi un ensemble de plus deux étiquettes à une donnée d'entrée. Dans ce cadre plus large de classification textuelle, si de nombreux efforts ont été déployés pour (1) affiner des modèles d'apprentissage pour des problèmes spécifiques de classification textuelle (Teng et al. 2016, Zhou and Li 2020) et (2) concevoir des architectures neuronales spécifiques à la classification de texte (Conneau et al. 2017, Yao et al. 2019, Zhang and Zhang 2020), moins de travaux se sont attelés à proposer des modèles génériques qui peuvent traiter des problèmes multi-classes indépendamment de la tâche et de l'objet à classer.

Dans ce cadre, trois approches principales ont été proposées par l'état de l'art pour traiter les problèmes multi-classes. Premièrement, ils peuvent être résolus en étendant des techniques de classification binaire; directement par les arbres de décision, les réseaux bayésiens ou les K plus proches voisins, avec quelques modifications pour les réseaux de neurones (Ou and Murphey 2007) et les machines à vecteurs de support (SVM) (Crammer and Singer 2001). Deuxièmement, des stratégies ont été développées qui reposent sur une décomposition du problème en un ensemble de sous-problèmes binaires, dont les résultats sont combinés pour déterminer la solution multi-classe finale. Dans ce cadre, deux stratégies principales ont été largement utilisées, à savoir la stratégie *one-vs-rest* et la stratégie *one-vs-one*. Selon (Hsu and Lin 2002), les deux stratégies offrent souvent des performances similaires, tandis que des résultats contrastés sont observés par (Pawara et al. 2020). Dans ce même contexte, on peut également mention-

ner la stratégie de *"error-correcting output-coding"* (Dietterich and Bakiri 1994) et sa généralisation (Allwein et al. 2000). La troisième approche consiste à construire une architecture d'apprentissage hiérarchique en supposant qu'il existe une certaine relation entre les étiquettes de classes. Ainsi, deux stratégies principales ont été proposées, à savoir l'approche basée sur les instances (Zupan et al. 1999), et la stratégie basée sur les prédictions (Godbole et al. 2002, Silva-Palacios et al. 2017).

Dans cette thèse, nous proposons de tirer le meilleur parti des stratégies par décomposition en intégrant les relations entre les étiquettes de classes dans une approche de type *one-vs-rest*. À cette fin, nous concevons une stratégie multi-tâche, dans laquelle chaque classificateur *one-vs-rest* est appris dans une configuration multi-tâche, ce qui permet de prendre en compte l'interdépendance des étiquettes de classes en une seule étape d'apprentissage, par opposition aux stratégies hiérarchiques qui nécessitent de deux étapes interdépendantes. Ainsi, chaque problème (ou tâche) *one-vs-rest* devrait bénéficier de l'apprentissage simultané des autres tâches comme le suggère (Caruana 1998), si celles-ci sont corrélées ou montrent des similarités cognitives. En particulier, nous proposons deux architectures multi-tâches de type *one-vs-rest* (tout-partagé et partagé-privé). Ces deux architectures sont comparées à des solutions de l'état de l'art en utilisant six jeux de données de référence pour l'analyse des sentiments (Socher et al. 2013, Nakov et al. 2016), la détection des émotions (Chatterjee et al. 2019), la classification thématique (Greene and Cunningham 2006) et la classification de relations lexico-sémantiques (Balikas et al. 2019), qui est l'objectif principal de notre thèse.

Organisation du manuscrit

Dans le chapitre 1, nous présenterons le matériel nécessaire pour appréhender le

travail développé tout au long de cette thèse. Tout d'abord, nous exposerons les bases de la théorie de l'apprentissage, puis nous nous concentrerons plus particulièrement sur les problèmes de classification traités tout au long de cette thèse. Nous expliquerons ensuite les plongements lexicaux non contextuels, plus particulièrement GloVe (Pennington et al. 2014) que nous utiliserons lors de nos expériences. Nous finirons ce chapitre par exposer les bases théoriques des réseaux de neurones.

Dans le chapitre 2, nous utiliserons des stratégies multi-tâches partagées et partagées-privées pour la détection des relations lexico-sémantiques. Nous testerons plus particuliérement différentes représentations continues distributionnelles pour les paires de mots ainsi que des patrons lexicaux qui peuvent améliorer cette classification.

Dans le chapitre 3, nous continuerons dans la même direction de recherche. En particulier, nous effectuerons une étude approfondie sur des caractéristiques spécialisées et nous étudierons ainsi comment l'ingénierie des caractéristiques couplée à des stratégies multi-tâches pourraient améliorer les performances de l'identification des relations lexico-sémantiques.

Dans le chapitre 4, nous adapterons des architectures multi-tâches pour résoudre des problèmes multi-classes. Nous concevrons des architectures multi-classes, dans lesquelles chaque classificateur *one-vs-rest* est appris dans une configuration multi-tâche. Ainsi, nous prendrons en compte les liaisons congnitives qui pourraient exister entre les étiquettes de classes pour améliorer la classification multi-classe, notamment dans le cadre des relations lexico-sémantiques mais aussi la classification textuelle.

Contexte de la thèse

Ma thèse s'est déroulée dans un cadre de thèse CIFRE avec le Crédit Agricole Brie

Picardie. Avec l'augmentation du nombre de clients bancaires, le Crédit Agricole Brie Picardie dispose d'une quantité de données très large. Il faut donc automatiser au maximum les tâches auxquelles l'intelligence humaine apporte le moins de valeur ajoutée, afin de maximiser le temps commercial utilement consacré au service du client. La caisse régionale Crédit Agricole Brie Picardie travaille actuellement sur différents algorithmes d'apprentissage qui permettent une connaissance approfondie du client à partir de données structurées présentes dans ses bases de données.

L'idée à l'origine de cette thèse est d'intégrer le traitement du langage naturel (données non structurées) dans les activités de la caisse en se basant sur le savoir-faire déjà acquis en matière d'intelligence artificielle.

Publications

La thèse a donné lieu aux publications suivantes :

- Chapitre 3 : Akhmouch, H., Dias, G. & Moreno, J. (2021). Understanding Feature Focus in Multitask Settings for Lexico-semantic Relation Identification. 59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing - Findings (ACL-IJCNLP). Bangkok (Thailande)

- Chapitre 4 : Akhmouch, H., Bouanani, H. Dias, G. & Moreno, J. (2021). Stratégie Multitâche pour la Classification Multiclasse. 28ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN). Lille, France.

D'autres publications ont été réalisées en dehors du contexte direct de la thèse :

- Bannour, N., Dias, G. Chahir, Y. & Moreno, J. (2020) Patch-Based Identifica-

tion of Lexical Semantic Relations.42nd European Conference on Information Retrieval (ECIR). Lisbon (Portugal)

- Balikas, G., Dias, G., Moraliyski, R., Akhmouch, H., & Amini, M. (2019) Learning Lexical-Semantic Relations Using Intuitive Cognitive Links. 41st European Conference on Information Retrieval (ECIR). Cologne (Allemagne)

La statistique est la première des sciences inexactes.

Edmond et Jules de Goncourt

1 Background

1.1 Introduction à l'apprentissage artificiel

L'apprentissage artificiel est une branche de l'intelligence artificielle qui permet aux machines d'apprendre à partir de données d'apprentissage et de s'améliorer au fil du temps, sans être explicitement programmées. Les algorithmes d'apprentissage artificiel sont capables de détecter des similitudes dans les données afin de faire leurs propres

prédictions sur des données non explorées.

L'apprentissage artificiel peut être utilisé sur des quantités massives de données et peut être beaucoup plus précis que les humains dans certaines situations particulières. Il peut être appliqué à différents domaines tels que le marketing, la gestion de la relation client, l'ingénierie et la médecine, mais on peut en énumérer davantage. Pour comprendre le fonctionnement de l'apprentissage artificiel, nous allons explorer brièvement les méthodes les plus courantes.

L'apprentissage supervisé: les algorithmes d'apprentissage supervisé font des prédictions sur la base de données d'apprentissage étiquetées. Chaque échantillon comprend une entrée et une sortie. Un algorithme d'apprentissage supervisé analyse cet échantillon de données et fait une prédiction sur les données non explorées. Il s'agit de l'approche la plus courante et la plus populaire de l'apprentissage automatique. Elle est dite "supervisée" parce que ces modèles doivent être alimentés par des échantillons de données étiquetées préalablement pour apprendre.

L'apprentisage non supervisé: les algorithmes d'apprentissage non supervisé permettent de découvrir des relations dans des données non étiquetées. Dans ce cas, les modèles reçoivent des données d'entrée, mais les résultats souhaités sont inconnus. Ils doivent donc faire des déductions sans aucune orientation. L'un des types les plus courants d'apprentissage non supervisé est le partitionnement[1], qui consiste à regrouper des données similaires. Cette méthode est surtout utilisée pour l'analyse exploratoire et permet de détecter des tendances cachées.

L'apprentissage semi-supervisé: dans ce cas, les données d'apprentissage sont

[1] Clustering.

divisées en deux groupes. Une petite quantité de données étiquetées et un ensemble plus important de données non étiquetées. Dans ce cas, le modèle utilise les données étiquetées en entrée pour faire des déductions sur les données non étiquetées, ce qui donne généralement des résultats plus précis que les modèles d'apprentissage supervisé ordinaires pour un même ensemble de données étiquetées. Cette approche gagne en popularité, notamment pour les tâches impliquant de grands ensembles de données, comme la classification d'images. Dans cette thèse, nous nous concentrerons plus particulièrement sur l'apprentissage supervisé.

1.1.1 Apprentissage supervisé

L'apprentissage supervisé traite des données étiquetées, généralement manuellement annotées, afin de trouver une fonction de prédiction qui associe au mieux l'observation d'entrée à sa sortie de telle sorte que l'erreur de généralisation pour toute nouvelle paire d'exemples et sa sortie (non vue par le classifieur) soit la plus faible. Nous supposons que l'entrée $\boldsymbol{X} \subseteq \mathbb{R}^d$ se trouve dans un espace vectoriel fini de dimension d et suit une distribution de probabilité $\mathbb{P}(x), x \in \boldsymbol{X}$ est une observation de l'entrée. Chaque $x \in \boldsymbol{X}$, a une valeur réelle $y \in \boldsymbol{Y}$ associée qui suit une distribution de probabilité $\mathbb{P}(y|x)$. L'objectif de l'apprentissage supervisé est de trouver la meilleure fonction $h^* \in \boldsymbol{H} : \boldsymbol{X} \mapsto \boldsymbol{Y}$ qui fait le moins d'erreurs possibles pour estimer $\boldsymbol{Y}$. Ceci est fait en suivant le principe de minimisation du risque empirique en considérant un échantillon étiqueté de taille m tiré de manière aléatoire identique et indépendante par rapport à la même distribution de probabilité $\mathbb{P}$.

1.1.2 Classification binaire

Il s'agit d'un processus ou d'une tâche de classification, dans lequel l'entrée $\boldsymbol{X}$ est classée en deux classes $\boldsymbol{Y} = \{1, -1\}$. Un exemple classique est la classification des emails en deux catégories possibles: spam vs non spam. Dans ce contexte, soit x un email donné, l'espace de sortie $\boldsymbol{Y} = \{1, -1\}$. -1 fait référence à un spam et $+1$ à un non spam. Dans les réseaux de neurones, la minimisation de la perte empirique est effectuée en considérant des fonctions erreur qui sont généralement convexes. Ces fonctions permettent l'utilisation de plusieurs algorithmes, comme la descente de gradient, qui s'appuie sur la convexité et la continuité de la fonction qu'il optimise. Les fonctions d'erreur convexes les plus populaires sont les suivantes pour un problème multi-classes à K classes [2] sont:

$$\mathcal{L}_{cross\ \ entropy}(\hat{y}, y) = -\frac{1}{K}\sum_{i=0}^{K} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{1.1}$$

$$\mathcal{L}_{squared}(\hat{y}, y) = \frac{1}{K}\sum_{i=0}^{K} max(0, 1 - y_i \hat{y}_i) \tag{1.2}$$

$$\mathcal{L}_{logistic}(\hat{y}, y) = \frac{1}{K}\sum_{i=0}^{K} log(1 + exp(-y_i \hat{y}_i)) \tag{1.3}$$

où $\hat{y}_i$ représente la prédiction faite par le réseau de neurone d'une observation x_i avec $\hat{y}_i$ est la vraie attendue. La fonction *log* représente la fonction logarithme et *max* la fonction maximum. Dans le cadre de cette thèse, nous utiliserons principalement la fonction d'erreur de la cross-entropy définie en équation (1.1).

[2] Dans le cas binaire, K=2

1.1.3 Classification multi-classes

La classification multi-classes peut être considérée comme une extension de la classification binaire, où le nombre de sorties possibles est strictement supérieur à deux. Formellement, elle désigne toute configuration de classification qui considère un espace de sortie $\boldsymbol{Y} = \{1, ..., K\}$ avec $K > 2$.

Pour résoudre le problème de la classification multi-classes plusieurs approches ont été proposées en réduisant le problème multi-classes à un problème binaire:

(1) Une première solution consiste à utiliser une approche *one-vs-one*. L'idée est de construire un classifieur pour chaque couple de classes afin de les discriminer. Par exemple, si nous considérons un problème multi-classes d'analyse de sentiments de Twitter avec trois classes possibles (Positif, Négatif et Neutre), il faudrait apprendre trois classifieurs pour discriminer les couples suivants:

- Positif versus Négatif;
- Négatif versus Neutre;
- Positif versus Neutre;

Dans cette méthode, on crée $\frac{K(K-1)}{2}$ classifieurs binaires, chaque classifieur apprend une paire de classes spécifique. La prédiction est ensuite effectuée par un vote majoritaire parmi tous les classifieurs. Cette méthode est très performante mais coûteuse en terme de calcul en raison du grand nombre de classifieurs à apprendre surtout pour les cas où K est grand.

(2) Une autre approche classique et moins coûteuse est la méthode *one-vs-rest*. L'idée de cette méthode est d'apprendre K classifieurs binaires, un pour chaque classe de

l'ensemble d'apprentissage. De manière similaire à l'approche *one-vs-one*, la prédiction finale se fait par un vote majoritaire parmi tous les classifieurs. Ainsi, si on reprend l'exemple du problème de l'analyse des sentiments, nous aurons trois classifieurs à apprendre:

- Positif versus (Négatif & Neutre)
- Négatif versus (Positif & Neutre)
- Neutre versus (Positif & Négatif)

Dans cette méthode, K classifieurs sont utilisés avec K le nombre de classes.

1.1.4 Evaluation des modèles

1.1.4.1 Métriques pour la classification binaire

Pour évaluer les performances d'un classifieur binaire, nous définissons les équations suivantes, où VP, VN, FP et FN représentent respectivement les vrais positifs, les vrais négatifs, les faux positifs et les faux négatifs.

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{1.4}$$

$$\text{Précision} = \frac{VP}{VP + FP} \tag{1.5}$$

$$\text{Rappel} = \frac{VP}{VP + FN} \tag{1.6}$$

$$\text{F}_\beta = (1 + \beta^2) \times \frac{\text{Précision} \times \text{Rappel}}{(\beta^2 \times \text{Pécision}) + \text{Rappel}} \tag{1.7}$$

L' *Accuracy* est la capacité du modèle à identifier correctement les observations, tandis que la précision mesure la capacité du modèle à distinguer les observations positives et négatives. Le rappel mesure le nombre de classifications positives trouvées parmi toutes les classifications positives disponibles. On introduit finalement deux autres métriques assez courantes: l'AUC et l'AUPR.

- L'**AUC** *(Area Under Curve)*, représente l'aire sous la courbe ROC[3] *(Receiver Operating Characteristic)*. Elle fournit une mesure agrégée des performances pour tous les seuils de classification possibles. Une façon d'interpréter l'AUC est la probabilité que le modèle classe un exemple positif aléatoire plus haut qu'un exemple négatif aléatoire

- **AUPR** *(Area Under the Precision-Recall)* de la même manière que pour l'AUC associé à la courbe ROC, l'AUPR représente l'aire sous la courbe Précision-Rappel.

1.1.4.2 Métriques pour la classification multi-classes

Lorsqu'il s'agit de classification multi-classes, et contrairement à la classifications binaire, les métriques doivent impliquer toutes les classes. De telles métriques peuvent être calcuées en agrégeant les performances de chaque classe. C'est là qu'interviennent les techniques de calcul de la moyenne.

- **Macro:** Il s'agit d'une moyenne arithmétique de toutes les métriques entre les classes. Cette technique donne un poids égal à toutes les classes, ce qui en fait une bonne option pour les tâches de classification équilibrée.

[3] représente le Rappel par rapport au FPR *(False Positive Rate)* à différents seuils de classification.

- **Micro:** La micro-moyenne est obtenue en divisant la somme de la diagonale de la matrice de confusion par la somme de toutes les valeurs de la matrice de confusion. Elle correspond à l'accuracy.

- **Pondérée:** Cette méthode tient compte du déséquilibre des classes en calculant la moyenne des mesures binaires pondérées par le nombre d'échantillons de chaque classe dans la cible. La moyenne pondérée sera calculée en multipliant chaque score par le nombre d'occurrences de chaque classe et en divisant par le nombre total d'échantillons.

Nous définissons enfin deux dernières métriques assez courantes pour la classification multi-classes:

- **AUNU** *(AUC of each class against the rest, using the uniform class distribution)*: est la moyenne harmonique des AUC de chaque classe.

- **AUNP** *(AUC of each class against the rest, using the a priori class distribution)*: est la moyenne pondérée des AUC de chaque classe.

1.2 Les plongements lexicaux

La création d'*embeddings* de mots ou plongements lexicaux[4] vise à capturer la signification des unités lexicales et leurs relations sémantiques. L'*embedding* des mots est une approche utilisée pour fournir une représentation vectorielle dense des mots qui capture leur contexte. Les modèles présentés dans cette section sont des versions améliorées de modèles discrets de sacs de mots (*bag-of-words*), comme les compteurs de mots et les

[4]A partir d'ici, nous utiliserons le mot "embedding" pour caractériser un plongement lexical.

compteurs de fréquence. Nous présenterons particulièrement trois modèles des *embeddings* non contextuels de l'état de l'art: (1) le modèle skip-gram (Mikolov et al. 2013), (2) le modèle CBOW (Mikolov et al. 2013) et (3) le modèle GloVe (Pennington et al. 2014).

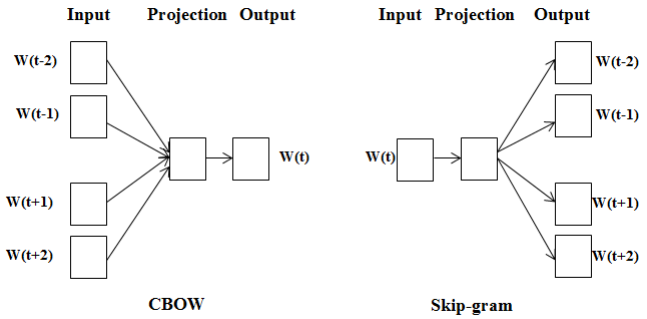

Figure 1.1: Modèles skip-gram et CBOW.

1.2.1 Le modèle général word2vec

Il existe deux modèles de word2vec (Mikolov et al. 2013): le skip-gram et le CBOW. Ils parviennent tous les deux à capturer les interactions entre un mot centré et ses mots contextuels. Cependant, ils le font différemment, et en quelque sorte de manière opposée. Alors que le skip-gram modélise la distribution des mots contextuels en fonction du mot centré, CBOW s'intéresse à la prédiction du mot centré en fonction de son contexte.

Dans cette section, nous ne ferons aucune distinction dans la notation entre un mot et son vecteur correspondant. Par exemple, w désigne à la fois le mot et son vecteur de mots.

Étant donné un vecteur de mot prédit $\hat{r}$ et un vecteur de mot cible w_t (selon son contexte), la probabilité du mot cible conditionnellement au mot prédit est calculée par

une fonction softmax:

$$P(w_t|\hat{r}) = \frac{exp(w_t^T \hat{r})}{\sum_{w \in W} exp(w^T \hat{r})} \tag{1.8}$$

où W est l'ensemble de tous les vecteurs de mots cibles.

Remarquez que $\hat{r}$ n'est ni un élément de W ni calculé à partir d'éléments de W. Comme nous le verrons plus tard, les éléments de W seront appelés *vecteurs de sortie* et $\hat{r}$ sera calculé à partir d'un ensemble différent constitué de *vecteurs d'entrée*.

Les fonctions de coût des modèles word2vec (pour un mot cible) minimisent la log-vraisemblance négative du vecteur mot cible étant donné le mot prédit correspondant:

$$\mathcal{L}(w_t, \hat{r}) = -\log P(w_t|\hat{r}) = \log \left(\sum_{w \in W} exp(w^T \hat{r}) \right) - w_i^T \hat{r} \tag{1.9}$$

Le gradient par rapport à w de $\mathcal{L}(w_t, \hat{r})$ que nous allons noter g_1 est:

$$g_1(w, w_t, W, \hat{r}) = \frac{\partial}{\partial w} \mathcal{L}(w_t, \hat{r}) = \hat{r}(P(w|\hat{r}) - I\{w = w_t\}) \tag{1.10}$$

où $I\{.\}$ est la fonction indicatrice.

Le gradient par rapport à $\hat{r}$ que nous allons noter g_2 est:

$$g_2(w_t, W, \hat{r}) = \frac{\partial}{\partial \hat{r}} \mathcal{L}(w_t, \hat{r}) = \sum_{w \in W} [P(w|\hat{r})w] - w_t \tag{1.11}$$

Le calcul des gradients est primordial pour la mise à jour des poids du réseau de neurones grâce à la méthode de rétropropagation du gradient que nous allons développer dans les sections suivantes.

1.2.1.1 Le modèle skip-gram

La figure 1.1 illustre le modèle utilisé pour apprendre les *embeddings* de mots par skip-gram. L'entrée du réseau est un mot, tandis que la sortie du réseau est une couche softmax sur les mots d'un corpus. Il s'ensuit que la couche de sortie est de dimensionnalité équivalente à V[5]. La tâche utilisée pour l'apprentissage de l'*embedding* des mots est la suivante: étant donné un mot, maximiser les probabilités des mots dans leur contexte. Le contexte (également appelé fenêtre) comprend jusqu'à c mots avant et après l'occurrence du mot d'entrée dans le texte. En raison de la forte dimensionnalité de la sortie, son calcul analytique est coûteux.

Pour un indice i et une taille de fenêtre c, le modèle skip-gram prédit les mots de contexte $\{w_j\}$, $(i - c \leq j \leq i + c, j \neq i)$ étant donné le mot centré r_i. Par conséquent, dans le modèle général, $w_t = w_j$ et $\hat{r} = r_i$ pour ce cas.

La fonction de coût est dérivée comme suit:

$$\mathcal{L}_{skipgram}(c, i) = \sum_{i-c \leq j \leq i+c, i \neq j} -\log P(w_j | r_i) \tag{1.12}$$

Les gradients de cette fonction sont:

$$\frac{\partial}{\partial w} \mathcal{L}_{skipgram}(c, i) = r_i \sum_{i-c \leq j \leq i+c, i \neq j} g_1(w, w_j, W, r_i) \tag{1.13}$$

$$\frac{\partial}{\partial r_i} \mathcal{L}_{skipgram}(c, i) = \sum_{i-c \leq j \leq i+c, i \neq j} g_2(w_j, W, r_i) \tag{1.14}$$

[5]On considère que nous disposons d'un vocabulaire de dimension V.

1.2.1.2 Le modèle CBOW

Intuitivement, ce modèle inverse le mécanisme de modélisation du skip-gram. CBOW prédit un mot en fonction de son contexte. Le vecteur du mot cible est maintenant le vecteur de sortie du mot à l'indice i, w_i ; le vecteur du mot prédit est la somme de tous les vecteurs d'entrée du contexte:

$$\hat{r} = \sum_{i-c \leq j \leq i+c, i \neq j} r_j \tag{1.15}$$

La fonction de coût du CBOW est la suivante:

$$\mathcal{L}_{CBOW}(c, i) = -\log P(w_i|\hat{r}) \tag{1.16}$$

et les gradients sont les suivants:

$$\frac{\partial}{\partial w}\mathcal{L}_{CBOW}(c, i) = g_1(w, w_i, W, \hat{r}) \tag{1.17}$$

$$\frac{\partial}{\partial r_j}\mathcal{L}_{CBOW}(c, i) = g_2(w_i, W, \hat{r}) \tag{1.18}$$

pour $i - c \leq j \leq i + c, i \neq j$. Sinon,

$$\frac{\partial}{\partial r_j}\mathcal{L}_{CBOW}(c, i) = 0 \tag{1.19}$$

1.2.2 Le modèle GloVe

GloVe est un modèle d'*embedding* de mots plus récent développé par (Pennington et al. 2014). Ce modèle a été comparé directement à word2vec. La version finale de l'article affirmait que GloVe surpassait toujours word2vec dans la tâche d'analogie des mots si on le laissait fonctionner suffisamment longtemps.

