DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE Nº 601
Mathématique et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par
François TORREGROSSA

REPRÉSENTATION DES MOTS ET DES CONNAISSANCES
CONSTRUCTION, UTILISATION ET ÉVALUATION DES PLONGEMENTS EUCLIDIENS ET HYPERBOLIQUES

Thèse présentée et soutenue à Rennes le 16 décembre 2021
Unité de recherche : IRISA — UMR6074

Rapporteurs avant soutenance :
Dr. Alexandre ALLAUZEN Professeur, ESPCI, Dauphine Université PSL
Dr. Elena CABRIO Maître de conférences, Université Côte d'Azur

Composition du jury :
Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition ne comprend que les membres présents

Examinateurs :	Dr. Alexandre ALLAUZEN	Professeur, ESPCI, Dauphine Université PSL
	Dr. Elena CABRIO	Maître de conférences, Université Côte d'Azur
	Dr. Céline HUDELOT	Professeure, CentraleSupélec - Laboratoire MICS
	Dr. Benjamin PIWOWARSKI	Chargé de recherche, CNRS, LIP6, Paris
Dir. de thèse :	Dr. Guillaume GRAVIER	Directeur de recherche, CNRS, IRISA, Rennes
Co-encadrants :	Dr. Vincent CLAVEAU	Chargé de recherche, CNRS, IRISA, Rennes

Invité :
Cyrille LEROUX Ingénieur R&D, Solocal, Rennes

Remerciements

Je tiens à remercier Guillaume Gravier, Vincent Claveau, Nihel Kooli, Robin Allesiardo et Cyrille Leroux pour leur encadrement et leur soutien durant ma thèse. J'ai toujours trouvé beaucoup de bienveillance et de ressources dans nos nombreuses discussions, merci pour cela.

Pour leur engagement dans l'évaluation de mes travaux, j'adresse mes sincères remerciements à Alexandre Allauzen et Elena Cabrio pour avoir accepté d'être les rapporteurs de ma thèse. Je remercie au même titre Céline Hudelot et Benjamin Piwowarski en tant que membre du jury. Enfin, je remercie Gwénolé Lecorvé, et à nouveau Alexandre Allauzen, pour leur implication dans les comités individuelles de suivi de thèse.

Je remercie cordialement l'entreprise Solocal pour avoir financé la thèse, et plus particulièrement pour m'avoir fait confiance. Merci à Christophe, Guillaume, Timothée, Cyrille, Mohamed et à l'ensemble des membres de la tribu Search pour leur accueil et leur accompagnement. Je préserve d'excellents souvenirs de nos discussions. Je remercie aussi l'ensemble des membres de l'équipe LinkMedia de l'IRISA pour leurs échanges.

Je remercie mes proches, en particulier mes parents, Philippe et Guylène, pour leur soutien et leur amour. Enfin Ismérie, je te remercie de m'accompagner et de m'épauler depuis 10 ans.

Sommaire

Remerciements **iii**

Introduction générale **11**

I Plongements de mots et gestion des connaissances **17**

1 État de l'art sur les plongements de mots **19**

1.1 Les plongements : construire un espace sémantique continu à partir des mots 20

1.1.1 Contexte et besoins 20

1.1.2 Formalisation 21

1.2 Plongements statiques 22

1.2.1 Plongements euclidiens statiques 22

1.2.2 Plongements non euclidiens statiques 27

1.3 Plongements contextuels 33

1.3.1 Plongements autorégressifs 34

1.3.2 Plongements autoencoding 35

1.4 Caractérisation des plongements 36

2 Evaluation des plongements de mots **39**

2.1 Evaluations, entre complexité et nécessité 40

2.2 Les modes d'évaluation 40

2.2.1 Evaluations globales 41

2.2.2 Evaluations intrinsèques 45

2.2.3 Evaluations extrinsèques 49

2.2.4 Discussion : l'évaluation parfaite 50
2.3 Corrélations entre les évaluations 51
2.3.1 Expériences et résultats 51
2.3.2 Perspectives 57

3 Ajout de connaissances a priori pour la détection et classification d'entités 59
3.1 Un moteur de recherche mono-champ 60
3.2 La tâche de *Slot filling* 61
3.2.1 Rappel de la tâche et adaptation pour PagesJaunes 61
3.2.2 Solutions disponibles 61
3.3 Solution sur mesure 62
3.3.1 Première version 62
3.3.2 Améliorations du modèle 63
3.3.3 Expériences 66
3.4 Conclusion 68

4 Application d'une métrique d'évaluation à l'alignement de bases de connaissances par plongement 69
4.1 Contexte et détails techniques 70
4.1.1 Plongement de Graphes de Connaissances 71
4.1.2 Tâche d'alignement 72
4.1.3 Historique des méthodes d'alignement 72
4.2 AlignD : méthode d'alignement globale par plongements 76
4.2.1 Mesurer l'alignement 76
4.2.2 Objectif total 79
4.2.3 Génération automatique de *seed* 79
4.3 Expériences 80
4.3.1 Protocole 80
4.3.2 *Seed* réelle 81
4.3.3 Taille de la *seed* et performance 82

4.3.4 *Seed* générée automatiquement 84
4.4 Conclusion 85

II Les espaces hyperboliques pour la représentation de connaissances hiérarchiques 87

5 Représentation de hiérarchies dans des espaces hyperboliques 91
5.1 Enjeux de la représentation des données 92
5.2 Plongement de hiérarchies en espace hyperbolique 93
5.2.1 Sur le principe des plongements de hiérarchies 93
5.2.2 Méthodes déterministes 95
5.2.3 Méthodes stochastiques 98
5.3 Conclusion 103

6 Paramétrisation continue d'arbres en variété riemannienne 105
6.1 Variantes algorithmiques 106
6.1.1 HyperE Rayon Variable (VR) 107
6.1.2 HyperE Initialisation par Barycentre (BI) 107
6.2 ASTRE 108
6.2.1 Modèle de base 108
6.2.2 Généralisation aux autres variétés riemanniennes 110
6.2.3 Optimisation 112
6.3 Expériences 113
6.3.1 Expériences de reconstruction 113
6.3.2 Expériences de fusion 120
6.4 Conclusion 125

7 Extraction d'arbre dans des plongements hyperboliques 127
7.1 Besoins 128
7.2 Etat de l'art 129

7.2.1 Historique des méthodes 129
7.2.2 Composants principaux 131
7.3 Méthodologie proposée 134
7.3.1 Combinaison de *features* syntaxiques et lexicales 134
7.3.2 Plongement des termes 136
7.3.3 Extraction d'arbre 137
7.3.4 Raffinage de la taxonomie 138
7.4 Expériences 138
7.4.1 Protocole 138
7.4.2 Evaluation 139
7.4.3 Reconstruction de taxonomies 141
7.4.4 Induction de taxonomie 142
7.5 Conclusion 145

8 Difficultés de la classification hiérarchique 147
8.1 Classification hiérarchique de textes 148
8.1.1 Description de la tâche 148
8.1.2 État de l'art 148
8.2 Élaboration d'un corpus sur-mesure 150
8.3 Méthodologie 151
8.3.1 Représentation des documents 151
8.3.2 Passage de vecteurs euclidiens vers des vecteurs hyperboliques 151
8.3.3 HMCN préfixé d'une projection hyperbolique (HYP-HMCN) 152
8.3.4 *Baseline* BERT 153
8.4 Expériences 153
8.4.1 Métriques d'évaluation 154
8.4.2 Résultats 154
8.5 Améliorations effectuées 155
8.5.1 Plongement des classes 155

8.5.2 Conversion séquentielle locale des vecteurs euclidiens 155
8.5.3 Résultats 157
8.6 Conclusion 158

Conclusion générale **159**

Table des matières **165**

Table des figures **171**

Liste des tableaux **175**

Liste des acronymes **177**

Annexes 180

A Résultats d'ASTRE dans d'autres dimensions (SemEval) **183**
A.1 2 dimensions 183
A.2 3 dimensions 185
A.3 10 dimensions 187

B Résultats d'ASTRE dans d'autres dimensions (WordNet) **189**
B.1 2 dimensions 189
B.2 3 dimensions 191
B.3 10 dimensions 192

C Un logiciel pour la transformation de plongements euclidiens en plongements hyperboliques et la découverte de connaissances hiérarchiques. **193**
C.1 Interface 194
C.2 Fonctionnement 195
C.3 Expériences 196
C.3.1 Découverte de hiérarchies 196

C.3.2 Réduction de dimensions 197
C.4 Limites 198
C.5 Conclusion 199

Bibliographie **203**

Bibliographie **203**

Résumé/Abstract **217**

Introduction générale

Les langues et les connaissances humaines décrivent la réalité par des symboles discrets artificiels. Ces symboles s'organisent dans un système de partage d'information, aboutissant à la communication de concepts dérivés du monde réel. Cependant, ces derniers, étant issus de la réalité, sont généralement de nature continue, ce qui établit un duel entre la discontinuité de la description (par les symboles) et la continuité du réel (les concepts). Comprendre et saisir le contenu de l'échange, c'est associer ces signes à une représentation imaginaire continue apprise par les interlocuteurs. À l'inverse, s'exprimer consiste à transmettre la séquence de signes correspondant à la représentation imaginaire que l'on souhaite partager. La qualité de cette modélisation linguistique a une incidence positive sur les tâches de description, de transmission et bon nombre de tâches coopératives. Jusqu'à récemment, les ordinateurs étaient incapables de traiter des tâches avec du langage naturel, les forçant à coopérer au travers de langages de programmation qui sont plus difficiles à maîtriser et limitent donc leur interaction avec les Hommes. Les succès récents du Traitement Automatique du Langage Naturel (TALN) par les machines reposent exactement sur le principe de fournir à celles-ci une représentation continue du monde à partir des symboles composant le langage ou un ensemble de connaissances organisées. On appelle ces représentations des plongements.

Les plongements sont donc un moyen de parvenir à une meilleure interface Homme-machine et à des technologies plus accessibles pour tous (e.g. les assistants vocaux), en produisant des représentations numériques des symboles linguistiques. En tant que tel, ils simplifient l'interaction des machines avec le langage écrit, en proposant de passer d'un signe symbolique (la chaîne de caractères) à un vecteur, contenant en lui-même des informations sur le concept désigné par le signe qu'il modélise. Les vecteurs des symboles coopèrent dans un espace continu offrant des fonctions de comparaison qui quantifient la proximité des vecteurs et leur concept sous-jacent. En effet, les symboles s'organisent dans un système linguistique et désignent des objets et des éléments syntaxiques utilisables dans des *relations sémantiques*. Par exemple, le mot «*personne*» associé au mot «*couronne*», se rapproche du mot «*roi*» ou «*reine*». Idéalement, l'organisation des vecteurs du plongement dans l'espace continue doit respecter l'organisation du système linguistique, et donc, modéliser ces relations sémantiques.

Dans cette thèse, nous étudions plusieurs plongements et espaces d'accueil de ces plongements, pour, en finalité, produire des représentations de meilleure qualité et accomplir des tâches relatives au TALN. Nous pensons que certains espaces d'accueil possèdent des propriétés avantageuses intrinsèquement incluses dans les

règles géométriques qui les régissent, et que le mélange d'informations issues de différentes sources permet la création de plongements plus polyvalents et utiles à des tâches industrielles. Les approches de plongements abordées ici se ressemblent toutes sur le principe, dans le sens où elles tentent de construire des représentations vectorielles à partir de données sans intervention humaine, ou plutôt, de façon auto-supervisée *via* une tâche fictive artificielle. L'intérêt principal est de pouvoir étendre à des échelles colossales leur principe en vue d'améliorer la qualité de la modélisation.

En particulier, les plongements de mots sont les premiers objets étudiés. Si les premières versions sont simples (MIKOLOV, SUTSKEVER et al., 2013 ; BOJANOWSKI et al., 2017 ; MIKOLOV, K. CHEN et al., 2013 ; VILNIS et MCCALLUM, 2015 ; C. SUN et al., 2018 ; NICKEL et KIELA, 2018 ; COLLOBERT et al., 2011 ; LU et al., 2020 ; IACOBACCI, PILEHVAR et NAVIGLI, 2015, un vecteur par mot, appelées plongements statiques), et passent à l'échelle simplement sur des ordinateurs communs, les derniers modèles récents (DEVLIN et al., 2019 ; YANG et al., 2019 ; RADFORD, 2018 ; RADFORD et al., 2019 ; BROWN et al., 2020 ; PETERS et al., 2018 ; AKBIK, BLYTHE et VOLLGRAF, 2018, dits plongements contextuels) requièrent des architectures bien plus lourdes, plus coûteuses et parfois inaccessibles. Pour leur simplicité, et aussi par la discrétion des modèles contextuels au début de nos trois années de thèse, nous nous focalisons principalement sur les premiers, en tentant de les évaluer pour les améliorer.

L'évaluation de la qualité des plongements de mots est assez acrobatique. D'un côté, on dispose souvent d'une tâche finale correspondant à un objectif pratique, qui, quand il est accompli satisfait le besoin. Si le plongement permet de résoudre l'objectif, y a-t-il vraiment besoin de remettre en cause la qualité de ce dernier ? De l'autre côté, un plongement efficace sur une tâche peut s'avérer inefficace sur une autre, démontrant un manque de polyvalence, ou l'omission de l'incorporation d'un aspect linguistique dans les vecteurs du plongement (CLAVEAU et KIJAK, 2016 ; SCHNABEL et al., 2015). La recherche active de critères d'évaluation caractérisant la qualité globale d'un plongement, de façon indépendante à des tâches requérant des couches de modélisation supplémentaires au plongement (les évaluations intrinsèques ou globales), est depuis longtemps une question poursuivie dans l'idée de pouvoir identifier les aspects du langage et les propriétés vectorielles indispensables à une modélisation solide, et de rechercher à les renforcer (SCHNABEL et al., 2015 ; TSVETKOV et al., 2015 ; HOULE, KASHIMA et NETT, 2012). Nous révèlerons, dans ce cadre, des caractéristiques vectorielles bénéfiques à la performance des plongements de mots statiques.

Imposer ces caractéristiques vectorielles est difficile et peu récompensé (LUKA, SOULIER et PICARD, 2021), ce qui pousse à la recherche d'autres axes d'améliorations de la qualité des plongements de mots. L'exploitation de données organisées compatibles avec des mots, peuvent s'intégrer dans leur vecteur, enrichissant les aspects linguistiques par des connaissances expertes. Nous verrons que la mixité informationnelle des vecteurs produits provoque l'amélioration significative de tâches industrielles, comme le *slot filling*. À partir de ce principe, nous avons recherché des

moyens d'acquérir, de représenter, de restructurer et d'utiliser des sources d'informations, dans le but de construire des vecteurs plus riches, quasi-universels, pour des tâches quelconques.

Ceci explique notre motivation à passer des plongements de mots aux plongements de graphes de connaissances (*Knowledge Graph*s, KGs) puis aux plongements de hiérarchies. Les plongements de graphes consistent en la représentation continue des nœuds du graphe, dont l'espace préserve idéalement les relations (les liens) de ce dernier. Cette représentation continue d'un graphe vient supplémenter la représentation des mots assimilables aux nœuds, ce qui augmente leur pertinence pour certains cas d'usage. Avant de procéder à cette étape de combinaison des types (mots et graphes), il convient de produire des plongements de KGs de qualité et porteur d'informations redondantes ou complémentaires. Les patrimoines communautaires telles que Wikidata, deviennent alors des ressources capitales dans l'enrichissement de KGs privés pour l'affinement des plongements des nœuds de ces graphes, faisant de l'alignement de KGs une tâche d'intérêt (Z. SUN, HU, Q. ZHANG et al., 2018 ; WU et al., 2019 ; CAO et al., 2019 ; ZHU et al., 2017 ; Y. LIN et al., 2015 ; M. CHEN et al., 2017 ; Z. SUN, HU et LI, 2017 ; SHVAIKO et EUZENAT, 2013). Cette tâche exploite les similitudes relationnelles ou symboliques entre deux KGs, pour identifier les nœuds analogues (référent à la même réalité conceptuelle), et enfin parvenir à la fusion des graphes. On peut chercher à résoudre ce problème d'alignement en produisant des plongements conjoints de ces deux KGs et forcer l'ensemble des entités analogues à s'aligner dans l'espace commun. Nous étudions ce problème et proposons une solution nouvelle, exploitant une métrique globale sur les plongement qui nous permet d'imposer des propriétés informationnelles sur les dimensions des plongements.

D'autres options (autres que l'enrichissement) existent pour obtenir des connaissances. C'est le cas des problèmes de découverte ou de restructuration de données, abordés dans cette thèse, visant à exploiter des sources polymorphes, aussi largement que possible, pour reformuler ou dévoiler des informations cachées. Les hiérarchies sont alors des graphes simples qui structurent les connaissances, et ce sont celles-ci que nous cherchons à découvrir. En apprenant à plonger les hiérarchies, on découvre des espaces et des propriétés (heuristiques) utiles plus tard pour l'induction de celles-ci à partir de données textuelles. Ainsi, les espaces hyperboliques s'imposent en fer de lance des méthodes de projections de hiérarchies, car ces derniers se démontrent plus efficaces pour incorporer les informations hiérarchiques que les traditionnels espaces euclidiens (NICKEL et KIELA, 2017 ; NICKEL et KIELA, 2018 ; GANEA, BECIGNEUL et HOFMANN, 2018a ; KLEINBERG, 2007 ; SARKAR, 2011 ; SALA et al., 2018). Nous proposons de fusionner deux grandes familles de méthodes pour produire une méthode de projection de hiérarchie propice à la mixité de l'information (hiérarchique et sémantique) et permettant la paramétrisation continue d'arbre, offrant la capacité d'appliquer des algorithmes avec descente de gradient stochastique (*Stochastic Gradient Descent*, SGD) sur ces derniers. Améliorer la représentation des hiérarchies délivre des nouvelles perspectives de visualisation et d'exploitation des données, et fournit d'autres axes de découverte d'informations.

La compréhension du processus de plongements nous permet de renverser

l'ordre des opérations. Ainsi, partant d'un plongement il apparaît possible de reconstruire la hiérarchie qui est à l'origine de son calcul. L'induction de hiérarchie se base sur ce principe, mais au lieu de fabriquer un plongement à partir de hiérarchies, on le fabrique à partir de plusieurs *features* linguistiques (syntaxiques, distributionnelles, lexicales) très répandus dans le domaine (MAO, REN et al., 2018 ; ALY et al., 2019 ; PANCHENKO et al., 2016 ; MAITRA et DAS, 2016 ; TAN, BOND et GENABITH, 2016 ; C. WANG, HE et ZHOU, 2017). Traditionnellement, les *features* s'emploient dans des classifieurs, mais nous proposons de former des plongements et de les interpréter comme des hiérarchies en renversant le processus de plongement. Ce principe peut s'appliquer à tous types de plongement et donc généraliser l'approche à toutes sources de données, pourvu qu'elles possèdent certaines propriétés.

Une fois en possession de plongements de connaissances (de hiérarchies ou de KG) se pose la question de fusion entre ces connaissances et des plongements plus traditionnels, de type euclidiens, sur les mots. La classification hiérarchique apparaît comme un moyen d'aborder ce problème, dans lequel, des documents textes sont classés dans une hiérarchie de classes. L'utilisation de modèles contextuels euclidiens donne une représentation classique du document qui doit idéalement se positionner dans une hiérarchie, représenté par un plongement hyperbolique. On cherche donc à bâtir un espace commun documents et classes, pour mettre en commun l'information des deux sources (hiérarchiques et sémantiques), et ainsi clore la problématique d'association classes-documents, et finalement, connaissances-mots.

La thèse est scindée en deux parties. La partie I traite des plongements de mots et de l'enrichissement des plongements par des connaissances extérieures. Plus spécifiquement,

— le chapitre 1 est un état de l'art des méthodes usuelles pour réaliser des plongements de mots,

— le chapitre 2 s'occupe de l'évaluation des plongements, et propose des métriques nouvelles qui tente de caractériser globalement la qualité des vecteurs,

— le chapitre 3 s'attaque à une problématique industrielle de PagesJaunes, et la résout en proposant deux principes : l'enrichissement des vecteurs des mots par des connaissances structurées dans des bases de données, et le *pseudo-labeling*,

— le chapitre 4 applique une métrique d'alignement, utilisée pour qualifier la qualité des plongements, en tant que critère d'optimisation de la tâche d'alignement de KGs, permettant l'enrichissement mutuel des deux KGs.

La partie II restreint la découverte et l'enrichissement des données aux hiérarchies avec des espaces hyperboliques. Précisément,

— le chapitre 5 est un état de l'art des méthodes de plongement des hiérarchies, et donne les idées clés pour procéder à l'optimisation de modèles dans des espaces hyperboliques,

— le chapitre 6 propose une nouvelle méthode de plongement de hiérarchies, qui combine deux grandes familles de méthodes et facilite la mixité d'information,

— le chapitre 7 parle de l'induction de hiérarchies, en exploitant des plongements hyperboliques de termes et en renversant le principe de plongement, ce qui fournit une nouvelle méthode non supervisée d'induction de taxonomies,

— le chapitre 8 envisage la possibilité de produire des espaces conjoints entre une hiérarchie de classes et des documents associés pour procéder à la classification hiérarchique, afin d'enrichir, à terme, des plongements euclidiens de mots avec des plongements hyperboliques de hiérarchies.

Première partie

Plongements de mots et gestion des connaissances

Chapitre 1

État de l'art sur les plongements de mots

Sommaire

1.1 Les plongements : construire un espace sémantique continu à partir des mots 20
1.1.1 Contexte et besoins 20
1.1.2 Formalisation 21
1.2 Plongements statiques 22
1.2.1 Plongements euclidiens statiques 22
1.2.2 Plongements non euclidiens statiques 27
1.3 Plongements contextuels 33
1.3.1 Plongements autorégressifs 34
1.3.2 Plongements autoencoding 35
1.4 Caractérisation des plongements 36

Ce chapitre introduit les notions essentielles pour calculer des plongements à partir de textes bruts, qu'on appelle des plongements de mots. Il organise la plupart des méthodes existantes pour produire ce type de représentation. Après une remise en contexte et une première formalisation dans la section 1.1, on aborde les plongements statiques dans la section 1.2. On distinguera dans cette section, les plongements reposant sur l'espace euclidiens (section 1.2.1), des autres qui utilisent d'autres géométries (section 1.2.2), comme la géométrie hyperbolique ou des espaces de probabilités. Par la suite on explique les plongements contextuels dans la section 1.3, des modèles de représentation du langage plus récents, plus complexes et, en principe, plus efficaces. Enfin, on propose des propriétés permettant la distinction des méthodes de plongements dans la section 1.4. Cet état de l'art a été publié dans le journal international IJDSA en 2021 (TORREGROSSA, ALLESIARDO, CLAVEAU, KOOLI et al., 2021).

1.1 Les plongements : construire un espace sémantique continu à partir des mots

1.1.1 Contexte et besoins

Les mots sont des éléments distincts du langage porteur d'un ou plusieurs sens. Les caractères, à eux seuls, ne permettent pas de retrouver le sens des mots qu'ils composent. Par exemple, *restaurant* et *nourriture* sont tous deux sémantiquement corrélés, cependant, il apparaît impossible de quantifier l'intensité de cette corrélation en se basant uniquement sur leur chaîne de caractères.

Les plongements de mots, sont des modèles appris automatiquement sur des corpus de textes, permettant de répondre à cette problématique de quantification sémantique, en produisant une représentation continue de l'espace des mots. En d'autres termes, un vecteur de $\mathbb{R}^n$ est attribué à chaque mot comme sa projection dans un espace continu, et ce vecteur se doit idéalement de préserver les relations sémantiques du mot qu'il représente par des relations géométriques dans l'espace de représentation. Sur la figure 1.1, par exemple, on obtient les capitales par translation depuis les pays.

La puissance des plongements repose sur leur capacité à représenter continûment le langage naturel par des vecteurs, mettant en évidence les relations sémantiques par des relations géométriques. Cela permet de mettre en place des architecture, dites de bout-en-bout, transposant le caractère discret du langage dans un espace continu. Les plongements sont ainsi largement utilisés en TALN, puisque cela facilite leur usage dans des modèles d'apprentissage profonds. L'analyse de sentiment, la reconnaissance d'entités nommés et d'autres tâches ont abondamment bénéficié des plongements de mots (BAKAROV, 2018 ; SCHNABEL et al., 2015 ; PENNINGTON, SOCHER et MANNING, 2014).



FIGURE 1.1 – Vecteurs des capitales et leur pays issus d'un plongement FastText (BOJANOWSKI et al., 2017) dont la dimension est réduite à 2 dimensions par une PCA.

Dans ce chapitre, nous nous intéressons particulièrement aux plongements appris «*from scratch*» (COLLOBERT et al., 2011). Ce type de plongements consiste à traiter un ensemble de textes bruts pour en extraire des vecteurs, ayant les propriétés évoquées précédemment, sans connaissances expertes et en auto-supervision. Ces méthodes se basent sur le principe d'HARRIS, 1954, affirmant que *des mots partageant un contexte commun ont le même sens*. Dans la suite, nous utiliserons ce principe, nommé *hypothèse distributionnelle*, pour définir approximativement la sémantique des mots. Ainsi, le sens d'un mot est essentiellement défini par les mots apparaissant autour de lui, ou autrement dit, son contexte. La pertinence des mots du contexte n'est évidemment pas uniforme pour définir le sens d'un mot, et c'est pourquoi, certains modèles de plongements proposent, entre autres bénéfices, de s'adapter à cette variabilité. Des modèles comme BERT ont, par leur architecture de type *transformer*, la faculté d'intégrer inégalement le contexte du mot représenté (DEVLIN et al., 2019). Un autre type de plongement peut aussi être réalisé en utilisant des bases de connaissances ou des graphes, mais nous y viendrons dans le chapitre 4. Ici, nous présentons des méthodes axées principalement sur le plongement pour le texte brut utilisant l'hypothèse distributionnelle.

1.1.2 Formalisation

Les plongements sont originellement des *lookup tables* qui associent à chaque mot un vecteur de $\mathbb{R}^n$ (COLLOBERT et al., 2011). Soit V un vocabulaire (ensemble de mots), les plongements prennent la forme suivante $e : V \rightarrow \mathbb{R}^n$.L'objectif d'un algorithme de plongement est de déterminer l'ensemble des vecteurs, *i.e.* la fonction e, de façon à satisfaire un critère d'optimisation. Les vecteurs des mots sont alors considérés comme des paramètres à optimiser au regard du critère désiré, devant aboutir à l'incorporation de connaissances sémantiques, c'est-à-dire, rapprocher les

mots partageant un sens similaire dans l'espace de représentation, les écarter à l'inverse, ou encore créer des propriétés géométriques. À nouveau, sur la Figure 1.1, la distance et la direction séparant les paires capitales-pays paraissent presque identiques. Des systèmes automatiques, tels que des réseaux de neurones ou des machines à vecteurs de support, peuvent par la suite exploiter ces aspects géométriques en trouvant des hyperplans, des portions de l'espace, etc. pour résoudre des tâches de TALN.

1.2 Plongements statiques

Les plongements statiques proposent, *a contrario* des plongements contextuels traités dans la section suivante, d'associer un vecteur fixé à chaque mot (graphène). Autrement dit, la représentation du mot ne varie pas entre les contextes, bien que parfois cela semble nécessaire. Par exemple, le mot *avocat* désigne à la fois un fruit et une profession, des usages bien différents distinguables grâce au contexte. Ces plongements ne traitent pas cette problématique polysémique, ils vont en revanche attribuer un vecteur global issu de l'agrégation de chaque usage local des mots dans le corpus de texte. Le terme *statique* fait référence à cette particularité : un vecteur est toujours le même et peut être statiquement pré-calculé.

En pratique, un plongement statique est représenté par une matrice $M = (m_{ij}) \in \mathbb{R}^{|V| \times n}$, dans laquelle, n est la dimension des vecteurs et la ligne i correspond au vecteur du i-ème mot du vocabulaire V, soit : $v^M_{w_i} = (m_{ij})_{j \in [\![1,D]\!]}$ où w_i est le i-ème mot de V.

1.2.1 Plongements euclidiens statiques

La géométrie euclidienne est utilisée pour construire ces plongements. Par exemple, Word2Vec (MIKOLOV, SUTSKEVER et al., 2013) et FastText (BOJANOWSKI et al., 2017) sont des algorithmes de plongements exploitant le produit scalaire euclidien entre les vecteurs des mots pour les construire.

1.2.1.1 Word2Vec

MIKOLOV, SUTSKEVER et al., 2013 ont mis au point un algorithme, nommé Word2Vec, incorporant le contexte des mots apparaissant dans un corpus de texte pour produire des plongements euclidiens statiques. L'intégration de relations paradigmatiques et syntagmatiques entre les mots du vocabulaire se fait en utilisant l'un des deux modèles suivants : *Skip-Gram* (SG) ou *Continuous Bag-of-Words* (CBoW). L'aspect paradigmatique est défini par l'ensemble des concepts individuels issus des différents mots du vocabulaire, et, l'aspect syntagmatique est relatif aux informations contenues dans la structure d'apparition. L'algorithme de plongement tente de traiter ces deux aspects du langage, en se reposant sur l'hypothèse distributionnelle, c'est-à-dire, en considérant que ces informations sont contenues dans le contexte d'occurrence d'un mot.

$$T = \ldots \; w_{i-(k+1)} \; \boxed{w_{i-k} \; w_{i-(k-1)} \; \ldots \; w_{i-1} \; \boxed{w_i} \; w_{i+1} \; \ldots \; w_{i+(k-1)} \; w_{i+k}} \; w_{i+(k+1)} \; \ldots$$

FIGURE 1.2 – Fenêtre de taille k autour d'un mot cible w_i dans un corpus T. Les mots du contexte sont alors $\{w_{i-k}, w_{i-(k+1)}, ..., w_{i-1}, w_{i+1}, ..., w_{i+(k-1)}, w_{i+k}\}$.

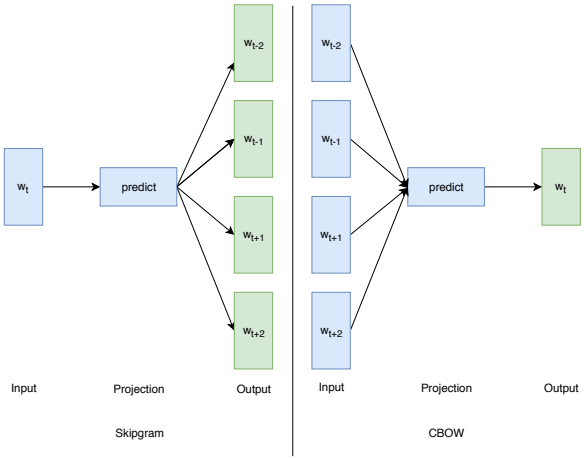


FIGURE 1.3 – Les modèles SG et CBoW. À gauche, le modèle SG qui prédit des mots de la fenêtre du contexte connaissant le mot cible. À droite, le modèle CBoW utilisant les mots du contexte pour prédire le mot cible (adaptée de MIKOLOV, K. CHEN et al., 2013).

Suivant les travaux de MIKOLOV, SUTSKEVER et al., 2013 et PENNINGTON, SOCHER et MANNING, 2014, le contexte d'un mot dans une phrase peut être défini en prenant les k mots précédents et k mots subséquents de ce mot. On définit alors une fenêtre de taille $2k$ contenant les mots du contexte pour un mot central appelé le mot cible (voir figure 1.2). Considérant des fenêtres contextuelles pour deux mots cibles différents, l'hypothèse distributionnelle revient à considérer que la proximité sémantique des mots cibles est corrélée à la proportion de mots en communs entre les fenêtres. Deux variantes de modèles sont proposées dans l'algorithme Word2Vec : SG et CBoW, montrés en figure 1.3.

La variante SG tente de prédire les mots du contexte en connaissant le mot cible. Cela revient à dire que l'on veut intégrer des informations dans le vecteur du mot cible de façon à permettre l'identification, dans le vocabulaire, des mots du contexte les plus probables. Plus formellement, soit un corpus d'entraînement T (i.e. une suite de mots) et une matrice représentant le plongement W, la maximisation de

$$\frac{1}{|T|}\sum_{t=1}^{|T|}\sum_{\substack{c=t-k \\ c\neq t}}^{t+k} \log p(w_c \mid w_t, W) = \frac{1}{|T|}\sum_{t=1}^{|T|}\sum_{\substack{c=t-k \\ c\neq t}}^{t+k} \log \frac{e^{\langle v^W_{w_c}, v^W_{w_t}\rangle}}{\sum_{w\in V} e^{\langle v^W_w, v^W_{w_t}\rangle}}, \tag{1.1}$$

revient à satisfaire cet objectif pour tous les mots du vocabulaire. Pour donner une

image à cette équation, elle revient à faire coulisser la fenêtre de contexte sur l'ensemble du corpus de texte, et à chaque fois on maximise la probabilité de détection de chacun des mots du contexte.

En pratique le dénominateur n'est pas calculable car le vocabulaire contient quelques centaines de milliers d'éléments, et calculer un produit scalaire à chaque étape d'optimisation sur cette quantité de vecteurs ne peut être accompli en un temps raisonnable. C'est pourquoi, l'échantillonnage négatif (*Negative Sampling*, NS) est utilisé en pratique pour résoudre ce problème, ce qui revient *grosso modo* à échantillonner des mots aléatoires du vocabulaire et à remplacer la somme sur le vocabulaire par la somme sur un échantillon. L'intuition étant que les mots aléatoires constituent un bruit et que les vecteurs du mot cible doivent faciliter l'identification du mot du contexte par rapport à ce bruit (MIKOLOV, SUTSKEVER et al., 2013). Un autre aspect pratique est qu'on utilise deux matrices différentes, une pour les mots du contexte et une autre pour les mots cibles pour refléter la différence de rôle entre les mots du contexte et les mots cibles.

Le modèle CBoW est opposé au modèle SG, puisqu'il revient à faire l'inverse : on utilise les mots d'une fenêtre contextuelle pour identifier le mot cible le plus probable. La tâche est plus simple, puisqu'un plus grand nombre de mot est connu pour faire la prédiction. Les plongements résultants de la méthode CBoW semble capturer en priorité des informations syntagmatiques plutôt que des informations paradigmatique, ceci est visible dans les résultats présentés par MIKOLOV, K. CHEN et al., 2013.

La taille de la fenêtre fait aussi varier le type d'informations incorporé dans les vecteurs, ainsi, une large fenêtre intégrera davantage d'informations paradigmatiques, et une fenêtre plus petite des informations syntagmatiques (PENNINGTON, SOCHER et MANNING, 2014). Une preuve a été proposée par LEVY et GOLDBERG, 2014, démontrant que ces modèles reviennent à créer une factorisation de la matrice d'information mutuelle entre les mots, justifiant formellement l'approche et la reliant aux travaux précédents d'analyse distributionnelle.

1.2.1.2 GloVe

La méthode GloVe, élaborée par PENNINGTON, SOCHER et MANNING, 2014, est aussi une approche pour produire des plongements statiques. Le principe de base utilisé est plus direct car il fait directement intervenir la matrice de co-occurrences entre les mots. On dit que deux mots co-occurrent si l'un apparaît dans la fenêtre contextuelle de l'autre. Le processus est divisé en deux étapes. D'abord, on calcule la matrice de co-occurrences sur l'ensemble du corpus. Cette étape peut être vue comme une compression résumant globalement l'ensemble des informations locales contenues dans chacune des fenêtres contextuelles. Cette compression n'a lieu qu'une seule fois et il est facile d'ajouter de nouveaux textes sans avoir à tout recalculer. La seconde étape consiste à factoriser cette matrice.

En reprenant les notations originales, on note X_{ij} est le nombre de fois où le mot w_j apparaît dans les fenêtres du mot w_i. Conformément à PENNINGTON, SOCHER et MANNING, 2014,

$$P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}} \tag{1.2}$$

est la probabilité que le mot w_j apparaisse dans le contexte du mot w_i. Cette probabilité nous aide à quantifier la pertinence d'un mot par rapport à d'autres. Si l'on considère deux mots cibles w_i, w_j et un mot du contexte w_k,

$$Q = \frac{P_{ik}}{P_{jk}} \tag{1.3}$$

agit comme une mesure discriminante :

— $Q \gg 1$ signifie que w_k est plus proche de w_i que de w_j,

— $Q \ll 1$ signifie que w_k est moins proche de w_i que de w_j,

— $Q \simeq 1$ signifie que w_k est aussi proche de w_i que de w_j.

L'idée soutenant GloVe est de parvenir à reconstruire cette métrique Q à partir du vecteur des mots, et donc, d'incorporer les informations contenues dans les probabilités de la matrice. Formellement, on cherche au moyen des vecteurs des $v^W_{w_i}$, $v^W_{w_j}$ et $v^C_{w_k}$ (les mots cibles et contextuels sont projetés avec des plongements différents W et C) à reconstruire Q :

$$F(v^W_{w_i}, v^W_{w_j}, v^C_{w_k}) = Q = \frac{P_{ik}}{P_{jk}}, \tag{1.4}$$

où F est une fonction devant satisfaire plusieurs critères. D'abord F doit traiter les vecteurs linéairement et sans ambiguïté sur les dimensions, donc on choisit d'écrire $F(v^W_{w_i}, v^W_{w_j}, v^C_{w_k})$ comme $F((v^W_{w_i} - v^W_{w_j})^T v^C_{w_k})$. En exploitant le produit scalaire, on préserve la consistence entre les dimensions des vecteurs. Une autre contrainte sur F est le fait de pouvoir permuter le rôle d'être un mot cible avec celui d'être un mot du contexte, et donc F doit satisfaire :

$$F((v^W_{w_i} - v^W_{w_j})^T v^C_{w_k}) = \frac{F((v^W_{w_i})^T v^C_{w_k})}{F((v^W_{w_j})^T v^C_{w_k})}. \tag{1.5}$$

ce qui implique que $F = \exp$ et

$$P_{ik} = F((v^W_{w_i})^T v^C_{w_k}) \iff (v^W_{w_i})^T v^C_{w_k} = \log\ X_{ik} - \log\ X_i. \tag{1.6}$$

Le terme $\log\ X_i$ est considéré comme un biais b^W_i, et on rajoute un autre biais b^C_k pour préserver la symétrie entre les rôles. En plus de cela, on se donne une fonction f pour amortir le logarithme lorsque X_{ik} atteint la valeur 0, ce qui arrive souvent

comme la matrice de co-occurrences est éparse. Le critère d'optimisation des vecteurs est donc le suivant,

$$J = \sum_{i,k} f(X_{ik})((v_{w_i}^W)^T v_{w_k}^C + b_i^W + b_k^C - \log\ X_{ik})^2. \tag{1.7}$$

GloVe est alors un autre modèle pour calculer des plongements de mots reposant sur du texte brut comme SG et CBoW. Ses avantages principaux étant qu'il ne fait pas intervenir de NS et calcule une compression du corpus sur lequel les vecteurs sont appris (ce qui engendre moins de calculs).

1.2.1.3 FastText

Le principal désavantage de Word2Vec et GloVe est leur incapacité à calculer des représentations pour des mots non présents dans le corpus d'entraînement, qu'on appelle des mots hors vocabulaire (*Out Of Vocabulary*, OOV). Pour résoudre ce problème, BOJANOWSKI et al., 2017 ont proposé un nouveau modèle de plongement : FastText. FastText décompose les mots en n-grammes de caractères et associe à chaque n-gramme un vecteur, ce qui permet de calculer la représentation d'un mot connu ou inconnu, en prenant la somme des vecteurs de ses n-grammes. En plus d'ajouter la capacité de gérer des mots OOV, FastText rajoute des informations morphologiques dans les vecteurs du plongement, une propriété pouvant s'avérer utile pour certaines langues dans lesquelles la morphologie du mot joue un rôle important, notamment pour les déclinaisons. En français par exemple, le bi-gramme *re* en préfixe indique une répétition, ou encore le 4-gramme *ment* en suffixe indique un adverbe. Ces informations incluses dans ces *sous-mots* (le sous-ensemble successif de caractères) sont extraites par l'intervention des n-grammes, ajoutant cet aspect aux vecteurs des mots.

La matrice du plongement est ainsi étendue pour contenir les vecteurs des n-grammes. Comme la quantité de n-grammes est très large, une fonction hachage permet à la fois de réduire le nombre de vecteurs nécessaires à la représentation des n-grammes, et aussi d'indexer les lignes correspondantes dans la matrice étendue. Soit W_{FT} la matrice du plongement FastText étendue, alors $W_{FT} \in \mathbb{R}^{(|V|+H)\times n}$, où H est la taille de la table de hachage.

Rappelons qu'un n-gramme est la suite consécutive de n caractères dans une chaîne. Le représentation FastText s'écrit

$$\underbrace{v_{w_i}^{W_{FT}}}_{\text{vecteur FastText}} = \underbrace{W_{FT}^{i,*}}_{\text{vecteur du mot}} + \underbrace{\sum_{n=\text{minn}}^{\text{maxn}} \sum_{z \in G^n(w_i)} W_{FT}^{|V|+h(z),*}}_{\text{somme des vecteurs des n-grammes}}, \tag{1.8}$$

avec :

— minn et maxn étant respectivement la taille minimum et maximum des n-grammes,

— $G^n(w_i)$ est l'ensemble des n-grammes dans w_i, $G^n(*)$ étant l'ensemble de tous les n-grammes, tel que $G^n(w_i) \subset G^n(*)$,

— $W_{FT}^{k,*}$ est la k-ème ligne de la matrice du plongement W_{FT},

— $h : G^n(*) \rightarrow [\![1, H]\!]$ est la fonction de hachage associant un n-gramme à un indice.

1.2.1.4 Plongements enrichis par des connaissances

Comme nous l'avons précédemment évoqué, les méthodes de plongements statiques ne sont pas capables de gérer la polysémie, ni de calculer la représentation d'un mot en fonction du contexte dans lequel il apparaît, étant donné que chaque mot est associé statiquement à un vecteur, et que le calcul de ce dernier se fait globalement sur un corpus de texte. SensEmbed (IACOBACCI, PILEHVAR et NAVIGLI, 2015) et CoupledCR (LU et al., 2020) ont été proposés pour tenter de rendre possible la gestion de la polysémie en exploitant des outils externes (au plongement) et des bases de connaissances.

La gestion de la polysémie peut être acquise en modifiant simplement les plongements. Au lieu de connecter un mot à un vecteur, on associe un vecteur par sens d'un mot. Par exemple, comme le mot *avocat* a deux sens (le fruit et la profession), on produira deux vecteurs (un pour chaque sens). On appelle cela des vecteurs de concepts, puisqu'on associe désormais les concepts des mots à des vecteurs, aboutissant à une représentation plus fine de la sémantique des mots.

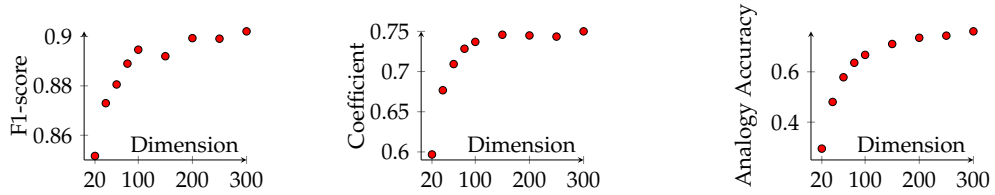
Pour déterminer le concept sous-jacent à l'usage d'un mot, on utilise un outil de désambiguïsation lexicale (comme Babelfy pour SensEmbed) ou les annotations de Wikipédia (pour CoupledCR). L'optimisation des vecteurs est identique à celle évoquée avant. Pour CoupledCR, un plongement du graphe des hyperliens Wikipédia est réalisé et incorporé à l'intérieur des vecteurs de concepts à l'aide d'une décomposition en valeurs singulières (*Singular Value Decomposition*, SVD).

Pour calculer la similarité entre deux mots, on calcule la similarité maximale parmi les similarités entre chaque concepts attachés à ces mots. Cela permet d'être plus fin dans la quantification sémantique entre deux mots, et les auteurs montrent que cette manière de faire produit des résultats proches de jugements humains.

Cependant, l'ajout de la gestion de la polysémie se fait par des outils externes qui ne sont pas infaillibles. L'utilisation de ces outils casse l'aspect bout-en-bout des approches par plongements ce qui peut créer de la propagation d'erreur.

1.2.2 Plongements non euclidiens statiques

L'implication peut se retrouver dans le langage pour décrire des relations asymétriques de type *est_un* ou *être_un_type_de*. Par exemple, un chien est un mammifère mais un mammifère n'est pas forcément un chien. Projeter ce type de relations dans des espaces euclidiens est compliqué, étant donné leur nature asymétrique. En effet, la plupart des métriques définies sur un espace euclidiens sont symétriques, comme



(A) F1-score sur les données CoNLL2003 (tâche de nomage d'entités) proposées par SANG et MEULDER, 2003.

(B) Coefficient de corrélation de Spearman avec WordSim-353 (FINKELSTEIN et al., 2002, tâche de similarité).

(C) Précision d'analogie sur les données Google proposées dans (MIKOLOV, K. CHEN et al., 2013).

FIGURE 1.4 – Performances de plongements FastText en fonction de leur dimension.

la similarité cosinus. Il est donc difficile de reconstruire des relations d'hyperonymie entre les mots, ou plutôt, entre les concepts qu'ils représentent (C. SUN et al., 2018).

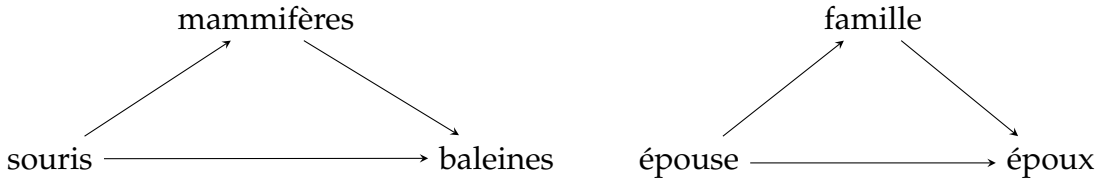
En plus, représenter des données complexes (comme le langage) est efficace, avec des espaces euclidiens, seulement lorsque la dimension est grande. On peut valider cette remarque en regardant la performance des plongements en fonction de leur dimension, toute chose égale par ailleurs, comme sur la figure 1.4c. À partir de 100 dimensions, le plongement semble atteindre des performances proches du maximum possible. La nécessité d'avoir un grand nombre de dimensions est expliquée par certaines contraintes évoquées par K. SUN et al., 2015 et illustrées en figure 1.5. Un des désavantages premier d'avoir un grand nombre de dimensions est l'ajout de complexité dans le processus d'optimisation des plongements, comme mentionné par NICKEL et KIELA, 2017.

Les propriétés des plongements euclidiens peuvent être utiles pour des tâches d'analogie ou reposant sur la similarité sémantique, cependant, ils apparaissent moins efficaces pour gérer des relations plus asymétriques comme des relations d'implications. Nous introduisons ici d'autres espaces pour gérer ces cas.

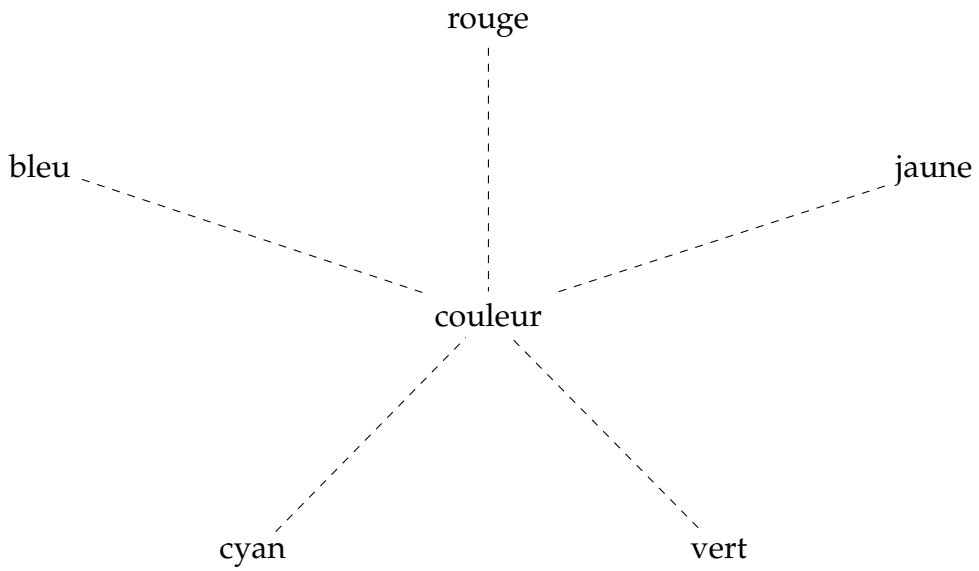
1.2.2.1 Plongements probabilistes

Nous appelons les plongements probabilistes, l'ensemble des plongements associant à chaque mot une loi de probabilité (par exemple une loi gaussienne). Au lieu de modéliser chaque mot par un vecteur, et ensuite inférer une probabilité $p(w_c|w_t)$ à partir de ces derniers comme pour Word2Vec ou GloVe, les plongements probabilistes travaillent directement sur les lois de probabilités associé à chaque mot. Par la suite, la «distance» entre deux mot est estimée en utilisant la divergence de Kullback-Leibler (divergence K-L), l'*Expected Likelihood* (EL), ou la distance de Wasserstein (C. SUN et al., 2018). Les deux premieres mesures peuvent s'écrire :

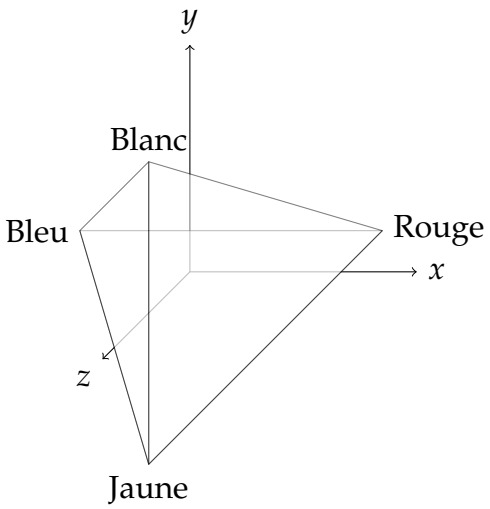
$$\begin{aligned} KL(p,q) &= \int p(x) \cdot \log \frac{p(x)}{q(x)} dx \\ EL(p,q) &= \int p(x) \cdot q(x) dx \end{aligned} \tag{1.9}$$



(A) Inégalité triangulaire. Cette propriété contraint les hyponymes à être proches les uns des autres dans un espace euclidien. *Souris* et *baleines* ne partagent pas de sens communs, mais, ils doivent être proches parce qu'ils partagent un hyperonyme commun : *mammifères*. Cela peut être vertueux, comme montré par l'exemple de la *famille*.



(B) Dans $\mathbb{R}^2$, le nombre maximum de points partageant le même plus proche voisin est 5, comme représenté par le pentagone. Par conséquent, plus de dimensions sont requises pour représenter des relations complexes (e.g. une hiérarchie).



(C) Le nombre maximum de points équidistants est 4 dans $\mathbb{R}^3$, comme représenté par le tétraèdre. Il n'est donc pas possible de représenter des familles de mots ayant une similarité homogène et plus de $d+1$ éléments dans $\mathbb{R}^d$ (code source pour le tétraèdre par Ignasi `https://tex.stackexchange.com/questions/174317/creating-a-labeled-tetrahedron-with-tikzpicture`).

FIGURE 1.5 – Example de contraintes de $\mathbb{R}^d$, se basant sur K. SUN et al., 2015.

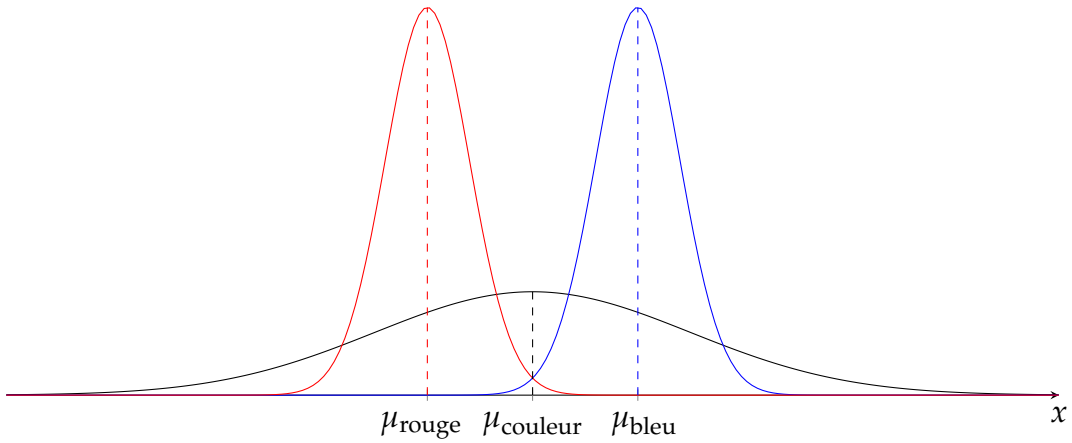


FIGURE 1.6 – Exemple d'une hiérarchie plongée dans un espace probabiliste, comme expliqué par VILNIS et MCCALLUM, 2015

où p et q sont des densités de probabilité et étant les représentations de deux mots que l'on cherche à comparer.

En particulier EL intervient pour gérer les relations symétriques (*e.g.* la synonymie) du plongement, et KL pour gérer les relations asymétriques (*e.g.* l'hyperonymie).

Les plongements gaussiens ou Word2Gauss, proposés par VILNIS et MCCALLUM, 2015, sont une implémentation de plongements probabilistes. Soit G un plongement de type Word2Gauss, chaque mot est associé à une loi normale $v^G_{w_i}(x) = \mathcal{N}(x, \mu_i, \sigma_i)$ avec $\mu_i \in \mathbb{R}^n$ et $\sigma_i \in \mathbb{R}^{n \times n}$ étant respectivement la moyenne et la matrice de covariance de la loi attribuée à w_i. L'interprétation de la moyenne est comparable à celle des vecteurs issus de Word2Vec. Cependant, la matrice de covariance incorpore l'incertitude des mots. Cela signifie qu'un mot ayant beaucoup d'hyponymes (et donc étant assez générique) aura davantage de variance qu'un mot avec peu d'hyponymes.

Un mot comme *fruit* aura une assez grande variance parce qu'il représente une grande gamme de fruits différents. Le mot *pomme* aura une variance plus faible puisqu'il représente moins de fruits. L'addition de cette notion d'incertitude, encodé dans la variance de la loi, confère à la modélisation la capacité de gérer plus facilement les hiérarchies, comme on peut le constater sur la figure 1.6.

Le processus d'optimisation de Word2Gauss est assez identique à celui de Word2Vec, à l'exception qu'il repose sur KL ou EL pour estimer la similarité entre les mots, au lieu de produit scalaire. Selon la nature de l'information à incorporer on choisira plutôt KL (hiérarchies) ou EL (sémantiques). On peut combiner les deux types sources en une seule, comme dans C. SUN et al., 2018, pour bénéficier des deux aspects.

1.2.2.2 Plongements hyperboliques

Dans un espace euclidien, le plus court chemin entre deux points est toujours une ligne droite. Dans le cas d'un espace hyperbolique, les plus courts chemins peuvent se courber et l'intensité de cette courbure est liée à l'emplacement de la destination par rapport à la source.

On introduit la boule de Poincaré ou le disque de Poincaré, un espace hyperbolique utilisé pour faire des plongements de mots ou de graphes (NICKEL et KIELA, 2017; TIFREA, BECIGNEUL et GANEA, 2019; GANEA, BECIGNEUL et HOFMANN, 2018a). Soit $\mathcal{B}^n = \{x \in \mathbb{R}^n, \|x\|_2 < 1\}$, la boule ouverte unité et $\|\cdot\|_2$ la norme $L2$. La boule de Poincaré est la variété riemannienne $\mathbb{D}^n = (\mathcal{B}^n, g_x)$, où

$$g_x = \left(\frac{2}{1 - \|x\|_2}\right)^2 g^E, \tag{1.10}$$

$x \in \mathcal{B}^n$ et g^E est le tenseur métrique euclidien soit I_n. Le tenseur métrique est la généralisation du produit scalaire euclidien, et définit localement la façon de mesurer un déplacement tangent dans l'espace (JOST, 2008; K. SUN et al., 2015). Il permet intuitivement de formaliser la géométrie de l'espace (SUZUKI, ENOKIDA et YAMANISHI, 2019). La distance est alors calculée en intégrant ce tenseur sur un chemin entre deux points, pour $x, y \in \mathcal{B}^n$

$$d_{\mathbb{D}^n}(x, y) = \operatorname{acosh}\left(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)}\right). \tag{1.11}$$

Une visualisation du comportement de cette distance est représentée pour une boule en 2 dimensions en figure 1.7. On y voit la courbure des plus courts chemins, et que celle-ci dépend de la destination à partir de la source.