Concrètement, soit X la matrice d'occurrences du corpus où l'entrée (i,j) est le nombre de fois où le mot i et le mot j cooccurrent. Soit $P_{ij} = \frac{X_{ij}}{X_i}$ la probabilité que le mot i et le mot j cooccurrent. La relation entre le mot i et le mot j est déterminée sur la base du rapport des probabilités de leurs cooccurrences avec un mot de contexte k modélisée par une fonction F:

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{P_{ik}}{P_{jk}} \tag{1.20}$$

Puisque dans la matrice d'occurrence, les rôles des lignes et des colonnes sont interchangeables, nous appliquons la symétrie à l'équation ci-dessus:

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{F(w_i^T w_k^{'})}{F(w_j^T w_k^{'})} \tag{1.21}$$

D'après l'équation 1.20 et l'équation 1.21, nous avons:

$$F(w_i^T w_k^{'}) = P_{ik} = \frac{X_{ik}}{X_i} \tag{1.22}$$

La solution à l'équation 1.20 est $F = Exp$. Ainsi:

$$w_i^T w_k^{'} = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{1.23}$$

À partir de l'équation 1.23, nous définissons la fonction objectif des moindres carrés pondérés suivante:

$$J = \sum_{i,j} f(X_{ij})(w_i^T w_j^{'} + b_i + b_j^{'} - \log X_{ij}) \tag{1.24}$$

où $f(.)$ est une fonction de poids et $b, b^{'}$ sont des termes de biais.

Rappelons la fonction objectif de formation de skip-gram:

$$J = -\sum_i \sum_{j \in context(i)} \log Q_{ij} \tag{1.25}$$

où $Q_{ij} = \frac{\exp(w_i^T w_j^{'})}{\sum_k exp(w_i^T w_k^{'})}$, qui peut être réécrit comme:

$$J = -\sum_{i,j} X_{i,j} \log Q_{i,j} = \sum_i X_i H(P_i, Q_i) \tag{1.26}$$

où H est l'entropie croisée entre deux distributions.

Le modèle GloVe est plus avancé que l'équation 1.26 de deux façons:

(a) La fonction de poids n'est pas X_i mais une fonction arbitraire $f(X_{ij})$.

(b) L'entropie croisée présente plusieurs inconvénients. Premièrement, elle attribue une masse de probabilité plus importante à un événement peu probable. Deuxièmement, le calcul de la constante de normalisation implique une sommation sur l'ensemble du

vocabulaire, ce qui est coûteux. Pour résoudre ces problèmes, le modèle GloVe choisit une perte quadratique à la place.

1.3 Les réseaux de neurones

1.3.1 Perceptron multi-couche

Nous voulons considérer un réseau de neurones assez général composé de L couches (bien sûr sans compter la couche d'entrée). Considérons une couche arbitraire, disons ℓ, qui possède N_ℓ neurones $X_1^{(\ell)}$, $X_2^{(\ell)}$, ..., $X_{N_\ell}^{(\ell)}$, chacune avec une fonction d'activation $f^{(\ell)}$. Remarquez que la fonction d'activation peut être différente d'une couche à l'autre. La fonction d'activation peut être donnée par toute fonction différentiable, mais n'a pas besoin d'être linéaire. Ces cellules reçoivent des signaux des neurones de la couche précédente, $\ell-1$. Par exemple, le neurone $X_j^{(\ell)}$ reçoit un signal de $X_i^{(\ell-1)}$, avec un facteur de pondération de $w_{ij}^{(\ell)}$. Nous avons donc une matrice de poids de $N_{\ell-1}$ par N_ℓ, $\mathbf{W}^{(\ell)}$, dont les éléments sont donnés par $W_{ij}^{(\ell)}$, pour i=1, 2, ..., $N_{\ell-1}$ et $j = 1, 2, \ldots, N_\ell$. Le neurone $X_j^{(\ell)}$ a également un biais donné par $b_j^{(\ell)}$, et son activation est de $a_j^{(\ell)}$.

Pour simplifier la notation, nous utiliserons $n_j^{(\ell)} (= y_j)$ pour désigner l'entrée nette dans le neurone $X_j^{(\ell)}$. Elle est donnée par

$$n_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}, \qquad j = 1, 2, \ldots, N_\ell. \tag{1.27}$$

Ainsi, l'activation du neurone $X_j^{(\ell)}$ est la suivante

$$a_j^{(\ell)} = f^{(\ell)}(n_j^{(\ell)}) = f^{(\ell)}(\sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}). \tag{1.28}$$

Nous pouvons considérer la couche zéro comme la couche d'entrée. Si un vecteur d'entrée $\mathbf{x}$ a N composantes, alors $N_0 = N$ et les neurones de la couche d'entrée ont des activations $a_i^{(0)} = x_i, i = 1, 2, \ldots, N_0$.

La couche L du réseau est la couche de sortie. En supposant que le vecteur de sortie $\mathbf{y}$ a M composantes, alors nous devons avoir $N_L = M$. Ces composantes sont données par $y_j = a_j^{(L)}, j = 1, 2, \ldots, M$.

Pour tout vecteur d'entrée donné, les équations ci-dessus peuvent être utilisées pour trouver l'activation pour chaque neurone pour tout ensemble donné de poids et de biais. En particulier, le vecteur de sortie du réseau $\mathbf{y}$ peut être trouvé. La question restante est de savoir comment entraîner le réseau à trouver un ensemble de poids et de biais afin qu'il puisse effectuer une certaine tâche.

L'algorithme de rétropropagation pour un réseau de neurones multi-couches est illustré en annexe A.

1.3.2 Encodage des séquences

Dans cette section, nous décrivons une méthode permettant d'apprendre une représentation de phrase (une série de mots) en fonction de la tâche étudiée. Cette méthode consite à apprendre un réseau de neurones profond appelé LSTM *(Long short-term memory)* (Hochreiter and Schmidhuber 1997). Pour détailler ces réseaux LSTM pour la suite, les

indices indiquent les pas de temps et les exposants indiquent les couches. Tous nos états sont de dimension n. Soit $h_t^l \in \mathbb{R}^n$ un état caché dans la couche l pour le pas de temps t. Soit $T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ une transformation affine ($Wx + b$). Soit $\odot$ une multiplication par éléments et soit h_t^0 un *embedding* de mot au pas de temps t. Nous utilisons les activations h_t^L pour prédire y_t, puisque L est le nombre de couches dans notre LSTM profond.

Les LSTM sont l'amélioration des réseaux RNN (réseaux de neurones récurrents) (Williams and Zipser 1989) pour lesquels les transitions déterministes entre les états cachés précédents et actuels sont modélisés comme suit:

$$\text{RNN} : h_t^{l-1}, h_{t-1}^l \to h_t^l \tag{1.29}$$

Pour les RNN classiques, la fonction de transition déterministe est donnée par:

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ où } f \in \{\text{sigm}, \tanh\} \tag{1.30}$$

et,

$$\text{sigm}(x) = \frac{1}{1 + e^{-x}} \tag{1.31}$$

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1.32}$$

Les LSTM disposent de transitions plus compliquées qui lui permettent de mémoriser des informations pendant un grand nombre de pas de temps. Cette mémoire est stockée dans un vecteur appelé *cellules de mémoire*, $c_t^l \in \mathbb{R}^n$. Bien que de nombreuses architectures LSTM diffèrent par leur structure de connectivité et leurs fonctions d'activation,

toutes les architectures LSTM possèdent des cellules de mémoire explicites pour stocker des informations pendant de longues périodes de temps. Le LSTM peut décider d'écraser la cellule mémoire, de la récupérer ou de la conserver pour le prochain pas de temps. L'architecture LSTM utilisée dans nos expériences est donnée par les équations suivantes (Graves et al. 2013) :

$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \tag{1.33}$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{1.34}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g \tag{1.35}$$

$$h_t^l = o \odot \tanh(c_t^l) \tag{1.36}$$

Dans ces équations, sigm et tanh sont appliqués par élément. La figure 1.2 illustre les LSTM et ses équations.

1.3.3 Les modèles du langage contextualisés

Les représentations contextualisées des mots tiennent compte des limites des *embeddings* de mots non contextualisés. Des travaux récents ont tenté de créer des représentations prenant en compte le contexte. ELMo (Peters et al. 2018), BERT (Devlin et al. 2019) et USE (Cer et al. 2018) sont des modèles de langage neuronaux profonds qui sont finement ajustés pour créer des modèles pour un large éventail de tâches. Leurs

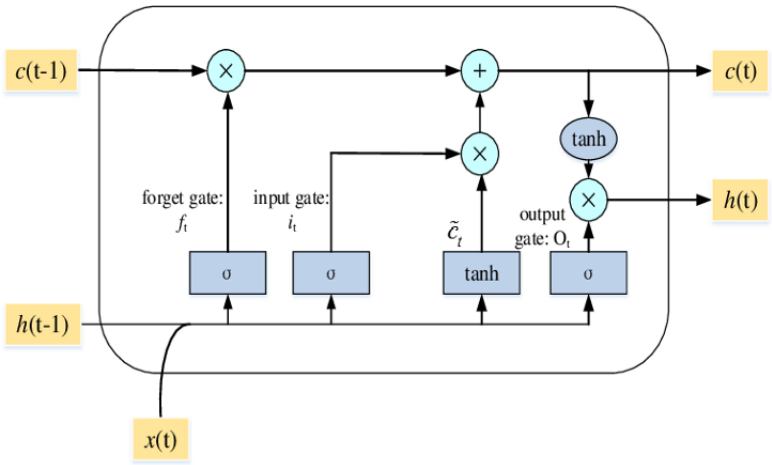

Figure 1.2: Architecture LSTM.

représentations internes des mots sont contextualisées car elles dépendent de l'ensemble de la phrase en entrée. Les représentations contextuelles ont atteint des performances intéressantes sur diverses tâches du traitement automatique du langage. Dans le cadre de la détection des relations lexico-sémantiques entre deux mots, ces *embeddings* contextuels vont être utilisés pour la représentation des patrons lexicaux reliant les deux mots étudiés.

La simplicité est la sophistication suprême.

Léonard De Vinci

2

Architectures multi-tâches pour l'identification des relations lexico-sématiques

2.1 Motivation

L'identification des relations lexico-sémantiques entre deux mots est une tâche d'une importance cruciale pour le traitement du langage naturel. À cette fin, il existe différentes méthodologies qui se concentrent sur une seule relation lexicale (c'est-à-dire la classification binaire), identifient plusieurs relations sémantiques (c'est-à-dire la classification multi-classes) ou apprennent simultanément des relations lexico-sémantiques liées cognitivement (c'est-à-dire la classification multi-tâches). La plupart des approches existantes se concentrent sur les architectures de classification binaire et multi-classes, bien que des études récentes montrent que la classification multi-tâches peut améliorer les performances de classification. Cependant, dans ce dernier cas, des architectures d'apprentissage totalement partagées *fully-shared* sont proposées et reposent sur des représentations basiques des mots. Dans ce chapitre, nous proposons et étudions des architectures neuronales multi-tâches partagées et privées *shared-private* qui combinent des représentations d'entrée continues distributives et basées sur des patrons lexicaux,

et qui se concentrent spécifiquement sur les caractéristiques des mots uniques.

2.2 Un aperçu de la littérature

L'identification des relations lexico-sémantiques a été abordée par différentes stratégies d'apprentissage: la classification binaire (Snow et al. 2004, Roller et al. 2014, Weeds et al. 2014, Shwartz et al. 2016, Nguyen et al. 2017a, Glavaš and Ponzetto 2017, Nguyen et al. 2017b), la classification multi-classes (Shwartz and Dagan 2016a, Santus et al. 2016a) et la classification multi-tâches (Attia et al. 2016, Balikas et al. 2019). Dans tous les cas, deux stratégies majeures ont été abordées pour capturer la sémantique entre deux mots: (1) la méthode basée sur les patrons lexicaux, et (2) l'approche distributionnelle.

La première approche repose sur l'analyse des patrons lexico-syntaxiques qui existent entre deux mots (par exemple, x **est un type de** y). Cette direction de recherche a été initiée par (Hearst 1992), qui définit manuellement des patrons spécifiques identifiant avec précision les relations sémantiques telles que l'hypernoymie. Cette direction de recherche est encore activement étudiée, notamment pour l'identification de l'hypernoymie (Roller et al. 2018). Des études ont également été proposées pour apprendre automatiquement de tels modèles (Snow et al. 2004; 2006, Kozareva and Hovy 2010). Dans ce cas, les paires de mots sont représentées comme un ensemble de tous leurs patrons, qui sont ensuite utilisés pour la classification. En tant que tels, les patrons pertinents sont ceux qui se voient attribuer des poids d'apprentissage élevés par le classifieur. Il est communément admis que les patrons lexico-syntaxiques présentent une précision élevée mais un rappel insuffisant car la matrice de l'espace des caractéristiques est creuse. Des études récentes (Shwartz et al. 2016, Nguyen et al. 2017b) proposent de

surmonter cette limitation en représentant les patrons lexicaux sous forme de vecteurs continus à l'aide de modèles neuronaux tels que les BiLSTM. Dans ce contexte, des résultats probants ont été obtenus prouvant l'efficacité de la généralisation de ces modèles pour le cas spécifique de l'identification d'une seule relation sémantique ((Shwartz et al. 2016) pour l'hypernoymie et (Nguyen et al. 2017b) pour l'antonymie). Nous proposons donc d'inclure des représentations continues des patrons lexicaux comme caractéristiques d'apprentissage pour la détection des relations lexico-sémantiques.

Contrairement aux travaux connexes qui s'attaquent à une seule relation sémantique, nous proposons un modèle d'encodage unique basé sur les réseaux de neurones BiLSTM qui représente tous les patrons lexicaux indépendamment de la relation sémantique abordée. L'idée sous-jacente est que dans un paradigme multi-tâches, si les tâches sont cognitivement liées, il est probable que certains patrons partagent de l'information commune avec différentes relations sémantiques. En particulier, nous démontrerons dans la section 2.5 qu'un tel encodage est relativement puissant lorsque les patrons lexico-syntaxiques sont peu fréquents pour une relation donnée.

La deuxième stratégie pour capturer la relation sémantique entre deux mots est basée sur l'hypothèse distributionnelle de Harris (Harris 1954). Dans ce contexte, la décision de savoir si un mot x est en relation avec un mot y est basée sur la représentation distributionnelle de chaque mot. Ainsi, les paires de mots peuvent être représentées par la concaténation des représentations individuelles des mots ((Baroni et al. 2012, Roller et al. 2014, Weeds et al. 2014) pour le cas discret et (Mikolov et al. 2013, Shwartz et al. 2016) pour le cas continu) ou par leur différence vectorielle ((Weeds et al. 2004) pour le cas discret et (Fu et al. 2015, Vylomova et al. 2016) pour le cas continu). En par-

ticulier, (Shwartz et al. 2016, Nguyen et al. 2017b) montrent que la combinaison des méthodes basées sur les patrons lexicaux avec l'approche distributionnelle améliore considérablement les performances de classification dans les espaces latents. Cependant, elles ne tiennent pas compte de la similarité sémantique entre les mots individuels comme le proposent (Weeds et al. 2014). En effet, la similarité cosinus entre les représentations de mots individuels est généralement un bon indicateur de classification. Contrairement aux travaux connexes qui n'abordent pas toutes les caractéristiques d'apprentissage possibles, nous proposons d'étudier l'influence de toutes les caractéristiques d'apprentissage distributionnelles (c'est-à-dire la concaténation, la différence vectorielle et le cosinus) en combinaison avec l'encodage continu de patrons.

Dans le paradigme de l'apprentissage multi-tâches, (Attia et al. 2016) et (Balikas et al. 2019) sont les deux seules études répertoriées à ce jour[1]. En particulier, (Attia et al. 2016) proposent un réseau de neurones convolutif multi-tâches où une tâche agit comme une adaptation au domaine (classification de la parenté entre deux mots) et la seconde tâche est un problème de classification multi-classes pour quatre relations sémantiques: hypernoymie, méronymie, synonymie et antonymie. D'autre part, (Balikas et al. 2019) proposent une méthodologie à grain fin qui vise à déterminer si le processus d'apprentissage d'une relation sémantique donnée peut être amélioré par l'apprentissage simultané d'une autre relation, où les relations sont la synonymie, la co-hyponymie, l'hypernoymie et la méronymie. En particulier, ils proposent un modèle de réseau de neurones à partage de paramètres dur *fully-shared*, où une paire de mots est encodée en utilisant la concaténation des plongements lexicaux de mots re-

[1]Au meilleur de notre connaissance.

spectifs. Les inconvénients communs de ces deux études sont les suivants: (1) elles proposent toutes des architectures *fully-shared* et ne tirent pas parti des modèles multi-tâches partagés-privés plus puissants; (2) elles ne s'appuient que sur l'approche distributionnelle pour encoder les paires de mots (en particulier, seulement la concaténation des représentations de mots individuels) et ne bénéficient pas de la combinaison fructueuse des représentations continues distributionnelle et celles basées sur les patrons lexicaux (Shwartz et al. 2016, Nguyen et al. 2017b); et (3) elles ne s'attaquent pas à la spécificité des relations asymétriques. En effet, alors que la synonymie, la co-hyponymie et l'antonymie sont des relations symétriques, l'hypernoymie et la méronymie sont des relations asymétriques qui ne sont généralement traitées qu'avec des caractéristiques d'apprentissage symétriques. Par conséquent, nous proposons de concevoir des architectures neuronales qui incluent des caractéristiques asymétriques spécifiques (c'est-à-dire des architectures conscientes des mots uniques) combinées à des encodages de patrons asymétriques par définition. Comme le code et les jeux de données de (Balikas et al. 2019) sont disponibles pour la reproductibilité[2], nous testons nos hypothèses dans ce même contexte.

2.3 Architecture d'apprentissage

Dans cette section, nous présentons la méthodologie globale et les décisions de conception de notre approche. Nous commençons par définir les caractéristiques d'entrée et nous poursuivons en décrivant l'architecture du réseau neuronal.

[2] https://github.com/Houssam93/MultiTask-Learning-NLP

2.3.1 Représentations continues

Nous proposons d'étudier trois caractéristiques distributionnelles de paires de mots différentes: la concaténation, la différence vectorielle et le cosinus. Soit (w_1, w_2) une paire de mots et w_1, w_2 leurs représentations distributionnelles respectives (ici GloVe (Pennington et al. 2014)) de dimension d. Les caractéristiques distributionnelles d'entrée de la paire de mots (w_1, w_2) seront notées comme: $w_1 \oplus w_2$ pour la concaténation, $w_1 \ominus w_2$ pour la différence vectorielle et $cos(w_1, w_2)$ pour la mesure de similarité cosinus telle que définie dans l'équation 2.1.

$$cos(w_1, w_2) = \frac{\sum_{i=1}^{d} w_1^i \times w_2^i}{\sqrt{\sum_{i=1}^{d} {w_1^i}^2} \times \sqrt{\sum_{i=1}^{d} {w_2^i}^2}} \tag{2.1}$$

2.3.2 Représentation des patrons lexicaux

Les patrons lexicaux peuvent être définis comme des séquences de mots qui se produisent entre deux mots (w_1, w_2) dans un intervalle de 10 mots maximum de la même phrase.[3] Le tableau 2.1 montre des exemples de patrons lexicaux entre paires de mots pour chacune des relations sémantiques étudiées. On peut remarquer que la simple syntaxe ou même l'absence de patron entre les deux mots peut donner une indication sur la nature de la relation sémantique. Par exemple, le mot *or* dans la phrase *error or fault* fournit une indication de similarité entre *error* et *fault*. Mais, toutes les séquences ne sont pas des patrons lexicaux et certaines séquences peuvent lier des mots de manière erronée. C'est le cas dans la phrase *burning fuel in the combustion*, où *burning* et *com-*

[3] Ce paramètre a été ajusté expérimentalement.

bustion sont liés à tort. Pour atténuer cet effet, nous utilisons la fréquence comme indicateur de l'informativité du patron. Nous avons constaté que la conservation des k patrons les plus fréquents pour chaque paire de mots améliore les résultats par rapport à l'échantillonnage aléatoire des patrons pour chaque paire. Ici, k est un hyper-paramètre.

De manière similaire à (Shwartz et al. 2016), nous transformons les patrons qui relient une paire de mots (w_1, w_2) en vecteurs continus à l'aide d'un réseau neuronal BiLSTM. Nous appelons ce réseau $BiLSTM^{Shared}$ car il est partagé entre différentes tâches. Notre intuition est que le réseau BiLSTM peut réussir à encoder simultanément l'information universelle des patrons qui relieraient les mots de M tâches conjointes. Formellement, étant donné le i-ième patron le plus fréquent entre (w_1, w_2), chacun de ses mots est représenté par son *embedding* GloVe pré-entraîné et l'ensemble des *embeddings* est introduit dans le $BiLSTM^{Shared}$. Pour s'assurer que chaque séquence ait la même longueur, nous utilisons la mise à zéro[4] de chaque patron afin que sa longueur soit de 10 mots. La sortie du $BiLSTM^{Shared}$ pour le i-ième patron le plus fréquent est notée $h_i^{w_1,w_2}$ et définie dans l'équation 2.2, où (w_1, PT^i, w_2) est la séquence d'*embedding* de mots commençant par l'*embedding* de x suivi des *embeddings* des mots du i-ième chemin et du mot y, notés w_1, PT^i et w_2 respectivement:

$$h_i^{w_1,w_2} = BiLSTM_i^{Shared}(w_1, PT^i, w_2) \;\; \forall i, \;\; 1 \leq i \leq k \tag{2.2}$$

Pour obtenir une représentation unique des k patrons d'une paire de mots, nous faisons la moyenne des k représentations $h_i^{w_1,w_2}$ comme indiqué dans l'équation 2.3.

[4] zero-padding

$$h_*^{w_1,w_2} = Average(h_1^{w_1,w_2}, ..., h_k^{w_1,w_2}) \tag{2.3}$$

Relation	Patron lexical
Synonymie	**error** or **fault** **change** as an **alteration** **burning** fuel in the **combustion**
Hyperonymie	**aircraft** firing rocket into an enemy **plane** **unit** that includes **screen** **act** was an unconscious **ritual**
Co-hyponymie	**pineapple** and **apricot** **chisel** usually used with **mallet** **horse** frightened by **lion**
méronymie	**bowl** from the world of **glass** **television** and **video** **couch** on **seat**
Aléatoire	**reference** in the book of **mormon** **nothing** to stop the **robber** **driver** was issued traffic **ticket**

Table 2.1: Exemples de patrons lexicaux entre deux mots (en gras).

2.3.3 L'architecture *shared-private*

L'architecture neuronale globale est présentée dans la Figure 2.1. L'architecture consiste en une seule partie partagée et plusieurs parties privées. La partie partagée sert à encoder des informations générales pour la paire de mots d'entrée. Cependant, on ne peut pas s'attendre à ce que ces informations générales soient suffisantes. Par conséquent, nous examinons si l'ajout de couches d'encodage privées (par rapport à une tâche) peut améliorer les performances. Dans les deux sous-sections suivantes, nous

présentons l'architecture en détail.

2.3.3.1 L'architecture générale

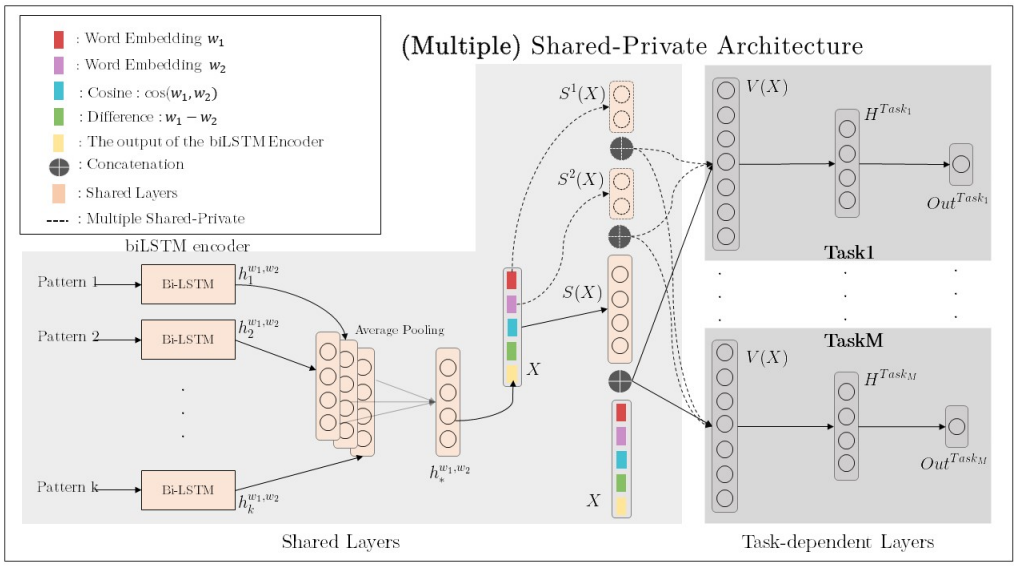

Figure 2.1: L'architecture générique partagée privée *shared-private*. Différentes configurations de $V(X)$ permettent de proposer des architectures tout partagé *fully-shared* pour M tâches.