Un autre type d'espace hyperbolique est utilisé pour essayer réduire les contraintes numériques engendrées par $\mathcal{B}^n$. En effet, $\mathbb{D}^n$ repose sur une boule ouverte pouvant causer des instabilités numériques à mesure qu'on s'approche de l'horizon 1 (NICKEL et KIELA, 2018). Si l'on s'approche trop près de la frontière, les points risques de passer de l'autre côté par erreur, générant des erreurs en cascade. On peut alors utiliser cet autre espace comme alternative à cela : l'espace de Lorentz. L'espace de Lorentz est la variété riemannienne $\mathcal{L}^n = (\mathcal{H}^n, g_l)$, avec $\mathcal{H}^n = \{x = (x_0, ..., x_n) \in \mathbb{R}^{n+1}, \langle x, x\rangle_{\mathcal{L}} = -1, x_0 > 0\}$ (voir figure 1.8), le produit scalaire lorentzien

$$\langle x, y\rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i, \tag{1.12}$$

et le tenseur métrique

$$g_l(x) = \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}. \tag{1.13}$$

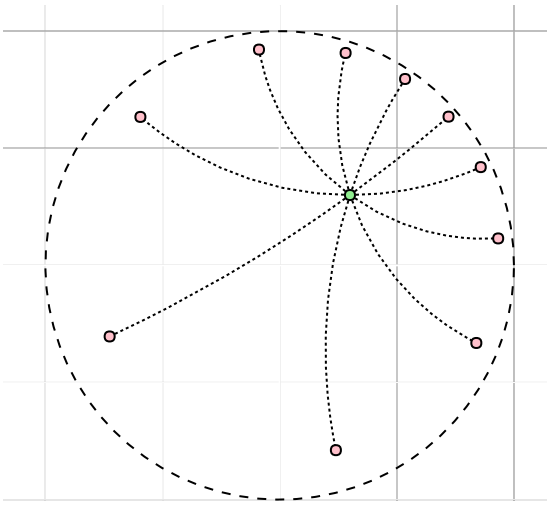


FIGURE 1.7 – Représentation des plus courts chemins (en pointillés) dans le disque de Poincaré $\mathbb{D}^2$. Chaque marque périphérique est à la même distance de la marque centrale, à noter que pour chacune le plus proche voisin est la marque centrale.

La distance s'écrit alors

$$d_{\mathcal{L}}(x, y) = \text{arcosh}(-\langle x, y \rangle_{\mathcal{L}}). \tag{1.14}$$

Même si $\mathcal{L}^n$ résout les instabilités numériques, il est moins facilement interprétable que $\mathcal{D}^n$, en particulier parce que $\mathcal{L}^n$ a une surface exponentielle, ce qui rend difficilement observable les vecteurs lointains. $\mathbb{D}^n$, quant à lui, représente cette surface exponentielle dans une surface restreinte (la boule ouverte), ce qui facilite la detection de comportements visuels. Par exemple, les concepts partageant le même hyperonyme auront tendance à se rassembler dans la même zone du disque et les concepts les plus spécifiques seront ceux avec les normes les plus grandes. À noter que l'on peut passer aisément d'une géométrie à l'autre en utilisant les fonctions suivantes (NICKEL et KIELA, 2018) :

$$x = (x_1, ..., x_n) \in \mathbb{D}^n, h(x) = \frac{(1 + \|x\|_2, 2x_1, ..., 2x_n)}{1 - \|x\|_2^2} \tag{1.15}$$

$$x = (x_0, ..., x_n) \in \mathcal{L}^n, p(x) = \frac{x}{x_0 + 1} \tag{1.16}$$

L'optimisation des paramètres dans ces espaces hyperboliques peut se faire en utilisant des méthodes d'optimisation riemannienne développée par BECIGNEUL et GANEA, 2019, ou en pratiquant une SGD sur des vecteurs d'un espace euclidien projetés dans l'espace hyperbolique souhaité.

Des généralisations de SG et GloVe ont été faite aux espaces hyperboliques (LEIMEISTER et WILSON, 2018 ; TIFREA, BECIGNEUL et GANEA, 2019), en remplaçant le

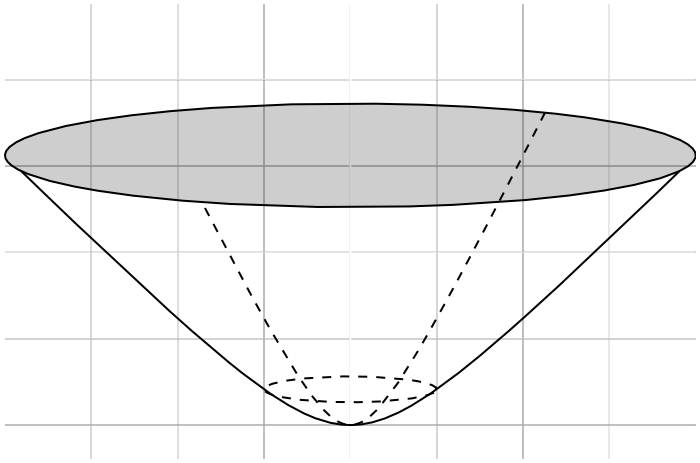


FIGURE 1.8 – Espace de Lorentz $\mathcal{L}^2$, constitué de l'ensemble des points de la surface conique.

produit scalaire par l'opposé de la distance et en utilisant une optimisation riemannienne. Pour des tâches de similarités et d'analogies, on n'observe pas d'améliorations par rapport aux espaces euclidiens en grande dimension, sauf pour GloVe (en faveur des espaces hyperboliques). Les plongements hyperboliques obtiennent, cependant, la même performance avec beaucoup moins de dimensions. Ces plongements sont largement plus performants pour projeter des graphes comme nous le faisons dans la partie II. Nous y reviendrons à ce moment là.

1.3 Plongements contextuels

Les plongements contextuels sont directement opposés aux plongements statiques par nature, puisqu'ils produisent des représentations variables des mots. En effet, le vecteur d'un mot n'est plus calculé globalement, suite à l'optimisation d'un critère sur corpus de texte, mais est obtenu en se basant sur le contexte d'apparition du mot. Ainsi, un modèle de langue est utilisé pour extraire les informations locales et produire un vecteur contextuel dépendant de l'usage local du mot qu'on cherche à représenter.

Nous identifions trois problèmes majeurs des plongements statiques résolus par l'utilisation des plongements contextuels. D'abord, dans leur formulation originale, les plongements statiques sont incapables de gérer la polysémie puisque qu'un vecteur unique global est pré-attribué à chaque mot. Dans le meilleur des cas, les plongements statiques enrichis par des connaissances (voir section 1.2.1.4) parviennent à traiter la polysémie grâce à la division des mots en concepts, mais cela repose sur des outils tiers et n'est pas bout-en-bout. À l'opposé, les plongements contextuels adaptent automatiquement leurs représentations à la situation contextuelle présentée, ce qui peut améliorer la qualité des vecteurs. Ensuite, les plongements contextuels peuvent étendre la représentation à une séquence de mots. Dans certains cas, le sens d'un groupe de mots diffère largement du sens de chacun des mots qui compose le groupe. Par exemple, le sens de l'expression *avoir le beurre et l'argent du beurre*

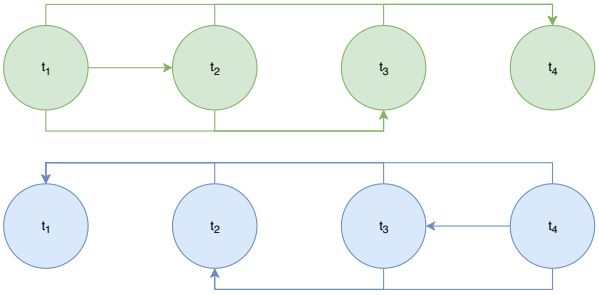


FIGURE 1.9 – Modélisation autorégressive d'une séquence de tokens (t_1, t_2, t_3, t_4). Le modèle du dessus illustre une factorisation en avant et celui du dessous une factorisation en arrière.

est tout à fait différent du sens du mot *beurre* ou *argent* bien qu'étant des éléments essentiels de l'expression. Le dernier problème concerne l'hypothèse distributionnelle qui suppose que le sens d'un mot est distribué dans son contexte, ce qui en pratique se caractérise par un traitement uniforme des fenêtres contextuelles. Comme souligné dans le travail de LING et al., 2015, certains mots du contexte participent davantage que d'autres, ou pire, ont un sens différent.

Une remarque importante est que les plongements contextuels utilisent toujours des plongements statiques pour initier la transition entre le texte et l'espace de représentation continu. Ce n'est qu'ensuite que la séquence des vecteurs, produite depuis la phrase (séquence de mots), est traité par un modèle temporel combinant les caractéristiques statiques des vecteurs pour les enrichir d'information contextuelle.

Le formalisme avancé section 1.1.2 est incorrect pour les plongements contextuels. Dans ce cas précis, le plongement est de la forme $e : V \times C \to \mathbb{R}^n$ avec $C \subseteq V^k$, et k, la taille de la fenêtre contextuelle prise en charge par le modèle.

Selon YANG et al., 2019, il y a deux types d'approches de plongements contextuels : *autoregressive* et *autoencoding*. Nous détaillons dans les deux parties suivantes ces modélisations. Nous considérons une suite de mots (phrase) $S = w_1, ..., w_T$, le but final étant de produire une représentation contextuelle pour chaque mot w_i en utilisant $(w_j)_{j \in [\![1, i-1]\!]}$ pour un modèle autorégressif et $(w_j)_{j \in [\![1,T]\!] \setminus \{i\}}$ pour un modèle d'autoencoding.

1.3.1 Plongements autorégressifs

Les plongements autorégressifs sont inspirés des modèles de langue basés sur des n-grammes (JÓZEFOWICZ et al., 2016 ; RADFORD, 2018). En effet, la modélisation du langage à partir des n-grammes propose une estimation de $p(w_t|w_{t-1}, ..., w_{t-n})$ (voir figure 1.9), c'est-à-dire la probabilité d'apparition d'un mot connaissant les n mots précédents. Les modèles autorégressifs étendent cette modélisation en représentant les mots précédents avec des vecteurs contextuels, en recherchant les informations pertinentes au calcul de la probabilité. L'objectif est de maximiser (YANG et al., 2019) au regard de paramètres θ :

$$p_\theta(S) = p_\theta(w_1, ..., w_T) = \prod_{i=1}^{T} p_\theta(w_i | w_{j<i}), \tag{1.17}$$

pour toutes les phrases S du corpus. On cherche à ce que chaque mot soit explicable par les mots qui le précèdent.

Les premiers plongements autorégressifs utilisent la combinaison : plongements statiques + réseaux de neurones récurrents (*Recurrent Neural Network*s, RNNs) + softmax (AKBIK, BLYTHE et VOLLGRAF, 2018 ; PETERS et al., 2018 ; MCCANN et al., 2017 ; JÓZEFOWICZ et al., 2016), pour représenter les séquences temporelles avec un vecteur de contexte caché et ensuite déterminer en sortie la distribution des mots connaissant le contexte. Aussi, la séquence est inversée de sorte à faire un traitement du langage bidirectionnel. Cependant, comme mentionné par DEVLIN et al., 2019 ; YANG et al., 2019, le plongement résultant est divisé, puisque le contexte précédent et subséquent du mot sont considérés indépendamment. Si cela n'était pas le cas, le modèle aurait accès aux mots suivants, et donc, l'objectif de modélisation serait caduque puisque trivial. Comme nous le montrerons dans la section suivante, les architectures de type autoencoding parviennent à faire une inférence à partir d'un contexte unifié, pouvant mener à une meilleure compréhension du langage (DEVLIN et al., 2019 ; YANG et al., 2019).

Les modèles plus récents reposent largement sur le décodeur du modèle transformer, introduit par VASWANI et al., 2017. C'est le cas des modèles de type *Generative Pre-Training* (GPT) (RADFORD, 2018 ; RADFORD et al., 2019 ; BROWN et al., 2020). Utiliser le modèle transformer plutôt que des LSTM semble bénéfique, malgré le fait que cela augmente grandement le nombre de paramètres du modèle.

La problématique OOV peut survenir avec des plongements autorégressifs, ou même contextuels de manière plus générale, puisqu'ils utilisent un vocabulaire préétabli pour associer les mots à leur vecteur statique. C'est pourquoi, des réseau de neurones de convolution (*Convolutional Neural Network*, CNN) de caractères dans JÓZEFOWICZ et al., 2016 et MCCANN et al., 2017, ou un vocabulaire basé sur du Byte-Pair-Encoding, dans RADFORD, 2018, ou même des modèles WordPiece pour segmenter les mots dans DEVLIN et al., 2019, ont été proposés pour essayer de prendre en charge la morphologie des mots et étendre la représentation aux mots inconnus. Un autre solution et d'entraîner un système se basant directement sur les caractères un-à-un plutôt que des mots, comme avec FLAIR proposé par AKBIK, BLYTHE et VOLLGRAF, 2018. EL BOUKKOURI et al., 2020 proposent CharacterBERT, un modèle qui repose sur une idée identique et qui réutilise le modèle BERT, remplaçant la segmentation WordPiece par des CNNs sur les caractères. Les auteurs démontrent une meilleure performance pour gérer les mots spécifiques à certains corpus (e.g. textes médicaux), tout en éliminant les problématiques d'OOV.

1.3.2 Plongements autoencoding

Les plongements utilisant l'autoencoding, comme BERT (DEVLIN et al., 2019), essaie d'encoder le contexte bidirectionnel pour prédire le mot cible. On s'assure

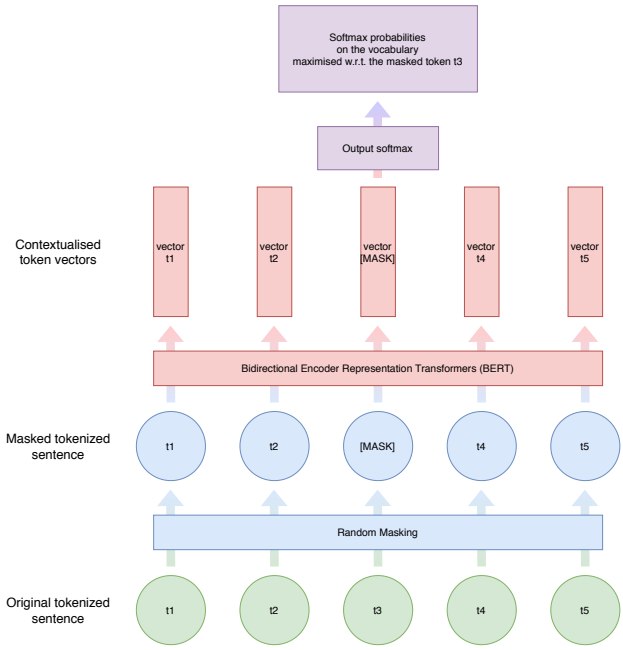


FIGURE 1.10 – Modélisation masquée du langage, utilisée pour entraîner des plongements autoencoding. Les tokens sont aléatoirement remplacé par un token `[MASK]`. À la fin, la représentation vectorielle de `[MASK]` est donnée à une couche de classification sur le vocabulaire. L'objectif est de prédire le token caché par `[MASK]`.

qu'aucune information à propos de ce dernier n'est donné au modèle pour ne pas rendre triviale la tâche de prédiction. On utilise pour cela un token `[MASK]` pour cacher le mot cible au modèle. Le modèle transformer inclut l'information contenue par les vecteurs des mots du contexte les uns dans les autres, de façon à ce que chaque mot ait une représentation dépendante de son contexte. L'objectif est de faire apprendre au modèle la distribution des mots cibles connaissant le contexte, c'est-à-dire que la représentation vectorielle de `[MASK]` permette de reconstruire cette distribution à travers la couche de classification du modèle. La figure 1.10 illustre cette tâche dite de modélisation masquée du langage. Après cette étape en principe le modèle est affiné sur une tâche finale réelle. Nous n'entrerons pas davantage dans les détails de ces méthodes, car ce n'est pas le coeur du sujet de notre travail.

1.4 Caractérisation des plongements

Nous proposons une analyse des principales propriétés des plongements introduits jusqu'ici. Sur le tableau 1.1, nous avons les propriétés suivantes :

— **OOV** : un plongement de mot gère l'OOV s'il est capable de donner des représentations à des mots inconnus. En principe, nous ne l'avons pas mentionné jusqu'ici, chaque méthode peut gérer cette problématique en introduisant un token `<UNK>` pour remplacer les mots qui sont inconnus. Cette solution n'est pas satisfaisante car tous les mots inconnus ont la même représentation, et apprendre cette représentation suppose d'avoir un a priori sur la distribution

TABLEAU 1.1 – Propriétés des algorithmes de plongement.

Algorithme de Plongement	OOV	Contextualisé	Auto-supervisé	Non-euclidien	Multi-source
GloVe (PENNINGTON, SOCHER et MANNING, 2014)			✓		
Word2Vec (MIKOLOV, SUTSKEVER et al., 2013)			✓		
FastText (BOJANOWSKI et al., 2017)	✓		✓		
SensEmbed (IACOBACCI, PILEHVAR et NAVIGLI, 2015)			✓		✓
CoupledCR (LU et al., 2020)			✓		✓
Word2Gauss EL (VILNIS et MCCALLUM, 2015)			✓	✓	
Word2Gauss Wasserstein (C. SUN et al., 2018)			✓	✓	✓
Poincaré GloVe (TIFREA, BECIGNEUL et GANEA, 2019)			✓	✓	
Lorentz Skip-Gram (LEIMEISTER et WILSON, 2018)			✓	✓	
ELMo (PETERS et al., 2018)	—	✓	✓		✓
FLAIR (AKBIK, BLYTHE et VOLLGRAF, 2018)	✓	✓	✓		✓
BERT (DEVLIN et al., 2019)	—	✓	✓		✓
CharacterBERT (EL BOUKKOURI et al., 2020)	✓	✓	✓		✓
GPT-3 (BROWN et al., 2020)	—	✓	✓		✓

des mots inconnus dans les phrases non apprises. Dans le tableau nous indiquons par une coche (✓) les algorithmes capables de généraliser à des mots inconnus de façon plus fine, en utilisant par exemple, l'information des sous-mots. Le symbole — indique les méthodes qui gère l'OOV mais pas de façon totale. En fait, BERT et XLNET travaillent sur les mots avec un tokenizer qui les segmente et ces segments peuvent être partagés entre les mots. Cette solution augmente grandement la couverture des mots (surtout des mots rares) sans qu'elle soit complète. Dans tous les cas, les méthodes contextuelles ont la possibilité d'utiliser les mots du contexte pour donner une représentation au mot si ses sous-mots ne sont pas connus par le modèle (dans ce cas, seul le plongement statique fait défaut).

— **Contextualisé** : quand la représentation d'un mot est calculé pour chaque occurrence en fonction de son contexte.

— **Auto-supervisé** : (BEYER et al., 2019) : lorsque la tâche utilisée pour entraîner le plongement est auto-supervisée, ce qui veut dire qu'une tâche supervisée de substitution est automatiquement extraite d'un jeu de données sans étiquettes. Par exemple pour Word2Vec SG, on essaye de prédire les mots de la fenêtre contextuelle.

— **Non-euclidien** quand la représentation utilise une géométrie non-euclidienne.

— **Multi-sources** : les plongements utilisent un critère d'optimisation principal. Cependant, il est possible d'enrichir les vecteurs des mots en utilisant plusieurs sortes de jeux de données et plusieurs critères. Par exemple dans C. SUN et al., 2018, les auteurs utilisent la divergence K-L et la distance de Wasserstein pour projeter simultanément un texte brut et une base de connaissances. Les plongements contextuels sont particulièrement multi-sources puisqu'ils requièrent une étape d'affinement.

Chapitre 2

Evaluation des plongements de mots

Sommaire

2.1 Evaluations, entre complexité et nécessité 40
2.2 Les modes d'évaluation 40
2.2.1 Evaluations globales 41
2.2.2 Evaluations intrinsèques 45
2.2.3 Evaluations extrinsèques 49
2.2.4 Discussion : l'évaluation parfaite 50
2.3 Corrélations entre les évaluations 51
2.3.1 Expériences et résultats 51
2.3.2 Perspectives 57

Les méthodes de plongements possèdent des structures différentes selon leur support géométrique, et recherchent à capturer des aspects abstraits du langage (la similarité, l'analogie ou la temporalité). Ce chapitre est voué à l'évaluation des méthodes de plongements statiques et à la recherche de qualité structurelle conduisant à l'amélioration des représentations. On présente dans les sections 2.1 et 2.2 une présentation des méthodes d'évaluation des plongements existantes et de celles introduites par cette thèse. La section 2.3 étudie les corrélations entre les métriques d'évaluation et offre des perspectives pour améliorer les plongements ainsi que la façon de les évaluer.

2.1 Evaluations, entre complexité et nécessité

Evaluer quantitativement l'intégration des caractéristiques linguistiques dans un plongement est une tâche ardue, principalement à cause de la pléthore d'axes d'évaluations possibles et sur lesquelles le plongement a souvent des résultats hétérogènes (SCHNABEL et al., 2015). Par conséquent, il est difficile de résumer par une seule valeur la qualité des plongements, expliquant de façon absolue la complexité du modèle. Il est plutôt d'usage d'analyser la performance des plongements sur plusieurs tâches ou sur la tâche d'intérêt.

L'évaluation des plongements est un sujet largement étudié. Beaucoup de ressources et de procédures, listés exhaustivement dans le travail de BAKAROV, 2018, ont été proposées pour comparer de nombreux plongements de l'état de l'art comme GloVe (PENNINGTON, SOCHER et MANNING, 2014) ou Word2Vec SG (MIKOLOV, SUTSKEVER et al., 2013). Une catégorisation a été vite proposée pour différencier les méthodes d'évaluation se focalisant soit sur la qualité structurelle (conformité au jugements humains, répartition du voisinage...) soit sur la performance concrète via des tâches utiles (ayant un intérêt industriel) comme la reconnaissance d'entités nommées, la classification de texte ou la traduction.

Nous proposons dans ce chapitre d'étendre la gamme des évaluations globales (mesurant la qualité structurelle des plongements) et de regarder la corrélation de celles-ci avec des tâches extrinsèques (celles ayant intérêt direct). L'objectif premier soutenant le calcul de cette corrélation est de déterminer des caractéristiques structurelles déterminantes pour la performance utile, et ainsi, les imposer à l'apprentissage.

2.2 Les modes d'évaluation

On différencie trois catégories de méthodes d'évaluations : globales, intrinsèques et extrinsèques. Les catégories globales et intrinsèques analysent des aspects structurels abstraits (orthogonalité, alignement) ou concrets (conformité aux jugements humains), et la catégorie des tâches extrinsèques juge la qualité du plongement sur une tâche tierce. Il est important de remarquer que les évaluations globales et intrinsèques s'appliquent essentiellement sur des plongements statiques plutôt que contextuels, pour lesquelles la représentation d'un mot hors de tout contexte n'a pas

vraiment de sens. Par conséquent, dans les deux sections traitant de ces évaluations, les plongements considérés sont statiques.

2.2.1 Evaluations globales

Les évaluations globales sont des mesures quantifiant des caractéristiques sur la distribution des vecteurs dans son ensemble, en omettant volontairement les spécificités individuelles des vecteurs (AMSALEG et al., 2015). En conséquence, ces mesures ne nécessitent aucune donnée pour être calculées.

2.2.1.1 Dimensionalité intrinsèque

La dimensionalité intrinsèque est une métrique locale introduite par HOULE, KASHIMA et NETT, 2012, ayant pour but de traduire la complexité d'un voisinage en un point x d'un espace de plongement. La complexité étant la dimensionalité minimale pour décrire l'ensemble des points se trouvant dans la couronne définie par deux sphères concentriques de centre x et d'épaisseur ε. Comme souligné par CLAVEAU, 2018, une couronne de haute dimension indique qu'un léger déplacement sur x changerait complètement les points contenus dans la couronne et donc le voisinage de x, et donc résulte d'une mauvaise qualité pour la recherche d'information autour de x.

Un estimateur local de la dimension intrinsèque d'un point x, dénoté ID_x, peut être calculé en suivant AMSALEG et al., 2015. En notant, $\mathcal{N}_{\|.\|_2}(x,k)$ les k plus proches voisins de x et $\|.\|_2$ la norme L2, alors :

$$\mathrm{ID}_x = -\left[\frac{1}{k}\cdot\sum_{y\in\mathcal{N}_{\|.\|_2}(x,k)}\ln\frac{\|y-x\|_2}{1+\max\limits_{z\in\mathcal{N}_{\|.\|_2}(x,k)}\|z-x\|_2}\right]^{-1} \tag{2.1}$$

estime la dimensionalité intrinsèque de x.

Comme évoqué, cet indicateur est local et ne décrit que la complexité du voisinage d'un point x. En revanche, nous pouvons créer des mesures locales en étudiant la distribution de ces valeurs pour l'ensemble des points du plongement, à savoir : la moyenne ou l'écart-type. L'intuition étant que la performance finale est corrélée à l'agrégation des comportements locaux de la dimensionalité intrinsèque (prenant alors la forme d'une mesure globale), en particulier pour les tâches faisant intervenir de la recherche d'information.

2.2.1.2 Rang effectif et dimension empirique

On peut réutiliser, pour l'évaluation globale des plongements statiques, des méthodes pour étudier des matrices issues du traitement de l'image et du traitement du signal. Les plongements statiques étant essentiellement des matrices, il est donc simple d'étendre des mesures globales sur des matrices à ces derniers.

Notant W la matrice d'un plongement de dimension D, on peut formuler le rang effectif de W (ROY et VETTERLI, 2007), noté erank(W), comme :

$$\text{erank}(W) = \exp\left(-\sum_{i=1}^{D}\left[\frac{s_i}{\sum_{j=1}^{D} s_j}\cdot\log\left(\frac{s_i}{\sum_{j=1}^{D} s_j}\right)\right]\right), \tag{2.2}$$

où $\boldsymbol{s} = (s_i)_{i\in[\![1,D]\!]}$ sont les valeurs singulières de W.

On peut discerner l'entropie de Shannon dans l'Eq. 2.2, qui correspond à la quantité d'information contenue dans les valeurs singulières de W. Idéalement, les valeurs singulières doivent contenir une quantité égale d'information (ce qui correspond à la fois au cas d'entropie élevée et au cas où les valeurs sont égales), puisqu'une valeur singulière prépondérante (associée donc à une entropie basse) indique un usage sous-optimal de la dimensionalité de l'espace. En effet, une entropie basse indique que certaines valeurs propres sont prépondérantes, et donc que l'ensemble de l'information de la matrice peut être exprimée par ces dernières. En d'autres termes, il y a trop de dimensions par rapport à l'information contenue dans la matrice.

Le rang d'une matrice peut être intuitivement compris comme les degrés de liberté des vecteurs colonnes, ou alors comme l'indépendance entre ces derniers. C'est une valeur discrète qui ne traduit pas l'intensité de cette indépendance. En effet, des vecteurs orthogonaux ont l'air plus indépendants que des vecteurs quasiment colinéaires, cependant, le rang nous dit qu'il y a deux vecteurs indépendants dans chaque cas.

Le rang effectif va plus loin en quantifiant cette notion. Il propose une interprétation continue du rang, en retournant notamment dans le cas de vecteurs quasi-colinéaires un rang quasiment égal à 1. La figure 2.1 illustre ces propos : à mesure que l'angle entre les vecteurs croît vers $\frac{\pi}{2}$, et donc que les vecteurs « *deviennent moins colinéaires* », le rang effectif augmente.

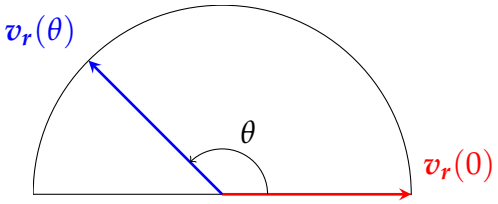
Sur cette Figure, on étudie aussi la dimension empirique (edim) de POLING et LERMAN, 2014 :

$$p \in [0,1], \text{edim}(W,p) = \frac{||\boldsymbol{s}||_p}{||\boldsymbol{s}||_{\frac{p}{1-p}}}, \tag{2.3}$$

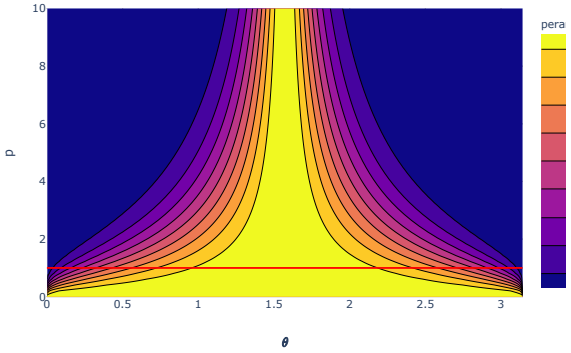
avec $\|\boldsymbol{x}\|_p = (\sum_i x_i^p)^{\frac{1}{p}}$, et on propose le rang effectif étendu (perank) :

$$p \in \mathbb{R}, \text{perank}(W,p) = \exp\left(-\sum_{i=1}^{D}\left[\frac{s_i^p}{\sum_{j=1}^{D} s_j^p}\cdot\log\left(\frac{s_i^p}{\sum_{j=1}^{D} s_j^p}\right)\right]\right). \tag{2.4}$$

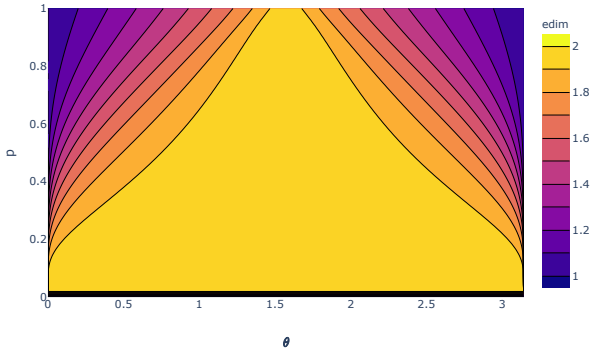
Ces deux extensions introduisent un paramètre p qui permet de contrôler la sensibilité du rang effectif à l'orthogonalité de la famille de vecteurs étudiée (la matrice).



(A) On considère un demi-cercle de rayon r. Un vecteur est inscrit dans ce demi-cercle et défini par son angle θ avec un vecteur référence $v_r(0)$.



(B) Rang effectif étendu, $p \in [0, 10]$. La ligne rouge horizontale correspond au rang effectif standard (le cas $p = 1$).



(C) Dimension empirique, $p \in [0, 1]$.

FIGURE 2.1 – Exemple pour montrer le comportement de perank et edim. Une matrice contenant deux vecteurs à deux dimensions est construite en prenant les vecteurs de 2.1a. Les valeurs de perank et edim sont calculées en fonction de l'angle $\theta \in [0, \pi]$.

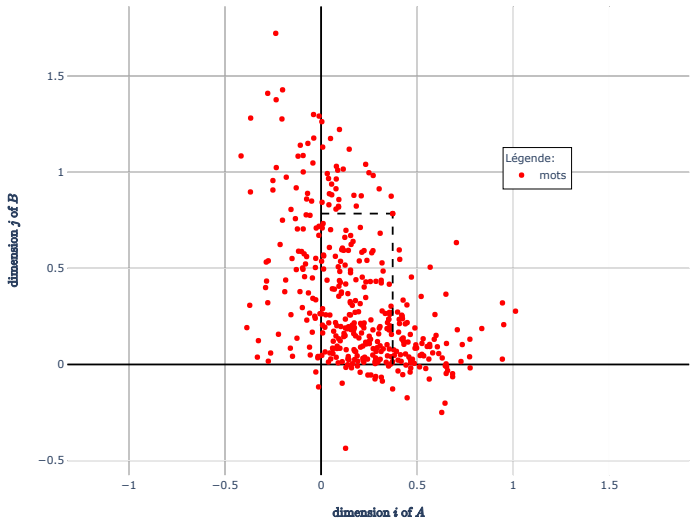


FIGURE 2.2 – Valeur de la dimension j d'un plongement B en fonction de celle de la dimension i d'un plongement A, pour des mots aléatoires (marques rouges).

Le rang effectif étendu n'étant pas limité à l'intervalle $[0, 1]$, c'est pour cette raison que nous l'avons construit et proposé.

Nous pouvons remarquer que l'ensemble des métriques de cette section varie dans l'intervalle $[1, D]$. 1 ou proche correspondent à des cas où l'entropie est faible et à des vecteurs quasiment colinéaires n'exploitant pas ou très peu les dimensions disponibles. À l'inverse D ou proche correspondent à l'orthogonalité ou à des vecteurs quasi-orthogonaux. Les valeurs intermédiaires doivent s'interpréter comme le nombre de dimensions avec lequel il est possible de compresser la matrice avec une perte d'information faible (ROY et VETTERLI, 2007).

2.2.1.3 Auto-alignement

Nous proposons une nouvelle mesure globale basée sur QVEC, une métrique définie par TSVETKOV et al., 2015. À la base, QVEC est une mesure intrinsèque qui quantifie le degré de conformité entre le plongement et des connaissances linguistiques humaines, comme WordNet (G. A. MILLER, 1995). Pour ce faire, on transforme manuellement les connaissances linguistiques en un plongement (en utilisant, par exemple, les catégories associées aux mots) et on regarde si les dimensions de ce plongement peuvent correspondre avec celle d'un plongement de type Word2Vec. Autrement dit, on cherche à savoir si la dimension d'un plongement A représente la même information qu'une dimension d'un autre plongement B, et de quantifier ce partage et leur intensité.

Cette quantification se fait grâce au coefficient de corrélation de Pearson entre

une dimension de A et une autre de B. Les mots sont considérés comme les observations. On observe, par exemple, sur la figure 2.2 une dépendance entre les dimensions (i et j) des deux plongements (A et B), signifiant un partage d'information sur ces axes.

En agrégeant les informations sur l'ensemble des dimensions, on obtient une matrice de corrélation dimension à dimension de A pour B (ou l'inverse). Ensuite, il s'agit de déterminer les paires de dimensions correspondantes pour avoir une corrélation totale maximale, en s'assurant que chaque dimension de A (resp. B) n'est alignée qu'une seule fois avec une dimension de B (resp. A).

L'auto-alignement reprend l'idée de QVEC en proposant de calculer la matrice de corrélation de A avec A, et de l'utiliser pour extraire des métriques résumant les informations de corrélation, en s'affranchissant, par conséquent, de la comparaison avec des données linguistiques humaines. On restreint la matrice à la partie triangulaire strictement supérieure pour enlever la répétition par symétrie et la diagonale de 1, et on prend les valeurs absolues des corrélations. On appelle cette partie de matrice l'Auto-Corrélation Dimensionnelle (ACD).

L'intérêt d'étudier cette partie de la matrice de corrélation est de mettre en exergue les redondances d'information d'une dimension à l'autre. Si une dimension est fortement corrélée à une autre dans le plongement, cela peut traduire que les deux dimensions encodent la même information et donc qu'une seule des deux est nécessaire.

Prendre la moyenne, la médiane ou autre de l'ACD peut fournir des indicateurs sur la redondance d'information et l'usage de l'espace par le plongement. Contrairement aux métriques globales de type rang effectif, l'ACD ne requiert pas de décomposition singulière ce qui facilite son intégration comme critère de régularisation.

2.2.2 Evaluations intrinsèques

Les évaluations intrinsèques (SCHNABEL et al., 2015) estiment la concordance entre les structures des plongements et des jugements humains. Pour cela, on doit se munir de données linguistiques permettant de mener cette comparaison. On présente trois évaluations intrinsèques largement utilisées (la similarité, l'analogie et la catégorisation), et on rappelle l'expression de la similarité cosinus pour deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$:

$$\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}. \tag{2.5}$$

2.2.2.1 Similarité

La tâche de similarité consiste à donner un score à une paire de mots, en utilisant le plongement, et à le comparer avec le score moyen donné par un panel de juges humains. Si les deux scores concordent, on peut en déduire que les concepts des mots du jeu de données et la relation entre les paires sont bien représentés. Certains jeux

TABLEAU 2.1 – Jeux de données de similarité (BAKAROV, 2018).

Nom	Quantité	Type de relations
WordSim353 (FINKELSTEIN et al., 2002)	353	Synonymy (common words)
MEN (BRUNI, TRAN et BARONI, 2014)	3000	Synonymy (common words)
RG (RUBENSTEIN et GOODENOUGH, 1965)	65	Synonymy (common words)
SimLex-999 (HILL, REICHART et KORHONEN, 2015)	999	Synonymy (common words)
SimVerb (GERZ et al., 2016)	3500	Synonymy (essentially verbs)
RW (LUONG, SOCHER et MANNING, 2013)	2034	Synonymy (low frequency words)
HyperLex (VULIĆ et al., 2017)	2616	Entailment (common words)

de données donnent un score de synonymie entre les paires, d'autres fournissent un score d'hyperonymie (le premier mot peut-il être le parent du second ?), le type de score est donc une caractéristique du jeu. Dans le tableau 2.1, on liste un ensemble de jeux de données qui sont souvent utilisés dans l'état de l'art, et pour chacun, nous donnons le type des relations et la nature des mots.

Les scores construits avec le plongement sont le plus généralement obtenus en prenant la similarité cosinus entre les paires. On peut cependant définir d'autres heuristiques lorsque la similarité cosinus n'est pas adaptée, en particulier dans le cas où on veut noter des relations asymétriques (NICKEL et KIELA, 2017 ; C. SUN et al., 2018). En finalité, pour obtenir un quantificateur de l'accord entre les scores issus du plongements et ceux du jeu de données, on utilise le coefficient de corrélation de Spearman.

2.2.2.2 Analogie

L'analogie, proposée par MIKOLOV, SUTSKEVER et al., 2013, évalue le plongement des relations linguistiques. Considérant trois mots w_A, w_B, et w_C, tels que w_A est relié avec w_B par une relation $\mathcal{R}$, la tâche consiste à trouver un quatrième mot w_D dans le plongement de façon à ce que w_C soit relié à w_D par $\mathcal{R}$. Par exemple, considérant la relation *Paris : France*, on souhaite trouver *Espagne* avec *Madrid*.

Plus précisément, w_D est retrouvé comme la solution d'un problème mathématique, présenté par LEVY, GOLDBERG et DAGAN, 2015 :

— 3CosAdd (1) : $\underset{w_D \in V \setminus \{w_A, w_B, w_C\}}{\operatorname{argmax}} \cos(v_{w_B} - v_{w_A} + v_{w_C}, v_{w_D})$,

— 3CosAdd (2) : $\underset{w_D \in V \setminus \{w_A, w_B, w_C\}}{\operatorname{argmax}} \cos(v_{w_D}, v_{w_B}) - \cos(v_{w_D}, v_{w_A}) + \cos(v_{w_D}, v_{w_C})$,

— 3CosMul : $\underset{w_D \in V \setminus \{w_A, w_B, w_C\}}{\operatorname{argmax}} \frac{\cos(v_{w_D} v_{w_B}) \cdot \cos(v_{w_D}, v_{w_C})}{\cos(v_{w_D}, v_{w_A}) + \epsilon}$, $\epsilon = 0.001$.

D'un point de vue géométrique, on considère que la relation linguistique prend la forme d'un vecteur constant $v_{\mathcal{R}}$ qui s'additionne au vecteur du mot pour appliquer cette relation, $v_{w_A} + v_{\mathcal{R}} = v_{w_B}$. On peut donc déterminer $v_{\mathcal{R}}$ par $v_{w_B} - v_{w_A}$ et rechercher l'élément le plus proche de $v_{w_C} + v_{\mathcal{R}}$ parmi les vecteurs du plongement pour retrouver l'autre membre de la relation w_D. La figure 1.1 du chapitre 1 est la

TABLEAU 2.2 – Jeux de données d'Analogie (BAKAROV, 2018).

Nom	Quantité	Relations
Google Analogy (MIKOLOV, K. CHEN et al., 2013)	19000	Capital, Country, Family, Currency, Cities, Morphology
MSR (MIKOLOV, YIH et ZWEIG, 2013)	8000	Morphology

TABLEAU 2.3 – Jeux de données pour la catégorisation (BAKAROV, 2018).

Nom	Quantité	Nombre de classes
Battig (BARONI, MURPHY et al., 2010)	5330	56
AP (ALMUHAREB, 2006)	402	21
BLESS (BARONI et LENCI, 2011)	200	17

représentation de ce problème. Les capitales sont espacées approximativement du même vecteur de leur pays.

Le tableau 2.2 contient des jeux de données utilisés pour réaliser cette tâche, ainsi que les types de relations qu'elles contiennent. À la différence des tâches de similarité, les tâches d'analogie se focalisent davantage sur les aspects géométriques du plongement, en considérant la relation comme une translation plutôt qu'une simple distance.

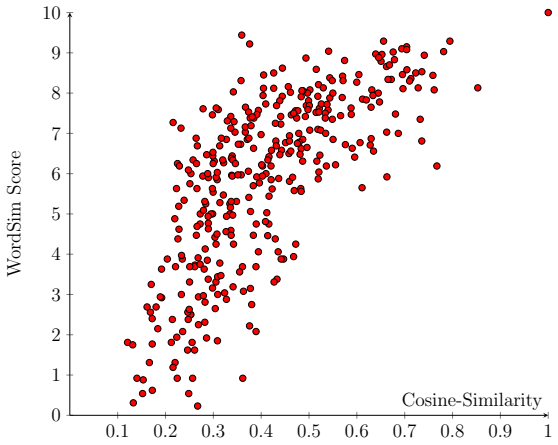
2.2.2.3 Catégorisation

La catégorisation quantifie la capacité du plongement à distinguer des ensembles de concepts. Un jeu de données de catégorisation se caractérise par un ensemble de N classes $\mathcal{C} = (C_i)_{[\![1,N]\!]}$, un ensemble de K mots $\mathcal{W} = (w_i)_{[\![1,K]\!]}$. Chaque mot appartient à une classe spécifique. Le but est de reconstruire à partir du plongement N classes $\mathcal{C}^r = (C_i^r)_{[\![1,N]\!]}$ avec les vecteurs des K mots. L'outil CLUTO (KARYPIS, 2003), avec les paramètres par défaut, est généralement utilisé pour réaliser cette opération (SCHNABEL et al., 2015 ; BARONI, DINU et KRUSZEWSKI, 2014). La pureté permet de quantifier l'accord entre $\mathcal{C}$ et $\mathcal{C}^r$ (MANNING, RAGHAVAN et SCHÜTZE, 2008) :

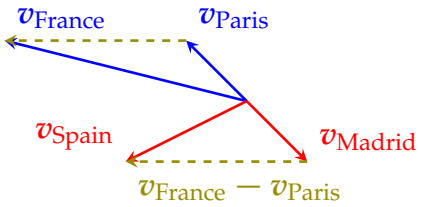
$$\text{Purity}(\mathcal{C}, \mathcal{C}^r) = \frac{1}{K} \sum_{i=1}^{N} \max_{j \in [\![1,N]\!]} \#(C_i \cap C_j^r). \tag{2.6}$$

Le tableau 2.3 contient des jeux de données de catégorisation ainsi que le nombre de classes qu'ils contiennent.

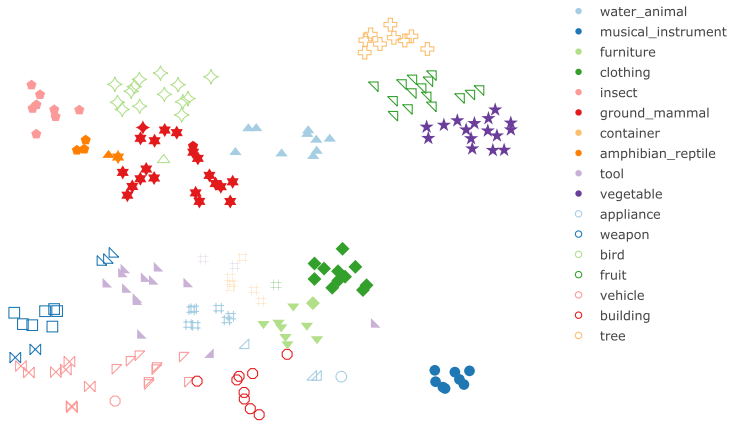
Enfin, nous donnons une représentation graphique des 3 tâches intrinsèques évoqués jusqu'ici, sur la figure 2.3. Cela permet de visualiser le processus d'évaluation avec : un graphe des paires de mots pour la similarité (2.3a), la relation sous forme géométrique pour l'analogie (2.3b), le partitionnement en classes pour la catégorisation (2.3c).



(A) Tâche de similarité avec WordSim (FINKELSTEIN et al., 2002). Les points rouges représentent les paires de mots, l'abscisse étant la similarité cosinus pour la paire et l'ordonnée étant le score de similarité du jeu de données. Le coefficient de Spearman vaut 0.75 pour ce graphe.



(B) Tâche d'analogie, la relation *être la capitale de* est supposée être linéaire et est extraite de l'exemple (*France*, *Paris*).



(C) Tâche de catégorisation sur le jeu de données Bless (BARONI et LENCI, 2011) (en suivant BAKAROV, 2018) avec un plongement de 300 dimensions projeté sur un plan 2D avec t-SNE : chaque couleur représente une catégorie réelle, les marques représentent les catégories reconstruites.

FIGURE 2.3 – Tâches intrinsèques

2.2.3 Evaluations extrinsèques

Les évaluations extrinsèques (SCHNABEL et al., 2015) sont différentes par nature car elles consistent à observer la performance d'un modèle (utilisant les plongements en entrée) sur des tâches concrètes et requièrent des représentations plus complexes que des plongements pour être résolues, parce qu'elles se focalisent sur des aspects plus abstraits du langage. Par exemple, le fait de devoir traiter la temporalité d'une phrase, de détecter des entités ou des intentions sont des informations qui ne sont généralement pas présentes lors de la phase d'entraînement des plongements, et qui nécessitent une phase d'affinement et une couche modélisation supplémentaire pour intégrer et ajouter ces informations. En pratique, des plongements statiques sont donnés à des RNNs pour classifier de façon séquentielle une phrase. Les plongements dynamiques auront aussi des couches finales affinées à la tâche, mais à l'inverse des plongements statiques ces derniers ont déjà une compréhension de la temporalité du langage. Dans les deux cas, les couches finales ont pour objectif d'extraire les caractéristiques cruciales pour la tâche d'intérêt.

On donne trois tâches extrinsèques pour l'exemple, celles-ci seront utilisées dans la section suivante pour le calcul des corrélations inter-tâches :

— **La reconnaissance d'entités nommées (*Named Entity Recognition*, NER)** est une tâche de classification séquentielle, dans laquelle une phrase est donnée et le modèle doit assigner à chaque mot de cette phrase une classe d'entité particulière (une personne, une organisation ou autre). CoNLL2003 (SANG et MEULDER, 2003) est un jeu de données qu'on utilisera pour la suite, et qui est composé d'articles de presse.
— **L'analyse de sentiment (*Sentiment Analysis*, SA)** est une tâche de classification dans lequel le modèle doit donner une classe de sentiment à chaque phrase. On utilise les données de *Stanford Sentiment Treebank* (SST) (SOCHER et al., 2013), qui est un ensemble d'avis sur des films et le sentiment très positif, positif, neutre, négatif ou très négatif associé à l'avis.
— **La classification de textes (*Text Classification*, TC)** est une tâche de classification au niveau du document, dans lequel le modèle doit classifier le texte dans diverses catégories. Les données AGNews [1] sont utilisées, c'est une ressource en ligne composée d'articles de presse dans quatre catégories (World, Sports, Business, et Sci/Tech).

Ces trois tâches reposent sur des aspects assez différents du langage, ce qui en fait à notre sens, une bonne référence. Comme nous l'avons dit, ces tâches ont besoin d'un modèle supplémentaire pour être résolues :

— Pour NER, on fournit des plongements statiques ou contextuels à un GRU bidirectionnel d'une couche suivie d'une couche CRF qui prédit les classes.
— Pour TC et SA, on utilise une séquence de plongements statiques ou contextuels avec un GRU bidirectionnel d'une couche suivie d'une couche linéaire sortant le bon nombre de classes.

1. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

TABLEAU 2.4 – Evaluations extrinsèques (Précision : P, Rappel : R, F1-score : F1).

Type de plongement	NER			SA			TC		
	P	R	F1	P	R	F1	P	R	F1
GloVe (768d)	0.833	0.809	0.820	0.381	0.358	0.346	0.908	0.908	0.908
Word2Vec SG (768d)	0.882	0.837	0.858	0.447	0.436	0.415	0.920	0.92	0.919
FastText (768d)	0.881	0.852	0.866	0.440	0.434	0.391	0.918	0.918	0.917
BERT	0.921	0.928	0.925	0.508	0.500	0.471	0.937	0.938	0.937

On propose une pré-évaluation des plongements sur cette référence dans le tableau 2.4. Toutes les méthodes utilisent les paramètres par défaut, excepté la dimension qu'on fixe à 768 pour tous les modèles, de façon à être sur la même dimension que BERT. Les modèles décrit ci-dessus sont entraînés avec l'un des quatre plongements pendant cinq époques. Nous utilisons le projet *transformers* de WOLF et al., 2019 pour les modèles BERT.

Sur l'ensemble des tâches, BERT retourne les meilleurs résultats, ce qui peut être expliqué par sa capacité à prendre en compte le contexte dans le calcul de la représentation des mots.

2.2.4 Discussion : l'évaluation parfaite

La qualité d'un plongement est difficile à définir en raison de l'abondance de tâches sur lesquelles il peut être utilisé. Certains seront très efficaces pour certaines tâches, mais beaucoup moins pour d'autres et inversement (CLAVEAU et KIJAK, 2016 ; SCHNABEL et al., 2015). D'un point de vue purement utilitariste, on pourrait s'arrêter à maximiser l'évaluation sur une tâche extrinsèque liée à notre problème concret. Cependant, cela ne reflète pas la qualité de projection de l'ensemble des aspects linguistiques, la tâche extrinsèque ne comptant que sur quelques aspects.

Il est souvent recommandé (SCHNABEL et al., 2015 ; BAKAROV, 2018) d'évaluer son plongement sur un ensemble complet de tâches, bien que la complétude de ces tâches soient difficiles à définir. Les tâches intrinsèques semblent fournir des indicateurs sur le respect des structures linguistiques via des relations géométriques et semblent donc moins spécifiques que des évaluations extrinsèques (BAKAROV, 2018). Les évaluations globales, quant à elles, nous donnent des indications sur l'usage de la dimensionalité du plongement et sur la répartition des informations dans les dimensions de manière agnostique à ce qui est représenté, à savoir le langage. Il semble donc important de se munir d'un grand panel de tâches indépendantes pour avoir une connaissance large de la qualité des aspects linguistiques du plongement, d'où l'intérêt d'étudier la corrélation entre les tâches pour savoir si on mesure et intègre largement ces aspects.

Lorsque cela est possible et que l'on veut qualifier la qualité des plongements de façon générale, les évaluations intrinsèques et globales sont souvent des indicateurs peu coûteux à calculer et moins spécifiques que les tâches extrinsèques. Cependant,

pour les plongements contextuels, il n'est pas facile d'obtenir ces indicateurs puisqu'ils n'ont pas directement des représentations globales des mots (en dehors de tout contexte).

Pour résoudre ce problème d'évaluations et de qualité globale, des références ont été proposée, comme GLUE (A. WANG, SINGH et al., 2018) qui contient neuf tâches extrinsèques dont le but est de couvrir l'ensemble des aspects du langage. SuperGLUE (A. WANG, PRUKSACHATKUN et al., 2019) et FLUE (H. LE et al., 2019) sont deux autres *benchmarks* d'évaluation, le premier se basant sur le principe de GLUE et rendant les tâches plus complexes, le second est similaire à GLUE, mais en français.

2.3 Corrélations entre les évaluations

Nous proposons d'investiguer les corrélations entre les nombreuses évaluations existantes des plongements. Nous nous restreignons à l'évaluation des plongements FastText (BOJANOWSKI et al., 2017), qui sont largement utilisés dans l'état de l'art, cependant notre approche peut être appliquée à d'autres plongements euclidiens statiques. La compréhension des évaluations des corrélations nous donnera plusieurs outils :

— Des évaluations fortement corrélées posent la question sur la pertinence de s'en servir. En fait, il suffirait de n'en garder qu'une seule plutôt qu'un ensemble évaluant le même aspect. De plus, comme dit précédemment, connaître les évaluations indépendantes nous assurent de la couverture de l'évaluation nécessaire au jugement fiable de la qualité globale du plongement.

— Les procédés d'évaluations peu coûteux corrélés à des procédés plus couteux pourraient être utilisés pour sélectionner plus rapidement les hyperparamètres du plongement, en contournant l'étape d'optimisation de la tâche extrinsèque.

— Certaines évaluations ne requièrent aucune donnée (évaluations globales). S'il y a corrélation entre ces évaluations et les autres plus coûteuses et nécessitant des données (évaluations extrinsèques), alors les critères d'évaluations globaux pourraient être transformés en critères d'optimisation utilisés à l'apprentissage pour améliorer les performances des tâches corrélées.

2.3.1 Expériences et résultats

Avec la librairie FastText[2], nous avons généré environ 140 plongements avec un ensemble d'hyper-paramètres différents. Ces paramètres ont été échantillonnés aléatoirement selon les distributions de le tableau 2.5. Avec cette méthode, nous avons créé plusieurs plongements avec différents handicaps ou atouts. Par exemple, la taille de la fenêtre contextuelle influence la capacité à comprendre les aspects syntagmatiques ou paradigmatiques. Le corpus d'entraînement pour les vecteurs

2. `https://github.com/facebookresearch/fastText`

TABLEAU 2.5 – Distributions des hyper-paramètres utilisées pour générer les plongements FastText.

Nom	Distribution (choix uniforme)
Dimension (-dim)	$[20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150]$
Learning rate (-lr)	$[5 \cdot 10^{-2}, 5 \cdot 10^{-3}, 5 \cdot 10^{-4}]$
Windows Size (-ws)	$[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]$
Number of Epoch (-epoch)	$[1, 2, 3, 4, 5]$
Negative Sampling Size (-neg)	$[2, 4, 6, 8, 10, 12, 14]$
Minimum Frequency (-minCount)	$[5, 50, 100, 250, 500]$
Number of Bucket (-bucket)	$[1, 5 \cdot 10^3, 10^4, 5 \cdot 10^4, 10^5, 5 \cdot 10^5, 10^6, 2 \cdot 10^6]$
Min N-gram (-minn)	$[2, 3, 4]$
Max N-gram (-maxn)	minn $+ [0, 1, 2, 3]$
Subsampling Threshold (-t)	$[5 \cdot 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$
Training Corpus (-input)	10%, 25% d'articles aléatoires de Wikipédia ou l'ensemble.

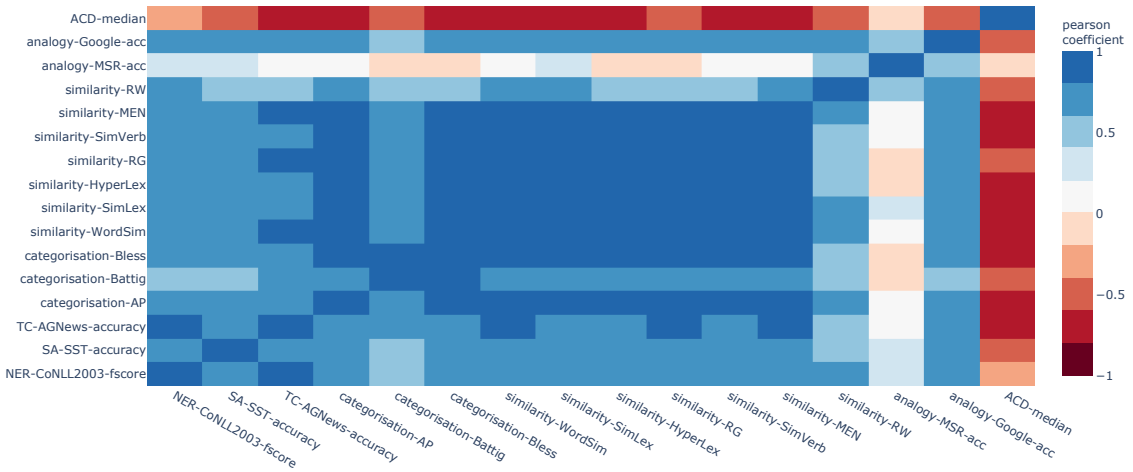


FIGURE 2.4 – Matrice de corrélation de Pearson entre les évaluations. Les trois premières colonnes sont des tâches extrinsèques, les suivantes sont intrinsèques, et la dernière est globale (ACD médiane).

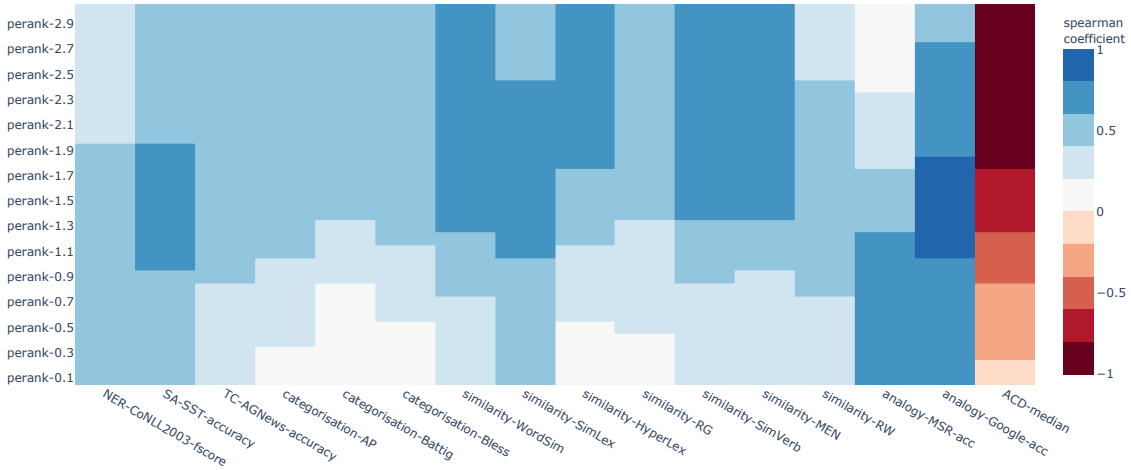
est issu des dumps de Wikipédia [3], et on choisit de prendre soit 10%, 25% ou 100% du corpus (ce choix est un hyper-paramètre).