Une paire de mots (w_1, w_2) est représentée par ses représentations distributionnelles et ses représentations basées sur des patrons. Formellement, l'entrée d'une instance peut être définie comme le vecteur X suivant:

$$X = (w_1 \oplus w_2, w_1 \ominus w_2, cos(w_1, w_2), h_*^{w_1,w_2}) \tag{2.4}$$

(Balikas et al. 2019) a montré l'efficacité d'une couche partagée lors de l'apprentissage simultané des fonctions de décision pour plusieurs tâches. Par conséquent, nous proposons d'encoder cette information dans notre architecture neuronale. La couche partagée

S encode l'information commune qui existe entre plusieurs tâches pour toutes les représentations d'entrée, comme défini dans l'équation (2.5):

$$S(X) = \sigma(W_S X + b_S) \tag{2.5}$$

La couche ci-dessus est partagée entre les tâches. Par conséquent, elle encode des informations génériques des mots, ce qui peut être considéré comme une limitation, en particulier pour les relations asymétriques où l'ordre des mots dans la paire d'entrée a de l'importance. Pour pallier à cette limitation, nous ajoutons deux couches supplémentaires partagées, une pour chaque mot d'entrée, c'est-à-dire S^1 pour w_1 et S^2 pour w_2. De cette façon, nous faisons l'hypothèse que les relations asymétriques peuvent être mieux modélisées si l'on tient compte explicitement de l'information asymétrique. L'idée sous-jacente est que les mots individuels peuvent contenir des informations utiles uniques pour décider si deux mots sont sémantiquement liés (par exemple, leur degré de généralité). Ces couches supplémentaires tenant compte d'un seul mot sont définies dans l'équation 2.6:

$$S^i(X) = \sigma(W_S^i w_i + b_S^i), \quad \forall i, \quad 1 \leq i \leq 2 \tag{2.6}$$

Notez que $S(X)$, $S^1(X)$ et $S^2(X)$ sont partagées pour toutes les tâches. Cependant, (Liu et al. 2017) ont démontré que des performances de classification accrues peuvent être atteintes si toutes les tâches concurrentes reçoivent des représentations à la fois partagées et privées. De tels modèles neuronaux sont appelés architectures partagées-privées. Nous étendons notre architecture afin d'intégrer des représentations privées

par tâche.

Un vecteur $V(X) = (S(X), S^1(X), S^2(X), X)$ représentant la concaténation des trois représentations partagées $S(X), S^1(X)$ et $S^2(X)$, plus la représentation d'entrée initiale X est alimentée par deux couches dépendantes de la tâche, à savoir H^{T_j} et Out^{T_j} respectivement définies dans les équations 2.7 et 2.8 pour une tâche spécifique T_j. Ces couches de sortie dépendent de la tâche, ce qui, dans les équations 2.7 et 2.8, est représenté par les identifiants de la tâche T_j.

$$H^{T_j} = \sigma(W_H^{T_j} V(X) + b_H^{T_j}) \tag{2.7}$$

$$Out^{T_j} = \sigma(W_O^{T_j} H^{T_j} + b_O^{T_j}) \tag{2.8}$$

Les paramètres du modèle de réseau neuronal sont mis à jour en minimisant la fonction d'entropie croisée binaire $\mathcal{L}(T_j)$ définie dans l'équation 2.9. Nous utilisons un apprentissage par mini-batch et b est la taille du batch. La fonction de perte prend en entrée Out^{T_j} et y^{T_j}, où $y^{T_j} = 1$ si x et y sont liés à T_j et $y^{T_j} = 0$ sinon.

$$\mathcal{L}(T_j) = -\frac{1}{b} \sum_{i=1}^{b} y_i^{T_j} . \log(Out_i^{T_j}) + (1 - y_i^{T_j}) . \log(1 - Out_i^{T_j}) \tag{2.9}$$

Concrètement, les paramètres partagés des couches partagées (tels que W_S et W_S^i dans les équations 2.5 et 2.6) sont mis à jour en minimisant $\mathcal{L}(T_j)$ en alternance avec les autres tâches. Alors que les paramètres privés des couches privées (tels que $W_H^{T_j}$ et $W_O^{T_j}$ dans les équations 2.7 et 2.8) sont uniquement mis à jour en minimisant $\mathcal{L}(T_j)$ pour leurs exemples d'apprentissage spécifiques. Ces étapes d'apprentissage sont également

détaillées dans l'algorithme 1.

Algorithm 1: Le processus d'entraînement de l'architecture *shared-private*

```
Data: La paire de mot de chaque M tâche, taille du batch b, itérations
it = 1 ;
while it < itérations do
    for i = 0; i < M; i = i + 1 do
        sélectionner aléatoirement un batch de taille b for T_i nommée batch_i
        for (w_1, w_2) in batch_i do
            calculer Out^{T_i}
        end
        Mettre à jour les paramètres en fonction de E(T_i) du batch_i ;
        Calculer la performance sur l'ensemble de validation de T_i.
    end
end
```

2.3.3.2 Architéctures spécifiques

A partir du modèle générique, nous pouvons générer différentes architectures en changeant les composantes du vecteur $V(X)$. Dans ce travail, nous expérimentons trois architectures de ce type.

Premièrement, le modèle *fully-shared* proposé par (Balikas et al. 2019) peut être défini en fixant $V(X) = S(X)$. Dans ce cas, toute l'information de l'entrée est partagée, et c'est la seule information utilisée pour la classification.

La deuxième architecture que nous évaluons est le modèle *single shared-private*. Dans ce cas, les représentations partagées et privées sont utilisées pour la classification, mais sans information sur les mots individuels. Dans ce cas, $V(X) = (S(X), X)$.

La troisième architecture est appelée le modèle *multiple shared-private* et repose sur la définition du vecteur $V(X)$ suivant: $V(X) = (S(X), S^1(X), S^2(X), X)$. Dans cette architecture, des représentations partagées et privées sont utilisées pour le processus de

décision. Ces représentations comprennent également des parties dédiées à l'intégration des mots uniques en entrée. Nous nous attendons à ce que le fait d'avoir des parties du réseau de neurones qui traitent explicitement chaque mot de l'entrée améliore les performances sur les relations asymétriques. En tant que tel, il s'agit du seul modèle conscient d'un seul mot, qui peut améliorer la classification des relations asymétriques.

Dans le contexte de notre évaluation dans la section 2.5.1.4, nous proposons une architecture spécifique, où les patrons lexicaux ne sont pas partagés par le réseau de neurones BiLSTM. L'idée sous-jacente est que les patrons lexicaux peuvent dépendre de la relation et ne devraient être utilisés que comme couches privées. Par conséquent, le vecteur d'entrée X donné dans l'équation 2.4 serait supprimé de son argument $h_*^{w_1,w_2}$, de sorte que $X = (w_1 \oplus w_2, w_1 \ominus w_2, cos(w_1, w_2))$. En conséquence, $h_*^{w_1,w_2}$ serait directement concaténé au vecteur $V(X)$, tel que $V(X) = (S(X), S^1(X), S^2(X), X, h_*^{w_1,w_2})$.

2.4 Expériences

Dans cette section, nous présentons les détails d'implémentation ainsi que les jeux de données utilisés pour la classification.

2.4.1 Jeux de données

Comme indiqué dans la section 2.2, il existe un grand nombre de travaux connexes pour l'identification des relations lexico-sémantiques. Le premier jeu de données de référence, Weeds, a été proposé par (Weeds et al. 2004) dans le cadre d'études sur les mesures de similarité lexicale. Dans le même but, (Baroni and Lenci 2011) a introduit le célèbre jeu de données Bless, et (Santus et al. 2016b) a compilé le jeu de données

ROOT9 [5], qui contient des paires de mots extraits aléatoirement d'EVALution (Santus et al. 2015), de Lenci/Benotto (Benotto 2015) et de Bless (Baroni and Lenci 2011). Dans le cadre de l'apprentissage concurrent des relations lexico-sémantiques, (Balikas et al. 2019) a récemment introduit le jeu de données RUMEN[6] pour inclure la synonymie. Tous les jeux de données sont résumés avec leurs caractéristiques spécifiques dans le tableau 2.2.

En particulier, pour chaque jeu de données, nous indiquons le pourcentage de paires de mots contenant soit 0. 1, 2 ou plus de 2 patrons. Nous verrons dans la section 2.5, que ces valeurs sont importantes pour l'interprétation des résultats. De plus, nous construisons artificiellement des versions équilibrées de ROOT9 et Weeds pour une comparaison équitable avec RUMEN qui est équilibré à l'origine. En particulier, toutes les paires de mots de la relation sémantique la moins fréquente ont été conservées, et un nombre similaire de paires a été sélectionné aléatoirement pour les autres ensembles de paires de mots. Enfin, pour tester les architectures à trois tâches, nous avons proposé de construire un jeu de données artificiel compilé à partir de RUMEN, ROOT9 et Weeds, appelé RRW[7]. En particulier, toutes les paires de mots de chaque relation sémantique sont ajoutées, et l'ensemble de paires aléatoires est construit à partir des paires aléatoires de RUMEN et ROOT9, uniquement, afin de maintenir l'équilibre autant que possible.

[5] https://github.com/esantus/ROOT9

[6] https://github.com/Houssam93/MultiTask-Learning-NLP/tree/master/data/RUMEN.

[7] disponible à https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP/blob/main/Code/prepare_data_multi_task.py.

2.4.2 Extraction des patrons lexicaux

Pour extraire les patrons lexicaux qui relient les paires de mots des jeux de données étudiés, nous avons téléchargé la base de données wikipedia dump en anglais[8] et extrait un corpus représentatif d'environ 4 Go. Tous les articles du corpus ont ensuite été divisés par phrase et stockés dans un fichier csv[9]. Nous avons enfin effectué un prétraitement pour supprimer les caractères spéciaux dans les phrases. Seuls les patrons lexicaux qui ne dépassent pas une longueur maximale de 10 mots[10] ont ensuite été considérés comme des séquences de mots valides.

2.4.3 Configurations d'apprentissage

La sortie de chaque couche $h_i^{w_1,w_2}$ contient 200 neurones, et la fonction d'activation de ces couches est la tangente hyperbolique. Pour les couches S, S^1, S^2 et H^{T_j}, le nombre de neurones par couche est respectivement de 300, 100, 100 et 50. Ces couches partagent la fonction sigmoïde comme fonction d'activation. Quant au processus d'apprentissage, Adam (Kingma and Ba 2014) est utilisé comme optimiseur avec les paramètres par défaut de Keras (Chollet 2015). Le réseau est entraîné avec des batchs de 64 exemples et le nombre d'itérations est optimisé pour maximiser le score F_1 sur l'ensemble de validation. Les plongements lexicaux des mots sont initialisées avec les représentations à 300 dimensions de GloVe (Pennington et al. 2014).

L'architecture globale présentée dans la figure 2.1 est mise en œuvre à l'aide de Keras avec Tensorflow comme back-end.

[8] https://dumps.wikimedia.org/enwiki/20190220/

[9] disponible à https://github.com/Houssam93/MultiTask-Learning-NLP/blob/master/data.

[10] Cette valeur a été réglée expérimentalement.

	Synonymes		Hyperonymes		Co-hyponymes		Méronymes		Aléatoire	
Dataset	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)
RUMEN (Balikas et al. 2019)	6326	44/14/8/34	6326	65/12/5/18	-	-	-	-	6326	93/4/1/2
ROOT9 (Santus et al. 2016b)	-	-	2447	21/9/6/64	3200	28/14/8/50	-	-	1100	78/9/4/9
WEEDS (Weeds et al. 2004)	-	-	1257	40/13/6/41	2083	60/11/5/24	-	-	6326	93/4/1/2
BLESS (Baroni and Lenci 2011)	-	-	1337	57/9/4/30	3565	34/12/7/40	2943	99/0/0/1	6702	97/1/0/2
ROOT9 (bal.)	-	-	1100	20/9/5/65	1100	29/15/8/48	-	-	1100	78/9/4/9
Weeds (bal.)	-	-	1257	40/13/7/40	1257	61/10/5/24	-	-	1257	94/4/1/1

Table 2.2: Les détails des jeux de données Rumen, ROOT9, Weeds et Bless. 0 / 1 / 2 / >2 représente le pourcentage de paires de mots n'ayant respectivement aucun patron, 1 patron, 2 patrons et plus de 2 patrons dans Wikipedia Dump.

2.5 Evaluation

Dans cette section, nous présentons les résultats d'évaluation de tous les modèles présentés dans la section 2.3. En particulier, ils sont testés sur trois jeux de données de référence pour la modélisation à deux tâches, à savoir RUMEN, ROOT9 et Weeds, et deux autres jeux de données de référence pour les architectures à trois tâches, à savoir Bless et RRW. Les résultats de la classification sont mesurés par les scores de l'*accuracy*, de la précision, du rappel et de F_1.

Il est important de noter que le fractionnement lexical est appliqué comme proposé dans (Levy et al. 2015), afin qu'il n'y ait pas d'intersection de vocabulaire entre l'ensemble de test et les ensembles d'entraînement et de validation. En effet, (Levy et al. 2015) souligne que l'utilisation des représentations distributionnelles dans le contexte de l'apprentissage supervisé tend à effectuer une mémorisation lexicale plutôt qu'un apprentissage de la relation entre les mots. Dans ce cas, le modèle apprend principalement des propriétés indépendantes de termes uniques par paires. Pour éviter cette situation, le fractionnement lexical est proposé. Dans le cadre de ces expériences, nous suivons exactement la même procédure définie par (Balikas et al. 2019) pour le fractionnement lexical.

2.5.1 L'apprentissage concurrent entre deux tâches

		Synonymie vs Aléatoire				Hypernym vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	(Balikas et al. 2019) - Baseline	0.731	0.734	0.721	0.747	0.780	0.750	0.739	0.761
	one-class + all inputs	0.778	0.777	0.786	0.768	0.784	0.770	0.817	0.730
	Fully shared (Balikas et al. 2019) + all inputs	0.806	0.806	0.812	0.801	0.786	0.776	0.807	0.748
	single shared-private + all inputs	**0.835****	**0.837****	**0.832****	**0.844****	0.820**	0.816**	0.829**	**0.803****
	multiple shared-private + all inputs	0.834**	0.836**	**0.832***	0.840**	**0.823****	**0.818****	**0.837****	0.801**
		Co-hyponymie vs Aléatoire				Hypernym vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	(Balikas et al. 2019) - Baseline	0.892	0.928	**0.952**	0.905	0.834	0.882	0.887	0.877
	one-class + all inputs	0.906	0.936	0.939	0.934	0.853	0.894	0.907	0.882
	Fully shared (Balikas et al. 2019) + all inputs	0.900	0.933	0.932	0.935	0.857	0.898	0.897	0.901
	single shared-private + all inputs	**0.919****	**0.945****	0.950**	**0.940**	0.865**	0.903	0.912	0.895
	multiple shared-private + all inputs	0.917**	0.943**	**0.952****	0.936	**0.866***	**0.904**	**0.913**	**0.896**
		Co-hyponymie vs Aléatoire				Hypernym vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds	(Balikas et al. 2019) - Baseline	0.738	0.455	0.450	0.460	0.825	0.451	0.478	0.426
	one-class + all inputs	0.779	0.522	0.569	0.484	0.882	0.611	0.685	0.555
	Fully shared (Balikas et al. 2019) + all inputs	0.791	0.585	0.584	0.590	0.880	0.641	0.648	0.638
	single shared-private + all inputs	**0.825****	**0.659****	**0.645****	**0.676****	**0.903****	0.716**	**0.702**	0.735**
	multiple shared-private + all inputs	0.822**	0.652**	0.640**	0.669**	0.902**	**0.718****	0.698	**0.742****

Table 2.3: Accuracy, F_1, précision et rappel pour les architectures à deux tâches. * et ** sont les valeurs p de 0.05 et 0.10 respectivement, basées sur le t-test supposant des variances d'échantillon inégales des valeurs métriques par rapport aux méthodes de référence (One class + all inputs et (Balikas et al. 2019) + all inputs").

2.5.1.1 Interprétation des résultats

Les premiers résultats pour deux tâches sont présentés dans le tableau 2.3 pour les trois modèles proposés dans la section 2.3, à savoir *fully-shared*, *single shared-private* et *multiple shared-private* sur les jeux de données originaux RUMEN, ROOT9 et Weeds. Les performances de ces modèles sont comparées aux travaux de (Balikas et al. 2019) et à la contrepartie à une classe de notre architecture (c'est-à-dire sans multi-tâches).

Pour les trois jeux de données, toutes les architectures surpassent l'architecture de référence, ce qui met clairement en évidence l'importance de la définition de nouvelles représentations d'entrée et les avantages de l'apprentissage multi-tâches. En moyenne

sur les deux tâches et les trois jeux de données, une amélioration maximale de 6,9% en termes de précision et de 12,0% en termes de score F_1 est obtenue par rapport à (Balikas et al. 2019).

Pour tenir compte de l'impact sur la représentation d'entrée, nous montrons les résultats de l'architecture *fully-shared* de notre modèle générique, qui correspond au modèle présenté par (Balikas et al. 2019) mais intégrant toutes les représentations continues distributionnelles et les représentations basées sur des patrons. Dans ce cas, en moyenne sur l'ensemble des jeux de données, l'amélioration maximale du modèle *single shared-private* ou du modèle *multiple shared-private* est évaluée à 2,0% en termes de précision et à 3,6% en termes de F_1.

Il ressort clairement de ces résultats que l'amélioration des représentations d'entrée et la définition de nouveaux modèles d'apprentissage multi-tâches améliorent les résultats de classification. La différence entre le modèle *single shared-private* et le modèle *multiple shared-private* n'est pas aussi flagrante qu'avec l'architecture *fully-shared*. Les résultats globaux du tableau 2.3 montrent que dans la majorité des cas, le modèle le plus performant est le modèle *multiple shared-private*, mais avec une petite marge. De plus, selon l'ensemble de données, ces chiffres peuvent changer. Par exemple, le modèle *multiple shared-private* a tendance à surpasser l'architecture *single shared-private* pour Weeds, mais la situation est inversée pour RUMEN.

À l'origine, le modèle *multiple shared-private* a été conçu pour mieux prendre en compte les relations asymétriques telles que l'hypernoymie ou la méronymie. Cette situation n'est pas clairement mise en évidence, en effet, même si pour certaines tâches le modèle à tendance à surpasser légèrement le modèle *single shared-private*. Cependant

les différences ne sont pas statistiquement pertinentes et nous ne permettent pas de tirer des conclusions claires.

Enfin, il est important de noter que tous les modèles multi-tâches obtiennent de meilleurs résultats que le modèle à une classe pour toutes les relations sémantiques sur tous les jeux de données. Ainsi, les avantages de l'apprentissage simultané sont clairs puisque les deux tâches s'entraident dans le processus de classification. Il s'agit d'une question importante car il a été démontré que dans certaines configurations multi-tâches, seule une tâche peut être améliorée, tandis que l'autre reste inchangée (Balikas et al. 2017).

2.5.1.2 Jeux de données équilibrés

Afin d'éviter le déséquilibre naturel des jeux de données, ROOT9 et Weeds ont été modifiés pour que toutes les classes aient le même nombre de paires. Les caractéristiques de ces jeux de données sont présentées dans le tableau 2.2. Notez que RUMEN a été conçu pour être équilibré, il n'a donc pas été modifié. Ces tests sont effectués afin d'avoir une vue d'ensemble du biais qui peut être introduit par un ensemble non équilibré d'exemples d'apprentissage, et de permettre des comparaisons plus directes entre les jeux de données. Les résultats globaux sont présentés dans le tableau 2.4 et montrent des tendances similaires aux résultats précédents avec des jeux de données non équilibrés. En effet, toutes les architectures multi-tâches restent les plus performantes lorsqu'elles sont comparées aux modèles de référence, en particulier les modèles *shared-private*. Bien que les tâches individuelles se comportent de manière similaire, les résultats moyens de deux tâches en termes de précision et de scores F_1 penchent

clairement en faveur de l'architecture multi-tâches *shared-private*, de manière similaire aux résultats précédents.

		Co-hyponyme vs Aléatoire				Hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec. .
ROOT9	(Balikas et al. 2019) - Baseline	0.811	0.818	0.822	0.814	0.818	0.809	0.787	0.831
	one-class + all inputs	0.870	0.866	0.889	0.846	0.825	0.823	0.843	0.807
	Fully shared (Balikas et al. 2019) + all inputs	0.863	0.865	0.851	0.881	0.832	0.839	0.816	**0.864**
	single shared-private + all inputs	0.897***	0.897***	0.897	0.897**	0.847**	0.847	**0.854**	0.843**
	multiple shared-private + all inputs	**0.900*** **	**0.899*** **	**0.902***	**0.897*** **	**0.848*** **	**0.850***	0.848	0.854
		Co-hyponyme vs Aléatoire				Hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds	(Balikas et al. 2019) - Baseline	0.587	0.602	0.573	0.634	0.661	0.658	0.635	0.682
	one-class + all inputs	0.651	0.653	0.641	0.665	0.779	0.737	**0.853**	0.648
	Fully-shared (Balikas et al. 2019) + all inputs	0.659	0.670	0.641	0.702	0.760	0.762	0.724	0.804
	single shared-private + all inputs	**0.711**	**0.717**	**0.693**	**0.743**	0.789	0.788	0.758	0.821
	multiple shared-private + all inputs	0.708	0.714	0.691	0.738	**0.797**	**0.799**	0.759	**0.844**

Table 2.4: Accuracy, F_1, précision et rappel sur les jeux de données ROOT9 et Weeds pour les architectures à deux tâches. Les jeux de données sont équilibrés. Le fractionnement lexical est appliqué.

2.5.1.3 Analyse d'ablation

Nous effectuons une analyse d'ablation de la meilleure architecture, c'est-à-dire le modèle *multiple shared-private* même si la différence avec le modèles *single shared-private* reste très minime. L'idée sous-jacente de l'analyse d'ablation est de tenir compte de l'importance de chaque caractéristique d'entrée individuelle dans le processus de classification. Elle a été largement utilisée dans l'évaluation des systèmes de réponse aux questions (Chen et al. 2017) et nous proposons de reproduire cette analyse dans le contexte de notre travail. Les résultats de l'ablation sont présentés dans le tableau 2.5. Nous commençons par supprimer la couche BiLSTM, puis la caractéristique cosinus $cos(w_1, w_2)$ et enfin, la caractéristique différence $(w_1 \ominus w_2)$. Notez que la suppression de la couche BiLSTM élimine toutes les informations basées sur les patrons qui sont résumées dans une caractéristique unique, comme le montre la figure 2.1.

Comme prévu, la suppression des représentations d'entrée semble avoir un impact direct sur les performances. La situation parfaite est décrite pour le jeu de données Weeds, où chaque caractéristique individuelle contribue de manière égale au processus de classification. Cependant, cette situation n'est pas vraie pour tous les jeux de données. Contrairement aux conclusions de (Shwartz et al. 2016), qui affirment que les caractéristiques continues basées sur des patrons lexicaux améliorent considérablement les performances[11], les résultats sont plus mitigés pour notre étude. En effet, la suppression de l'encodage BiLSTM augmente globalement les résultats pour RUMEN. Cette situation peut facilement s'expliquer en se basant sur le Tableau 2.2. En effet, la plupart des paires de mots dans RUMEN ne contiennent pas de patrons lexicaux. Plus précisément, 44% des paires de synonymes, 65 % des paires d'hypernoms et 93% des paires aléatoires ne sont liées par aucun patron. Par conséquent, il est probable que l'encodage continu ne soit pas en mesure de généraliser correctement des informations limitées. L'autre effet important qui doit être souligné est la qualité des patrons découverts. En effet, le jeu de données Weeds présente des caractéristiques similaires à celles du jeu de données RUMEN, avec 40% des paires d'hypernoymies, 60% des paires de co-hyponymes et 93% des paires aléatoires, qui n'ont aucun patron les reliant. Cependant, comme le jeu de données Weeds est construit manuellement et contient plus de concepts concrets que le jeu de données RUMEN. Il est donc plus probable que les modèles extraits soient cohérents avec les définitions communes des relations sémantiques étudiées. A l'inverse, RUMEN est construit automatiquement à partir de WordNet (Miller 1995) et peut contenir des concepts généraux qui peuvent être difficiles à trouver dans des corpus même

[11] Notez qu'ils ne présentent pas d'analyse d'ablation.

importants, ou qui peuvent être liés fortuitement. En tant que tel, il est probable que la qualité des modèles ne soit pas garantie, donnant lieu à des encodages bruités.