À la fin de la phase d'entraînement nous prenons au plus les 200 000 mots les plus fréquents des plongements, de façon à pouvoir calculer les évaluations globales et intrinsèques. Pour les évaluations extrinsèques, les représentations sont calculées avec les plongements FastText (utilisant les n-grammes). On utilise les mêmes réseaux que précédemment (voir section 2.2.3).

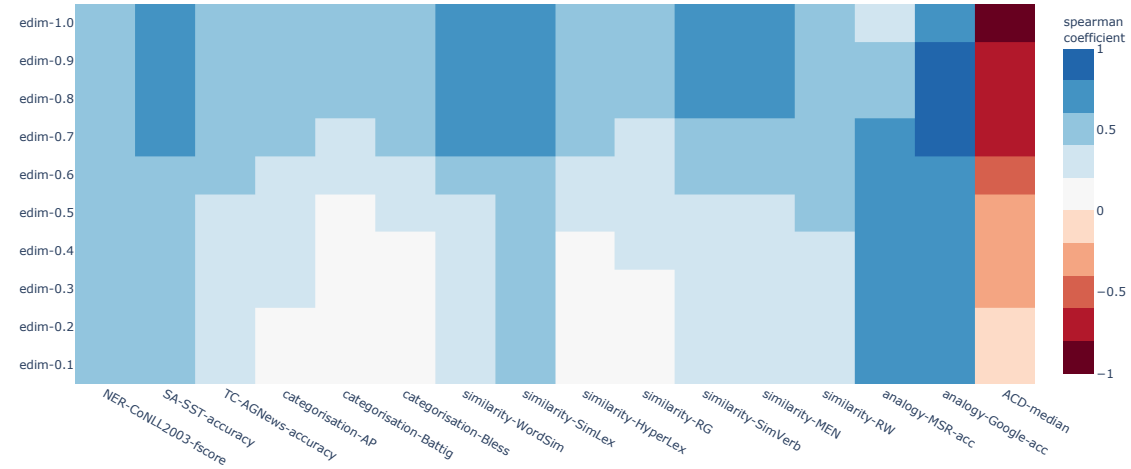
Pour simplifier l'analyse des résultats nous présentons les figures suivantes (à chaque fois le nom de la tâche correspond à son type suivi du jeu de données utilisée) :

— La figure 2.4 résume les corrélations de Pearson entre chaque paire d'évaluations (intrinsèques, extrinsèques et l'ACD médiane). Le coefficient de Pearson

3. `https://dumps.wikimedia.org/wikidatawiki/`

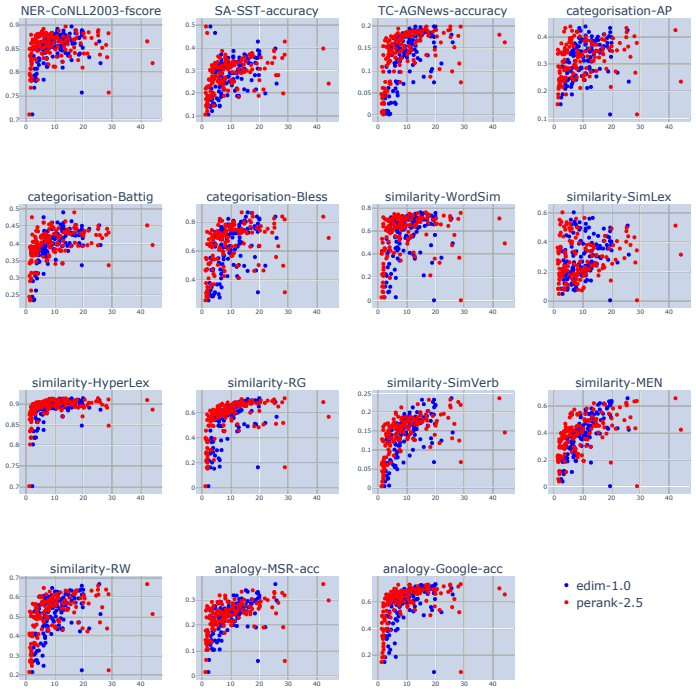


(A) Corrélation de Spearman pour le perank.

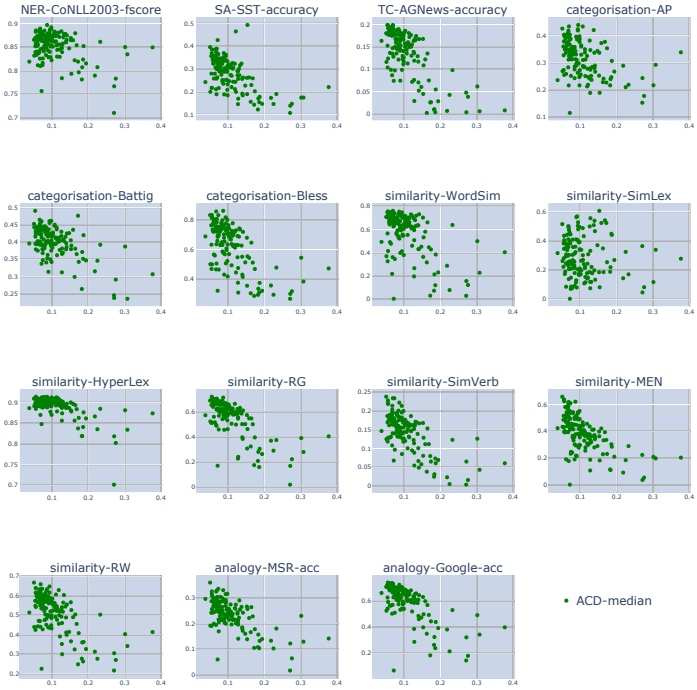


(B) Corrélatioon de Spearman pour l'edim.

FIGURE 2.5 – Matrices de corrélation de perank (2.5a) et edim (2.5b) avec les autres évaluations. Chaque ligne correspond à une valeur de p différente au formatperank$-\{p\}$ et edim$-\{p\}$.



(A) En fonction de l'edim ou du perank.



(B) En fonction de l'ACD médiane

FIGURE 2.6 – Performances des tâches extrinsèques et intrinsèques en fonction des évaluations globales.

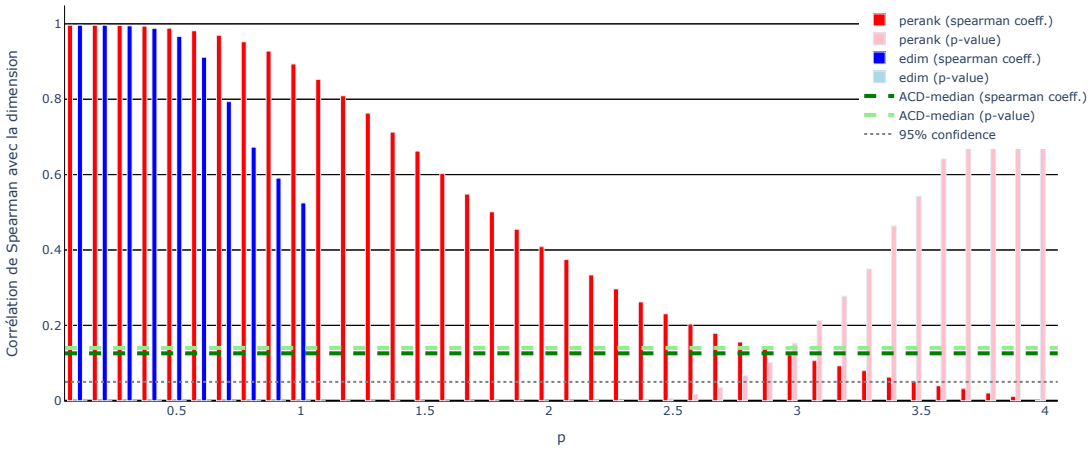


FIGURE 2.7 – Corrélations de Spearman des métriques globales avec la dimension du plongement (en fonction du paramètre p pour edim et perank).

est choisi car on estime que la dépendance entre ces scores doit être linéaire.

— La figure 2.5 donne les coefficients de Spearman pour le perank et l'edim avec les autres évaluations. On choisit Spearman car le lien entre le perank et l'edim ne semble pas linéaire, du fait de l'intervention de fonction puissance ou exponentielle. L'autre raison de la non linéarité est que le gain en performance lorsque l'on obtient une dimension supplémentaire semble dépendre du nombre de dimensions déjà utilisées. Autrement dit, si on a peu de dimensions le gain d'une dimension aura des répercussions positives plus importantes sur les autres évaluations que lorsque l'on a déjà beaucoup de dimension.

— La figure 2.6 représente les nuages de points par évaluation en fonction des évaluations globales.

— La figure 2.7 informe sur la corrélation de Spearman entre les métriques globales et la dimension du plongement. On cherche à savoir si ces métriques capturent autre chose qu'un hyper-paramètre du modèle.

On ne retourne pas les résultats de l'évaluation de la dimensionalité intrinsèque, car elle n'a pas démontré des résultats pertinents dans notre cadre expérimental. On extraits les trois aspects suivants des résultats.

Indépendance des tâches. La figure 2.4 montre des corrélations linéaires entre les tâches. Un grand nombre de tâches intrinsèques sont fortement corrélées (de coefficient $>$ 0.9). Sept tâches semblent plus indépendantes que les autres (de gauche à droite) : NER, SA, TC, categorisation-Battig, similarity-RW, analogy-MSR et analogy-Google. On peut expliquer cela par le fait que les aspects linguistiques évalués par ces tâches ne sont pas couverts par les autres. Par exemple, similarity-RG et similarity-WordSim sont particulièrement dépendantes puisqu'elles évaluent la même chose : la similarité entre des mots courants. À l'opposé, similarity-RW mesure la similarité entre des mots rares. De la même façon les tâches extrinsèques se

détachent du lot. Cependant, elles ont l'air assez dépendantes entre elles. Cela peut être dû au fait que bien qu'elles aient l'air indépendantes sur le principe, ces tâches reposent sur des aspects linguistiques similaires. Une autre hypothèse serait que les tâches sont bien indépendantes, mais que l'algorithme FastText est bon à capturer l'ensemble des aspects utilisés par ces tâches. Par conséquent, si le plongement est bon sur une tâche, il sera aussi bon sur les autres. Pour aller plus loin, il faudrait évaluer plusieurs algorithmes de plongements différents ou rajouter plus de tâches extrinsèques et intrinsèques (comme dans le travail de SCHNABEL et al., 2015; B. WANG et al., 2019).

Sélection Rapide. Un problème commun dans les tâches extrinsèques est l'optimisation d'hyper-paramètres. Cette étape est coûteuse et ignore souvent l'étape d'optimisation des paramètres du plongement. En effet, il est plus simple de se focaliser sur les hyper-paramètres du modèle utilisant le plongement. Les Figures 2.4 et 2.5 montrent des corrélations modérées entre les évaluations globales ou intrinsèques et les tâches extrinsèques. Ce résultat est important car les tâches intrinsèques et globales sont plus rapides à exécuter que les autres. Par conséquent, en considérant un ensemble de plongements entraînés avec différents hyper-paramètres, on pourrait se reposer sur ces tâches pour choisir l'ensemble de paramètres le plus probable de produire de bons résultats. Cela semble être confirmé par la figure 2.6 : les meilleures performances sont obtenues pour les valeurs maximales (minimales pour l'ACD médiane) des métriques globales. Cependant, on doit admettre que les évaluations intrinsèques et globales ne donnent que la tendance vers le meilleur plongement, sans lien de causalité. Si possible, il est préférable de faire l'optimisation des hyper-paramètres en utilisant l'objectif final.

Critère d'optimisation. Pour certaines valeurs de p, le rang effectif étendu et la dimension empirique semblent positivement corrélés avec la plupart des évaluations (voir figure 2.5). Concernant l'ACD médiane, la corrélation semble négative. Cela implique que maximiser perank/edim ou minimiser l'ACD médiane aurait pour impact d'augmenter les scores de l'ensemble des évaluations. Répartir l'information sur le maximum de dimensions du plongement semble guider vers un meilleur plongement. Il est aussi important de noter que le perank et l'edim sont principalement sensibles à l'orthogonalité (voir figure 2.1). Par conséquent, pour intégrer ces résultats, il faudrait soit imposer le perank ou l'edim comme un critère d'optimisation, soit utiliser d'autres critères pour imposer cette orthogonalité (comme le SRIP de BANSAL, X. CHEN et Zhangyang WANG, 2018) ou des paramétrisations orthogonales comme présentés par J. ZHANG, LEI et DHILLON, 2018 ou ARORA et al., 2019. Le plus gros problème est d'appliquer ces critères ou paramétrisations qui rallongent le processus d'optimisation.

Ces observations peuvent être affaiblies en invoquant le fait que les évaluations globales sont en lien direct avec des paramètres du plongement déjà reconnus pour leur lien avec la performance, comme la dimension. Par conséquent, regarder directement la dimension permettrait de répondre aux problèmes du *Critère d'optimisation* et de *Sélection rapide*. Nous tentons de répondre à cet argument via la figure 2.7 qui

montre une forte corrélation de l'edim et du perank avec la dimension du plongement pour des valeurs faibles de p dans les deux cas, mais qui diminue à mesure que p augmente, jusqu'à devenir négligeable. Pour l'ACD médiane, la corrélation tend vers de faibles valeurs sans que nous ayons une p-valeur permettant de conclure plus fortement à cette observation. La corrélation avec la dimension doit idéalement être faible, puisque sinon cela revient à mesurer la dimension du plongement. Or, nous savons déjà que l'augmentation de la dimension entraîne une amélioration des performances. Par conséquent, il faudra plutôt prendre des mesures globales ayant une corrélation faible avec la dimension et étant corrélées aux autres évaluations pour être certains de mesurer des qualités structurelles. Le coefficient étant celui de Spearman, vers $p \geq 2$ pour perank, on observe cela pour perank. L'ACD médiane semble aussi être un bon indicateur sur la structure du plongement.

Une autre faiblesse est le manque d'un lien de causalité entre ces corrélations. Les points *Indépendance des tâches* et *Critère d'optimisation* souffrent de cette remarque. Il apparaît que la non-corrélation n'entraîne pas nécessairement une indépendance, ni que l'optimisation du critère entraînera une amélioration, car la causalité n'est pas établie. Cependant, il apparaît légitime de penser que l'orthogonalité et la maximisation de l'usage de l'espace du plongement peut permettre une amélioration de la tâche, et la corrélation est là pour appuyer cette idée. Nous indiquons des tendances, qu'il apparaît de confirmer par des liens causaux au cas par cas.

2.3.2 Perspectives

En étudiant les évaluations globales, intrinsèques et extrinsèques, nous avons mis en évidence des liens non négligeables entre évaluations globales et extrinsèques. Aussi, nous avons proposé et mis au point des métriques globales permettant la mise en valeur de propriétés structurelles corrélées à la performance. Les évaluations globales, ne portant que sur la structure du plongement et étant indépendante de données linguistiques, peuvent s'intégrer en tant que critères d'optimisation pour tenter de les renforcer et ainsi augmenter la performance des autres tâches.

En tentant d'intégrer ces critères lors de l'apprentissage de plongement FastText, nous avons rencontré plusieurs problèmes. D'abord, certains critères sont très difficiles à intégrer et à optimiser car, ils font intervenir des décompositions de matrices en valeurs singulières (comme le perank où l'edim). Cela augmente le coût de calcul de chaque étape d'optimisation, ce qui ralentit fortement le processus d'optimisation.

Malgré cela, en réussissant à appliquer le critère (après des compromis sur la taille du jeu de données), nous avons observé des effets négatifs (baisse de performance sur la plupart des évaluations) de l'application de ces critères au plongement. Cela peut être dû à la trop forte intégration du critère lié à l'edim ou au perank, effaçant la tâche d'apprentissage principale de FastText.

Forcer l'orthogonalité entre les vecteurs du plongement, avec d'autres critères que perank ou edim, a été étudiée par LUKA, SOULIER et PICARD, 2021, et aboutit au constat que lorsque le critère d'orthogonalité est trop fortement introduit la performance est réduite, mais qu'un léger gain est possible lorsqu'il l'est moins fortement. Il faudrait étudier davantage l'edim, le perank et l'ACD médiane, lorsqu'ils sont intégrés en tant que critère d'optimisation dans les plongements, pour déterminer à partir de quand ils deviennent utiles, s'ils le deviennent.

Les résultats présentés dans ce chapitre ont été publiés à la conférence internationale LREC 2020 (TORREGROSSA, CLAVEAU et al., 2020). L'évaluation des plongements est une problématique que nous retrouverons dans d'autres chapitres. Par exemple, lorsque nous traitons du plongement de hiérarchies et que nous cherchons à évaluer leur qualité (voir chapitre 6). Au-delà d'une mesure de qualité, nous verrons qu'il est possible d'intégrer une métrique globale présenté ici (l'ACD) en tant que critère d'optimisation d'une tâche utilisant des plongements (voir chapitre 4). Enfin, dans l'objectif d'améliorer la performance des tâches à base de plongements, nous présentons dans le chapitre suivant l'intégration de connaissances métiers dans des plongements de mots.

Chapitre 3

Ajout de connaissances a priori pour la détection et classification d'entités

Sommaire

3.1 Un moteur de recherche mono-champ 60
3.2 La tâche de *Slot filling* 61
3.2.1 Rappel de la tâche et adaptation pour PagesJaunes 61
3.2.2 Solutions disponibles 61
3.3 Solution sur mesure 62
3.3.1 Première version 62
3.3.2 Améliorations du modèle 63
3.3.3 Expériences 66
3.4 Conclusion 68

Dans l'objectif d'améliorer la performance des tâches extrinsèques, nous avons essayé dans le chapitre 2 de trouver des métriques structurelles (appelées globales) permettant idéalement, *via* leur optimisation, d'aboutir à l'augmentation des résultats sur des tâches extrinsèques utiles pour l'industrie. Sans réussir à appliquer les résultats positifs des corrélations inter-évaluations, nous avons élaboré d'autres stratégies pour réaliser cet objectif, Solocal étant particulièrement intéressée par des innovations en ce sens, notamment pour son moteur de recherche PagesJaunes [1]. Cela nous a aussi donné l'opportunité de valoriser partiellement nos connaissances et réalisations techniques du chapitre 2.

Ce contexte particulier à notre thèse en lien avec Solocal, nous fournit un accès à des données expertes acquises par la connaissance du métier de PagesJaunes. Cette connaissance est cruciale pour maintenir ou améliorer le fonctionnement des outils de PagesJaunes comme son moteur de recherche. Nous proposons dans ce chapitre d'élaborer davantage la tâche de *slot filling*, proche de celle de NER (voir section 2.2.3), dans ce cadre industriel.

Ce projet était déjà commencé avant le début de la thèse, et nous rappelons dans les sections 3.2.2 et 3.3.1 les travaux déjà accomplis avant notre intervention, pour simplement donner un contexte à nos travaux. C'est pourquoi certains choix technologiques comme FastText, bien que non mis en défaut dans ce projet jusqu'à présent, paraissent désormais désuets (notamment depuis l'apparition des plongements contextuels) mais étaient à l'époque les plus pertinents.

3.1 Un moteur de recherche mono-champ

PagesJaunes est un moteur de recherche français spécialisé dans la recherche de professionnels locaux. Ce moteur est composé de deux champs en entrée (moteur bi-champs) : le *QuiQuoi* et le *Où*. Par exemple dans le champ *QuiQuoi* on saisit *plombier* et dans le champ *Où*, on entre *Rennes*. Un projet a visé à combiner ces deux champs pour former un moteur mono-champ dans lequel les utilisateurs pouvaient formuler leurs requêtes en langage naturel (voir figure 3.1). Cette modification a été motivée par l'essor des automates conversationnels et autres systèmes de dialogues présents dans les mobiles multifonctions (Google Assistant, Siri), la domotique (Google Home, Amazon Echo), mais aussi par la création de Jackbot (l'agent conversationnel de PagesJaunes). En effet, traiter des requêtes en langage naturel est un problème difficile à résoudre, qui permet cependant, d'assurer une présence sur ces nouveaux supports et donc motive le développement d'un tel module.

La compréhension et l'extraction des entités de la requête sont d'abord utiles au développement du moteur mono-champ pour l'identification des deux anciens champs mais aussi à ces agents. Pour évaluer l'usage du mono-champ, PagesJaunes souhaite utiliser sa solution technique actuelle bi-champs en nourrissant les champs par les entités *QuiQuoi* et *Où* extraites par la compréhension. C'est ce que nous allons faire en utilisant le *slot filling*.

1. `https://www.pagesjaunes.fr/`

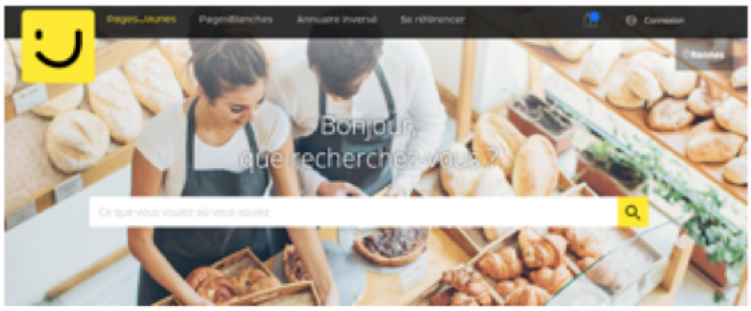


FIGURE 3.1 – Interface du moteur de recherche mono-champ de Pages-Jaunes

TABLEAU 3.1 – Exemple de *slot filling* (mode IOB) sur une requête.

Requête	Je	recherche	un	restaurant	végétarien	à	Paris
Classes	O	O	O	B-QuiQuoi	I-QuiQuoi	O	B-Où

3.2 La tâche de *Slot filling*

3.2.1 Rappel de la tâche et adaptation pour PagesJaunes

La tâche de *Slot filling* est une tâche de classification séquentielle dont le but est de déterminer la classe de chaque mot d'une phrase, ces classes étant vu comme des champs à remplir par le système. Dans le cadre de PagesJaunes, une phrase est la requête de l'utilisateur et les champs sont *QuiQuoi*, *Où*, ou rien (O). Dans le tableau 3.1, on donne un exemple d'une telle classification pour une requête donnée. On utilise généralement le mode de classification IOB, pour distinguer le début du reste de l'entité (l'ensemble des mots contenus dans la case). Cette tâche est résolue communément en supervisant un modèle d'apprentissage automatique avec la séquence de classes à prédire sur un jeu d'entraînement. La tâche est parfois associée avec de la détection d'intentions, qui consiste à trouver l'intention générale de la phrase (tâche de classification globale).

3.2.2 Solutions disponibles

Plusieurs solutions commerciales peuvent être utilisées pour accomplir cette tâche, à savoir Google DialogFlow [2] ou Microsoft Luis [3]. Ces logiciels se reposent sur la définition des types d'entités et des intentions par des exemples, en fournissant pour le premier cas une liste d'entités par champ et pour le second cas la phrase et son intention. Par exemple *Restaurant Végétarien* est une entité du champ *QuiQuoi*.

Dans le tableau 3.2, on reporte la performance de ces solutions commerciales, et on remarque qu'elles ne sont pas satisfaisantes. En plus de cela, des contraintes techniques sur le nombre d'entités maximum (limité à une dizaine de milliers) par

2. `https://dialogflow.com/`
3. `https://www.luis.ai/`

TABLEAU 3.2 – Performance des solutions commerciales sur la tâche de *slot filling* sur des requêtes manuellement étiquetés. La précision est la proportion de requêtes dans lesquelles les champs ont été correctement identifiés.

Algorithme	**Précision**
Google DialogFlow	0.74
Microsoft Luis	0.64

champ ne satisfont pas notre cas d'usage, étant donné que notre système doit pouvoir gérer plusieurs millions d'entités. En plus, ces solutions sont difficilement utilisables en symbiose avec les connaissances PagesJaunes. Pour ces raisons, il fut nécessaire de développer une solution sur mesure.

Heureusement, la tâche de *slot filling* profite d'un état de l'art conséquent. Une approche commune est d'utiliser des connaissances expertes, des patrons et des dictionnaires (A. SUN et al., 2011). Cela a l'avantage d'être simple à mettre en place, mais ces méthodes généralisent plus difficilement à des domaines sur lesquelles les connaissances sont manquantes. Une alternative est d'utiliser des modèles conditionnels comme les champs aléatoires conditionnels (*Conditional Random Field*s, CRFs) (YE-YI WANG et ACERO, 2011 ; LAFFERTY, MCCALLUM et PEREIRA, 2001). Plusieurs travaux ont démontré la supériorité des RNN pour le *slot filling* en combinaison avec des CRF (MA et HOVY, 2016 ; MESNIL et al., 2013).

3.3 Solution sur mesure

Comme évoqué ci-dessus, les solutions commerciales existantes ne suffisent pas aux besoins du projet de Solocal. Par conséquent, il a été décidé de développer une solution sur mesure, se basant sur les travaux de l'état de l'art concernant le *slot filling*.

3.3.1 Première version

Pendant ce projet, plusieurs algorithmes différents de l'état de l'art ont été implémentés : un algorithme basé sur la base de connaissance de PagesJaunes, un CRF et plusieurs modèles utilisant des RNN. Nos expérimentations confirment la supériorité des RNN par rapport aux autres algorithmes.

3.3.1.1 Données d'apprentissage synthétiques

Un des problèmes majeurs rencontré dans les débuts du projet fut le manque de données réelles relatives à la problématique, à savoir, l'absence de requêtes en langage naturel pour la recherche de professionnels locaux. Cependant, PagesJaunes dispose de bases de données exhaustives sur les professionnels, les mots-clefs et les localités, utilisées pour la recherche bi-champs. En s'inspirant de DialogFlow, PagesJaunes a implémenté un générateur de texte basé sur des patrons syntaxiques dans lesquelles les champs sont remplis par les éléments de ces bases.

(Donne moi | C'est quoi) le (téléphone | numéro) de (NOM_PROFESSIONNEL).

FIGURE 3.2 – Exemple d'un patron syntaxique.

Environ cinquante patrons furent définis par des experts et utilisés pour la génération de millions de requêtes types. L'avantage de cette méthode est que l'on connaît les classes ou champs à remplir par construction (on procède à l'envers).

3.3.1.2 Modèle initial

Voici les étapes successives de traitement des requêtes utilisateurs :

— ***Tokenization*** : cela consiste à, dans notre cas, extraire les mots d'une requête et à les normaliser en enlevant la ponctuation, les accents et les majuscules. Ces caractères spéciaux contiennent, en quelque sorte, une part non négligeable de l'information en démarquant les champs *QuiQuoi* et *Où* du reste. Cependant, les requêtes saisies par des utilisateurs réels ne contiennent pas toujours ces caractères, il a donc été décidé de faire sans.

— **Plongement de mots** : FastText (BOJANOWSKI et al., 2017) est utilisé ici pour sa capacité à créer des vecteurs pour des mots inconnus pendant la phase d'entraînement, mais pouvant s'avérer proches de mots connus. Cela est assez courant dans ce cas puisque les utilisateurs commettent parfois des erreurs orthographiques en écrivant leurs requêtes.

— **RNN** : ces réseaux sont utilisés pour leurs bonnes performances sur la tâche de *slot filling*. En fait, leur capacité à capturer les informations contextuelles est bénéfique pour cette tâche.

3.3.1.3 Collecte de données réelles

Un modèle reprenant les éléments de la section 3.3.1.2 a été déployé en production sur 5% du trafic. Cela a permis de collecter des millions de requêtes de vrais utilisateurs. Une part de ces requêtes a été sélectionnée aléatoirement et manuellement étiquetée par des experts. Cela a permis de constituer un ensemble de 6000 requêtes mono-champ. Le tableau 3.3 montre la précision des différentes solutions sur ce jeu de données. Le modèle utilisant le RNN est largement plus efficace que les deux autres solutions. L'implémentation de l'algorithme basé sur les connaissances PagesJaunes est assez naïve et consiste à simplement regarder les expressions correspondantes aux termes de la requête dans une base de données. Au vu des performances assez bonnes qu'il a démontrées, il a été décidé d'exploiter ces connaissances directement dans le modèle avec RNN.

3.3.2 Améliorations du modèle

Après l'étiquetage des données de productions, nous avons recherché des moyens d'améliorer la solution. Deux améliorations ont été apportées : l'ajout d'un vecteur de connaissances aux vecteurs des mots de la requête (voir section 3.3.2.1) et le pseudo-labeling (voir section 3.3.2.2).

TABLEAU 3.3 – Performances de plusieurs algorithmes de *slot filling* évaluées dans le cadre de PagesJaunes (les détails expérimentaux sont dans la section 3.3.3). La précision est la proportion de requêtes dans lesquelles les champs ont été correctement identifiés.

Algorithme	**Précision**
Algorithme basé sur les connaissances PagesJaunes	0.764
CRF	0.778
RNN	**0.840**

3.3.2.1 Vecteurs de connaissances

Nous avons essayé de déterminer les causes des requêtes mal comprises par le système, et pour une certaine quantité, les experts avaient besoin d'utiliser des bases de données pour savoir si les entités de la requête correspondaient à des professionnels ou à des localités. Comme le modèle ne parvenait pas à des performances similaires aux experts, nous avons eu l'intuition que pour augmenter la performance, il fallait inclure ces informations issues de nos bases de données en entrée du réseau. En effet, les informations contenues dans les documents Wikipedia (servant de support au calcul des vecteurs FastText) ne suffisent peut être pas à résoudre la tâche.

***n*-grammes de mots.** Une liste de chaque n-gramme de mots est générée depuis la requête de l'utilisateur, et on retrouve ces n-grammes dans chaque base de données de PagesJaunes s'ils y sont présents. Chacun d'eux est alors associé avec sa fréquence dans les champs *QuiQuoi* et *Où* ainsi qu'à sa présence dans chaque base de données (nom de professionnels, activités ou mots-clés, les localités). La figure 3.3 donne un exemple de décomposition par n-grammes.

Vecteur de Connaissances. Pour chaque mot de la requête on fabrique un vecteur de connaissances en agrégeant toutes les informations possédées sur lui. Pour chaque valeur $n = 1, 2, 3, 4+$ de n-grammes, on possède un vecteur composé de la fréquence des champs *QuiQuoi* et *Où* ainsi que sa présence dans les différentes bases. Il est important de noter que chaque mot possède l'information des n-grammes auxquels il participe. Ces vecteurs sont agrégés pour former le vecteur de connaissances. Chaque vecteur de connaissances est passé dans une couche linéaire avant d'être fourni en entrée du RNN, en charge d'incorporer les informations dans un espace continu. Le vecteur résultant de cette dernière opération est appelée *knowledge embedding* ou *plongement de connaissances* et il est concaténé au vecteur FastText du mot.

3.3.2.2 Pseudo-Labeling

Une autre limite du modèle se situe dans les données d'entraînement qui sont jusqu'ici synthétiques et ne reflètent pas identiquement la distribution des requêtes du monde réel. Les requêtes manuellement étiquetées ne sont pas en nombre suffisant pour être utilisées à l'entraînement et sont donc exclusivement réservées à la

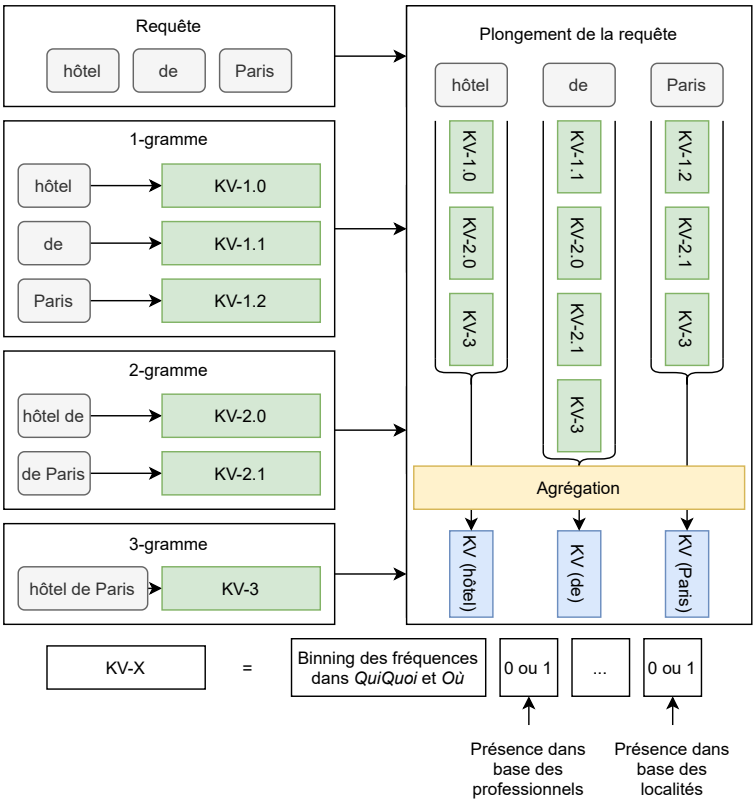


FIGURE 3.3 – Décomposition d'une requête en n-grammes et création des vecteurs de connaissances à partir des bases de données de Solocal.

.

TABLEAU 3.4 – Distribution des hyper-paramètres.

Paramètres	Echantillonnage	Description
batch_size	choice([128,256,512])	Taille des batches
learning_rate	log_uniform(10^{-5},10^{-1})	Facteur d'apprentissage
knowledge_emb_size	int.uniform(32, 512)	Dimension du vecteur de connaissance
concat_dropout	float.uniform(0.01, 0.4)	Dropout appliqué à la concaténation
rnn_layers	choice([1, 2])	Nombre de couche RNN (bi-GRU)
rnn_size	int.uniform(32, 1024)	Taille des couches RNN (bi-GRU)
rnn_dropout	float.uniform(0.1, 0.4)	Dropout appliqué à la sortie RNN
dense_size	int.uniform(32,1024)	Taille de la couche linéaire en sortie du RNN
dense_dropout	float.uniform(0.1, 0.4)	Dropout appliqué à la couche de sortie du RNN
pseudo_label_ratio	choice([0, 0.1, 0.2, 0.3, 0.4, 0.5])	Taux de données pseudo-labelisés par batch
use_knowledge_vector	choice([True, False])	Ajout du Vecteur de connaissances

validation et au test. Pour inclure des données réelles, on fait intervenir du pseudo-labeling (GRANDVALET et BENGIO, 2005), en faisant étiqueter les données de production (contenant des millions de requêtes) par notre modèle entraîné sur les patrons syntaxiques. Cela permet d'inclure les données réelles lors de l'entraînement du modèle.

3.3.3 Expériences

Le modèle est implémenté avec la libraire PyTorch (PASZKE et al., 2019) et on utilise l'optimisation Adam. Les requêtes étiquetées manuellement sont conservées pour la validation et le test en 50/50. On entraîne plusieurs modèles avec des hyper-paramètres différents tirés aléatoirement des distributions du tableau 3.4, de façon à explorer l'espace des hyper-paramètres de façon aléatoire : un processus connu sous le nom de *Random Search* (BERGSTRA et BENGIO, 2012) et renommé pour être efficace à trouver rapidement les hyper-paramètres optimaux.

Plus de 300 modèles avec des hyper-paramètres différents furent entraînés et évalués, le tableau 3.5 répertorie la précision des meilleurs modèles pour chaque amélioration proposée : Pseudo-Labelling (PL) et Knowledge Vector (KV).

Comme attendu, le modèle de base n'exploitant que le RNN est le moins performant. Le modèle manque d'informations sur la distribution réelle des requêtes ainsi que sur les entités entrées dans les champs recherchés. Chaque amélioration prise individuellement améliore les résultats sur notre test, et la combinaison des deux améliorations (KV+PL) retourne les meilleurs résultats. Pour renforcer cette observation nous retournons, en figure 3.4, la distribution (sous forme de boîte à moustache) de la précision parmi les 300 modèles évalués par type. On voit clairement que plus de la moitié des modèles faisant intervenir KV+PL ont une meilleure performance que les autres solutions.

Nous pensons que ce mode d'évaluation recherchant la distribution de la métrique d'intérêt (ici la précision) dans l'ensemble des modèles disponibles est pertinent car il permet de mieux situer l'apport d'une méthode. En effet, un modèle peut s'avérer très performant sur certaines données ou avec des hyper-paramètres spécifiques, mais peut aussi être sensible à la moindre modification de ces derniers. Ici, nous montrons que nos propositions ont une distribution de précision robuste à la

TABLEAU 3.5 – Résultats obtenus avec le RNN, avec le vecteur de connaissances (KV) et/ou avec le pseudo-labeling (PL). La précision est calculée sur les patrons syntaxiques, l'ensemble de validation et test.

	Précision		
Algorithme	**Patrons syntaxiques**	**Validation**	**Test**
RNN	0.936	0.824	0.840
RNN+PL	0.928	0.856	0.867
RNN+KV	**0.939**	0.857	0.866
RNN+KV+PL	0.921	**0.875**	**0.886**

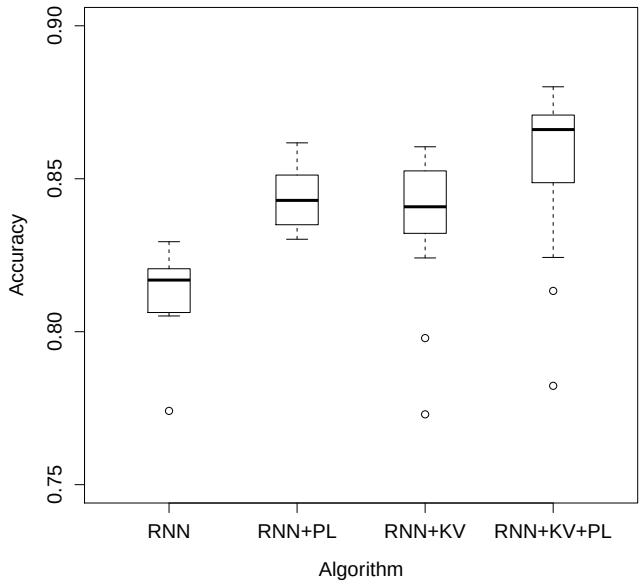


FIGURE 3.4 – Performances des modèles sur l'ensemble de validation en fonction des améliorations proposées.

plupart des perturbations de leurs hyper-paramètres et sont donc généralisables à d'autres cas. Nous ajoutons également que la performance maximale observée n'est pas due à une combinaison unique d'hyper-paramètres, mais bien consistante.

Il est aussi important de remarquer l'influence attendue du pseudo-labeling sur les résultats des modèles. En effet, les modèles avec le pseudo-labeling sont moins performants sur les patrons syntaxiques, en comparaison avec des modèles identiques (RNN avec RNN+PL et RNN+KV avec RNN+KV+PL), mais plus performants sur la validation et le test. Cela s'explique par le fait que les patrons syntaxiques diffèrent de la distribution réelle des requêtes. Le pseudo-labeling a donc, dans notre cas, l'effet d'augmenter la précision sur les ensembles de validation et de test.

3.4 Conclusion

L'ajout de connaissances dans la représentation vectorielle d'un mot a augmenté significativement la performance de la tâche de *slot filling*. Ce résultat souligne l'importance des connaissances *a priori*. Nous proposons ici une manière de les intégrer dans un modèle de classification, et nous observons qu'elles contribuent à l'amélioration du modèle.

Cela provoque l'apparition de nouveaux besoins potentiels comme la nécessité d'enrichir ses bases de données ou d'améliorer la façon de représenter ces bases et ces connaissances de façon continue. Pour cela, nous verrons dans les chapitres suivants des solutions, en particulier du côté de la représentation des connaissances dans des espaces hyperboliques. Plus directement, le chapitre suivant traitera de ces deux aspects : l'enrichissement des connaissances et les représentations euclidiennes de bases de connaissances.

Ce travail sur le *slot filling* reste limité aux plongements statiques, et il faudrait étudier l'effet de l'usage des plongements contextuels comme BERT (DEVLIN et al., 2019). En particulier sur l'usage des n-grammes de mots, dans notre solution actuelle, nous créons artificiellement les vecteurs de connaissances qui étendent la représentation du mot à ses mots alentours. Ceci pourrait être directement pris en charge par des plongements contextuels, à condition de réussir à intégrer les connaissances de PagesJaunes dans ces vecteurs (via une tâche de pré-entraînement, par exemple).

Les travaux présentés dans ce chapitre ont été publiés à la conférence internationale ICONIP 2019 (TORREGROSSA, KOOLI et al., 2019). Concernant les contributions de la thèse sur ce travail, elles se situent plutôt sur la partie expérimentation et industrialisation du modèle. La construction du *plongement de connaissances* a été mise au point par Robin Allesiardo.

Chapitre 4

Application d'une métrique d'évaluation à l'alignement de bases de connaissances par plongement

Sommaire

4.1 Contexte et détails techniques 70
4.1.1 Plongement de Graphes de Connaissances 71
4.1.2 Tâche d'alignement 72
4.1.3 Historique des méthodes d'alignement 72
4.2 AlignD : méthode d'alignement globale par plongements 76
4.2.1 Mesurer l'alignement 76
4.2.2 Objectif total 79
4.2.3 Génération automatique de *seed* 79
4.3 Expériences 80
4.3.1 Protocole 80
4.3.2 *Seed* réelle 81
4.3.3 Taille de la *seed* et performance 82
4.3.4 *Seed* générée automatiquement 84
4.4 Conclusion 85

Nos travaux sur les métriques globales ont introduit la notion d'alignement entre des plongements (voir section 2.2.1.3). Le chapitre 3 a montré l'intérêt d'avoir des bases contenant des connaissances dans l'objectif d'améliorer la performance d'une tâche comme celle du *slot filling*. Posséder des bases avec des connaissances adaptées à la tâche est complexe car cela requiert, entre autres, des acteurs experts ou une expérience du terrain. Beaucoup de ressources en ligne gratuites (présentes dans les *Linked Open Data*) pourraient être utilisées pour collecter ou enrichir les bases de données d'acteurs privés et enfin accélérer leurs innovations, ou les affiner pour les rendre plus efficaces.

Pour procéder à l'enrichissement des données privées par des données publiques, il faut procéder à l'alignement entre les entités de chaque membre, c'est-à-dire à l'identification des entrées analogues pour pouvoir faire correspondre les informations. Dans la suite, nous considérons les bases de connaissances, représentées comme des graphes de connaissances (*Knowledge Graph*s, KGs) utilisant le format de représentation *Resource Description Framework* (RDF), puisqu'il s'agit du format hégémonique pour les *Linked Open Data*.

Nous présentons dans ce chapitre un processus d'alignement reposant sur le plongement des KGs dans des espaces continus, à l'instar du plongement de mots, et sur la métrique de mesure d'alignement présentée précédemment. Les deux se combinant pour proposer une manière non supervisée efficace d'alignement entre KGs.

4.1 Contexte et détails techniques

Le formalisme RDF permet de formaliser la description de ressources composées de ce qu'on appellera des *entités*. Ces entités possèdent des attributs, étant des propriétés propres ou des relations avec d'autres entités décrites. L'objectif plus général du RDF est d'offrir un format générique de description aux machines pour leur faciliter la compréhension des interactions entre les ressources, et ainsi, automatiser et améliorer des processus comme la recherche d'information, le catalogage ou **le partage et l'échange de connaissances** [1].

En effet, le format utilise des *Uniform Resource Identifier* (URI) pour cibler les ressources, ce qui autorise un ensemble de ressources de pointer les éléments d'un autre et ainsi unifier la connaissance des différents systèmes disponibles (DBpedia, PubMed, Freebase...). D'un point de vue pratique, lorsque deux systèmes de données fusionnent (après l'acquisition d'une entreprise par une autre), le format RDF est particulièrement efficace car il facilite l'extension des connaissances d'un système à l'autre et permet de faire coopérer les informations, pourvu qu'on connaisse les entités à aligner (SHVAIKO et EUZENAT, 2013), c'est-à-dire, qu'on puisse identifier les entités référant au même concept/objet réel. Une grande variété de tâches peut alors bénéficier de l'extension et de l'unification des connaissances, notamment

1. https://www.w3.org/TR/PR-rdf-syntax/Overview.html

la découverte de connaissances, les systèmes de questions-réponses ou de raisonnement, et la recherche sémantique (BRILL et al., 2001 ; GUHA, MCCOOL et E. MILLER, 2003).

Les systèmes de ressources étant nombreux et variés, faisant souvent appel à des sémantiques différentes pour décrire une même ressource (SHVAIKO et EUZENAT, 2013, problème nommé *hétérogénéité sémantique*), cela rend l'alignement difficile à résoudre et requiert l'intervention humaine ou celle d'un système automatisé performant. Ces systèmes automatiques disposent de plusieurs méthodes pour procéder à cette tâche comme l'exploitation des éléments symbolique des entités (le nom, le type ou les attributs de l'entité). Plus récemment, les plongements de KGs ont été utilisés pour tenter de résoudre cette tâche avec succès, en apprenant la représentation continue des entités et des relations et en utilisant la similarité cosinus (voir Eq 2.5) pour identifier les vecteurs alignés (Z. SUN, HU et LI, 2017 ; ZHU et al., 2017 ; M. CHEN et al., 2017 ; Z. SUN, HU, Q. ZHANG et al., 2018).

Dans la suite, nous expliquons les méthodes existantes pour produire des plongements de KGs (section 4.1.1), puis nous présentons la tâche d'alignement en elle-même (section 4.1.2). Nous proposons ensuite un historique des méthodes d'alignement par plongement et leurs principales étapes (section 4.1.3), avant de présenter notre nouveau critère d'alignment (section 4.2) et son évaluation (section 4.3).

4.1.1 Plongement de Graphes de Connaissances

Le format RDF présente des listes de relations sur les entités sous forme de triplets (h, r, t), indiquant une entité sujet h (pour *head*), une relation r et un objet t (pour *tail*). Les plongements de KGs utilisent ces triplets pour calculer les représentations vectorielles pour chaque membre (entités et relations) via la minimisation d'un objectif. L'objet de la relation peut être une entité ou la valeur d'un attribut, mais dans le cas de l'alignement, on considère généralement seulement les objets qui sont des entités (bien que certaines méthodes ne le fassent pas). Pour notre système, on ne tient compte aussi que des objets qui sont des entités.

Ici nous utilisons comme base la méthode TransE, de BORDES et al., 2013, commun à la plupart des méthodes d'alignement reposant sur les plongements de KGs. Cette méthode associe à chaque élément un vecteur et transcrit la relation sous forme géométrique, considérant un triplet (h, r, t) et leur vecteur respectif v_h, v_r et v_t, on souhaite avoir :

$$v_h + v_r = v_t. \tag{4.1}$$

Pour obtenir cela, on minimise la distance L2 entre $v_h + v_r$ et v_t, tout en maximisant la distance de $v_h + v_r$ ou v_t avec des éléments négatifs. L'objectif s'écrit :

$$O_e = \sum_{\tau \in T^+} [f(\tau) - \lambda_1]_+ + \mu_1 \sum_{\tau' \in T^-} [\lambda_2 - f(\tau')]_+ \ , \tag{4.2}$$

où $f(\cdot)$ est une fonction sur les triplets : $f((h,r,t)) = \|v_h + v_r - v_t\|_2^2$; $[x]_+ = \max(x,0)$; et λ_1, λ_2 et μ_1 sont des hyper-paramètres du modèle pour émarger la minimisation et équilibrer l'objectif positif avec l'objectif négatif. T^+ et T^- correspondent respectivement aux triplets positifs et négatifs. Les triplets négatifs sont obtenus en remplaçant le sujet ou l'objet par une entité sans lien avec, respectivement, l'objet ou le sujet.

La minimisation de O_e (Eq. 4.2) permet d'incorporer les connaissances du KG dans les vecteurs et ainsi obtenir une représentation continue des entités et des relations le composant.

4.1.2 Tâche d'alignement

On considère deux KGs, KG_1 et KG_2, et leurs entités respectives X_1 et X_2. L'objectif de la tâche d'alignement est de déterminer un ensemble

$$A = \{(x,y) \in X_1 \times X_2 | x \sim y\} \tag{4.3}$$

où $\sim$ dénote l'équivalence des entités x et y (Z. SUN, HU, Q. ZHANG et al., 2018). On connaît en général un sous-ensemble $A' \subseteq A$ appelé *seed* (amorce) qui agit comme un ensemble d'entraînement pour calibrer l'alignement et s'assurer que la distance entre les entités reflètent aussi le lien sémantique entre KG_1 et KG_2. Le reste des entités $A \setminus A'$ est appelé *ref* et sert à l'évaluation de l'alignement.

Plus spécifiquement, la *seed* donne la correspondance entre KG_1 et KG_2 pour un petit nombre d'entités (environ 30%), et doit fournir l'information a priori nécessaire pour assurer la proximité des entités équivalentes. La *seed* est aujourd'hui le goulot d'étranglement des techniques d'alignement par plongements, puisque la précision de l'alignement final dépend substantiellement de sa taille et de sa qualité. Dans un cas réel, l'intervention d'un expert humain sera donc nécessaire pour obtenir une *seed* avec de telles propriétés, ce qui entraîne des coûts supplémentaires et ruine le caractère automatisé de l'approche.

Du côté de l'évaluation, la *ref* est l'ensemble des entités pour lesquelles on recherche une correspondance. En principe, aucune hypothèse ne doit être supposée sur ces entités mais, en pratique, les méthodes actuelles comme BootEA, de Z. SUN, HU, Q. ZHANG et al., 2018, ou RDGCN, de WU et al., 2019, supposent respectivement l'uniformité des distances ou font de l'échantillonnage négatif (*Negative Sampling*, NS) sur les entités de la *ref*. Notre méthode ne fait aucune hypothèse sur les entités de la *ref*, car nous considérons que supposer l'uniformité des distances peut affecter les entités réellement alignées ou proches de l'être.

4.1.3 Historique des méthodes d'alignement

Les premières solutions automatiques d'alignement utilisaient les informations symboliques sur les entités présentes dans les KGs (e.g. leurs attributs). Ces approches se heurtent à l'hétérogénéité sémantique due à des choix sémantiques ou

terminologiques différents ou à l'ambiguïté de description de certaines entités. L'apport de structures externes (liens Wikipédia ou WordNet (G. A. MILLER, 1995)) est une solution utilisée pour gérer ce problème d'hétérogénéité (SHVAIKO et EUZENAT, 2013). Les hyperonymes, synonymes ou hyponymes d'une entité dans une taxonomie pouvant servir de base pour le calcul de similarité entre les entités des deux KGs, comme peut le faire l'algorithme *Anchor-flood* de SEDDIQUI et AONO, 2009.

Les méthodes par plongement sont, en principe, capable de directement gérer l'hétérogénéité sémantique en faisant abstraction des conventions logiques et sémantiques de chaque KG (Z. SUN, HU, Q. ZHANG et al., 2018). Pour TransE par exemple, ce sont seulement les liens entre les entités et relations qui comptent pour le calcul des représentations, plutôt que leurs étiquettes. Les plongements permettent aussi de fabriquer des caractéristiques indépendantes de l'intervention humaine, et par conséquent, en construire des plus adaptées au problème à résoudre, contrairement aux méthodes symboliques (WU et al., 2019). Le passage à l'échelle est aussi simple avec les méthodes par plongement.

Pour ces raisons, plusieurs solutions d'alignements utilisant des plongements TransE ou des Réseau Convolutif de Graphe (*Graph Convolutional Network*, GCN) ont été proposé. Nous ne détaillons pas ici les GCN, car ils ne sont pas utilisés dans le reste de la thèse ni dans ce chapitre autrement que pour distinguer les types de méthodes (voir les travaux de KIPF et WELLING, 2017 pour davantage détails sur la méthode GCN).

Dans la première famille exploitant la méthode TransE, MTransE (M. CHEN et al., 2017) établit le plongement de chaque KG avec TransE, puis propose différentes transformations pour faire l'alignement entre les plongements. IPTransE (ZHU et al., 2017) apprend aussi un plongement de chaque KG, mais utilise PTransE (Y. LIN et al., 2015) (un algorithme similaire à TransE) et intègre trois modules (traduction, transformations linéaires et partage de paramètres) pour aligner les KGs. JAPE (Z. SUN, HU et LI, 2017) fait la jointure des attributs entre les entités des deux KGs pour augmenter la précision de l'alignement. BootEA (Z. SUN, HU, Q. ZHANG et al., 2018) rajoute un critère explicite minimisant la distance entre les entités alignées de la *seed*. BootEA infère progressivement les liens entre les entités de la *ref*, pour les utiliser lors de l'entraînement et faire augmenter graduellement la prédiction de l'alignement, similaire à de l'apprentissage semi-supervisé.

La deuxième famille exploite les GCN plutôt que la méthode TransE (CAO et al., 2019 ; Zhichun WANG et al., 2018 ; WU et al., 2019). RDGCN (WU et al., 2019) est la technique la plus performante parmi celles que nous connaissons et reposant sur les GCN. Cette technique utilise des procédés similaires au *parameter swapping* (voir section 4.1.3.1) en construisant un graphe dual à celui portant sur les entités (nommé alors graphe primal). Le graphe dual est un graphe où les nœuds sont les relations du graphe primal, et les liens entre relations sont pondérés par la quantité d'entités communes. Le graphe dual et primal sont intriqués pour fournir une représentation des entités censée mieux contenir l'information des relations du graphe primal.

Ensuite, on exploite la *seed* pour calculer l'objectif, qui consiste à rapprocher directement des vecteurs des entités alignées et l'écartement aléatoire des entités non alignées, rendant l'approche vulnérable aux alignements incorrects. Enfin, il faut aussi noter que RDGCN utilise des vecteurs pré-entraînés GloVe [2] de taille 300 pour son initialisation. Nous proposons de voir l'incidence de cette initialisation.

Globalement, la principale faille des systèmes actuels est le fait qu'ils reposent lourdement sur la *seed*, faisant de la qualité et la taille de cette dernière des déterminants majeurs sur la précision de l'alignement. Dans les sections suivantes, nous détaillons les principes fondamentaux permettant le fonctionnement de ces méthodes (notamment ceux de BootEA) et que nous réutilisons pour notre modèle AlignD.

4.1.3.1 Parameter Swapping

Cette étape, présentée par Z. SUN, HU, Q. ZHANG et al., 2018, consiste à augmenter le nombre de triplets positifs en se servant de la *seed* A'. Les entités présentes dans KG_1 sont injectées dans des triplets où les objets ou sujets sont des entités de KG_2. Cela se fait en utilisant la connaissance a priori d'alignement d'entités présentes dans A'. En créant des triplets mixtes, on favorise le partage de paramètres entre les deux KGs, via leurs vecteurs. De façon plus formelle, pour $(x, y) \in A'$, on remplacera un triplet (x, r, t) (resp. (h, r, x)) de KG_1 par (y, r, t) (resp. (h, r, y)), et inversement pour les triplets de KG_2. Ces triplets nouvellement créés intègrent l'ensemble des triplets naturels aux KGs et sont alors utilisés pour l'entraînement (en tirant des exemples négatifs adaptés).

4.1.3.2 Objectif d'alignement

Pour pouvoir imposer l'alignement, l'objectif de plongement O_e (voir Eq 4.2) est combiné avec un objectif d'alignement O_a mesurant le désaccord (à minimiser) entre les entités à aligner. La forme de O_a dépend de la méthode (M. CHEN et al., 2017 ; Y. LIN et al., 2015 ; ZHU et al., 2017 ; Z. SUN, HU et LI, 2017 ; Z. SUN, HU, Q. ZHANG et al., 2018) mais toutes les propositions d'objectif reposent directement sur la *seed* et certaines font des hypothèses sur les entités de la *ref*. Pour les entités de la *seed*, la distance entre les paires doit être minimisée, au travers de la similarité cosinus par exemple. Pour les entités de la *ref*, on cherche à uniformément minimiser (ou maximiser selon la méthode) la distance entre elles, via le NS (WU et al., 2019) ou une matrice de similarité (Z. SUN, HU, Q. ZHANG et al., 2018). La Figure 4.1 représente le calcul de l'objectif O_a en utilisant la similarité plutôt que la distance comme cela est fait en pratique. Procéder ainsi pose deux problèmes résolus par notre approche :

— la distance entre paire d'entités est minimisée par paquets et non globalement rendant le processus sensible aux erreurs dans la *seed*,

— l'hypothèse sur la *ref* paraît très forte, car des entités alignées dans la *ref* sont considérées de la même façon que des entités non alignées, pénalisant l'incorporation sémantique de ces entités.

2. https://nlp.stanford.edu/projects/glove/

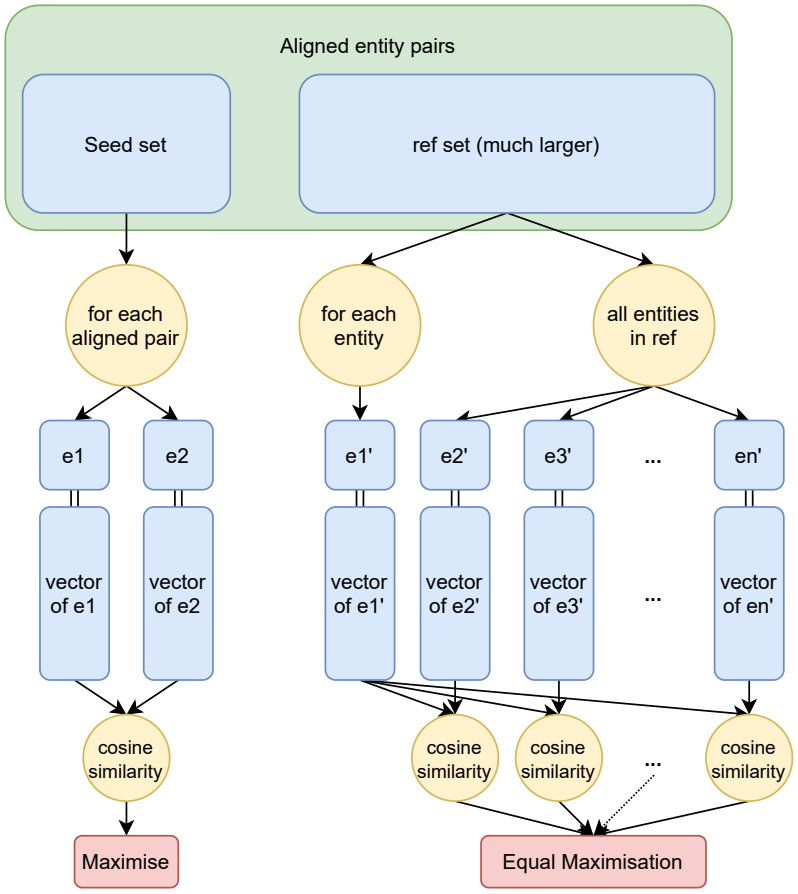


FIGURE 4.1 – Calcul de l'objectif d'alignement des méthodes actuelles.