En ce qui concerne les autres caractéristiques d'entrée, certains résultats intéressants sont obtenus. Il est clair que pour tous les jeux de données et toutes les tâches (symétriques ou asymétriques), la caractéristique cosinus influence positivement les performances finales. En effet, c'est la caractéristique qui présente la plus grande perte de performance lorsqu'elle est supprimée. Bien que cette situation puisse être comprise pour les relations symétriques, il est surprenant de constater que la mesure de similarité symétrique profite clairement à la classification des relations asymétriques telles que l'hypernoymie. De plus, la différence vectorielle s'est généralement révélée moins pertinente que la concaténation à l'instar de ce que (Shwartz et al. 2016) mentionnent. Cette situation est clairement décrite pour ROOT9 où la suppression de la différence vectorielle augmente les résultats. Cependant, cette situation n'existe pas pour les autres jeux de données. Enfin, sur la base des résultats du tableau 2.5, nous pouvons conclure que toutes les entrées sont importantes mais qu'une attention particulière doit être accordée aux modèles et à la différence vectorielle en fonction de l'ensemble de données utilisé.

2.5.1.4 BiLSTM privé

Comme discuté dans la section 2.3, on peut argumenter que les patrons doivent être dépendants de la tâche et ne peuvent partager aucune information entre les tâches. En effet, à l'origine, les patrons ont été définis différemment pour chaque relation sémantique. Cependant, dans notre configuration générique, nous faisons l'hypothèse que les patrons peuvent partager certaines informations latentes entre les tâches. Pour vérifier

		Synonymie vs Aléatoire				Hypernym vs Aléatoire			
	multiple shared-private Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	All inputs	0.836	0.839	0.820	**0.860**	0.826	**0.805**	0.780	**0.832**
	w/o BiLSTM	**0.841**	**0.842**	0.830	0.855	**0.828**	0.802	**0.797**	0.806
	w/o BiLSTM w/o cos	0.789	0.795	0.770	0.822	0.804	0.775	0.769	0.781
	w/o BiLSTM w/o cos w/o vdiff	0.778	0.782	0.764	0.802	0.805	0.776	0.770	0.782
	All inputs w/ Private-only BiLSTM	0.831↓	0.834↓	0.817↓	0.851↓	0.834↑	0.810↑	0.799↑	0.821↓
		Co-hyponymie vs Aléatoire				Hypernym vs Aléatoire			
	multiple shared-private Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	All inputs	**0.925**	**0.950**	**0.960**	0.940	**0.875**	**0.911**	0.914	**0.909**
	w/o BiLSTM	0.923	**0.950**	0.950	**0.949**	0.865	0.903	**0.922**	0.884
	w/o BiLSTM w/o cos	0.897	0.932	0.939	0.925	0.843	0.891	0.874	0.909
	w/o BiLSTM w/o cos w/o vdiff	0.912	0.942	0.949	0.934	0.856	0.898	0.901	0.895
	All inputs w/ Private-only BiLSTM	0.930↑	0.954↑	0.951↓	0.958↑	0.885↑	0.919↑	0.915↑	0.923↑
		Co-hyponymie vs Aléatoire				Hypernym vs Aléatoire			
	multiple shared-private Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds	All inputs	**0.838**	**0.685**	**0.638**	**0.740**	**0.904**	**0.720**	**0.710**	**0.730**
	w/o BiLSTM	0.817	0.643	0.598	0.695	0.895	0.680	0.694	0.667
	w/o BiLSTM w/o cos	0.783	0.557	0.541	0.574	0.867	0.594	0.611	0.578
	w/o BiLSTM w/o cos w/o vdiff	0.776	0.546	0.528	0.565	0.854	0.501	0.589	0.436
	All inputs w/ Private-only BiLSTM	0.826↓	0.661↓	0.615↓	0.714↓	0.891↓	0.689↓	0.664↓	0.716↓

Table 2.5: Analyse d'ablation pour l'architecture *multiple shared-private* et ses variations: Encodage BiLSTM privé uniquement. Accuracy, F_1, precision et rappel sont donnés pour les architectures à deux tâches sur les jeux de données RUMEN, ROOT9 et Weeds. Les jeux de données ne sont pas équilibrés. Le découpage lexical est appliqué. Notez que ↓ (resp. ↑) signifie que le score est inférieur (resp. supérieur) à celui obtenu par l'algorithme *multiple shared-private* avec toutes les entrées.

cette hypothèse, nous proposons de tester notre modèle générique avec la définition de $V(X) = (S(X), S^1(X), S^2(X), X, h_*^{w_1,w_2})$, qui correspond à un encodage BiLSTM privé uniquement, c'est-à-dire que les patrons ne sont pas partagés entre les tâches. Les résultats sont présentés dans le tableau 2.5.

Différentes situations sont mises en évidence. Pour le jeu de données Weeds, il est clair que le retrait de l'information partagée contenue dans les patrons est préjudiciable à la performance globale. Ainsi, notre hypothèse initiale est confirmée. Cependant, la situation est légèrement différente pour RUMEN. Dans ce cas, le partage des informations contenues dans les patrons est toujours bénéfique en moyenne, sauf pour la Précision. Cependant, certaines différences sont mises en évidence entre les relations

sémantiques. Dans ce cas, l'impact de l'information partagée est positif pour la synonymie mais pas pour l'hypernoymie. Cela peut s'expliquer par le fait que les modèles jouent généralement un rôle moins important pour la détection de la synonymie, alors qu'ils se sont avérés jouer un rôle important pour la découverte de l'hypernoymie (Roller et al. 2018). En tant que tel, comme la synonymie est liée à l'hypernoymie par la similitude en cosinus, il est probable qu'ils partagent certains patrons qui peuvent être utiles pour la synonymie. Bien que l'inverse puisse ne pas être vrai, car les patrons lexicaux entre synonymes peuvent inclure des données d'entrée bruitées. Enfin, la situation pour ROOT9 est complètement différente. En effet, dans ce cas, la configuration BiLSTM privée uniquement surpasse la solution avec des représentations partagées des patrons, sauf pour la précision. Ce résultat peut être expliqué par le fait que ROOT9 est le jeu de données qui contient le moins de paires sans patrons. Il existe donc une grande quantité d'informations privées potentielles, qui peuvent suffire à elles seules à classer correctement les paires de mots, sans tirer parti des informations partagées des patrons.

En résumé, il est clair que si les jeux de données manquent d'évidences de patrons, le modèle générique avec des informations partagées de patrons peut conduire à une meilleure performance de classification. Cependant, si de nombreux patrons existent, avec une qualité élevée, il est probable que la configuration BiLSTM privée uniquement soit la plus adaptée. Cependant, cette dernière situation est loin d'être raisonnable dans les applications du monde réel, où les données contiennent beaucoup de bruit, et où le vocabulaire est diversifié.

Il y a de légères différences entre les deux architectures pour RUMEN et ROOT9, mais pas de vainqueur clair. Cependant, les résultats obtenus à l'aide de Weeds ne sont

pas performants si l'on utilise une architecture uniquement privée, ce qui est similaire à la suppression de la couche BiLSTM. Cela suggère que nos couches partagées parviennent à comprendre les patrons sous-jacents en les partageant entre différentes tâches et pas seulement en utilisant une couche BiLSTM.

2.5.2 Apprentissage concurrent de trois tâches

Les résultats des modèles à trois tâches sont présentés dans le tableau 2.6 pour tous les modèles proposés dans la section 2.3, à savoir *fully-shared*, *single shared-private* et *multiple shared-private* sur le jeu de données original Bless et le jeu de données RRW, une compilation des jeux de données originaux RUMEN, ROOT9 et Weeds. Comme dans la section précédente, nous comparons les performances de ces modèles avec l'architecture de (Balikas et al. 2019) et l'architecture *one-class* (c'est-à-dire sans multi-tâches). En moyenne, sur les deux jeux de données, presque toutes les architectures multi-tâches surpassent l'architecture de référence, à l'exception (1) du modèle *fully-shared* pour RRW, qui présente des résultats inférieurs à ceux de (Balikas et al.

		Co-hyponymie vs Aléatoire		Hyperonymie vs Aléatoire		méronymie vs Aléatoire	
	Algorithm	Acc.	F_1	Acc.	F_1	Acc.	F_1
Bless	(Balikas et al. 2019) - Baseline	0.904	0.858	0.900	0.666	0.860	0.773
	one-class + all inputs	0.908	0.865	0.907	0.693	0.849	0.754
	Fully shared (Balikas et al. 2019) + all inputs	0.918	0.881	0.905	0.686	0.855	0.759
	single shared-private + all inputs	**0.938****	**0.911****	0.923**	**0.757****	0.878**	0.804**
	multiple shared-private + all inputs	0.937**	**0.911****	**0.923****	0.753**	**0.880****	**0.806****
		Co-hyponymie vs Aléatoire		Hyperonymie vs Aléatoire		Synonymie vs Aléatoire	
	Algorithme	Acc.	F_1	Acc.	F_1	Acc.	F_1
RRW	(Balikas et al. 2019) - Baseline	0.843	0.812	0.794	0.814	0.817	0.802
	one-class + all inputs	0.799	0.742	0.773	0.794	0.768	0.742
	Fully shared (Balikas et al. 2019) + all inputs	0.821	0.791	0.783	0.804	0.804	0.789
	single shared-private + all inputs	0.855**	0.836**	0.813**	0.836**	**0.831****	**0.824****
	multiple shared-private + all inputs	**0.857****	**0.837****	**0.815****	**0.837****	0.830**	0.823**

Table 2.6: Accuracy et F_1 sur les jeux de données Bless et RRW pour les architectures à trois tâches. Les jeux de données ne sont pas équilibrés. Le fractionnement lexical est appliqué.

2019), et (2) du modèle *fully-shared* pour Bless, qui présente un score F_1 inférieur à celui du modèle à classe unique. Ainsi, les résultats montrent principalement la supériorité des architectures partagées-privées. En particulier, en moyenne sur les trois tâches et les deux jeux de données, une amélioration maximale de 2,2% en termes de précision et de 6,8% en termes de score F_1 peut être obtenue par rapport à (Balikas et al. 2019).

De la même manière que pour les résultats à deux tâches, nous présentons les améliorations apportées par les architectures partagées-privées par rapport au modèle proposé par (Balikas et al. 2019), lorsqu'il intègre toutes les représentations continues distributionnelles et les représentations basées sur des patrons. Dans ce cas, en moyenne sur tous les jeux de données, l'amélioration maximale du modèle *single shared-private* ou du modèle *multiple shared-private* est évaluée à 3,4% en terme de précision et à 6,5% en terme de F_1. Ces résultats montrent que l'introduction de représentations d'entrée plus complexes ainsi que la définition de nouveaux modèles d'apprentissage multi-tâches favorisent les résultats de classification.

Cependant, la différence entre le modèle *single shared-private* et le modèle *multiple shared-private* ne semble pas être significative, et est susceptible de dépendre des jeux de données testés. En effet, la différence entre les deux modèles est moins claire que pour les modèles à deux tâches, ce qui suggère que le processus de généralisation des informations sur un seul mot peut être plus difficile à réaliser, lorsque davantage de relations sémantiques (symétriques et asymétriques) sont abordées. Cependant, de nouvelles idées peuvent être découvertes en analysant soigneusement les résultats du tableau 2.6. En particulier, l'architecture *multiple shared-private* qui inclut des informations d'un seul mot donne de meilleurs résultats pour l'ensemble de données com-

prennent plus de relations asymétriques. En effet, les meilleurs résultats sont obtenus pour Bless, qui comprend deux relations asymétriques, à savoir l'hypernoymie et la méronymie, sur trois relations. A l'inverse, les meilleurs résultats sont obtenus par le modèle conscient du mot unique (i.e. *multiple shared-private*) testé sur RRW, qui ne contient qu'une seule relation asymétrique, à savoir l'hypernoymie. Ces résultats semblent confirmer notre hypothèse initiale selon laquelle l'information sur un seul mot doit être prise en compte lorsqu'on traite des relations asymétriques.

Enfin, il est important de noter que presque tous les modèles multi-tâches obtiennent de meilleures performances que les modèles à classe unique respectifs, à l'exception de l'architecture *fully-shared* sur l'ensemble de données Bless. En particulier, les architectures *shared-private* affichent des résultats supérieurs pour toutes les tâches en cours par rapport au classifieur à classe unique. En tant que tel, il est clair que toutes les relations sémantiques sont corrélées et qu'elles se renforcent mutuellement dans leurs décisions individuelles.

2.5.3 Conclusion

Nous avons présenté une architecture générique multi-tâches pour l'identification de relations lexico-sémantiques. En particulier, ce modèle neuronal encode à la fois des caractéristiques distributives et des caractéristiques basées sur des patrons lexicaux, et introduit des configurations conscientes du mot unique qui permettent un meilleur traitement des relations asymétriques dans certains cas. Les résultats de l'évaluation pour deux et trois tâches confirment que les architectures *shared-private* surpassent les modèles récents proposés dans (Balikas et al. 2019) ainsi que les modèles à une seule classe.

Il est intéressant de noter que le modèle BiLSTM partagé proposé pour coder les motifs lexicaux améliore la qualité du codage lorsque peu de motifs existent.

La première étape consiste à établir que quelque chose est possible ; alors la probabilité se produira.

Elon Musk

3

Etude approfondie des caractéristiques pour l'identification des relations lexico-sémantiques

Comme evoqué dans le chapitre 2, l'application d'architectures multi-tâches ont permis d'améliorer les performances pour l'identification des relations lexico-sémantiques. Dans ce chapitre, nous proposons d'étudier l'utilisation de caractéristiques (*feature egineering*) dans un contexte d'apprentissage mutli-tâches pour l'extraction des relations lexico-sémantiques.

Dans le contexte d'identification des relations lexico-sémantiques, de nombreuses études existent pour le cas discret (Snow et al. 2004, Roller et al. 2014, Weeds et al. 2014), mais nous concentrons notre étude sur les stratégies incluant les espaces sémantiques latents.

Dans le premier cas, Levy et al. (2015) et (Vylomova et al. 2016) ont proposé des évaluations similaires pour combiner les vecteurs d'entrée de mots ($\overrightarrow{w_1}$, $\overrightarrow{w_2}$), suite aux expériences initiales de (Baroni et al. 2012, Roller et al. 2014, Weeds et al. 2014). En particulier, les paires de mots sont codées comme la concaténation des représentations des mots constitutifs ($\overrightarrow{w_1} \oplus \overrightarrow{w_2}$), leur différence vectorielle ($\overrightarrow{w_1} \ominus \overrightarrow{w_2}$) ou leur somme

($\overrightarrow{w_1} + \overrightarrow{w_2}$). Une autre direction de recherche vise à calculer un cosinus généralisé (Vu and Shwartz 2018) défini comme la multiplication élément par élément des vecteurs d'entrée ($\overrightarrow{w_1} \otimes \overrightarrow{w_2}$). Leurs résultats montrent clairement la supériorité de la combinaison de ($\overrightarrow{w_1} \oplus \overrightarrow{w_2}$) et ($\overrightarrow{w_1} \otimes \overrightarrow{w_2}$). Notre intuition est qu'un cosinus généralisé en entrée d'un réseau de neurones, permetterait de donner des poids différents à chaque dimension de l'espace de l'*embedding*, permettant ainsi un apprentissage plus fin que la simple application de la valeur de cosinus.

Une autre direction de recherche vise à construire des espaces sémantiques latents affinés. (Nguyen et al. 2017a) a proposé HyperVec, où les *embeddings* sont appris dans un ordre spécifique afin de capturer la hiérarchie distributionnelle des relations hyperonyme-hyponyme à partir d'une base de connaissance externe. (Vulić and Mrkšić 2018) ont plutôt proposé une stratégie de post-traitement qui réadapte la base de connaissance (ici les liens hyperonymiques de WordNet) dans un espace sémantique latent original. En tant que telle, leur procédure rapproche les vraies paires hyperonyme-hyponyme dans l'espace euclidien transformé. Cependant, ces méthodes souffrent d'une couverture limitée car elles n'affectent que les vecteurs de mots vus dans les ressources externes. Pour faire face à cette limitation, (Kamath et al. 2019) ont présenté une méthode de post-traitement qui spécialise les vecteurs de tous les mots du vocabulaire en apprenant une fonction de spécialisation globale explicite. Suivant la même idée, (Wang and He 2020) ont appris deux fonctions de projection de *embeddings* (la première des hyperonymes vers les hyponymes et la deuxième des hyponymes vers les hyperonymes) à partir d'une base de connaissance externe de sorte qu'une paire de mots est représentée par la concaténation des *embeddings* projetés (représentation BiRRE). Dans la même

ligne de recherche, (Bouraoui et al. 2020) ont récemment introduit un cadre qui affine BERT (Devlin et al. 2019) pour inclure des informations relationnelles. À partir de quelques instances d'une relation donnée provenant d'une base connaissance externe, ils trouvent des phrases qui sont susceptibles d'exprimer cette relation dans un corpus donné, puis utilisent un sous-ensemble de celles-ci comme modèles. Le modèle de langage est ainsi affiné en prédisant si une paire de mots donnée est susceptible d'être une instance d'une certaine relation.

Bien que les *embeddings* à réglage fin aient donné de meilleurs résultats que les *embeddings* à usage général, ils dépendent des relations étudiées et des connaissances, ce qui réduit intrinsèquement le champ d'application de cette approche. Une exception intéressante est proposée par (Meng et al. 2019) qui apprennent des *embeddings* de texte dans un espace sphérique (JoSE) adapté aux informations relationnelles qui ne dépendent d'aucune ressource externe.

D'un autre point de vue, l'ingénierie des caractéristiques offre une augmentation de performance très peu coûteuse (Vu and Shwartz 2018) dans les environnements sans ressources. Mais, une étude complète de la combinaison des caractéristiques manque encore, ainsi que la définition de caractéristiques asymétriques dans le contexte des espaces continus, bien que de nombreux travaux existent pour le cas discret (Kotlerman et al. 2010, Santus et al. 2017). Dans ce chapitre, nous proposons donc de traiter les limitations susmentionnées dans une configuration sans ressources en:

- Définissant un ensemble de caractéristiques asymétriques basées sur la divergence de Kullback-Leibler dans un espace continu;
- Etudiant la combinaison de caractéristiques dans et entre les ensembles de famille

de caractéristiques;

- Définissant une architecture multi-tâches partagée-privée avec des ensembles de caractéristiques optimisés pour les couches spécifiques à la tâche et inter-tâches;
- Inspectant l'impact des *embeddings* sphériques combinés à l'ingénierie des caractéristiques.

3.1 Analyse des caractéristiques

En plus de la concaténation d'*embeddings* de mots, nous définissons trois familles de caractéristiques basées sur l'hypothèse distributionnelle (caractéristiques symétriques et asymétriques) et l'approche paradigmatique (caractéristiques basées sur des patrons lexicaux) dans des espaces sémantiques continus.

3.1.1 Caractéristiques symétriques

Des études ont mis en évidence l'intérêt de coupler les *embeddings* de mots avec des caractéristiques spécifiques pour améliorer l'identification des relations. En particulier, la mesure de similarité cosinus *cos* a montré des résultats prometteurs (Garten et al. 2015, Barkan 2017). Cependant, Vu and Shwartz (2018) ont démontré l'efficacité de l'intégration de la multiplication par éléments des vecteurs d'entrée, qui peut être vue comme un cosinus généralisé (*cosG*, alias *Mult*), qui est défini dans l'équation 3.1.

$$cosG(\overrightarrow{w_1}, \overrightarrow{w_2}) = \bigoplus_{i=1}^{d} w_1^i w_2^i \tag{3.1}$$

Alors que le cosinus ne fournit qu'une valeur unique en entrée, *cosG* se réfère à une entrée de dimension d, mettant ainsi en évidence un problème dimensionnel. Par conséquent, nous proposons de transformer le *cosG* en une valeur unique en utilisant une couche d'activation linéaire comme dans l'équation 3.2. Le *cosG*1*D* peut être considéré comme une valeur de contrôle du *cosG* en tenant compte du biais dimensionnel (de la haute à la basse dimension).

$$cosG1D(\overrightarrow{w_1}, \overrightarrow{w_2}) = \sum_{i=1}^{d} \lambda_i w_1^i w_2^i \tag{3.2}$$

La contrepartie de l'équation 3.2 est la duplication (d fois) de la valeur du cosinus. Cette métrique appelée diffusion du cosinus (*cosBr*) définie dans l'équation 3.3 vise à contrôler la question dimensionnelle d'une dimension basse à une dimension élevée.

$$cosBr(\overrightarrow{w_1}, \overrightarrow{w_2}) = \bigoplus_{i=1}^{d} cos(\overrightarrow{w_1}, \overrightarrow{w_2}) \tag{3.3}$$

Ainsi, dans l'équation 3.4, nous définissons une famille de caractéristiques de distribution symétriques.

$$CosF = (cos, cosG, cosBr, cosG1D) \tag{3.4}$$

Dans la sous-section suivante, nous détaillons la conception de nouvelles mesures distributionnelles asymétriques basées sur la divergence de Kullback-Leibler.

3.1.2 Caractéristiques distributionnelles asymétriques

L'asymétrie a donné de bons résultats pour le cas discret (Kotlerman et al. 2010, Santus et al. 2017), l'idée sous-jacente étant que la relation entre les mots peut être déséquili-

brée de telle sorte qu'un mot attire l'autre plus que l'inverse. Ici, nous définissons différentes caractéristiques asymétriques dans l'espace continu sur la base de la divergence de Kullback-Leibler (Kullback and Leibler 1951). Pour s'adapter au cas continu, nous transformons chaque dimension d'un vecteur de mots avec la fonction sigmoïde (σ) de sorte que toutes les valeurs se situent entre 0 et 1. Ainsi, chaque mot peut être considéré comme une distribution de probabilité et la métrique asymétrique *Kull* est définie dans les équations 3.5.

$$Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) = \sum_{i=1}^{d} log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \tag{3.5}$$

Pour prendre en compte les deux directions de l'asymétrie, nous proposons de concaténer les valeurs *Kull* pour les deux directions comme défini dans l'équation 3.6.

$$kull(\overrightarrow{w_1}, \overrightarrow{w_2}) = Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus Kull(\overrightarrow{w_2}|\overrightarrow{w_1}) \tag{3.6}$$

De la même manière que pour le *cosG*, nous proposons de définir la version multiplicative du *kull*, de telle sorte que *kullG* intègre la multiplication élément par élément des vecteurs d'entrée tels que définis dans les équations 3.7 (asymétrie unique) et 3.8 (concaténation des deux asymétries).

$$KullG(\overrightarrow{w_1}|\overrightarrow{w_2}) = \bigoplus_{i=1}^{d} log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \tag{3.7}$$

$$kullG(\overrightarrow{w_1}, \overrightarrow{w_2}) = KullG(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus KullG(\overrightarrow{w_2}|\overrightarrow{w_1}) \tag{3.8}$$

De la même manière que pour *cosG*1*D* et pour prendre en compte le problème di-

mensionnel de la version multiplicative du Kullback-Leibler, nous définissons *kullG*1*D* dans les équations 3.9 et 3.10 .

$$KullG1D(\overrightarrow{w_1}|\overrightarrow{w_2}) = \sum_{i=1}^{d} \lambda_i log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \tag{3.9}$$

$$\begin{aligned} kullG1D(\overrightarrow{w_1},\overrightarrow{w_2}) &= KullG1D(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus \\ & \quad KullG1D(\overrightarrow{w_2}|\overrightarrow{w_1}) \end{aligned} \tag{3.10}$$

De même que pour *cosBr*, nous proposons de définir *kullBr* sur la base du dédoublement (*d* fois) de la valeur de Kulback-Leibler pour les deux directions, comme dans l'équation 3.11.

$$\begin{aligned} kullBr(\overrightarrow{w_1},\overrightarrow{w_2}) &= \bigoplus_{i=1}^{d} Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) \\ & \quad \bigoplus_{i=1}^{d} Kull(\overrightarrow{w_2}|\overrightarrow{w_1}) \end{aligned} \tag{3.11}$$

Ainsi, dans l'équation 3.12, nous définissons une famille de caractéristiques distributionnelles asymétriques.

$$KullF = (kull, kullG, kullBr, kullG1D) \tag{3.12}$$

Dans la sous-section suivante, nous présentons la stratégie d'encodage des patrons lexicaux incarnant l'approche paradigmatique.

3.1.3 Caractéristiques paradigmatiques basées sur les patrons

Les patrons font partie de l'approche paradigmatique (Hearst 1992), qui suggère qu'il peut exister des séquences de mots spécifiques qui relient deux mots dans une relation donnée. Quelques exemples de séquences entre paires de mots sont donnés dans le tableau 2.1 du chapitre 2.