Nous proposons de traiter de ces deux points.

4.1.3.3 Alignement Itératif

Aussi connu sous le nom d'*alignement par bootstrapping*, ce processus consiste à réaliser, de façon itérative, l'alignement des entités de la *ref*, en prenant à chaque étape l'alignement le plus probable. On produit un alignement 1 pour 1 (Z. SUN, HU, Q. ZHANG et al., 2018), et on utilise ce dernier pour améliorer la prédiction dans l'itération suivante. Cela se fait en intégrant ces nouvelles entités alignées dans le *parameter swapping* et l'objectif global $O_e + O_a$.

4.2 AlignD : méthode d'alignement globale par plongements

Pour résoudre les limites soulevées dans la section précédente, on propose un nouvel objectif d'alignement ne faisant pas d'hypothèse sur la *ref* et qui gère l'alignement de façon plus globale. L'idée est venue de la métrique d'évaluation intrinsèque QVEC, présenté dans la section 2.2.1.3, et proposée par TSVETKOV et al., 2015. Comme on peut mesurer le degré d'alignement entre des plongements, on peut réutiliser cette mesure en l'intégrant directement comme objectif et ainsi optimiser l'alignement de leurs dimensions. L'alignement ne se fait donc plus au niveau atomique des paires d'entités, mais sur le niveau global en forçant les dimensions des plongements des deux KGs à encoder le même type d'information. Dans le premier cas, on s'assure de la proximité spatiale entre les entités alignées, dans le second cas on s'assure que les dimensions sont alignées, et les entités ne deviennent plus qu'un moyen pour calculer le degré de correspondance entre les dimensions. Les deux plongements finiront par avoir des dimensions homogènes et cela nous permettra d'unifier les informations des deux KGs, et d'identifier les entités alignées dans la *ref*.

L'alignement des entités devient un résultat indirect provoqué par l'alignement des dimensions. Comme on travaille sur le niveau global de représentation des informations au travers des dimensions, les hypothèses de distribution des distances pour les entités ne sont plus utilisées. Aussi, le fait d'élever l'alignement au niveau global limite l'impact d'entités alignées par erreur, puisque ces erreurs seront noyées dans le flot d'entités correctement alignées lors du calcul de corrélation dimensionnelle. Ceci est à l'opposé des méthodes antérieures, considérant individuellement et directement l'alignement des entités de la *seed*. Nous appelons cette méthode AlignD pour Alignement Dimensionnel, car c'est le résultat de l'optimisation de l'objectif introduit ci-après.

4.2.1 Mesurer l'alignement

QVEC est une mesure permettant de juger la concordance des dimensions d'un plongement de mots avec les dimensions d'un plongement issu de connaissances linguistiques humaines. Comme son rôle n'est pas d'assurer l'alignement entre deux

plongements, il faut l'adapter pour qu'elle rende compte de cela. En particulier, QVEC va choisir les paires de dimensions les plus corrélées, ce qui provoque l'association de dimensions d'indices différents. Or nous voulons pouvoir faire le produit scalaire entre les deux plongements, par conséquent la dimension i du plongement de KG_1 doit correspondre (être la plus corrélée) à la dimension i de KG_2, ce que nous appelons *alignement des dimensions*. De plus, il faut transformer QVEC pour qu'il devienne un critère d'apprentissage.

Notons $E_1(A')$ le plongement de KG_1 sur les entités de A', $E_2(A')$ celui de KG_2 sur A' de façon respective. Cela signifie que pour tout $i \in [\![1, |A'|]\!]$, la i-ème ligne de $E_1(A')$ est le vecteur d'une entité x, la i-ème ligne de $E_2(A')$ celui d'une entité y, et qu'on $(x, y) \in A'$. On note également la j-ème colonne (ou dimension) de $E_1(A')$ (resp. $E_2(A')$) par $E_1(A')_{*,j}$ (resp. $E_2(A')_{*,j}$), et $r(X, Y)$ le coefficient de corrélation de Pearson entre X et Y deux vecteurs de même dimension. Il est important de noter que là où QVEC utilisait les mots pour calculer le coefficient de Pearson, nous utilisons ici la *seed* A' pour accorder les valeurs des dimensions des entités équivalentes de KG_1 et KG_2.

L'idée est de mesurer la corrélation entre toutes les paires de dimensions $(i, j) \in [\![1, D]\!]$, où D est la dimension des plongements, afin de connaître l'alignement des dimensions. On peut écrire une matrice de corrélation

$$A_D = (r(E_1(S)_{*,i}, E_2(S)_{*,j}))_{(i,j)\in[\![1,D]\!]}. \tag{4.4}$$

et on propose une représentation du processus sur la Figure 4.2.

Par la suite, on considère que l'alignement des dimensions est réalisé lorsque les deux éléments suivants sont vérifiés : la dimension i de $E_1(A')$ est le plus corrélée à la dimension i de $E_2(A')$ (consistance du produit scalaire), et cette corrélation doit être prépondérante par rapport à des dimensions distinctes ($j \neq i$). En d'autres termes, les valeurs non diagonales de A_D doivent être petites par rapport aux valeurs diagonales, ce qui fait que la distance de A_D à la matrice identité I_D est une bonne mesure de l'alignement des dimensions. Plus cette distance est faible, plus l'alignement est fort et vice-versa. L'objectif s'écrit donc

$$O_d = \|A_D - I_D\|_2 \tag{4.5}$$

où $\|\cdot\|_2$ est la norme L2. Minimiser O_d force l'alignement des dimensions. D'autres normes pourraient être utilisées, nous faisons ce choix par défaut.

O_d traite les coordonnées des vecteurs pour chaque couple simultanément au travers du coefficient de corrélation de Pearson, ce qui force les représentations à être globalement consistantes. Le but est double : (1) cela assure que les dimensions alignées soient proportionnelles et encourage l'encodage d'informations différentes sur des dimensions non alignées ; (2) cela réduit la sensibilité aux erreurs quand elles sont minoritaires.

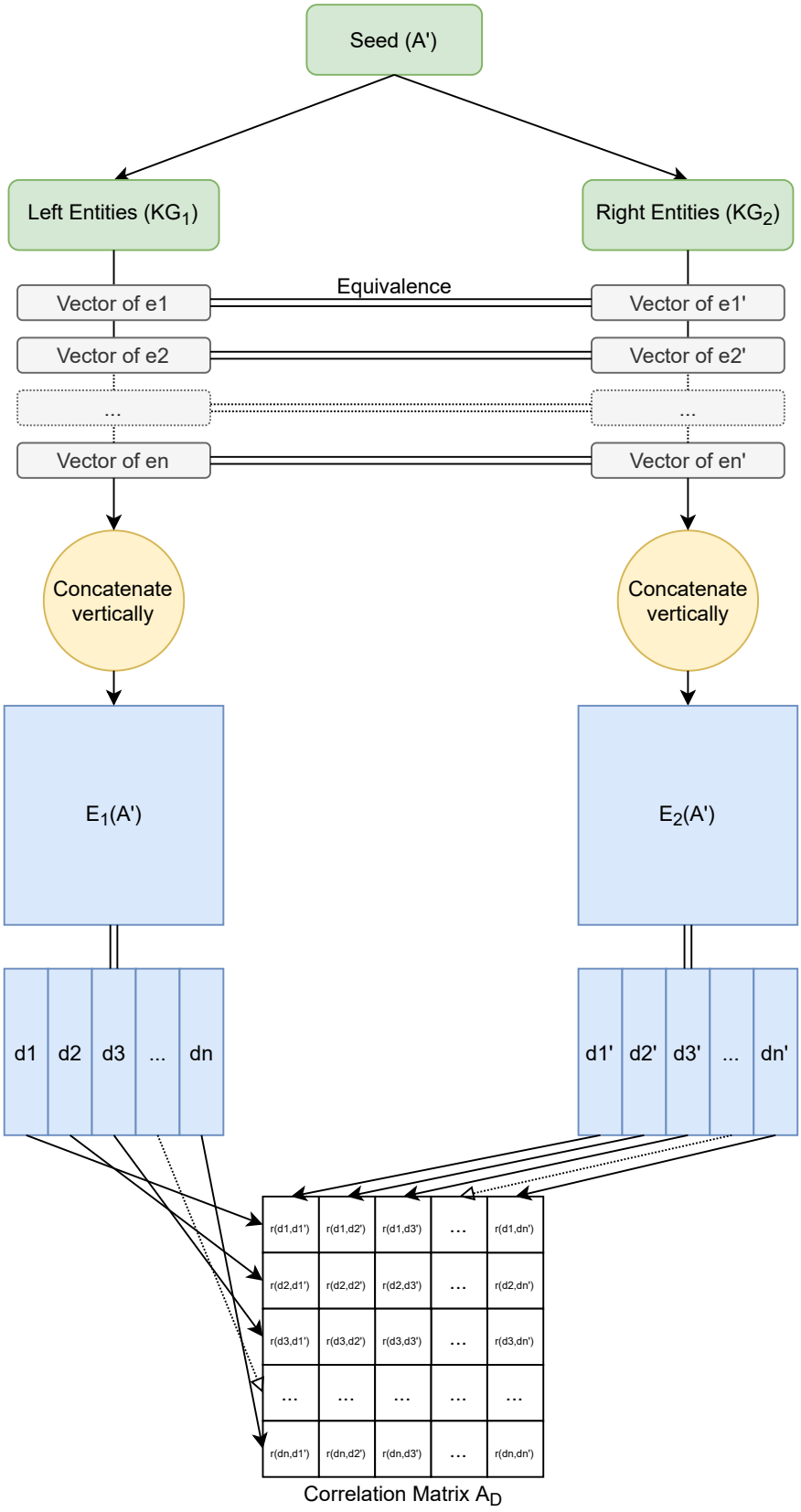


FIGURE 4.2 – Processus de calcul de la matrice de corrélation A_D à partir d'une *seed* A'.

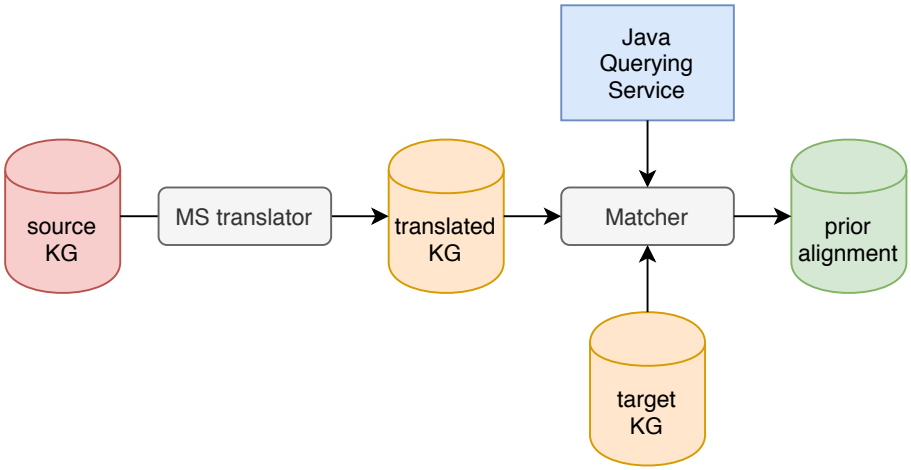


FIGURE 4.3 – Processus de génération automatique de *seed*.

4.2.2 Objectif total

On propose d'utiliser ce nouveau critère à la place des autres critères d'alignement, et de reprendre la philosophie de BootEA (Z. SUN, HU, Q. ZHANG et al., 2018). Le plongement des deux KGs se fait conjointement par l'objectif global

$$O = O_e + \alpha O_d, \tag{4.6}$$

où O_e est la fonction d'objectif de TransE de l'Eq 4.2, O_d est la fonction d'alignement des dimensions de l'Eq 4.5 et α est un coefficient d'équilibre entre les deux objectifs. La combinaison de cet objectif O avec le *parameter swapping* et l'alignement itératif aboutit à notre méthode d'alignement AlignD.

Il faut noter que O exploite l'ensemble des relations via O_e et ne fait aucune hypothèse sur les entités non présentes dans la *seed*, puisque O_d n'utilise que A' pour calculer l'alignement.

Il est connu que la *seed* a une influence considérable sur la précision de l'alignement, en particulier, lorsque la taille de la *seed* augmente, la précision suit. Cela est particulièrement vrai pour AlignD qui ne suppose rien des entités non présentes dans la *seed*, à l'inverse des autres méthodes. C'est pourquoi, augmenter la taille de la *seed* n'aura aucun effet sur les autres entités et ne peut être que bénéfique. Cela suppose que la *seed* peut être étendue automatiquement par les autres méthodes d'alignements. Grâce à la vision globale apportée par le critère d'alignement de dimension, AlignD est prédestiné à tirer profit d'une grande *seed* pas forcément très précise. Les autres méthodes comme RDGCN, ou BootEA seront, quant à elles, moins capables et plus affectées par des *seed* peu précises, puisqu'elles forcent les entités alignées à se correspondre parfaitement et supposent l'uniformité ailleurs.

4.2.3 Génération automatique de *seed*

Aligner des KGs avec des méthodes de plongements est difficile sans l'apport d'une *seed* précise et grande. En poussant l'idée de l'extension automatique de *seed*

TABLEAU 4.1 – Quantité d'éléments dans les KGs de DBP15K.

Jeu de données		Nombre d'entités	Nombre de relations	Nombre de triplets
ZH-EN	Chinois	66 469	2 830	379 864
	Anglais	98 125	2 317	567 755
JA-EN	Japonais	65 744	2 034	354 619
	Anglais	95 680	2 096	497 230
FR-EN	Français	66 858	1 379	528 665
	Anglais	105 889	2 209	576 543

à l'extrême, on propose de générer automatiquement une *seed* de zéro, c'est-à-dire sans connaissances expertes ni intervention humaine. En particulier, on utilise la prédiction des méthodes conventionnelles symboliques, qui ne requiert pas de *seed* pour cela. Les méthodes que l'on utilise reposent uniquement sur du *string matching* (algorithme d'association de chaînes de caractères identiques) entre les noms des entités. Toute l'idée repose sur le principe d'identifier les couples d'entités ayant des noms les plus proches possibles. Le processus d'alignement automatique des noms d'entités, pour prédire une *seed*, est montré sur la Figure 4.3, où on fixe KG_1 à être le KG source et KG_2 à être la cible. Puisque les KGs ne sont pas nécessairement dans la même langue, un module de traduction est employé, Microsoft Translator (MS) dans nos expériences. Pour chaque nom d'entité de la source, on retrouve une liste d'entités de la cible avec un nom proche, en calculant la similarité des chaînes de caractères. Plusieurs algorithmes de comparaison de chaînes de caractères sont utilisés : fuzzy, Sørensen–Dice coefficient (DSC), Levenshtein, Jaro–Winkler. Cette méthodologie permet de faire un alignement de KGs par plongement sans connaissances *a priori*.

4.3 Expériences

On conduit à présent plusieurs expériences pour valider l'approche AlignD, en comparaison de l'état de l'art.

4.3.1 Protocole

L'ensemble de données standard DBP15K (Z. SUN, HU et LI, 2017) est utilisé pour réaliser la validation d'AlignD, en comparaison avec l'état de l'art. Il contient trois alignements de KGs obtenus à partir des versions multi-langues de DBpedia : chinois vers anglais (ZH-EN), japonais vers anglais (JA-EN), et français vers anglais (FR-EN). Chaque couple possède 15 000 entités alignées. On donne dans le tableau 4.1 des détails sur le contenu de chaque KG.

En suivant les travaux antérieurs sur l'alignement de KG (Z. SUN, HU et LI, 2017 ; Z. SUN, HU, Q. ZHANG et al., 2018), la *seed* a 30% des entités alignées (pour les expériences *seed* réelle), le reste est utilisé pour construire la *ref*. Pour l'évaluation, on réutilise les métriques de Z. SUN, HU, Q. ZHANG et al., 2018, le Mean Reciprocal Rank (MRR), moyennant les inverse des rangs des entités alignées dans le voisinage

des entités correspondantes, et le Hits@k, le ratio d'entités correctement alignés avec pour marge d'erreur les k plus proches voisins. Ces métriques mesurent la qualité de l'ordonnancement des entités de KG_2 dans le voisinage des entités de KG_1, en exploite la similarité cosinus pour trier les voisinages (Eq. 2.5). Par conséquent, le Hits@1 indique la qualité de l'alignement (l'entité de KG_2 la plus proche, doit être l'entité équivalente), et Hits@10 montre la qualité plus souple de la représentation jointe des KGs.

On étudie plusieurs cas d'initialisation d'AlignD en combinaison de plusieurs types de *seed* :

— AlignD et AlignD(glove,300), où on utilise la *seed* réelle composée de 30% des entités. AlignD est initialisé aléatoirement et AlignD(glove,300) l'est avec des plongements GloVe de 300 dimensions, comme le fait RDGCN (WU et al., 2019). Pour initialiser, RDGCN utilise le nom des entités et un module de traduction vers l'anglais pour les autres langues.

— AlignD[X] où AlignD est entraîné avec une *seed* générée automatiquement par un algorithme X.

Pour toutes les expériences, nous utilisons les hyper-paramètres $\lambda_1 = 0.01$, $\lambda_2 = 2$ et $\mu_1 = 0.2$. Le paramètre α, de l'Eq 4.6, dépend de la taille de la *seed* et est fixé manuellement. Le taux d'apprentissage est réglé à 0.01, pour 500 époques d'entraînement, en mettant à jour l'*alignement par bootstrapping* toutes les 10 époques. Le choix de ces hyper-paramètres repose sur les études précédents, notamment BootEA (Z. SUN, HU, Q. ZHANG et al., 2018). En particulier la dimension du plongement est de 75, sauf pour les expériences avec AlignD(glove,300), où la dimension vaut 300.

4.3.2 *Seed* réelle

On se compare d'abord sur un protocole identique aux études précédentes sur l'alignement de KGs, précisément sur DBP15K. On rappelle l'historique des algorithmes récents ayant montrés une performance à l'état de l'art :

— MTransE (M. CHEN et al., 2017) apprenant une transformation linéaire entre les plongements ;

— IPtransE (ZHU et al., 2017) qui établit de façon itérative des plongements via PTransE (Y. LIN et al., 2015) ;

— JAPE (Z. SUN, HU et LI, 2017) qui calcule des plongements de KGs joints tout en préservant les attributs ;

— Les approches basées sur GCN, comme MuGCN et RDGCN (CAO et al., 2019 ; WU et al., 2019). RDGCN est notre référence pour ces méthodes ;

— BootEA (Z. SUN, HU, Q. ZHANG et al., 2018), une technique semi-supervisée pour apprendre itérativement les entités alignées, que nous considérons comme notre référence pour les méthodes itératives basées sur TransE.

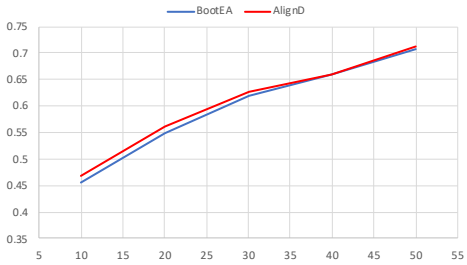
TABLEAU 4.2 – Hits@1, Hits@10 et MRR avec une *seed* réelle (représentant 30% de la taille total des alignements) sur DBP15K. La partie supérieure du tableau reprend les résultats publiés dans les études respectives. Les résultats des deux parties en-dessous sont produits via nos expériences.

Méthodes	ZH-EN			JA-EN			FR-EN		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE (M. CHEN et al., 2017)	30.83	61.41	0.364	27.86	57.45	0.349	24.41	55.55	0.335
IPTransE (ZHU et al., 2017)	40.59	73.47	0.516	36.69	69.26	0.474	33.30	68.54	0.451
JAPE (Z. SUN, HU et LI, 2017)	41.18	74.46	0.490	36.25	68.50	0.476	32.39	66.68	0.430
MuGCN (CAO et al., 2019)	49.56	87.03	0.621	50.10	85.70	0.621	49.50	87.00	0.621
BootEA (Z. SUN, HU, Q. ZHANG et al., 2018)	61.89	84.01	0.695	57.43	82.93	0.661	58.31	84.83	0.676
AlignD	62.68	84.70	0.701	58.88	83.06	0.671	60.77	85.30	0.691
RDGCN (WU et al., 2019)	70.75	84.55	–	76.74	89.54	–	88.64	95.72	-
AlignD(glove,300)	82.30	93.93	0.864	84.90	94.24	0.882	90.85	97.26	0.913
BootEA(glove,300)	83.11	93.84	0.869	84.60	93.93	0.879	92.38	97.56	0.942

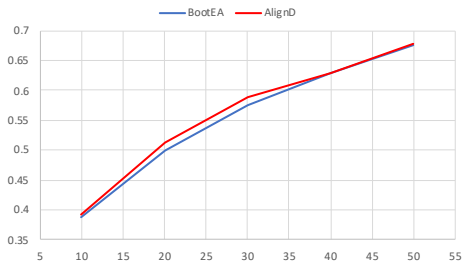
Les résultats sur la *seed* réelle de 30% sont rassemblés dans la Table 4.2. AlignD apparaît être le meilleur candidat des 5 références (ligne 1 à 5), en notant tout de même que BootEA et AlignD ont des performances proches. RDGCN n'est, quant à lui, pas comparable aux lignes 1 à 6, comme il utilise un autre mode d'initialisation avec des vecteurs pré-entraînés de taille 300. Pour rendre la comparaison équitable, on se propose d'initialiser les méthodes BootEA et AlignD de la même façon (ligne 8 et 9). Dans cette configuration, RDGCN se montre moins efficace. Il faut aussi remarquer que BootEA(glove,300) retourne la meilleure performance, cependant en pratique sur des données réelles, il peut être difficile d'obtenir une telle initialisation du plongement sur les entités. En effet, les plongements des entités sont initialisés en prenant le plongement du mot satisfaisant une certaine similarité morphologique avec l'entité, en partant du principe que les textes sur lesquelles ont été appris les vecteurs des mots coïncident avec la sémantique de l'entité, ce qui n'est pas toujours vrai en pratique.

4.3.3 Taille de la *seed* et performance

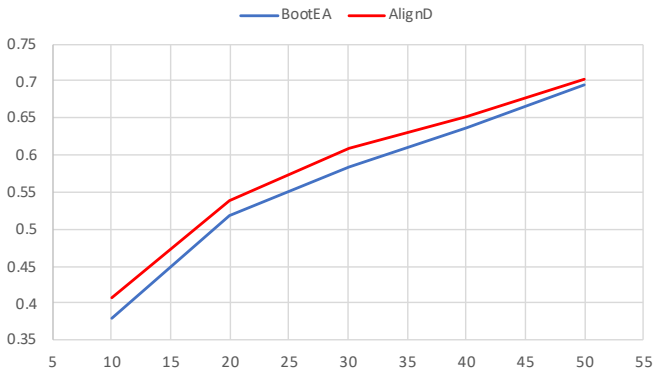
Comme reporté dans plusieurs études (Z. SUN, HU, Q. ZHANG et al., 2018 ; WU et al., 2019 ; Zhichun WANG et al., 2018 ; Z. SUN, HU et LI, 2017), le pourcentage d'entités pré-alignées (la *seed*) influence largement la précision. On reporte sur la Figure 4.4 la précision en fonction de plusieurs valeurs de ce pourcentage (de 10% à 50%). D'abord, on observe que les deux méthodes (BootEA et AlignD) sont fortement dépendantes de la taille de la *seed*. Cependant, pour la plupart des valeurs AlignD a une performance supérieure, avec un écart plus grand pour les petites tailles, et ce de façon consistante sur les trois version de DBP15k. Cette dernière observation peut traduire la robustesse d'AlignD aux petites *seed* et au bruit dans l'alignement. Lorsque la *seed* se réduit, le pourcentage d'*alignement par bootstrapping* augmente, ce qui entraîne nécessairement davantage d'erreurs d'alignement fournies à l'entraînement (via la semi-supervision). AlignD semble mieux profiter de cette situation que BootEA.



(A) ZH-EN



(B) JA-EN



(C) FR-EN

FIGURE 4.4 – Influence de la taille de la *seed* (pourcentage de la totalité des alignements disponibles, en abscisse) sur la performance (Hits@1, en ordonné) pour BootEA et AlignD.

TABLEAU 4.3 – Hits@1 pour des méthodes symboliques seules ou en combinaison avec AlignD.

Approches	symbolique seule			AlignD[symbolique]		
	ZH-EN	JA-EN	FR-EN	ZH-EN	JA-EN	FR-EN
DSC	25.22	44.64	54.10	80.28	84.01	89.33
Fuzzy	35.00	44.93	46.57	71.05	81.81	84.97
SubString	24.75	44.29	57.00	70.79	76.84	85.95
Word2Vec	24.38	37.49	39.86	79.61	80.81	84.47
Levenshtein	25.07	44.27	54.05	79.95	84.08	89.2
JaroWinkler	29.06	49.01	54.73	81.21	85.30	89.61

TABLEAU 4.4 – Hits@1 pour des méthodes d'alignement par plongement utilisant une *seed* générée automatiquement par approche symbolique.

Approches	ZH-EN	JA-EN	FR-EN
BootEA[JaroWinkler]	61.35	57.56	59.06
RDGCN[JaroWinkler]	65.10	75.76	83.52
AlignD[JaroWinkler]	81.21	85.30	89.61
BootEA(glove,300)[JaroWinkler]	82.99	84.39	91.72
AlignD(glove,300)[JaroWinkler]	83.12	84.58	92.15

4.3.4 *Seed* générée automatiquement

Pour démontrer la robustesse d'AlignD aux alignements erronés, et aussi pour mettre en évidence une nouvelle méthodologie d'alignement de KGs, basée sur la combinaison entre méthodes symboliques conventionnelles et méthodes par plongements, ne requérant aucune connaissance *a priori* (pas de *seed* réelle fournie par des experts), on se propose d'étudier le cas d'une *seed* automatiquement et complètement créée *from scratch*.

La Table 4.3 reporte les résultats des méthodes symboliques seules et en combinaison avec AlignD pour toutes les versions de DBP15K. L'alignement symbolique est utilisé, dans les trois dernières colonnes, pour fournir une *seed* initiale de basse qualité (mais plus grande) à AlignD. AlignD augmente à chaque fois la précision de l'alignement, mais réduit aussi l'écart entre les méthodes symboliques. Les meilleurs Hits@1 sont obtenus avec la méthode Jaro-Winkler.

On compare ensuite, dans la Table 4.4, le système non-supervisé AlignD[JaroWinkler] avec les méthodes d'alignement par plongement de l'état de l'art utilisant la même *seed* générée automatiquement. AlignD[JaroWinkler] surpassent toutes les méthodes (sans initialisation), grâce à sa robustesse aux erreurs.

AlignD(glove,300)[JaroWinkler] obtient les meilleurs résultats. Il est important de noter que les résultats d'AlignD sans initialisation sont proches de ceux d'AlignD(glove,300)[JaroWinkler]. Cela signifie que dans ce cadre, l'initialisation n'apporte qu'un gain limité pour AlignD, bien que celle-ci soit difficile à réaliser en pratique.

4.4 Conclusion

AlignD est une nouvelle méthode d'alignement pour les KGs, se focalisant sur l'alignement dimensionnel global plutôt que l'alignement individuel par entités. Son apport est la concentration d'informations similaires dans les mêmes dimensions des deux plongements tout en réduisant la corrélation entre dimensions distinctes, ce qui favorise l'incorporation d'informations différentes sur des dimensions différentes. Un autre intérêt est son fonctionnement global qui évite une sensibilité aux erreurs lorsque ces dernières sont peu nombreuses.

Notre proposition va aussi plus loin en faisant interagir des méthodes conventionnelles basiques avec des méthodes par plongements, ce qui donne un cadre d'alignement de KGs non supervisé et donc entièrement automatique. Cela peut faciliter le passage à l'échelle de certaines méthodes conventionnelles et l'hétérogénéité sémantique en utilisant les plongements, et cela permet aux méthodes par plongements de pouvoir justifier certains alignements via la méthode conventionnelle. En effet, on reproche souvent aux méthodes par plongements d'être peu explicables (SHVAIKO et EUZENAT, 2013).