Ici, nous proposons de mettre en œuvre la méthodologie de (Shwartz et al. 2016)[1] pour encoder les patrons lexicaux dans des espaces continus. Ainsi, nous transformons les k patrons lexicaux[2] les plus fréquents se produisant entre w_1 et w_2 en utilisant soit des BiLSTM, soit une représentation contextuelle continue grâche au *Universal Sentence Encoder* (USE) (**?**), puis nous effectuons un regroupement moyen pour obtenir la représentation finale en entrée. Le i-ième patron le plus fréquent encodé est défini dans l'équation 3.13, où $j \in \{\text{BiLSTM}, \text{USE}\}$, $i \in [1..k]$, et la représentation moyenne des k patrons lexicaux est notée $\overline{pat}^{w_1,w_2}_{*,j}$.

$$pat^{w_1,w_2}_{i,j} = encoder_j(w_1, path_i, w_2) \tag{3.13}$$

De la même manière que pour *CosF* et *KullF*, nous définissons une famille de caractéristiques basées sur les patrons lexicaux *PatF* dans l'équation 3.14.

$$PatF = (\overline{pat}_{*,\text{USE}}, \overline{pat}_{*,\text{BiLSTM}}) \tag{3.14}$$

Dans la section suivante, nous présentons les paramètres multi-tâches qui ont été implémentés pour prendre en compte les relations lexico-sémantiques.

[1] Méthodologie similaire à celle proposée au chapitre 2.
[2] k permet de traiter les séquences parasites.

3.2 Paramètres multi-tâches

Les architectures multi-tâches se sont révélées capables de combiner avec succès des relations lexico-sémantiques étroitement liées comme nous l'avons vu dans le chapitre 2.

Ici, nous proposons d'implémenter à la fois des architectures entièrement partagées et partagées-privées pour différentes combinaisons de représentations d'entrée et de caractéristiques $X = (\overrightarrow{w_1} \oplus \overrightarrow{w_2}, CosF, KullF, PatF)$. En particulier, la sélection directe (Kohavi and Sommerfield 1995) est utilisée pour la sélection des caractéristiques, car l'espace de recherche est très grand, 2^{10} combinaisons possibles[3].

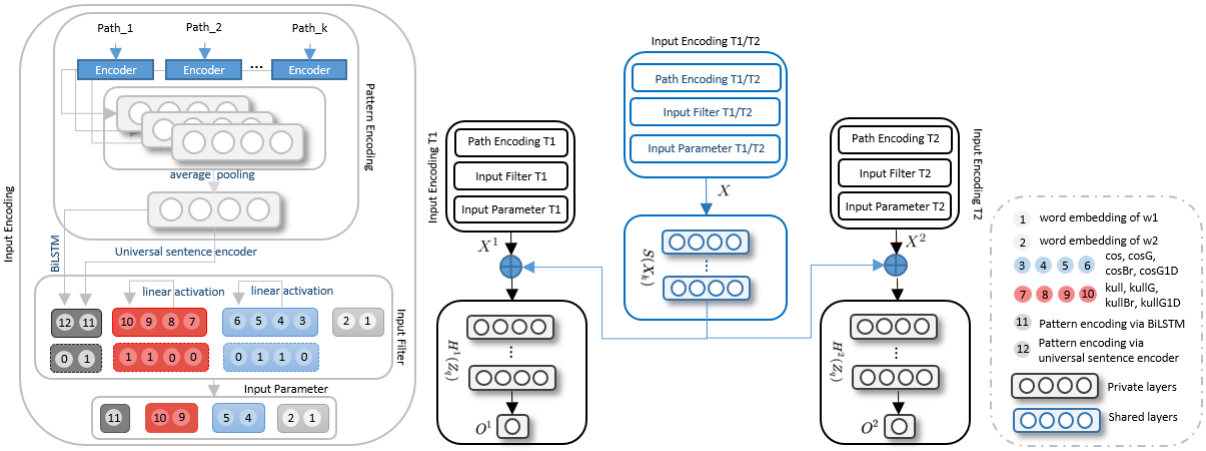

Figure 3.1: Les architectures entièrement partagées et partagées-privées avec de multiples combinaisons de caractéristiques d'entrée. Le réseau entièrement partagé ne comprend que la couche bleue, c'est-à-dire $S(X_b)$.

[3] La concaténation est l'entrée obligatoire.

3.2.1 Architectures multi-tâches

Les architectures sont présentées dans la figure 3.1 pour deux tâches. Formellement, soit X_k un vecteur d'entrée[4], nous calculons une couche partagée $S(X_k)$ comme dans l'équation 3.15, où W_{S^k} est une matrice de poids, b_{S^k} un vecteur de biais, et $k \in [1, K]$ (K le nombre de couches partagées).

$$S(X_k) = \sigma(W_{S^k} X_k + b_{S^k}) = X_{k+1} \tag{3.15}$$

Une couche privée $H^j(Z_q)$, qui résout la tâche T_j ($j \in [1, N]$) est définie dans l'équation 3.16, où $q \in [1, Q]$ (Q est le nombre de couches privées).

$$H^j(Z_q) = \sigma(W^j_{H^q} Z_q + b^j_{H^q}) = Z_{q+1} \tag{3.16}$$

Pour l'architecture entièrement partagée $Z_1 = S(X_K)$ et pour le modèle *shared-private* $Z_1 = S(X_K) \oplus X^i$, où X^i est le vecteur d'entrée spécifique à la tâche T_i. Enfin, les N décisions sont définies dans l'équation 3.17.

$$O^j = \sigma(W^j_O H^j(Z_Q) + b^j_O) \tag{3.17}$$

Les paramètres sont mis à jour en minimisant l'entropie croisée binaire. Ainsi, les poids de la couche partagée sont mis à jour en minimisant la fonction de perte de chaque tâche alternativement, tandis que les couches privées sont mises à jour pour leur tâche spécifique.

[4] $X_1 = X$, où X est le vecteur d'entrée initial qui combine à la fois des *embeddings* et un ensemble de caractéristiques spécifiques à la tâche en cours.

3.2.2 Jeux de données

Nous avons utilisé les mêmes jeux de données de référence qu'au chapitre 2, à savoir : Weeds (Weeds et al. 2004), Bless (Baroni and Lenci 2011), ROOT9 (Santus et al. 2015) et Rumen (Balikas et al. 2019).

Tous les jeux de données[5] sont résumés avec leurs caractéristiques spécifiques dans le tableau 2.2 du chapitre 2.

3.2.3 Les configurations d'apprentissage

La dimension de sortie de l'encodeur universel de phrases (USE) est égale à 512. La dimension de sortie du BiLSTM $\in \{100, 200, 300, 400, 500\}$, le nombre de patrons lexicaux les plus fréquents ($k \in [1...5]$), le nombre de couches cachées ($K \in \{1, 2\}$ et $Q \in \{1, 2\}$), le nombre de neurones ($\in \{5, 20, 50, 100, 150, 200, 300\}$) et le nombre d'itérations ($[1..100]$) sont des hyperparamètres libres qui sont réglés en utilisant un grid search. Les poids sont initialisés avec une distribution uniforme mise à l'échelle comme dans (Glorot and Bengio 2010) et mis à jour en utilisant Adam (Kingma and Ba 2014) avec un taux d'apprentissage fixé à $0,001$. Le réseau est entraîné avec des lots de 64 exemples et le nombre d'itérations est optimisé pour maximiser le score F_1 sur l'ensemble de validation. Les *embeddings* de mots sont initialisés avec les représentations à 300 dimensions de GloVe (Pennington et al. 2014) ou JoSE (Meng et al. 2019). Tous les modèles de l'état de l'art présentés dans le tableau 3.1 ont été implémentés pour fournir des résultats moyens et effectuer des tests statistiques[6].

[5] https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP/tree/main/Data

[6] https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP

3.2.4 Le fractionnement lexical

Comme suggéré dans (Levy et al. 2015), le fractionnement lexical est appliqué à toutes nos expériences afin qu'il n'y ait aucune intersection de vocabulaire entre l'ensemble de test et les ensembles d'entrainement et de validation. Notez qu'à des fins d'apprentissage, chaque jeu de données est divisé en sous-ensembles d'entrainement (50%), de validation (20%) et de test (30%).

3.3 Evaluation

Tous les résultats comparatifs contre quatre modèles de référence (Shwartz et al. 2016, Vu and Shwartz 2018, Balikas et al. 2019, Bannour et al. 2020) sont présentés dans le tableau 3.1 pour une moyenne de 25 exécutions. Des tests statistiques ont été appliqués sur quatre jeux de données de référence.

3.3.1 Modèles privés

Nous commençons par analyser l'impact de la combinaison de caractéristiques sur les modèles privés, c'est-à-dire lorsqu'une relation lexico-sémantique unique est prise en compte dans le processus d'apprentissage. Cela correspond aux quatre premières lignes du tableau 3.1. Sans surprise, l'introduction d'une combinaison de caractéristiques (éventuellement nouvelles) à un réseau de neurone multi-couches *(Multi Layer Perception : MLP)* qui forme le modèle *Best MLP* surpasse les modèles existants (Shwartz et al. 2016, Vu and Shwartz 2018) et le perceptron multicouche (MLP) qui ne comprend que la concaténation des mots incorporés (c'est-à-dire la ligne de base la plus

simple). Notez que le modèle Best MLP inclut les architectures de (Shwartz et al. 2016) et (Vu and Shwartz 2018) car il permet la combinaison de toutes les caractéristiques de la famille comme entrée.

		Synonyme vs Aléatoire				hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	MLP	0.754	0.754	0.750	0.759	0.750	0.757	0.731	0.786
	Shwartz and Dagan (2016b)	0.713	0.731	0.685	0.783	0.770	0.776	0.754	0.798
	Vu and Shwartz (2018)	0.851	0.847	0.864	0.831	0.842	0.843	0.832	0.854
	Best MLP	0.867 ⋆	0.865 ⋆	0.871 ⋆	0.859⋆†	0.863 ⋆	0.862 ⋆	0.860 ⋆	**0.865**⋆
	Balikas et al. (2019)	0.758	0.759	0.750	0.769	0.759	0.762	0.747	0.778
	Bannour et al. (2020)	0.854	0.850	0.873	0.827	0.819	0.784	0.812	0.756
	Best Fully-shared	0.861	0.864	0.843	**0.887**	0.860	0.859	0.861	0.856
	Best shared-private	**0.870**†+	**0.866**+	**0.889** †+	0.844+	**0.869** †+	**0.867** †+	**0.871** †+	0.864+
		Co-hyponyme vs Aléatoire				hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	MLP	0.909	0.939	0.954	0.925	0.904	0.936	0.944	0.929
	Shwartz and Dagan (2016b)	0.919	0.946	0.955	0.938	0.842	0.901	0.860	0.946
	Vu and Shwartz (2018)	0.940	0.961	0.962	0.959	0.943	0.962	0.961	0.964⋆
	Best MLP	0.950 ⋆	0.967 ⋆	**0.973**⋆	0.959	**0.947**⋆†	**0.965**⋆†	**0.971**†⋆	0.959†
	Balikas et al. (2019)	0.909	0.940	0.949	0.931	0.911	0.941	0.949	0.932
	Bannour et al. (2020)	0.949	0.966	0.964	**0.969**+	0.908	0.932	0.941	0.923
	Best Fully-shared	0.947	0.965	0.971	0.959	0.944	0.963	0.964	**0.962**
	Best shared-private	**0.951**+	**0.968** +	0.971+	0.964†	0.943+	0.962+	0.969+	0.955+
		Co-hyponyme vs Aléatoire				hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
WEEDS	MLP	0.720	0.449	0.422	0.479	0.726	0.457	0.432	0.485
	Shwartz and Dagan (2016b)	0.769	0.532	0.513	0.552	0.716	0.474	0.423	0.539
	Vu and Shwartz (2018)	0.848	0.691	0.669	0.714	0.833	0.661	0.641	0.682
	Best MLP	0.873 ⋆	0.737⋆	0.729⋆	0.746⋆†	0.886 ⋆	0.746⋆	0.797⋆	0.701⋆
	Balikas et al. (2019)	0.721	0.443	0.422	0.466	0.724	0.462	0.431	0.498
	Bannour et al. (2020)	0.871	0.713	0.754	0.678	**0.924**+	**0.751** +	**0.854**+	0.669
	Best Fully-shared	0.864	0.737	0.685	**0.796**	0.873	0.736	0.727	**0.746**
	Best shared-private	**0.890**†+	**0.761**†+	**0.789**†+	0.736+	0.882+	0.743	0.771	0.717†
		Meronym vs Aléatoire				hyperonyme vs Aléatoire			
	Algorithme	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
BLESS	MLP	0.839	0.748	0.805	0.698	0.845	0.762	0.797	0.731
	Shwartz and Dagan (2016b)	0.855	0.781	0.807	0.756	0.842	0.754	0.804	0.709
	Vu and Shwartz (2018)	0.886	0.820	0.883	0.765	0.882	0.811	0.891⋆	0.744
	Best MLP	0.909 ⋆	0.864 ⋆	0.883	**0.847** ⋆	0.905⋆	0.859 ⋆	0.872	0.847 ⋆†
	Balikas et al. (2019)	0.846	0.759	0.814	0.711	0.848	0.764	0.812	0.721
	Bannour et al. (2020)	0.896	0.837	**0.913**+	0.770	**0.954** +	0.821	0.883	0.769
	Best Fully-shared	0.903	0.850	0.895	0.810	0.906	0.862	0.861	**0.864**
	Best shared-private	**0.912** †+	**0.868**†+	0.890†	**0.847**+	0.916	**0.873**†+	**0.906**†+	0.843+

Table 3.1: Résultats globaux pour toutes les architectures avec l'*embedding* GloVe. Le fractionnement lexical est appliqué. ⋆, † et + indiquent une p-valeur ≤ 0.05 basée sur le t-test supposant des variances d'échantillon inégales des métriques entre respectivement (*Best MLP*) et (Vu and Shwartz 2018), (*Best shared-private*) et (*Best MLP*), et (*Best shared-private*) et (Bannour et al. 2020).

Pour mieux comprendre l'impact de l'ingénierie des caractéristiques, nous illustrons

les résultats impliquant toutes les combinaisons de caractéristiques au sein de chaque famille et toutes les combinaisons de caractéristiques entre les meilleures familles dans la figure 3.2 (a). Au sein de la famille *cosF* uniquement (c'est-à-dire que seules les métriques basées sur le cosinus sont utilisées pour le processus d'apprentissage)[7], les résultats mettent clairement en évidence la question dimensionnelle, étant *cos* et *cosG*1*D* les métriques unidimensionnelles qui présentent les pires résultats individuellement. La deuxième constatation importante réside dans le fait que la combinaison de métriques améliore régulièrement les performances par rapport aux métriques individuelles. En particulier, la combinaison (*cosG*, *cosBr*, *cosG*1*D*) donne les meilleurs résultats dans la grande majorité des cas, et notamment pour l'hyperonymie.

Au sein de la famille *KullF* seule[8], les résultats semblent indiquer que *kullBr* est la caractéristique la moins performante (seule et en combinaison), bien que les régularités soient difficiles à établir car des résultats différents peuvent être observés selon le jeu de données. De même que l'observation précédente, la combinaison de caractéristiques asymétriques fournit de meilleurs résultats pour la grande majorité des cas, ce qui suggère que les valeurs individuelles codent des informations complémentaires.

Dans la famille *PatF*[9], l'encodage BiLSTM semble fournir des résultats supérieurs à l'encodage USE, mais plus important encore, les résultats montrent clairement que les caractéristiques basées sur les patrons lexicaux peuvent être un indice fort pour le processus de classification à condition qu'un grand nombre de patrons lexicaux puisse être extrait, comme cela est montré pour ROOT9 (voir tableau 2.2 pour le nombre de

[7]Points bleus dans la figure 3.2.
[8]Points rouges dans la figure 3.2.
[9]Points noirs dans la figure 3.2.

patrons lexicaux).

De manière plus surprenante, les caractéristiques *CosF* indiquent régulièrement des résultats plus forts que les caractéristiques *KullF* et *PatF* pour les relations asymétriques (hyperonymie et méronymie), suggérant ainsi que la symétrie est une caractéristique importante pour toutes les relations.

Enfin, les résultats montrent clairement que les combinaisons des meilleures caractéristiques par famille[10] surpassent régulièrement les résultats des caractéristiques individuelles des familles, démontrant ainsi leur complémentarité. En particulier, les caractéristiques distributionnelles symétriques et asymétriques se combinent avec succès pour les relations asymétriques, et la combinaison la plus réussie est celle des caractéristiques basées sur les patrons lexicaux et sur le cosinus pour la co-hyponymie. Cependant, seules les caractéristiques distributionnelles symétriques permettent une performance maximale pour la synonymie, ce qui peut facilement se comprendre puisqu'il s'agit d'une relation symétrique. Pour renforcer nos commentaires, nous donnons la distribution des caractéristiques pour les meilleures configurations dans le tableau 3.2 (première ligne) pour tous les jeux de données et toutes les relations.

3.3.2 Modèles multi-tâches

Les résultats des architectures multi-tâches sont présentés dans les lignes 5 à 8 du tableau 3.1. En particulier, le réseau *Best fully-shared* représente le modèle de Balikas et al. (2019) avec un ensemble optimisé de caractéristiques d'entrée, contrairement à leurs paramètres qui reposent sur la concaténation unique des *embeddings* de mots. Les ré-

[10] Points verts dans la figure 3.2.

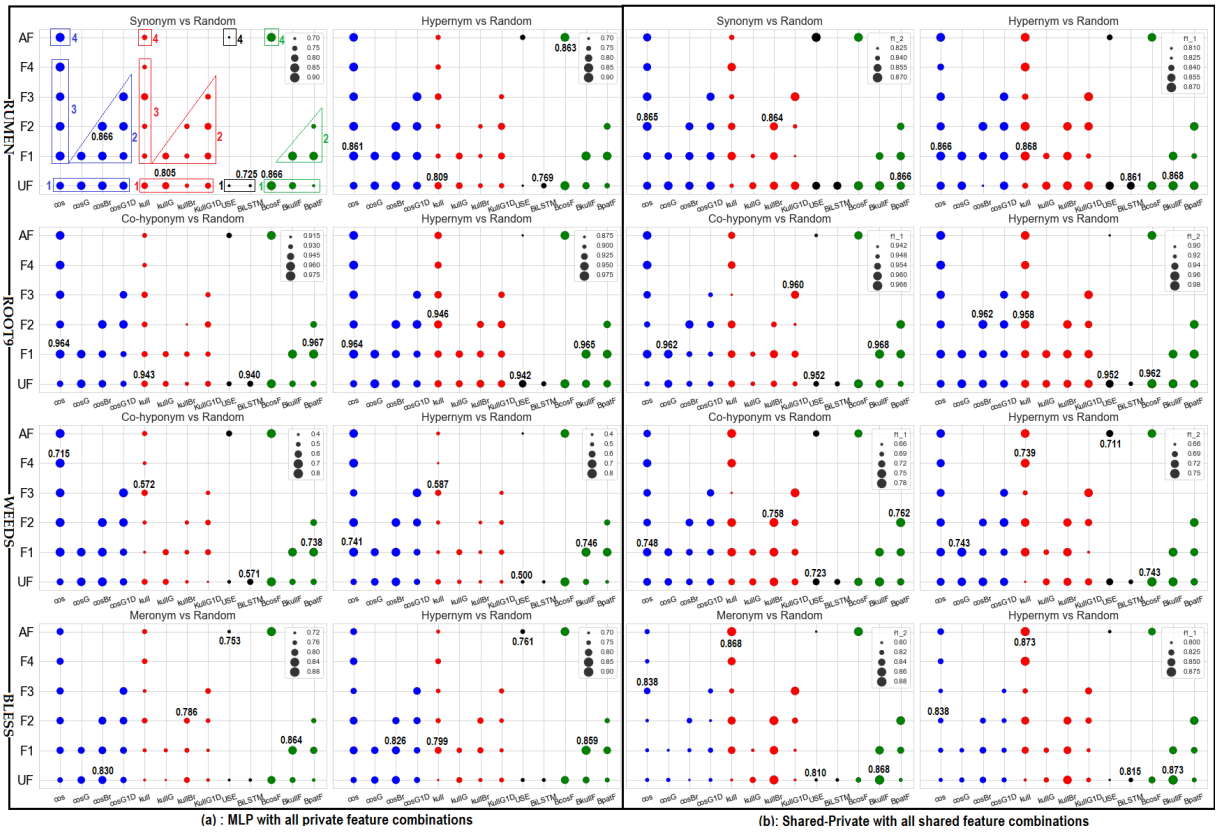

Figure 3.2: Résultats du score F_1 pour toutes les combinaisons de caractéristiques. Le cadre 1 représente toute caractéristique individuelle seule, par exemple (*cosG*,UF) signifie uniquement *cosG*. Le cadre 2 représente toute combinaison 2 par 2 de caractéristiques, par exemple (*cosBr*,F2) correspond à (*cosBr*,*cosG*). Le cadre 3 désigne la suppression d'une caractéristique de l'ensemble de toutes les caractéristiques, par exemple (Cadre bleu 3,F2) désigne (*cos*,*cosBr*,*cosG1D*). Le cadre 4 représente la combinaison de toutes les caractéristiques pour une famille donnée (AF). *BcosF*, *BKullF* et *BpatF* représentent la meilleure combinaison de caractéristiques dans leur famille respective.

sultats montrent clairement la supériorité du réseau *shared-private* (*Best shared-private*) sur le modèle entièrement partagé (*Best fully-shared*) pour la plupart des cas, ce qui suggère que la combinaison d'informations privées et partagées est bénéfique au processus de décision. Cependant, le *Best MLP* est un modèle difficile à battre car le *Best shared-private* surpasse statistiquement l'ancienne architecture 4 fois sur 8, et 2 fois sur 8 sans signification statistique. Mais le contraire n'est vrai que pour la ROOT9 (par rapport au

		Synonyme vs Aléatoire			Hyperonyme vs Aléatoire		
	Algorithme	CosF	KullF	PatF	CosF	KullF	PatF
RUMEN	*Best MLP*	0 1 1 0	0 0 0 0	0 0	0 1 1 1	1 0 0 0	0 1
	Best fully-shared	0 1 1 0	0 0 0 0	0 0	1 1 1 1	0 0 0 0	0 0
	Best shared-private	0 0 0 0	0 0 0 0	1 1	0 0 0 0	0 1 1 1	0 0
		Co-hyponymee vs Aléatoire			Hyperonyme vs Aléatoire		
	Algorithme	CosF	KullF	PatF	CosF	KullF	PatF
ROOT9	*Best MLP*	0 1 1 1	0 0 0 0	1 1	0 1 1 1	1 0 1 1	0 0
	Best fully-shared	0 1 1 0	0 0 0 0	0 0	1 1 1 1	0 0 0 0	0 0
	Best shared-private	0 1 1 0	0 1 1 0	0 0	1 1 1 1	0 0 0 0	1 0
		Co-hyponymee vs Aléatoire			Hyperonyme vs Aléatoire		
	Algorithme	CosF	KullF	PatF	CosF	KullF	PatF
WEEDS	*Best MLP*	1 1 1 0	0 0 0 0	0 1	0 1 1 1	1 1 0 1	0 0
	Best fully-shared	1 1 1 0	0 0 0 0	0 0	0 1 1 1	0 0 0 0	0 0
	Best shared-private	0 0 0 0	0 1 1 0	1 0	1 1 0 0	0 0 0 0	0 0
		Meronym vs Aléatoire			hyperonyme vs Aléatoire		
	Algorithme	CosF	KullF	PatF	CosF	KullF	PatF
BLESS	*Best MLP*	0 0 1 0	0 1 1 0	0 0	1 0 1 0	0 1 1 1	0 0
	Best fully-shared	0 0 1 0	1 0 1 1	0 1	0 0 1 0	0 0 0 0	0 1
	Best shared-private	0 0 0 0	1 1 1 1	0 0	0 0 0 0	1 1 1 1	0 0

Table 3.2: Meilleures combinaisons de caractéristiques pour tous les modèles. 0 et 1 représentent l'absence ou la présence, respectivement, de la caractéristique donnée au sein de sa famille, où l'ordre est donné par des équations 3.4, 3.12 and 3.14.

score F_1), où *Best MLP* dépasse statistiquement *Best shared-private*.

Le problème important des architectures privées et partagées est de comprendre comment elles distribuent l'espace des caractéristiques entre les couches privées et partagées. À cette fin, nous analysons la figure 3.2 (b), qui montre les combinaisons de caractéristiques pour la couche partagée, c'est-à-dire lorsque deux tâches sont apprises simultanément. Notez que dans ce cas, les meilleures combinaisons des modèles privés (appris séparément) limitent le processus d'apprentissage. La première conclusion principale est que les caractéristiques asymétriques distributionnelles (*KullF*) concurrencent régulièrement les caractéristiques basées sur le cosinus (*CosF*), et surpassent même clairement ces dernières pour Bless, ce qui n'est absolument pas le cas dans les modèles privés. La même conclusion peut être tirée pour les caractéristiques basées sur les patrons lexicaux *PatF*, dont l'impact est beaucoup plus important dans les couches partagées que dans

		Synonyme vs Aléatoire		Hyperonyme vs Aléatoire	
	Algorithme	GloVe	JoSE	GloVe	JoSE
RUMEN	MLP	0.754	0.730	0.757	0.731
	Best MLP	0.865	**0.870**	0.862	0.863
	Best shared-private	0.866	0.869⋆	**0.867**⋆	0.865†
		Co-hyponymee vs Aléatoire		Hyperonyme vs Aléatoire	
	Algorithme	GloVe	JoSE	GloVe	JoSE
ROOT9	MLP	0.939	0.927	0.936	0.922
	Best MLP	0.967	0.967	0.965	0.963
	Best shared-private	**0.968**⋆	0.966	0.962	**0.966**⋆†
		Co-hyponymee vs Aléatoire		Hyperonyme vs Aléatoire	
	Algorithme	GloVe	JoSE	GloVe	JoSE
WEEDS	MLP	0.449	0.458	0.457	0.455
	Best MLP	0.737	0.758	0.746	0.754
	Best shared-private	0.761	**0.764**⋆†	0.743	**0.759**⋆†
		Meronyme vs Aléatoire		hyperonyme vs Aléatoire	
	Algorithme	GloVe	JoSE	GloVe	JoSE
BLESS	MLP	0.748	0.810	0.762	0.798
	Best MLP	0.864	0.865	0.859	0.850
	Best shared-private	**0.868**⋆	0.865	0.873	**0.874**†

Table 3.3: F_1 scores with GloVe and JoSE. ⋆ and † denote p-value ≤ 0.05 based on the t-Test assuming unequal sample variances of metric values between respectively (*Best shared-private* JoSE) vs. (*Best shared-private* GloVe) and (*Best shared-private* JoSE) vs. (*Best MLP* JoSE).

les modèles privés par rapport à *CosF.* Cela suggère que lorsque les modèles privés se concentrent davantage sur les caractéristiques symétriques, les modèles privés-partagés tirent parti des caractéristiques asymétriques pour capturer la dissimilarité des tâches (en effet, dans les tâches concurrentes, il y a toujours au moins une tâche asymétrique).

Une autre observation intéressante est que les meilleurs modèles ne sont généralement pas une combinaison de différentes caractéristiques de la même famille. Seuls 2 cas sur 8 montrent des résultats améliorés par la combinaison de caractéristiques. En fait, ces résultats suggèrent que les couches privées et partagées équilibrent de manière distincte l'espace des caractéristiques de la famille. Nous voyons clairement cette situation dans le tableau 3.2 en examinant la complémentarité des vecteurs de caractéristiques d'entrée des modèles privés (ligne 1) et partagés-privés (ligne 3). Par exemple, lors de la maximisation de la tâche d'hyperonymie dans le modèle privé et partagé sur

Rumen, les vecteurs d'entrée privés sont ($cosG, cosBr, cosG1D, kull, \overline{pat}_{*,\text{BiLSTM}}$) pour l'hyperonymie et ($cosG, cosBr$) pour la synonymie, tandis que le vecteur d'entrée partagé est ($kullG, kullBr, kullG1D$)[11]. Il est intéressant de noter que cette situation ne se vérifie pas pour les modèles entièrement partagés, car ils sont clairement biaisés en faveur des métriques basées sur le cosinus et incluent rarement des caractéristiques asymétriques distributionnelles et basées sur les patrons.

3.3.3 *Embeddings* sphériques

Nous proposons de comparer nos architectures basées sur les caractéristiques avec les *embeddings* relationnels, à savoir JoSE (Meng et al. 2019), l'idée sous-jacente étant de comprendre comment les stratégies basées sur les caractéristiques peuvent se comparer et éventuellement s'ajouter à des espaces sémantiques neuronaux finement réglés. Les résultats sont illustrés dans le tableau 3.3.

Les résultats du modèle de base MLP ne mettent pas en évidence un avantage clair des *embeddings* relationnelles par rapport aux *embeddings* généraux comme GloVe, Bless étant la seule exception avec une légère améliorations pour Weeds. Cependant, il est intéressant de noter que la proportion d'amélioration est beaucoup plus importante pour les *embeddings* JoSE lors de l'introduction de combinaisons de caractéristiques. En effet, alors que le modèle MLP avec GloVe dépasse la version JoSE 5 fois sur 8, le modèle *Best MLP* avec JoSE dépasse la version GloVe 5 fois sur 8, ce qui suggère que les *embeddings* sphériques sont sensibles à l'ingénierie des caractéristiques.

Enfin, si les architectures partagées et privées fournissent globalement les meilleurs

[11] Si on maximise le modèle pour l'hypéronymie.

résultats, il est difficile d'établir une distinction claire entre les deux types d'*embeddings*, bien qu'une petite tendance en faveur des *embeddings* JoSE semble émerger. En effet, alors que la relation d'hyperonymie est mieux traitée par les *embeddings* relationnels (3 configurations sur 4), la méronymie est mieux traitée par GloVe bien qu'étant une relation asymétrique. En ce qui concerne les relations symétriques (synonymie et co-hyponymie), la situation converge légèrement vers les *embeddings* relationnels avec de meilleurs résultats dans 2 expériences sur 3.

3.4 Conclusion

Dans ce chapitre, nous avons proposé la définition de caractéristiques distributives asymétriques dans des espaces continus, basée sur la divergence de Kullback-Leibler, et suggéré de les combiner avec des familles de caractéristiques distributives symétriques et basées sur des patrons, en utilisant l'ingéniérie des caractéristiques. Nous avons proposé d'analyser l'impact de la combinaison de caractéristiques dans des contextes multitâches, qui combinent des couches privées et partagées. Les résultats ont mis en évidence les avantages de la combinaison de caractéristiques dans les modèles privés, et ils ont souligné l'importance des caractéristiques asymétriques (distributionnelles et paradigmatiques) dans les couches partagées. De plus, les architectures partagées-privées ont montré la capacité d'équilibrer les familles de caractéristiques entre les couches privées et partagées, tirant ainsi pleinement parti de la plupart des caractéristiques dans le processus de décision.

Une personne qui n'a jamais commis d'erreurs n'a jamais innové

Albert Einstein

4

L'apprentissage multi-classes

Jusqu'à présent, nous n'avons vu que les cas de classification binaire, mais ce n'est pas la situation réelle. En effet, nous aimerions distinguer pour une paire de mots s'il existe une relation lexico-sémantique qui les relie, i.e impliquant synonymie, hyperonymie, méronymie ou co-hyponymie dans un seul apprentissage. Dans ce cadre, nous proposons une nouvelle architecture d'apprentissage multi-classes mutli-tâches.

La classification multi-classes traite des problèmes impliquant plus que deux classes, par exemple la détection des émotions (positif, neutre ou négatif). Dans ce cadre, une multitude d'approches a été proposée pour résoudre ces problèmes comme des arbres de décision multi-classes, l'algorithme des K-plus proches voisins, etc. D'autres approches reposent sur une décomposition du problème en un ensemble de sous-problèmes binaires, dont les résultats de chaque classifieur sont combinés par un vote majoritaire pour déterminer la solution multi-classes finale. Dans ce cadre, deux stratégies principales ont été largement utilisées, à savoir la stratégie *one-vs-rest* et la stratégie *one-vs-one*. Selon (Hsu and Lin 2002), les deux stratégies offrent souvent des performances similaires, tandis que des résultats contrastés sont observés par (Pawara et al. 2020). Une autre approche très courante consiste à trouver une hiérarchie entre les classes pour construire une architecture d'apprentissage hiérarchique, en supposant qu'il existe une certaine relation qui lie les différentes étiquettes de classe. Dans ce cadre, deux stratégies principales ont été proposées, à savoir l'approche basée sur les instances (Zupan et al. 1999), et la stratégie basée sur les prédictions (Godbole et al. 2002, Silva-Palacios et al. 2017).

Dans ce chapitre, nous proposons d'améliorer les architectures *one-vs-rest* et hiérarchique de l'état de l'art en intégrant des stratégies multi-tâches à l'apprentissage multi-classes. À cette fin, nous proposons une architecture multi-tâches, dans laquelle chaque classifieur *one-vs-rest* est appris dans une configuration multi-tâches, ce qui permet de prendre en compte l'interdépendance des classes en une seule étape d'apprentissage. Ainsi, chaque classifieur *one-vs-rest* devrait bénéficier de l'apprentissage concurrent des autres tâches. En particulier, nous proposons deux architectures multi-tâches de type

one-vs-rest, (*fully-shared* et *shared-private*). En effet, les architectures multi-tâches *shared-private* ont montré leur capacité à surpasser d'autres architectures de l'état de l'art, notamment pour l'identification des relations lexico-sémantiques comme nous l'avons démontré dans les chapitres précédents. Enfin, nous proposerons des architectures multi-tâches inter-classes dans lesquelles nous tenterons de trouver des groupes de classes cognitivement liées parmi les classes étudiées. Des expériences sur six jeux de données de référence pour l'analyse des sentiments (Socher et al. 2013, Nakov et al. 2016), la détection des émotions (Chatterjee et al. 2019), la classification thématique (Greene and Cunningham 2006) et la classification de relations lexico-sémantiques (Balikas et al. 2019) montrent que nos architectures améliorent régulièrement les performances par rapport aux stratégies *one-vs-rest* de l'état de l'art, et concurrencent fortement les autres stratégies multi-classes.

4.1 Stratégie *one-vs-rest* multi-tâches

4.1.1 Architecture générale

La stratégie par décomposition *one-vs-rest* transforme un problème à N classes en N problèmes binaires (ici, tâches), où chaque classe doit être discriminée de toutes les autres classes. Ainsi, la décision finale est donnée par le classifieur avec la probabilité d'inférence maximale (vote majoritaire). Dans notre approche *one-vs-rest* multi-tâches, les N classifieurs binaires sont appris dans un cadre multi-tâches, ce qui permet d'obtenir une performance accrue pour chacune des tâches si celles-ci sont liées, i.e. s'il existe une relation (de similarité ou cognitive) entre les classes (Caruana 1998). Nous proposons d'introduire deux modèles multi-tâches multi-classes: le modèle *fully-shared* et le mod-

èle *shared-private*. Dans l'architecture *fully-shared* (figure 4.1), un réseau de neurones apprend une représentation partagée commune à toutes les tâches, à partir de laquelle toutes les décisions des classifieurs binaires doivent être apprises. Dans l'architecture *shared-private*, une couche privée est concaténée à la couche partagée, permettant à chaque classifieur binaire d'apprendre sa fonction de décision à partir d'informations spécifiques à la tâche mais aussi communes à toutes les tâches (Liu et al. 2017).

Les deux modèles sont formellement définis comme suit pour exactement une couche générique partagée et une couche privée par classifieur binaire. Soit X_k un vecteur d'entrée[1], la couche partagée $S(X_k)$ est calculée comme dans l'équation 4.1, où W_{S^k} est une matrice de poids, b_{S^k} est un vecteur de biais, et $k \in [1, K]$ où K est le nombre de couches partagées.

$$S(X_k) = \sigma(W_{S^k} X_k + b_{S^k}) = X_{k+1} \tag{4.1}$$

La couche privée $H^j(Z_q)$ de chaque classifieur binaire indépendant j, qui résout la tâche T_j ($j \in [1, N]$) est définie dans l'équation 4.2, où $q \in [1, Q]$ et Q est le nombre de couches cachées. Notez que pour l'architecture *fully-shared*, $Z_1 = S(X_K)$, et $Z_1 = S(X_K) \oplus X$ pour le modèle *shared-private*, où $\oplus$ représente la concaténation.

$$H^j(Z_q) = \sigma(W^j_{H^q} Z_q + b^j_{H^q}) = Z_{q+1} \tag{4.2}$$

La classe prédite est donnée par le maximum de toutes les probabilités prédites $O^1, ..., O^N$, qui sont définies dans l'équation 4.3.

[1] $X_1 = X$, où X est le vecteur de texte encodé.

$$O^j = \sigma(W^j_O H^j(Z_Q) + b^j_O) \tag{4.3}$$

Les paramètres sont mis à jour en minimisant l'entropie croisée binaire $E(O^j, y^j)$, où y^j est l'étiquette de référence. Ainsi, les poids de la couche partagée sont mis à jour en minimisant alternativement l'entropie croisée binaire de chaque classifieur, tandis que les couches privées ne sont mises à jour que pour une tâche donnée.

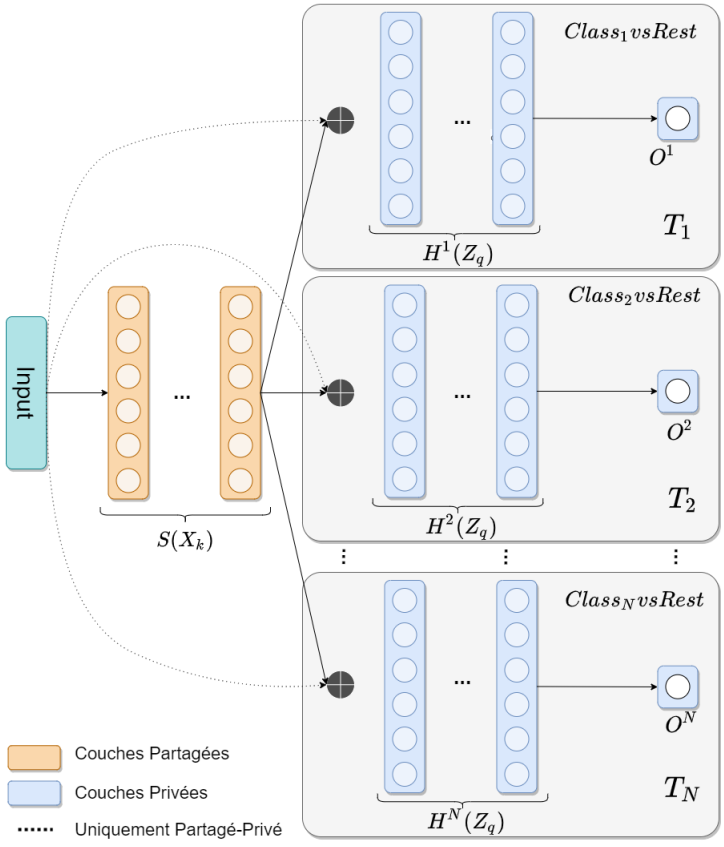

Figure 4.1: Architectures *fully-shared* et *shared-private* de la stratégie *one-vs-rest* multi-tâches. Les connexions en pointillés ne sont présentes que dans l'architecture *shared-private*.

4.1.2 Architectures inter-classes

Les architectures *fully-shared* et *shared-private* de la stratégie *one-vs-rest* multi-tâches supposent que toutes les tâches interagissent entre elles et sont liées, i.e les couches partagées encodent l'information commune de toutes les classes. Dans cette section, nous supposons qu'il peut exister un groupe de $1 < k < N$ classes qui sont liées entre elles étant donné k un hyper-paramètre à optimiser. Formellement, chaque couche partagée encode l'information commune d'une combinaison de k classes. Pour l'architecture partagée-privée k par k, nous aurons $\frac{N!}{k!(N-k)!}$ couches partagées à apprendre. Pour la simplification des explications, nous supposons que $k = 2$, soient i et j deux classes étudiées. Nous aurons ainsi une couche partagée pour chaque combinaison de deux classes. Formellement, de manière similaire à l'équation 4.1, nous avons :

$$S^{ij}(X_k^{ij}) = \sigma(W_{S_k^{ij}} X_k^{ij} + b_{S_k^{ij}}) = X_{k+1}^{ij} \tag{4.4}$$

Pour la partie privée de l'architecture, la seule différence par rapport à l'architecture précédente est la définition de Z_1^i. En effet, pour chaque entrée de la partie privée, nous concatenons le vecteur d'entrée de la tâche associée avec les couches partagées qui encodent l'information de la tâche en question. Formellement, nous avons:

$$Z_1^i = X^i \bigoplus_{j=1, j\neq i}^{j=N} S^{ij}(X_K^{ij})$$

L'architecture complète est illustrée pour le cas $k = 2$ et N=3 dans la figure 4.2.

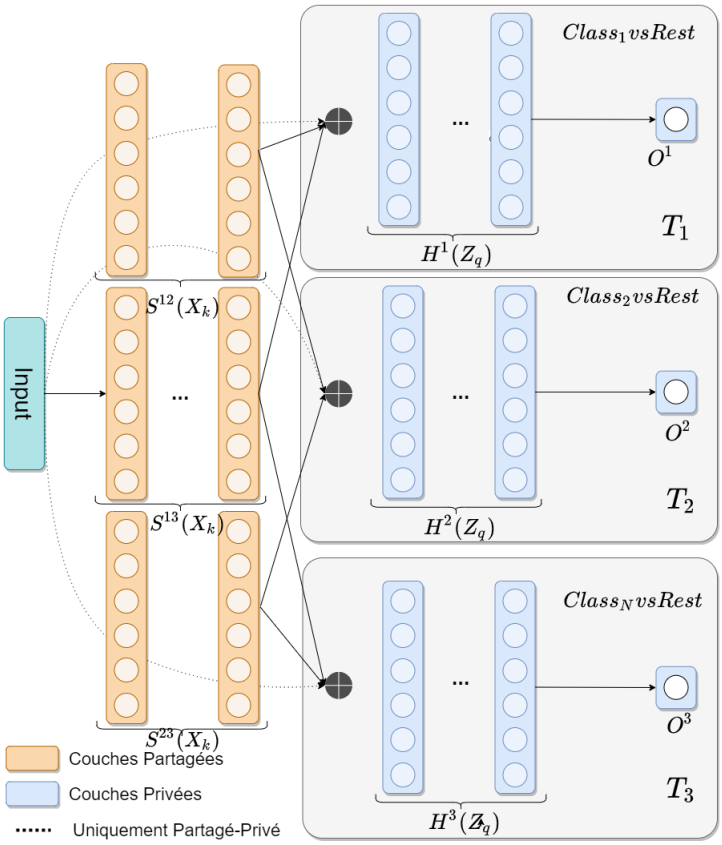

Figure 4.2: Architectures *fully-shared* et *shared-private* de la stratégie K par K pour le cas spécifique 3 tâches et $K = 2$.

4.2 Jeux de données

Notre évaluation porte sur six jeux de données qui s'articulent autour de quatre applications connues de classification textuelle: l'analyse des sentiments, la détection des émotions, la classification thématique, l'identification de relations lexico-sémantiques. Pour l'analyse des sentiments, nous utilisons le jeu de données Rotten Tomatoes [2] (Socher et al. 2013), qui consiste en des critiques de films notées sur une échelle de cinq valeurs,

[2]https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews

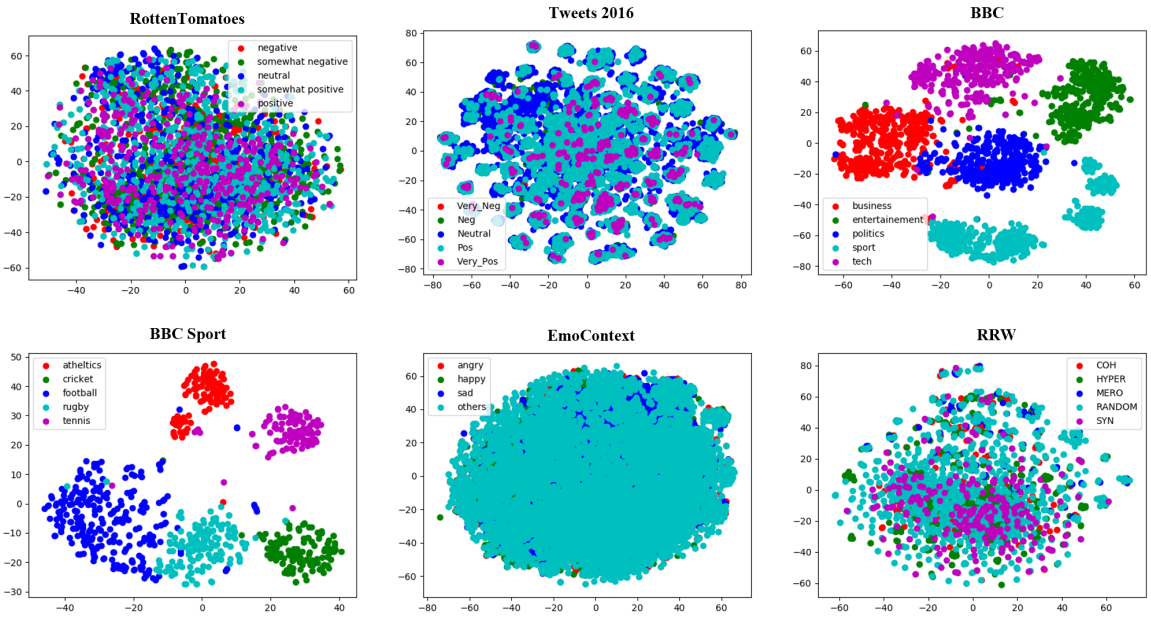

Figure 4.3: Représentation T-SNE des donnée.

et le jeu de données Tweet2016[3] (Nakov et al. 2016), qui consiste en des tweets portant un avis sur les principales compagnies aériennes américaines (notation sur une échelle de cinq valeurs). Pour la détection des émotions, nous utilisons le jeu de données EmoContext[4] (Chatterjee et al. 2019), qui consiste en des dialogues agent/utilisateur classés suivant quatre émotions (*happy*, *sad*, *angry*, *others*). Pour la classification thématique, nous utilisons les jeux de données BBC et BBC Sport[5] (Greene and Cunningham 2006), qui consistent en des articles de presse classés suivant cinq domaines d'actualité, soit généraux pour BBC soit sportifs pour BBC Sport. Pour l'identification des relations lexico-sémantiques, nous utilisons le jeu de données spécifique RRW créé

[3] http://alt.qcri.org/semeval2016/task4/
[4] https://www.humanizing-ai.com/emocontext.html
[5] https://www.kaggle.com/c/learn-ai-bbc

dans le chapitre 2 en concaténant trois jeux existants: Rumen[6] (Balikas et al. 2019), ROOT9[7] (Santus et al. 2016c) et Weeds[8] (Weeds et al. 2014). RRW conjugue ainsi les relations d'hyperonymie, de co-hyponymie, de méronymie et de synonymie. Le détail est présenté dans le tableau 4.1 ainsi que leur représentation T-SNE[9] dans la figure 4.3. Nous remarquons, la simplicité des problèmes BBC et BBC Sport, les jeux de données Rotten Tomatoes, Tweet2016, EmoContext et RRW semblent beaucoup plus complexes à résoudre.

Jeu de données	Distribution des classes				
Rotten Tomatoes	*negative* 313	*somewhat negative* 686	*neutral* 508	*somewhat positive* 718	*positive* 402
Tweet2016	*very negative* 138	*negative* 2201	*neutral* 10081	*positive* 7830	*very positive* 382
BBC	*business* 510	*entertainment* 386	*politics* 417	*sport* 511	*tech* 401
BBC Sport	*athletics* 101	*cricket* 124	*football* 265	*rugby* 147	*tennis* 100
EmoContext	*happy* 4243	*sad* 5463	*angry* 5506	*others* 14948	- -
RRW	*co-hyponymy* 5283	*hypernymy* 12004	*meronymy* 2943	*random* 25741	*synonymy* 6728

Table 4.1: Distribution des classes par jeu de données.

4.3 Configurations expérimentales

En ce qui concerne les configurations expérimentales, chaque jeu de données est aléatoirement divisé en trois sous-ensembles, à savoir l'ensemble d'entraînement (50 %), de validation (20 %) et de test (30 %). Afin de prendre en compte le déséquilibre des

[6] https://github.com/Houssam93/MultiTask-Learning-NLP/tree/master

[7] https://github.com/esantus/ROOT9

[8] https://github.com/SussexCompSem/learninghypernyms

[9] Le T-SNE *(t-Distributed Stochastic Neighbor Embedding)* est une technique de réduction de la dimensionnalité particulièrement bien adaptée à la visualisation d'ensembles de données à haute dimension.

classes lors de la phase d'apprentissage[10], nous utilisons Adasyn (He et al. 2008) pour équilibrer les instances sur les classes[11]. Au vu du Tableau 4.1, seul le jeu de données Tweet2016 reçoit ce pré-processus. Afin de coder l'entrée textuelle, nous utilisons l'architecture d'encodage pré-entraînée *Universal Sentence Encoder*[12] (Cer et al. 2018) avec une dimension de 512 pour l'espace de représentation. Pour les relations lexico-sémantiques, chaque paire de mots est encodée par la concaténation des représentations GloVe (Pennington et al. 2014) avec une dimension de 300.

Trois architectures sont implémentées comme bases de référence, une pour chaque paradigme multi-classes: stratégie de référence (*MultiClass*), par décomposition (*one-vs-rest*), et hiérarchique (*Hierarchical*). *MultiClass* est un réseau de neurones de type *feed-forward* avec des couches cachées activées par une fonction sigmoïde, et la couche de sortie par une fonction *softmax*. L'entropie croisée binaire est utilisée comme fonction de perte. *One-vs-rest* implémente une série de N réseaux de neurones de type *feed-forward* avec des couches cachées indépendantes. Toutes les couches sont activées par une fonction sigmoïde, l'entropie croisée binaire est utilisée comme fonction de perte, et le processus de décision se base sur la fonction maximum. *Hierarchical* correspond à l'algorithme proposé par (Silva-Palacios et al. 2017), que nous avons implémenté spécifiquement pour notre recherche.

Nos deux architectures *one-vs-rest* multi-tâches (*one-vs-rest fully-shared* et *one-vs-rest shared-private*) reçoivent exactement la même configuration que le *one-vs-rest* pour permettre une comparaison équitable. Le nombre de couches cachées ($K \in [1, 2]$ et

[10] Un problème bien connu des problèmes multi-classes.
[11] Les ensembles de validation et de tests restent déséquilibrés.
[12] https://tfhub.dev/google/universal-sentence-encoder/4

$Q \in [1, 2]$), d'époques ($[1..100]$) et de neurones ($[5, 20, 50, 100, 150, 200, 300]$) sont des hyperparamètres optimisés par un *grid search*. Les poids sont initialisés avec une distribution uniforme échelonnée (Glorot and Bengio 2010) et mis à jour à l'aide de Adam (Kingma and Ba 2014) avec un taux d'apprentissage fixé à 0,001[13].

4.4 Résultats

4.4.1 Modèles multi-tâches

Chaque architecture est évaluée sur 25 exécutions différentes (partitions d'entraînement et de validation différentes mais avec l'ensemble de test identique) et la signification statistique est calculée pour six métriques différentes afin de refléter au mieux les différences entre les architectures. En particulier, nous calculons le F1 Micro, le F1 Macro, le F1 Pondérée, l'ACC Macro (précision), l'AUNU (surface moyenne sous la courbe) et l'AUNP (surface pondérée sous la courbe) (Hand and Till 2001, Hossin and Sulaiman 2015). Les résultats sont présentés dans le Tableau 4.2. De plus, nous présentons dans la figure 4.5 les boxplots associées aux 25 exécutions pour le jeu de données RRW. Les résultats des boxplots de tous les jeux de données sont disponibles en annexe A.

Les résultats montrent clairement que la stratégie *one-vs-rest* multi-tâches dépasse l'approche classique *one-vs-rest* en tenant compte de la relation entre les classes. Ceci est particulièrement pertinent pour Tweet2016, EmoContext et RRW, où les architectures *fully-shared* et *shared-private* dépassent statistiquement les résultats de la stratégie *one-vs-rest*. En particulier, pour Tweet2016, des améliorations maximales de 2,7% F1 Micro, 2,4% F1 Macro et 2,9% F1 pondérée peuvent être atteintes. Les améliorations

[13]Tous les codes sources sont disponibles à l'adresse suivante :https://github.com/Houssam93/Multi-classes/tree/main/Code

	Algorithme	F1 Micro	F1 Macro	ACC Macro	AUNU	AUNP	F1 Pondérée
Tweet2016	*MultiClass*	0.514	0.362	0.805	0.663	0.655	0.537
	one-vs-rest	0.495	0.346	0.798	0.653	0.641	0.517
	Hierarchical	0.492	0.351	0.797	0.656	0.634	0.510
	one-vs-rest fully-shared	**0.522**⋆+	**0.370**⋆†+	**0.809**⋆+	**0.666**⋆†+	**0.661**⋆†+	**0.546**⋆†+
	one-vs-rest shared-private	0.514⋆+	0.362⋆+	0.806⋆+	0.662⋆+	0.653⋆+	0.539⋆+
Rotten Toma.	*MultiClass*	**0.403**	0.339	**0.761**	**0.598**	**0.605**	0.369
	one-vs-rest	0.386	0.348	0.754	0.595	0.598	0.367
	Hierarchical	0.388	0.280	0.755	0.580	0.592	0.326
	one-vs-rest fully-shared	0.400⋆+	0.338+	0.760⋆+	0.596+	0.603⋆+	0.368+
	one-vs-rest shared-private	0.393	**0.353**†+	0.757	0.600⋆+	0.602⋆+	**0.372+**
BBC	*MultiClass*	0.954	0.951	0.982	0.971	0.971	0.952
	one-vs-rest	0.968	0.967	0.987	**0.980**	0.980	0.968
	Hierarchical	**0.969**	0.968	0.987	**0.980**	0.980	**0.969**
	one-vs-rest fully-shared	**0.969**⋆	**0.969**⋆	**0.988**⋆	**0.980**⋆	**0.981**⋆	**0.969**⋆
	one-vs-rest shared-private	**0.969**	0.968	**0.988**	**0.980**	**0.981**	**0.969**⋆
BBC Sport	*MultiClass*	0.947	0.941	0.979	0.963	0.965	0.946
	one-vs-rest	0.964	0.966	0.986	0.978	0.976	0.964
	Hierarchical	0.970	**0.974**	**0.988**	**0.983**	**0.980**	0.970
	one-vs-rest fully-shared	0.951	0.952	0.981	0.970	0.968	0.951
	one-vs-rest shared-private	**0.971**⋆†	0.973⋆†	**0.988**⋆†	**0.983**⋆†	**0.980**⋆†	**0.971** ⋆†
EmoContext	*MultiClass*	0.825	0.808	0.912	0.869	0.866	0.825
	one-vs-rest	0.824	0.806	0.912	0.866	0.864	0.823
	Hierarchical	0.821	0.804	0.911	0.868	0.864	0.820
	one-vs-rest fully-shared	**0.828**⋆†+	**0.810**⋆†+	**0.914**⋆†+	**0.872**⋆+	**0.869**⋆†+	**0.827**⋆†+
	one-vs-rest shared-private	0.826⋆†+	0.809⋆+	0.913⋆†+	0.870	0.867⋆+	0.826⋆†+
RRW	*MultiClass*	0.587	0.494	0.835	0.686	0.704	0.583
	one-vs-rest	0.597	0.51	0.839	0.687	0.702	0.59
	Hierarchical	0.608	0.512	0.843	0.688	0.708	0.59
	one-vs-rest fully-shared	0.617⋆†+	0.519†+	**0.847**⋆†+	0.697⋆†+	0.716⋆†+	0.602⋆†+
	one-vs-rest shared-private	**0.618**⋆†+	**0.521**⋆†+	**0.847**⋆†+	**0.698**⋆†+	**0.717**⋆†+	**0.609**⋆†+

Table 4.2: Moyenne des scores F1 Micro, F1 Macro, ACC Macro, AUNU, AUNP et F1 Pondérée sur 25 exécutions pour tous les jeux de données. La pertinence statistique est basée sur le test t en supposant des variances d'échantillon inégales, et ⋆, † et + mettent en evidence une valeur de $p \leq 0.05$ par rapport aux algorithmes de références *one-vs-rest*, *Multiclass* et *Hierarchical* respectivement.

pour RRW sont du même ordre, alors qu'elles sont moins expressives pour EmoContext, mais néanmoins statistiquement pertinentes. Pour Rotten Tomatoes et la classification thématique (c'est-à-dire BBC et BBC Sport), la situation diffère légèrement car seule une des architectures multi-tâches *one-vs-rest* dépasse statistiquement le *one-vs-rest* :

one-vs-rest fully-shared pour Rotten Tomatoes et BBC, et *one-vs-rest shared-private* pour BBC Sport. En particulier, pour BBC Sport, des améliorations maximales de 0,7% F1 Micro, 0,7% F1 Macro et 0,7% F1 pondérée peuvent être atteintes. La différence entre les architectures *fully-shared* et *shared-private* peut être due à la force de la relation entre les classes comme expliqué dans (Qureshi et al. 2020). En effet, les architectures *fully-shared* semblent être plus performantes que les architectures *shared-private* lorsque les classes sont fortement liées, alors que le contraire se produit lorsque les classes sont moins fortement liées.

La comparaison entre la stratégie multi-tâches de type *one-vs-rest* et les implémentations *MultiClass* et *Hierarchical* nécessite une analyse approfondie. Pour Tweet2016, EmoContext et RRW, la stratégie *fully-shared* dépasse statistiquement les deux approches. En particulier, pour Tweet2016, des améliorations maximales de 0,8% (resp. 3%) F1 Micro, 0,8% (resp. 1,9%) F1 Macro et 0,9% (resp. 3,6%) F1 pondérée peuvent être atteintes par rapport à *MultiClass* (resp. *Hierarchical*). Ces résultats sont confirmés pour l'architecture *shared-private* dans le contexte de EmoContext et de RRW, mais pas pour Tweet2016, où les résultats ne peuvent pas être distingués de *MultiClass*. Pour BBC Sport, l'architecture *shared-private* est statistiquement plus performante que les résultats de *MultiClass*, mais les résultats sont similaires à ceux de *Hierarchical*. Il convient de noter que pour ce jeu de données, l'architecture *MultiClass* affiche les pires résultats, ce qui indique clairement la présence d'une relation entre les classes. Pour BBC, les résultats des stratégies multi-tâches *one-vs-rest* ne sont pas statistiquement supérieurs à ceux des stratégies *MultiClass* et *Hierarchical*, ce qui montre que ce jeu de données est certainement celui qui met en évidence le moins de relations entre les

classes. Enfin, pour Rotten Tomatoes, notre modèle *shared-private* démontre des résultats statistiquement supérieurs par rapport à *MultiClass* pour le F1 Macro, où une amélioration de 1,4% peut être atteinte. Mais c'est la seule mesure où cela se produit. En ce qui concerne les autres métriques, nous obtenons des résultats similaires à ceux du *MultiClass* pour les deux versions *one-vs-rest* multi-tâches. Cependant, des améliorations significatives sont obtenues par les deux architectures multi-tâches par rapport à *Hierarchical*, à une exception près, avec des améliorations maximales de 1,2% de F1 Micro, 7,3% de F1 Macro et 4,6% de F1 pondérée.

Afin de mieux comprendre le comportements des différentes architectures, nous présentons leurs performances (F1 Micro) à nombre de paramètres identiques pour tous les jeux de données dans la figure 4.4. Les résultats montrent que l'architecture hiérarchique est celle qui donne de moins bons résultats quelques soit le niveau de paramètres. Inversement, l'architecture *one-vs-rest shared-private* met en évidence les résultats les plus élevés et stables indépendemment du nombre de paramètres, en particulier par rapport à la version multi-classes standard (*MultiClass*), surtout pour un niveau de paramètres faible. Ces résultats confirment les hypothèses des réseaux multi-tâches qui agissent comme des auto-régulateurs et permettent une meilleure généralisation (Caruana 1998).

4.4.2 Analyse des boxplots pour le cas des relations lexico-sémantiques

Les résultats obtenus pour RRW illustrés en figure 4.5 montrent que la stratégie multi-tâches de type *one-vs-rest* est toujours plus performante que l'algorithme classique de type *one-vs-rest*. L'approche multi-tâches est également statistiquement plus perfor-

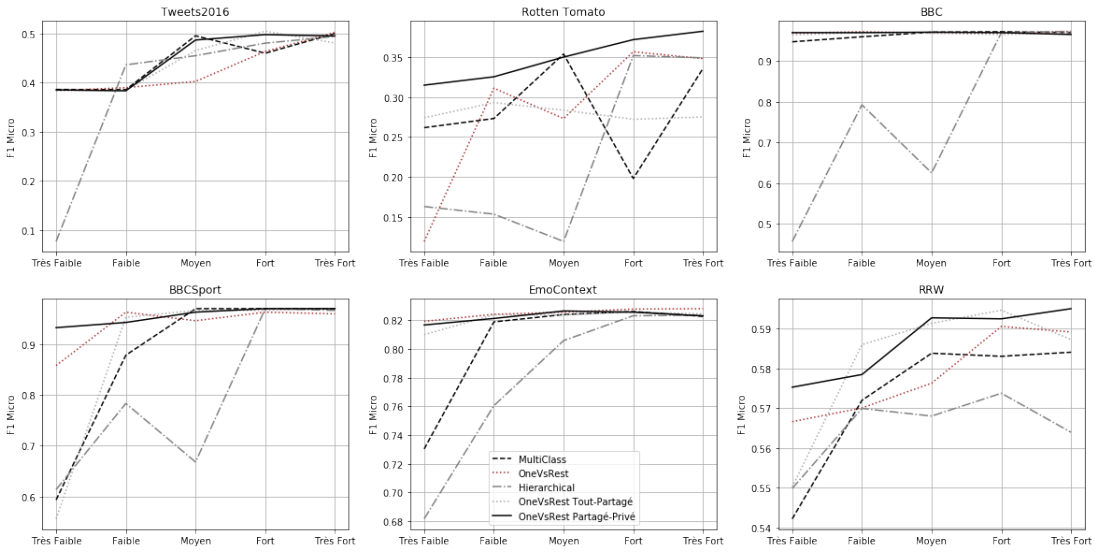

Figure 4.4: Évolution du F1 Micro pour toutes les architectures avec des paramètres fixes pour tous les jeux de données. Afin de faciliter la lecture et du fait de l'impossibilité de garantir exactement le même nombre de paramètres, 5 niveaux ont été définis pour regrouper la taille des architectures explorées : *Très faible* entre 10k et 30k paramètres, *Faible* entre 30k et 100k, *Moyen* entre 100k et 200k, *Fort* entre 200k et 500k, et finalement, *Très Fort* entre 500k et 1M.

mante que les réseaux de neurones classiques et les algorithmes hiérarchiques (*Hierarchical*). Il est important de noter que ces derniers algorithmes donnent des résultats avec une variance élevée pour la majorité des jeux de données, alors que notre approche est plus robuste à la variation. Les modèles inter-classes *k* par *k* ne parviennent pas à apporter des améliorations claires et constantes par rapport aux stratégies multi-tâches où K=1. Il est probable que ce type d'architectures complexes nécessite des réglages plus spécifiques afin d'améliorer leur performance. En annexe, nous montrons tous les boxplots pour tous les jeux de données et notons que les modèles multi-tâches K par K donnent des résultats interessants pour Tweet2016.

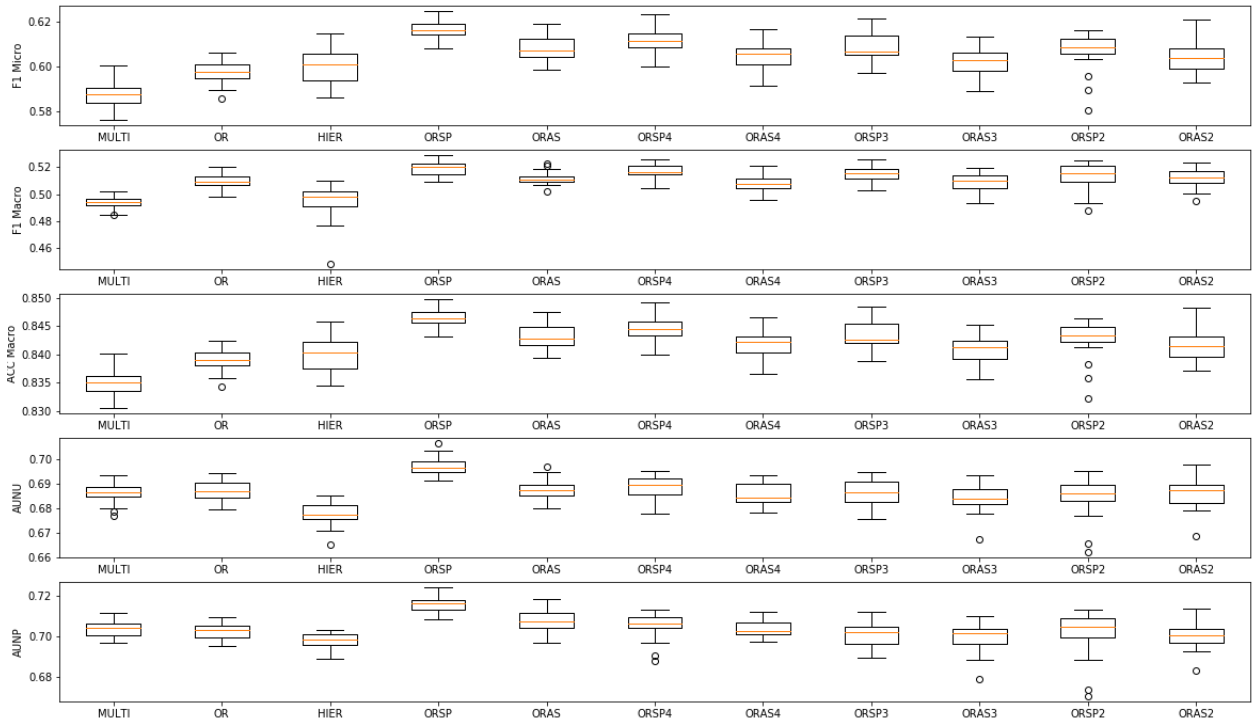

Figure 4.5: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée RRW. (1) MULTI : Modèle MLP multi-classes; (2) OR: Modèle *one-vs-rest*; (3) HIER : Modèle *Hierarchical*; (4) ORSPK : Modèle *one-vs-rest shared-private* K par K; (5) ORASK : Modèle *one-vs-rest all-shared* K par K.

5

Conclusion

Dans ce manuscrit, nous avons présenté une architecture générique multi-tâches pour l'identification de relations lexico-sémantiques. En particulier, ce modèle neuronal encode à la fois des caractéristiques distributionnelles et des caractéristiques basées sur des patrons lexico-sémantiques, et introduit des configurations conscientes du mot unique qui permettent de mieux traiter les relations asymétriques. Les résultats de l'évaluation

sur quatre jeux de données de référence (RUMEN, ROOT9, Weeds et Bless) pour deux et trois tâches confirment que les architectures partagées-privées surpassent les modèles récents proposés dans (Balikas et al. 2019) ainsi que les modèles à classe unique. Il est intéressant de noter que le modèle BiLSTM partagé proposé pour coder les patrons lexicaux renforce la qualité de l'encodage lorsque peu de patrons existent.

Dans le chapitre 3, nous avons proposé la définition de caractéristiques distributionnelles asymétriques dans des espaces continus basés sur la divergence de Kullback-Leibler, et suggéré de les combiner avec des familles de caractéristiques distributionnelles symétriques et basées sur des patrons à l'aide d'un processus de sélection de caractéristiques. Nous avons proposé d'analyser l'impact de la combinaison de caractéristiques dans des contextes multi-tâches, qui combinent des couches privées et partagées. Les résultats ont mis en évidence les avantages de la combinaison de caractéristiques dans les modèles privés, et ils ont souligné l'importance des caractéristiques asymétriques (distributionnelles et paradigmatiques) dans les couches partagées. En outre, les architectures partagées-privées ont montré la capacité d'équilibrer les familles de fonctionnalités entre les couches privées et partagées, tirant ainsi pleinement parti de la plupart des fonctionnalités dans le processus de décision.

Dans le chapitre 4, nous avons proposé des nouvelles stratégies *one-vs-rest* multi-tâche, pour la résolution de problèmes multi-classe. Les améliorations restent statistiquement pertinentes mais un peu faible. Cependant, des gains de performances non négligeables sont obtenus pour l'identification des relations lexico-sémantiques. L'introduction des architectures *shared-private k* par *k* ont néanmoins montré des difficultés pour améliorer les performances. une nouvelle direction de recherche consiste à trouver des stratégie

qui permettent à décider en amont les classes à partager. En effet, un tel travail permetterait de réduire les paramètres d'apprentissage ainsi que les données bruitées.

Finalement, une stratégie de classification basée sur des patchs pour aborder l'identification de relations sémantiques lexicales a montré sa capacité à améliorer les performances de l'état de l'art.

Travail futur :

- Intégrer l'ingéniérie des caractéristiques dans les architectures multi-classe
- Sélection des tuples de classe pour les architectures k par k afin de réduire les paramètres d'apprentissage dans un cadre de classificaiton multi-classe multi-tâche.
- Généraliser la notion de patch à toutes les architectures proposées dans ce mémoire, en particulier en adaptant l'ingéniérie des caractéristiques.
- Créer des *embeddings* conceptuels (comme ConceptNet) sans l'apport des resspirces externes (ontologies).
- Multimodalité: Intégrer ll'aspect visuel des mots avec des images et profiter des architectures multi-tâches pour améliorer les performances.

A

Annexes

A.1 Rétropropagation du gradient

Nous allons maintenant envisager l'entraînement d'un perceptron multicouche assez général pour l'association de formes. en utilisant l'algorithme BP. L'apprentissage est effectué de manière supervisée et nous supposons donc qu'un ensemble de paires de motifs (ou associations) (ou associations) : $\mathbf{s}^{(q)} : \mathbf{t}^{(q)}, q = 1, 2, \ldots, Q$ est donné. Les

vecteurs d'apprentissage $\mathbf{s}^{(q)}$ ont N composantes,

$$\mathbf{s}^{(q)} = \begin{bmatrix} s_1^{(q)} & s_2^{(q)} & \dots & s_N^{(q)} \end{bmatrix},$$

et leurs cibles $\mathbf{t}^{(q)}$ ont M composantes,

$$\mathbf{t}^{(q)} = \begin{bmatrix} t_1^{(q)} & t_2^{(q)} & \dots & t_M^{(q)} \end{bmatrix}.$$

Comme dans la règle du Delta, les vecteurs d'apprentissage sont présentés un par un au réseau pendant l'apprentissage. Supposons qu'à l'étape t du processus d'apprentissage, un vecteur d'apprentissage $\mathbf{s}^{(q)}$ pour un q particulier est présenté en entrée, $\mathbf{x}(t)$, au réseau. Le signal d'entrée peut être propagé vers l'avant à travers le réseau en utilisant les équations de la dernière section et l'ensemble actuel de poids et de biais pour obtenir la sortie correspondante du réseau, $\mathbf{y}(t)$. Les poids et les biais sont ensuite ajustés à l'aide de l'algorithme de descente la plus abrupte afin de minimiser le carré de l'erreur pour cet apprentissage. le carré de l'erreur pour ce vecteur d'apprentissage :

$$E = \|\mathbf{y}(t) - \mathbf{t}(t)\|^2,$$

où $\mathbf{t}(t) = \mathbf{t}^{(q)}$. est le vecteur cible correspondant au vecteur d'apprentissage choisi $\mathbf{s}^{(q)}$.

Cette erreur carrée E est une fonction de tous les poids et biais de l'ensemble du réseau puisque $\mathbf{y}(t)$ en dépend. Nous devons trouver l'ensemble des règles de mise à

jour de ces derniers en nous basant sur l'algorithme de la descente la plus abrupte :

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha \, ; \frac{\partial E}{\partial w_{ij}^{(\ell)}(t)}$$

$$b_{j}^{(\ell)}(t+1) = b_{j}^{(\ell)}(t) - \alpha \, ; \frac{\partial E}{\partial b_{j}^{(\ell)}(t)},$$

où $\alpha (> 0)$ est le taux d'apprentissage.

Pour calculer ces dérivées partielles, nous devons comprendre comment E dépend des poids et des biais. Premièrement, E dépend explicitement de la sortie du réseau $\mathbf{y}(t)$. (les activations de la dernière couche, $\mathbf{a}^{(L)}$), qui dépend ensuite de l'entrée nette dans la L-*ime* couche, $\mathbf{n}^{(L)}$. A son tour, $\mathbf{n}^{(L)}$ est donné par les activations de la couche précédente et les poids et biais de la couche L. La relation explicite est la suivante : par souci de brièveté, la dépendance au pas t est omise

$$\begin{aligned} E &= \|\mathbf{y} - \mathbf{t}(t)\|^2 = \|\mathbf{a}^{(L)} - \mathbf{t}(t)\|^2 = \|f^{(L)}(\mathbf{n}^{(L)}) - \mathbf{t}(t)\|^2 \\ &= \left\| f^{(L)} \left(\sum_{i=1}^{N_{L-1}} a_i^{(L-1)} w_{ij}^{(L)} + b_j^{(L)} \right) - \mathbf{t}(t) \right\|^2 . \end{aligned}$$

Il est alors facile de calculer les dérivées partielles de E par rapport aux éléments de $\mathbf{W}^{(L)}$ et $\mathbf{b}^{(L)}$ en utilisant la règle de la chaîne pour la différentiation. Nous avons

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = \sum_{n=1}^{N_L} \frac{\partial E}{\partial n_n^{(L)}} \frac{\partial n_n^{L)}}{\partial w_{ij}^{L)}}.$$

Remarquez que la somme est nécessaire dans l'équation ci-dessus pour une application correcte de la règle de la chaîne. Nous définissons maintenant le vecteur de sensibilité

pour une couche générale ℓ comme ayant des composantes

$$s_n^{(\ell)} = \frac{\partial E}{\partial n_n^{(\ell)}} \qquad n = 1, 2, \ldots, N_\ell.$$

On appelle cela la sensibilité du neurone $X_n^{(\ell)}$ car elle donne la variation de l'erreur de sortie, E, par unité de changement dans l'entrée nette qu'il reçoit.

Pour la couche L, il est facile de calculer directement le vecteur de sensibilité en utilisant la règle de la chaîne pour obtenir

$$s_n^{(L)} = 2\left(a_n^{(L)} - t_n(t)\right)\dot{f}^{(L)}(n_n^{(L)}), \qquad n = 1, 2, \ldots, N_L.$$

où $\dot{f}$ désigne la dérivée de la fonction de transfert f. Nous savons également que

$$\frac{\partial n_n^{(L)}}{\partial w_{ij}^{(L)}} = \frac{\partial}{\partial w_{ij}^{(L)}}\left(\sum_{m=1}^{N_{L-1}} a_m^{(L-1)} w_{mn}^{(L)} + b_n^{(L)}\right) = \delta_{nj} a_i^{(L-1)}.$$

Par conséquent, nous avons

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = a_i^{(L-1)}\, ; s_j^{(L)}.$$

De même,

$$\frac{\partial E}{\partial b_j^{(L)}} = \sum_{n=1}^{N_L} \frac{\partial E}{\partial n_n^{(L)}} \frac{\partial n_n^{L)}}{\partial b_j^{L)}},$$

et puisque

$$\frac{\partial n_n^{(L)}}{\partial b_j^{(L)}} = \delta_{nj},$$

nous avons

$$\frac{\partial E}{\partial b_j^{(L)}} = s_j^{(L)}.$$

Pour une couche générale, ℓ, on peut écrire

$$\frac{\partial E}{\partial w_{ij}^{(\ell)}} = \sum_{n=1}^{N_\ell} \frac{\partial E}{\partial n_n^{(\ell)}} \frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \sum_{n=1}^{N_\ell} s_n^{(\ell)} \frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}}.$$

$$\frac{\partial E}{\partial b_j^{(\ell)}} = \sum_{n=1}^{N_\ell} \frac{\partial E}{\partial n_n^{(\ell)}} \frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}} = \sum_{n=1}^{N_\ell} s_n^{(\ell)} \frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}}.$$

Puisque

$$n_n^{(\ell)} = \sum_{m=1}^{N_{\ell-1}} a_m^{(\ell-1)} w_{mn}^{(\ell)} + b_n^{(\ell)}, \qquad j = 1, 2, \ldots, N_\ell,$$

nous avons

$$\frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \delta_{nj} a_i^{(\ell-1)}$$

$$\frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}} = \delta_{nj},$$

et donc

$$\frac{\partial E}{\partial w_{ij}^{(\ell)}} = a_i^{(\ell-1)} \; ; s_j^{(\ell)}, \qquad \frac{\partial E}{\partial b_j^{(\ell)}} = s_j^{(\ell)}.$$

Par conséquent, les règles de mise à jour pour les poids et les biais sont (maintenant nous nous remettons la dépendance à l'indice d'étape t)

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha \; ; a_i^{(\ell-1)}(t) \; ; s_j^{(\ell)}(t)$$

$$b_j^{(\ell)}(t+1) = b_j^{(\ell)}(t) - \alpha \; ; s_j^{(\ell)}(t),$$

Afin d'utiliser ces règles de mise à jour, nous devons être en mesure de calculer les vecteurs de sensibilité $\mathbf{s}^{(\ell)}$ pour $\ell = 1, 2, \ldots, L-1$. D'après leur définition

$$s_j^{(\ell)} = \frac{\partial E}{\partial n_j^{(\ell)}} \qquad j = 1, 2, \ldots, N_\ell,$$

nous devons savoir comment E dépend de $n_j^{(\ell)}$. La clé pour calculer ces dérivées partielles est de noter que $n_j^{(\ell)}$ dépend à son tour de $n_i^{(\ell-1)}$ pour $i = 1, 2, \ldots, N_{\ell-1}$, car l'entrée nette de la couche ℓ dépend de l'activation de la couche précédente, $\ell - 1$, qui à son tour dépend de l'entrée nette de la couche $\ell - 1$. Plus précisément,

$$n_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} f^{(\ell-1)}(n_i^{(\ell-1)}) w_{ij}^{(\ell)} + b_j^{(\ell)}$$

pour $j = 1, 2, \ldots, N_\ell$ Nous avons donc pour la sensibilité de la couche $\ell - 1$

$$\begin{aligned} s_j^{(\ell-1)} &= \frac{\partial E}{\partial n_j^{(\ell-1)}} = \sum_{i=1}^{N_\ell} \frac{\partial E}{\partial n_i^{(\ell)}} \frac{\partial n_i^{(\ell)}}{\partial n_j^{(\ell-1)}} \\ &= \sum_{i=1}^{N_\ell} s_i^{(\ell)} \frac{\partial}{\partial n_j^{(\ell-1)}} \left(\sum_{m=1}^{N_{\ell-1}} f^{(\ell-1)}(n_m^{(\ell-1)}) w_{mi}^{(\ell)} + b_i^{(\ell)} \right) \\ &= \sum_{i=1}^{N_\ell} s_i^{(\ell)} \dot{f}^{(\ell-1)}(n_j^{(\ell-1)}) w_{ji}^{(\ell)} = \dot{f}^{(\ell-1)}(n_j^{(\ell-1)}) \sum_{i=1}^{N_\ell} w_{ji}^{(\ell)} s_i^{(\ell)}. \end{aligned}$$

Ainsi, la sensibilité d'un neurone de la couche $\ell - 1$ dépend des sensibilités de tous les neurones de la couche ℓ. Il s'agit d'une relation de récursion pour les sensibilités du réseau puisque les sensibilités de la dernière couche L sont connues. Pour trouver les activations ou les entrées nettes d'une couche donnée, nous devons alimenter l'entrée

depuis la gauche du réseau et avancer vers la couche en question. Cependant, pour trouver les sensibilités d'une couche donnée, nous devons commencer par la dernière couche et utiliser la relation de récursion pour revenir à la couche en question. C'est pourquoi l'algorithme de formation est appelé rétro-propagation.

En résumé, l'algorithme de rétropropagation pour la formation d'un perceptron multicouche est le suivant

1. Définir α. Initialiser les poids et les biais.

2. Pour le pas $t = 1, 2, \ldots$, répétez les étapes a-e jusqu'à convergence.

   a Définissez $\mathbf{a}^{(0)} = \mathbf{x}(t)$ pris au hasard dans l'ensemble d'apprentissage.

   b Pour $\ell = 1, 2, \ldots, L$, calculez

$$\mathbf{n}^{(\ell)} = \mathbf{a}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)} \qquad \mathbf{a}^{(\ell)} = f^{(\ell)}(\mathbf{n}^{(\ell)}).$$

   c Calculez pour $n = 1, 2, \ldots, N_L$

(boîte à couleurs)

$$s_n^{(L)} = 2\left(\mathbf{a}_n^{(L)} - \mathbf{t}_n(t)\right)\dot{f}^{(L)}(\mathbf{n}_n^{(L)}).$$

   d Pour $\ell = L-1, \ldots, 2, 1$ et $j = 1, 2, \ldots, N_\ell$, calculez

$$s_j^{(\ell)} = \dot{f}^{(\ell)}(n_j^{(\ell)}) \sum_{i=1}^{N_{\ell+1}} w_{ji}^{(\ell+1)} s_i^{(\ell+1)}.$$

   e Pour $\ell = 1, 2, \ldots, L$, mettre à jour

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha\, ; a_i^{(\ell-1)}(t)\, ; s_j^{(\ell)}(t),$$

$$b_j^{(\ell)}(t+1) = b_j^{(\ell)}(t) - \alpha\, ; s_j^{(\ell)}(t).$$

A.2 Boxplots des architectures multi-tâches pour des problèmes multi-classes

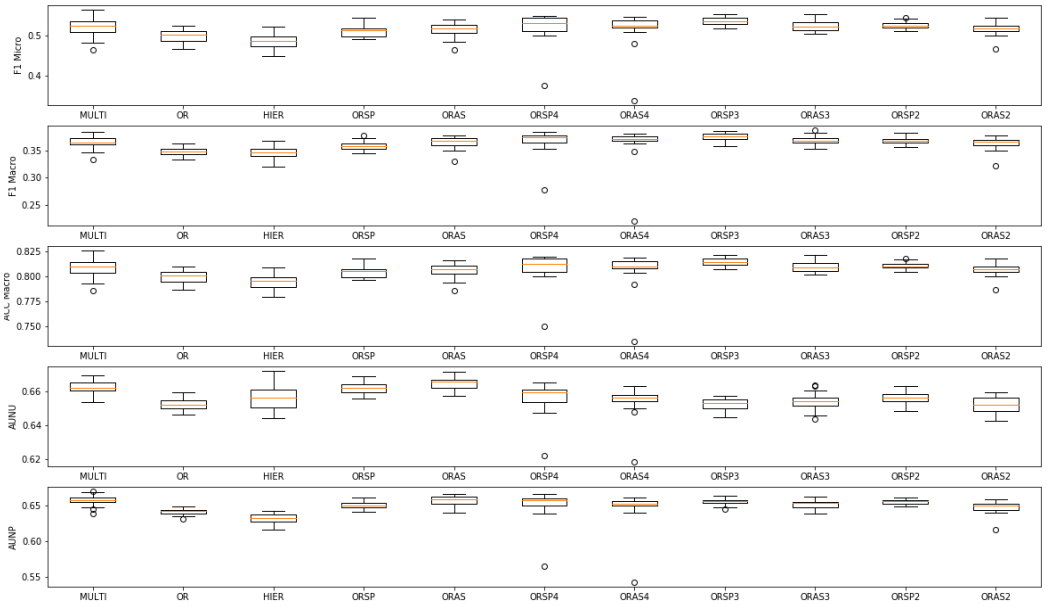

Figure A.1: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée Tweet2016

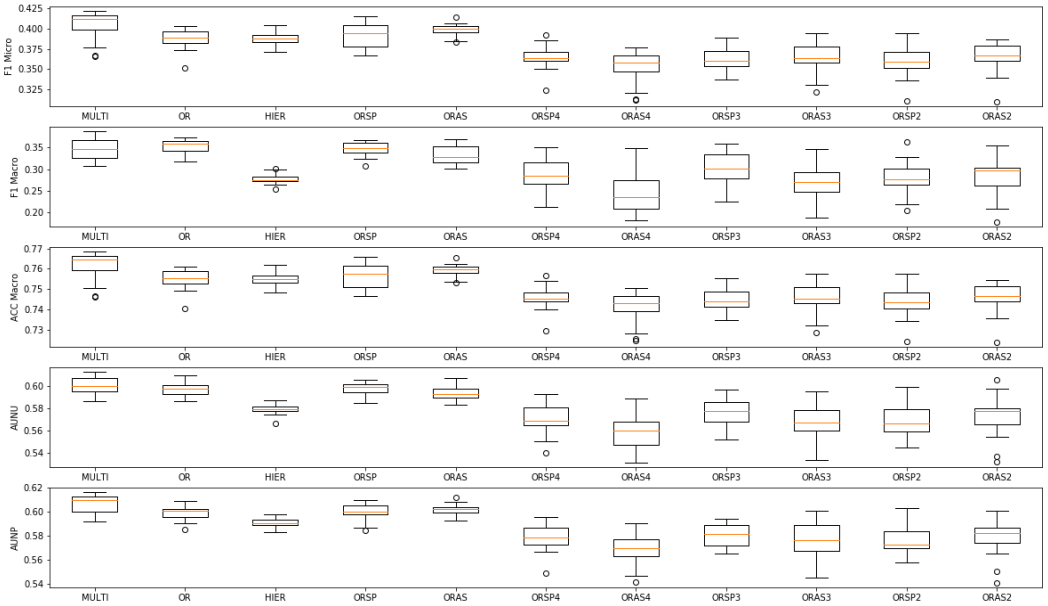

Figure A.2: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée Rotten Toma.

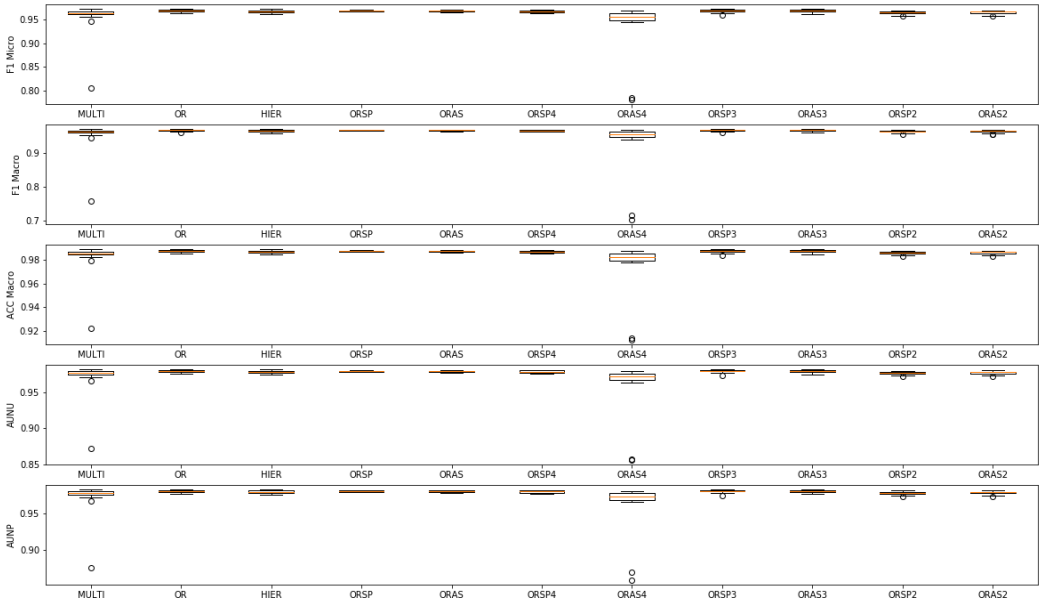

Figure A.3: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée BBC

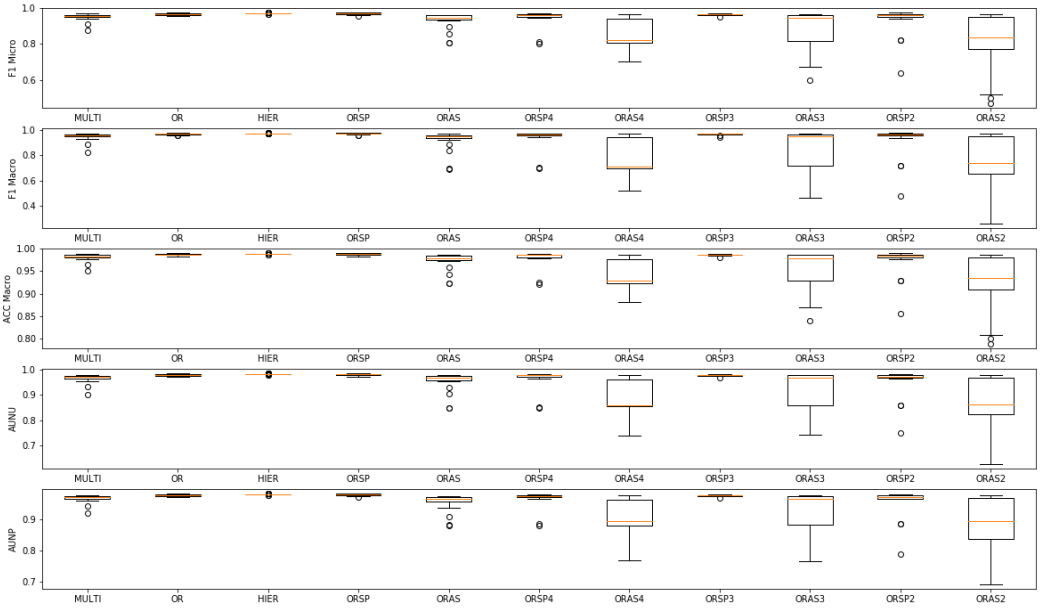

Figure A.4: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée BBC Sport

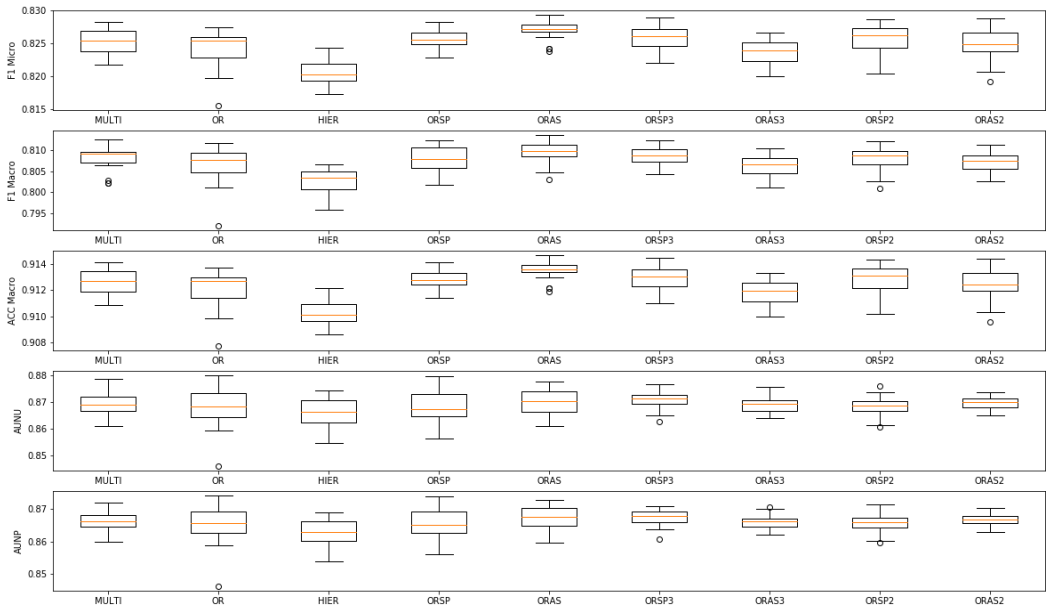

Figure A.5: Boxplots des différents modèles sur les 25 exécutions pour le jeu de donnée EmoContext

Références

Erin Allwein, Robert Schapire, and Yoram Singer. 2000. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141.

Mohammed Attia, Suraj Maharjan, Younes Samih, Laura Kallmeyer, and Thamar Solorio. 2016. Cogalex-v shared task: Ghhh - detecting semantic relations via word embeddings. In *Workshop on Cognitive Aspects of the Lexicon*, pages 86–91.

Georgios Balikas, Gaël Dias, Rumen Moraliyski, Houssam Akhmouch, and Massih-Reza Amini. 2019. Learning lexical-semantic relations using intuitive cognitive links. In *41st European Conference on Information Retrieval (ECIR)*, pages 3–18.

Georgios Balikas, Simon Moura, and Massih-Reza Amini. 2017. Multitask learning for fine-grained twitter sentiment analysis. In *40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1005–1008.

Nesrine Bannour, Gaël Dias, Youssef Chahir, and Houssam Akhmouch. 2020. Patch-based identification of lexical semantic relations. In *42nd European Conference on Information Retrieval (ECIR)*, pages 126–140.

Oren Barkan. 2017. Bayesian neural word embedding. In *31st AAAI Conference on Artificial Intelligence*, volume 31.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics (GEMS)*, pages 1–10.

Giulia Benotto. 2015. *Distributional Models for Semantic Relations: A Sudy on Hyponymy and Antonymy*. Ph.D. thesis, University of Pisa.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *34th AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 7456–7463.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *13th International Workshop on Semantic Evaluation (SemEval)*, pages 39–48.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

François Chollet. 2015. Keras. `https://keras.io`.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1107–1116.

Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, page 4171–4186.

Thomas Dietterich and Ghulum Bakiri. 1994. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 886–897.

R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu. 2015. Learning semantic hierarchies: A continuous vector space approach. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):461–471.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining distributed vector representations for words. In *1st workshop on vector space modeling for natural language processing*, pages 95–101.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1758–1768.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

Shantanu Godbole, Sunita Sarawagi, and Soumen Chakrabarti. 2002. Scaling multiclass support vector machines using inter-class confusion. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 513–518.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *23rd International Conference on Machine Learning (ICML)*, pages 377–384.

David Hand and Robert Till. 2001. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Haibo He, Bai Yang, Garcia Edwardo, and Li Shutao. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1322–1328.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th Conference on Computational Linguistics (COLING)*, pages 539–545.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mohammad Hossin and Mohammad Nasir Sulaiman. 2015. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1–11.

Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Abhik Jana, Nikhil Reddy Varimalla, and Pawan Goyal. 2020. Using distributional thesaurus embedding for co-hyponymy detection. In *12th Language Resources and Evaluation Conference (LREC)*, pages 5766–5771.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *4th Workshop on Representation Learning for NLP (RepL4NLP)*, pages 72–83.

Neha Kathuria, Kanika Mittal, and Anusha Chhabra. 2017. A comprehensive survey on query expansion techniques, their issues and challenges. *International Journal of Computer Applications*, 168(12).

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *2nd International Conference on Learning Representations (ICLR)*.

Ron Kohavi and Dan Sommerfield. 1995. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *1st International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 192–197.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1110–1118.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 970–976.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8206–8215.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR Workshops*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *10th International Workshop on Semantic Evaluation (SemEval)*, pages 1–18.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017a. Hierarchical embeddings for hypernymy detection and directionality. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017b. Distinguishing antonyms and synonyms in a pattern-based neural network. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 76–85.

Guobin Ou and Yi Lu Murphey. 2007. Multi-class pattern classification using neural networks. *Pattern Recognition*, 40(1):4–18.

Pornntiwa Pawara, Emmanuel Okafor, Marc Groefsema, Sheng He, Lambert Schomaker, and Marco Wiering. 2020. One-vs-one classification for deep neural networks. *Pattern Recognition*, 108:107528.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Syed Qureshi, Gaël Dias, Sriparna Saha, and Mohammed Hasanuzzaman. 2020. Improving depression level estimation by concurrently learning emotion intensity. *IEEE Computational Intelligence Magazine*, 15:47–59.

Marek Rei, Daniela Gerz, and Ivan Vulić. 2018. Scoring lexical entailment with a supervised directional similarity network. In *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 638–643.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *25th International Conference on Computational Linguistics (COLING)*, pages 1025–1036.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363, Melbourne, Australia. Association for Computational Linguistics.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016a. The cogalex-v shared task on the corpus-based identification of semantic relations. In *5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V)*, pages 69–79.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016b. Nine features in a random forest to learn taxonomical semantic relations. In *10th International Conference on Language Resources and Evaluation (LREC)*, pages 4557–4564.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016c. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4557–4564, Portorož, Slovenia. European Language Resources Association (ELRA).

Enrico Santus, Vered Shwartz, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 65–75.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *4th Workshop on Linked Data in Linguistics (LDL) associated to Association for Computational Linguistics and Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 64–69.

Vered Shwartz and Ido Dagan. 2016a. Cogalex-v shared task: Lexnet - integrated path-based and distributional method for the identification of semantic relations. *CoRR*, abs/1610.08694.

Vered Shwartz and Ido Dagan. 2016b. Path-based vs. distributional information in recognizing lexical semantic relations. In *5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2389–2398.

Daniel Silva-Palacios, Cesar Ferri, and María José Ramírez-Quintana. 2017. Improving performance of multiclass classification by inducing class hierarchies. *Procedia Computer Science*, 108:1692–1701.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *17th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 1297–1304.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 801–808.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Zhiyang Teng, Duy-Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1629–1638.

Tu Vu and Vered Shwartz. 2018. Integrating multiplicative features into supervised distributional methods for lexical entailment. In *7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 160–166.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1134–1145.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1671–1682.

Chengyu Wang and Xiaofeng He. 2020. BiRRE: Learning bidirectional residual relation embeddings for supervised hypernymy detection. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3630–3640.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *5th International Conference on Computational Linguistics (COLING)*, pages 2249–2259.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *20th International Conference on Computational Linguistics (COLING)*, pages 1015–1021.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *33rd Conference on Artificial Intelligence (AAAI)*, pages 7370–7377.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327.

Sijia Zhou and Xin Li. 2020. Feature engineering vs. deep learning for paper section identification: Toward applications in chinese medical literature. *Information Processing & Management*, 57(3):102206.

Blaž Zupan, Marko Bohanec, Janez Demšar, and Ivan Bratko. 1999. Learning by discovering concept hierarchies. *Artificial Intelligence*, 109(1-2):211–242.