Ces travaux ont été publiés dans à la conférence internationale ICTAI 2020 (VAIGH et al., 2020) en commun avec Cheikh Brahim El Vaigh. La contribution de la thèse à ce travail se situe sur l'élaboration du critère d'alignement entre les plongements.

On donne dans ce chapitre une manière de représenter et d'enrichir ces connaissances via des KGs et la représentation RDF. Il existe cependant d'autres manières de représenter des connaissances en particulier lorsqu'il s'agit de connaissances hiérarchisées et dans un format moins contraignant que le format RDF. Souvent on dispose de données qui peuvent contenir des connaissances complexes (graphes, hyperonymie, ...), mais qui ne sont pas sous la forme RDF : soit par choix, soit parce qu'il n'est pas adapté (données textuelles, tabulaires, bases de données comme les n-grammes de mots du chapitre 3).

Ces sources de données peuvent contenir des informations utiles aux tâches industrielles, comme nous l'avons vu, ou pourrait venir enrichir des KGs. Nous souhaitons dans la suite proposer un moyen d'extraire ces informations sous forme de connaissances hiérarchisées, et d'améliorer leur représentation. Pour cela nous étudions une géométrie adaptée à ces connaissances : la géométrie hyperbolique.

Deuxième partie

Les espaces hyperboliques pour la représentation de connaissances hiérarchiques

La découverte et la représentation de connaissances invoquant des relations multiples est aussi un sujet de recherche, aboutissant potentiellement à des KGs plus riches mais étant aussi plus difficiles à établir, *via* un traitement du langage plus avancé faisant intervenir plusieurs étapes de classification (MALLART et al., 2020). Nous nous restreignons au cadre mono-relation qui est plus proche des usages de Solocal (comme pour organiser les différentes catégories d'établissements professionnels) mais est aussi adaptable à des données désordonnées de nature hétérogène (textes, tables, vecteurs) dans le but de construire des connaissances depuis toutes sources. L'avantage est aussi de pouvoir limiter l'intervention des classificateurs en produisant un plongement, directement issu d'un critère de similarité, puis en l'interprétant comme une hiérarchie continue, pour au final en extraire la hiérarchie discrète en jouant sur les liens entre arbre et espaces hyperboliques.

Pour constituer ou représenter un ensemble de connaissances partant d'un corpus de données quelconques, on utilise dans cette partie des hiérarchies. Les hiérarchies sont des KGs dans lesquelles un unique type de relation inter-entités est considéré, et sont donc analogues à des graphes dirigés acycliques (*Directed Acyclic Graph*s, DAGs) connexes (SUZUKI, ENOKIDA et YAMANISHI, 2019). Les hiérarchies sont par conséquent représentées comme des graphes standards (V, E) composés d'un ensemble de nœuds V et d'un ensemble de liens $E \subseteq V \times V$, assimilés à l'unique relation représentée et satisfaisant les contraintes acycliques et de connexité. La singularité des hiérarchies les rapproche des espaces hyperboliques, considérés (NICKEL et KIELA, 2017) comme l'extension naturelle continue des structures d'arbres (graphes acycliques non dirigés connexes), et nous utilisons donc dans cette thèse la géométrie hyperbolique pour représenter continûment ces hiérarchies. Par exemple, WordNet (G. A. MILLER, 1995) est une hiérarchie sur des mots et groupes nominaux anglais, chaque mot ou groupe est inclus dans un ensemble de synonyme (*synset*). Ainsi, le *synset musical_instrument.n.01* est le représentant des mots *instrument* et *musical instrument*. Les *synset* sont ensuite reliés par des liens caractérisant l'inclusion ou la généralisation du concept *synset* avec d'autres. Ici, *musical_instrument.n.01*, représentant les instruments de musiques est le parent (entre autres), qu'on appellera l'hyperonyme, des *synset wind_instrument.n.01* (les instruments à vent), *string_instrument.n.01* (les instruments à cordes) et *percussion_instrument.n.01* (les percussions). Ces enfants sont appelés des hyponymes.

Nous montrons donc dans un premier temps (chapitre 5 et 6) l'intérêt et la manière de procéder à la représentation de hiérarchies dans des espaces hyperboliques, puis nous proposons un logiciel pour produire des plongements hyperboliques à partir de similarités (chapitre C), avant de détailler une façon d'extraire des hiérarchies discrètes de ces plongements (chapitre 7). Enfin, nous tenterons d'appliquer ces espaces pour la tâche d'étiquetage hiérarchique (chapitre 8).

Chapitre 5

Représentation de hiérarchies dans des espaces hyperboliques

Sommaire

5.1 Enjeux de la représentation des données 92
5.2 Plongement de hiérarchies en espace hyperbolique 93
5.2.1 Sur le principe des plongements de hiérarchies 93
5.2.2 Méthodes déterministes 95
5.2.3 Méthodes stochastiques 98
5.3 Conclusion 103

5.1 Enjeux de la représentation des données

Une bonne représentation des données est cruciale si l'on veut pouvoir extraire de bonnes caractéristiques de données désordonnées, puisque cela facilite à la fois la visualisation de ces données et leur integration dans des modèles d'apprentissage automatique. Par conséquent, le choix de l'espace de représentation est important pour mettre en valeur les informations cachées dans ces données. Certains espaces, comme l'espace hyperbolique, viennent ajouter des contraintes particulières, simplifiant l'accès à ces informations qui resteraient invisibles autrement.

En particulier, les graphes sont des structures exponentielles qui interviennent dans la résolution de problèmes réels (GOYAL et FERRARA, 2018). Ils sont connus pour être des structures complexes à représenter, puisque leur nature exponentielle et leur grand nombre d'attributs par nœud en font une source d'information de très grande dimensionalité. Produire de bons vecteurs pour les nœuds du graphe est par conséquent difficile, puisqu'il faut compresser ces informations discrètes, parfois éparses, en une représentation dense, qui préserve idéalement la structure du graphe et ses atouts sémantiques. Selon GOYAL et FERRARA, 2018, les plongements de graphes doivent jongler avec trois contraintes : le passage à l'échelle (les graphes sont souvent très larges nécessitant des algorithmes efficaces et rapides), la préservation des propriétés du graphes (quels attributs/liens doivent être préservés ?), et la complexité (des vecteurs de grandes dimensions sont plus coûteux à calculer). Nous pensons que la géométrie hyperbolique peut faciliter l'équilibre entre ces contraintes, dans le cas où le graphe est une hiérarchie.

La géométrie euclidienne est utilisée avec des arbres, pour que l'arbre couvrant géométrique minimal des nœuds plongés (l'arbre couvrant les nœuds dont les arêtes sont pondérés par la distance euclidienne, et dont la somme du poids des arêtes est minimal) corresponde à l'arbre à plonger (MONMA et SURI, 1992). Le degré (nombre de voisins directs) maximal des nœuds restreint alors la possibilité de faire ce type de plongements, ce qui se traduit par l'impossibilité de préserver l'information de l'arbre vers l'espace. La géométrie euclidienne est donc bornée par des caractéristiques du graphe pour parvenir à le représenter convenablement. Pour agrandir cette borne, il faut augmenter la dimension de la représentation entraînant un surplus de complexité numérique, sans réellement être à l'échelle de la nature exponentielle des graphes (K. SUN et al., 2015 ; NICKEL et KIELA, 2017). À l'inverse, la géométrie hyperbolique se montre plus apte à gérer la complexité des graphes (KRIOUKOV et al., 2009). KLEINBERG, 2007 démontre la supériorité de la géométrie hyperbolique en prouvant que tout graphe connexe admet un *greedy embedding* sur $\mathbb{D}^2$. En comparaison à la géométrie euclidienne, on a besoin d'au moins $\Omega(\log n)$ dimensions pour que chaque graphe contenant n nœuds en admette un. Cela est dû aux propriétés inhérentes des espaces hyperboliques, la plus importante étant son volume de croissance exponentielle, s'accordant avec la complexité spatiale des graphes, et le fait qu'un point dans l'espace peut disposer d'une infinité de voisins (K. SUN et al., 2015).

Pour ces raisons, la géométrie hyperbolique a été récemment utilisée sur des

hiérarchies et a montré des performances supérieures à la géométrie euclidienne, lorsqu'il s'agit de découvrir ou préserver des liens pertinents dans les données (NICKEL et KIELA, 2017 ; GANEA, BECIGNEUL et HOFMANN, 2018a). En outre, ces espaces présentent d'autres comportements avantageux comme le fait que la norme du vecteur semble corrélée à la profondeur de l'entité dans la hiérarchie (c'est-à-dire sa spécificité), quand le plongement est calculé comme dans le travail de NICKEL et KIELA, 2017.

Le plongement de hiérarchies dans des espaces hyperboliques trouve plusieurs applications comme la classification hiérarchique (B. CHEN, HUANG et al., 2020), l'induction de taxonomies (ALY et al., 2019), l'apprentissage zero-shot (LIU et al., 2020), et le clustering hiérarchique (MONATH et al., 2019). Dans chaque cas, l'intégration de la représentation continue de la hiérarchie facilite l'ajout de connaissances hiérarchiques améliorant la performance finale des modèles. Les récents réseaux de neurones hyperboliques, proposés par GANEA, BECIGNEUL et HOFMANN, 2018b, ouvrent la porte à l'usage des plongements hyperboliques pour n'importe quelle tâche dans des réseaux profonds, mais créent aussi le besoin de plongements de bonne qualité.

Dans ce qui suit, nous présentons le principe sous-jacent à l'incorporation continue des connaissances hiérarchiques et aux méthodes actuelles regroupées en deux catégories : déterministes et stochastiques. Ce chapitre est un état de l'art des méthodes de plongement de hiérarchies en espace hyperbolique, et permet de mieux appréhender ces espaces.

5.2 Plongement de hiérarchies en espace hyperbolique

5.2.1 Sur le principe des plongements de hiérarchies

Pour les plongements de mots, nous avons rappelé la notion de sémantique (*hypothèse distributionelle*) et la façon dont cette dernière s'intègre via des fenêtres contextuelles (voir chapitre 1). Pour un KG multi-relationel, TransE associe à chaque relation un vecteur et utilise l'addition pour transcrire ces relations sous forme géométrique (voir chapitre 4). Dans le cas d'une hiérarchie, nous disposons essentiellement d'un graphe dans lequel les liens entre les nœuds sont homogènes. Représenter de l'information provenant de hiérarchies est donc une tâche plus spécifique qui s'assimile à l'origine comme un problème de préservation des distances.

En effet, comme les liens ne portent pas d'informations propres, il s'agit essentiellement de minimiser la déformation des distances entre des nœuds dans la hiérarchie (vus comme les chemins de poids minimaux entre eux) et les vecteurs de ces nœuds dans l'espace du plongement. C'est, en tout cas, en minimisant cette déformation que l'on considère préserver l'information d'un graphe non dirigé (SARKAR, 2011 ; LEE, NAOR et PERES, 2009). Dans le cas d'une hiérarchie, il faut aussi prendre en compte l'aspect dirigé des liens et l'organisation par niveau. La norme apparaît comme une manière d'incorporer cette information soit en l'imposant directement

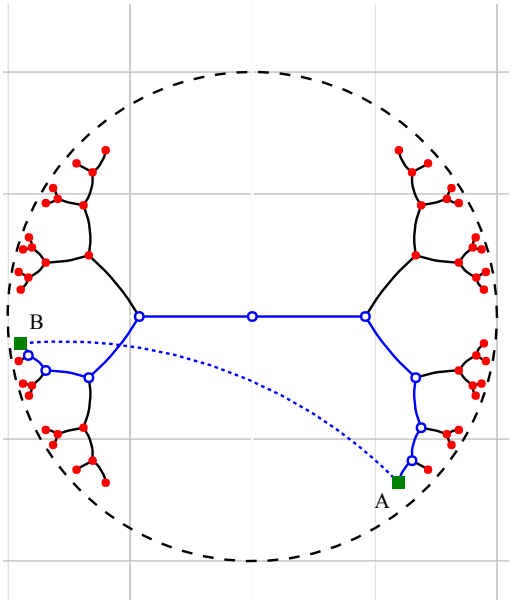


FIGURE 5.1 – Plongement d'un arbre binaire dans la boule de Poincaré $\mathbb{D}^2$ (la limite est illustrée par la ligne en tiret noir). Le chemin le plus court de A à B dans l'arbre (en bleu plein) est montré en relation de la distance dans $\mathbb{D}^2$ entre A et B (en pointillés bleus). La distortion est le produit des rapports maximaux (des deux sens) entre les distances de chaque paire de nœuds.

par construction soit en laissant l'algorithme l'encoder naturellement (NICKEL et KIELA, 2017).

Le plongement d'une hiérarchie doit donc préserver deux informations : la distance entre les nœuds et l'organisation des niveaux. L'organisation des niveaux peut s'encoder dans la norme des vecteurs, et, la préservation des distances est réalisée en optimisant la distortion, qui est un objectif mathématique défini pour mesurer la modification des distances lors du passage d'un espace à l'autre.

On considère pour cela le graphe non dirigé issu de la hiérarchie (obtenu en omettant la direction des liens) comme un espace métrique fini $(V, d_{\mathcal{G}})$ où V est l'ensemble des nœuds et $d_{\mathcal{G}}$ la distance définie sur le graphe $\mathcal{G} = (V, E)$ (à savoir la somme des poids du chemin de poids minimal entre deux nœuds). On se donne ensuite un espace métrique (Y, d_Y) accueillant les vecteurs d'un plongement dans Y, $f : V \to Y$, et muni d'une distance d_Y. On définit les quantités suivantes :

— l'expansion : $\delta_e(f) = \max\limits_{(x,y)\in V} \frac{d_Y(f(x),f(y))}{d_{\mathcal{G}}(x,y)}$, le facteur d'agrandissement des distances,

— la contraction : $\delta_c(f) = \max\limits_{(x',y')\in \mathrm{Im} f(V)} \frac{d_{\mathcal{G}}(f^{-1}(x'),f^{-1}(y'))}{d_Y(x',y')}$, le facteur de rétrécissement des distances,

— la distortion : $\delta(f) = \delta_c(f) \cdot \delta_e(f)$.

Dans le cas d'un espace hyperbolique en sortie on pourrait avoir $Y = \mathbb{D}^n$ et $d_Y = d_{\mathbb{D}^n}$ (voir section 1.2.2.2). On donne sur la figure 5.1 une représentation du calcul pour une paire de nœuds. L'idée de la distortion est de calculer la déformation maximale de la distance sur l'ensemble des paires possibles. Plus le ratio est proche de 1, moins il y a de déformations, et plus on considère que le plongement est de bonne qualité.

5.2.2 Méthodes déterministes

Les méthodes déterministes reposent sur des algorithmes qui positionnent les nœuds de la hiérarchie en veillant à respecter ou à minimiser la distortion. Ces méthodes retourneront donc toujours les mêmes résultats pour un graphe donné, et sont adaptées à cette seule problématique de préservation des distances.

Algorithme 5.1 Construction de SARKAR, 2011 (adapté de SALA et al., 2018)

Entrée: Un arbre dirigé $T = (V, E)$ enraciné en r_0, et un plongement vide f

function SARKAR_CONSTRUCTION(a, b, f, T) ▷ a est un nœud, b son parent, f le plongement partiel, T l'arbre
  $\theta \leftarrow 0$
  si $b \neq$ null **alors**
    $(0, z) \leftarrow \text{reflect}_{f(a)\to 0}(f(a), f(b))$
    $\theta \leftarrow \arg(z)$ ▷ angle entre z et l'axe x dans le plan
  sinon
    $f(a) \leftarrow (0, 0)$
  fin si
  pour c_i in successors(T, a) **faire** ▷ i énumère les successeurs, partant de 1
    $y_i \leftarrow \frac{e^\tau - 1}{e^\tau + 1} \cdot \left(\cos(\theta + \frac{2\pi i}{\text{degree}(T,a)}), \sin(\theta + \frac{2\pi i}{\text{degree}(T,a)})\right)$
    $f(c_i) \leftarrow \text{reflect}_{0\to f(a)}(y_i)$
    $f \leftarrow$ SARKAR_CONSTRUCTION(c_i, a, f, T)
  fin pour
  return f
fin function
$f \leftarrow$ SARKAR_CONSTRUCTION(r_0, null, f, T)

En particulier, SARKAR, 2011 a proposé une construction (voir Algorithme 5.1) pour plonger n'importe quel arbre dans $\mathbb{D}^2$ avec une distortion de $(1 + \varepsilon)$, ε étant arbitrairement petit. Une illustration de la construction est proposée sur la figure 5.2. On dispose les successeurs de x_0 sur le cercle 0, puis partant d'un successeur x_1, on trace le cercle 1 et on dispose les successeurs de x_1 et ainsi de suite. Tous les cercles ont le même rayon. Les cercles en géométrie hyperbolique diffèrent de ceux de la géométrie euclidienne car leur centre est décalé. Pour pouvoir réutiliser les propriétés des cercles euclidiens, on configure le cercle en $\vec{0}$ qu'on déplace en v_{x_i} au travers d'une fonction $\text{reflect}_{\vec{0}\to v_{x_i}}$ (voir figure 5.3). La longueur du rayon garantit la distortion minimale de $(1 + \varepsilon)$, autrement dit, plus τ est grand, plus la distortion

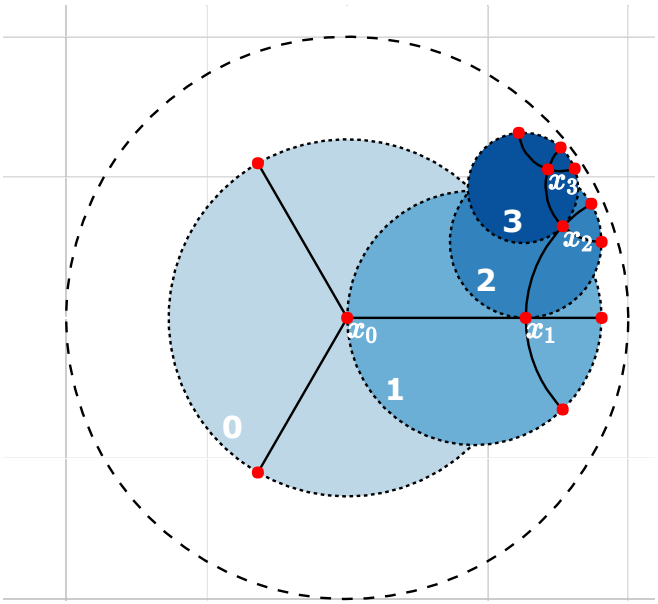


FIGURE 5.2 – Construction de SARKAR, 2011. Tous les cercles ont la même surface et sont de même rayon.

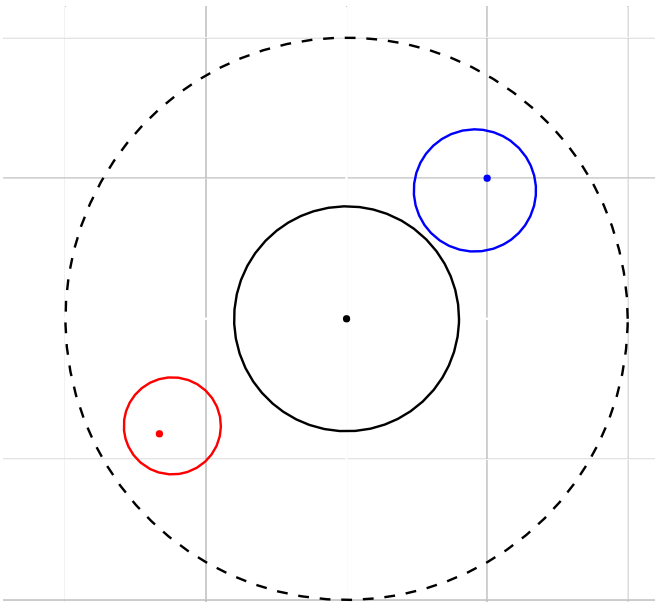
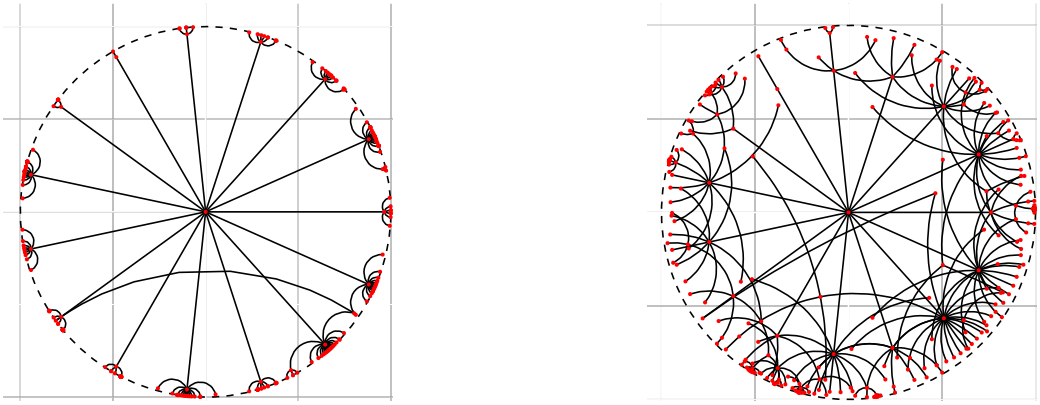


FIGURE 5.3 – Déplacement d'un cercle centré sur $\vec{0}$ (en noir) vers deux autres emplacements (rouge et bleu) dans $\mathbb{D}^2$. Les distances sont préservées.



(A) $\tau = 4$ (B) $\tau = 2$

FIGURE 5.4 – Plongement de la hiérarchie *environnement* de la tâche 14 de SemEval-2016 (BORDEA, LEFEVER et BUITELAAR, 2016) dans $\mathbb{D}^2$, avec plusieurs rayons (2 et 4). Les plongements des nœuds sont les points rouges et les liens de la hiérarchie sont représentés par des lignes pleines noires.

sera proche de 1 (voir la figure 5.4). τ est un paramètre important qui contrôle la performance de la reconstruction (ou la préservation de la hiérarchie), mais demande aussi des vecteurs d'une grande précision numérique : plus τ grandit plus les points se rapprochent de la bordure du disque sans jamais l'atteindre, et cela requiert une précision numérique très grande. Par conséquent, même si la théorie garantit une distortion optimale, en pratique, cela requiert un grand τ pour des hiérarchies avec un degré maximal élevé, ce qui limite cette approche. La profondeur de la hiérarchie influence aussi le besoin d'une grande précision.

SALA et al., 2018 ont proposé récemment une solution, HyperE, à ces problèmes pratiques directement liés à des caractéristiques de la hiérarchie à représenter. Cela consiste à réduire la tension sur la précision numérique en généralisant la construction de Sarkar dans des dimensions supérieures : les nœuds successeurs sont positionnées sur les sommets d'un hypercube. HyperE permet de projeter des hiérarchies mais aussi tout type de graphes en proposant de le transformer en un arbre préservant ses distances.

À notre connaissance, pour les espaces hyperboliques et les hiérarchies, il n'existe pas d'autres méthodes déterministes. Ces méthodes sont satisfaisantes d'un point de vue théorique puisqu'elles offrent un contrôle de la distortion via des paramètres du plongement (τ ou la dimension). D'un point de vue purement pratique, elles nécessitent une grande précision, principalement parce qu'elles sont conçues pour satisfaire le pire des cas. En effet, certaines hiérarchies peuvent avoir des branches moins peuplées ou moins profondes, et n'ont pas forcément besoin de l'espace alloué par l'algorithme. Ainsi sur la figure 5.4a, les branches faiblement peuplées ont autant de place pour se développer que les autres, bien que cela ne semble pas nécessaire. De la même manière, il est possible d'avoir des informations sur les attributs que l'on souhaite préserver dans l'espace. Comme les algorithmes déterministes sont conçus pour ne satisfaire que le critère de distortion, il apparaît

difficile d'intégrer d'autres critères. Les algorithmes stochastiques peuvent être utilisés comme un secours à cette situation.

5.2.3 Méthodes stochastiques

Nous appelons les méthodes stochastiques l'ensemble des méthodes se basant sur un processus aléatoire pour construire une solution au problème de préservation d'informations entre la hiérarchie et le plongement. Pour les méthodes considérées ici, elles reposent sur la définition d'un critère d'optimisation et de la minimisation de celui-ci via SGD. Nous proposons de voir d'abord le moyen d'optimiser des paramètres dans un espace hyperbolique.

5.2.3.1 Approche intuitive sur les variétés

Avant de plonger dans l'optimisation dans des espaces hyperboliques, nous voulons d'abord présenter un regard intuitif sur les variétés, un objet mathématique essentiel au travail dans ces espaces. On définit, généralement, les variétés comme une généralisation de la notion de courbe ou de surface à des dimensions supérieures (BECIGNEUL et GANEA, 2019), sans pour autant que cela facilite la visualisation d'un tel object.

Considérons une feuille de papier. Cette feuille est un espace euclidien à deux dimensions, dans lequel, des points ou des vecteurs peuvent exister. On peut courber cette feuille, sans marquer de plis, et former plusieurs types de surface, comme un cylindre. Ce cylindre est alors une variété. Si on place un point sur la feuille et qu'on désire le déplacer sur celle-ci, il faut suivre continûment la courbure du cylindre. Pour procéder à cette opération, on dispose de fonction mathématique, notamment l'expmap. Pour visualiser cette opération, déplions la feuille à plat, et traçons un vecteur amenant à la position souhaitée. Il suffit ensuite de replier la feuille sur la courbe précédente pour obtenir la position sur le cylindre du point. Cette opération de pliage, est exactement ce que l'expmap permet de faire.

On parle de variété différentiable quand ce qui se passe localement sur la feuille dépliée, est quasi-identique à ce qui se passe sur la feuille pliée. Cela nous permet d'appliquer les théories de différentiations sur le plan euclidien et de les transposer à la feuille courbée, via plusieurs techniques explicitées ci-après.

Dans la modélisation 3D, on utilise la cartographie UV par exemple pour replier une image 2D sur la surface d'un solide, et ainsi lui donner une texture. La surface du solide est la variété et la façon de plier un plan 2D (une image) sur celle-ci est l'expmap. La forme du solide, et donc de la surface, conditionne le pliage et peut donner un aspect étrange à l'image dépliée, traduisant un peu la géométrie de l'objet 3D. Ces aspects étranges sont en fait des caractéristiques qui nous intéressent pour les espaces hyperboliques, car ils facilitent la représentation de données hiérarchiques.

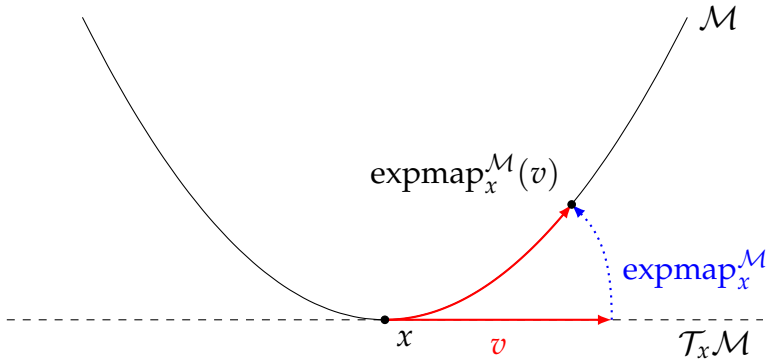


FIGURE 5.5 – Illustration de l'application d'une expmap en un point x sur une variété $\mathcal{M}$. La longueur des segments rouges est identique.

5.2.3.2 Optimisation dans un espace hyperbolique

Rappelons qu'un espace hyperbolique est en réalité une variété riemannienne $(\mathcal{M}, g)$, composée d'une variété différentiable $\mathcal{M}$ et d'un tenseur métrique g (BECIGNEUL et GANEA, 2019 ; NICKEL et KIELA, 2018). Une variété différentiable est une variété qui se comporte localement comme un espace euclidien classique. On définit souvent l'espace tangent d'un point x, noté $\mathcal{T}_x\mathcal{M}$ permettant le calcul des gradients en ce point x et qui s'apparente à l'approximation locale de la variété (BECIGNEUL et GANEA, 2019). La distance et les géodésiques (généralisation du plus court chemin sur la variété) entre des paires de points sont calculées à l'aide du tenseur métrique g (BECIGNEUL et GANEA, 2019).

On définit des fonctions permettant l'aller-retour entre le plan tangent $\mathcal{T}_x\mathcal{M}$ en x et la variété $\mathcal{M}$: l'$\exp_x^{\mathcal{M}} : \mathcal{T}_x\mathcal{M} \to \mathcal{M}$ qui plie le plan sur $\mathcal{M}$, et la $\log_x^{\mathcal{M}} : \mathcal{M} \to \mathcal{T}_x\mathcal{M}$ qui est son inverse. Quand on parle de plier le plan sur la variété, on peut aussi voir un vecteur de vélocité $v \in \mathcal{T}_x\mathcal{M}$ appliqué à x, ce dernier est alors poussé d'une distance $\|v\|$ en restant collé sur la variété, et la direction du déplacement est donnée par la direction de v au départ (BONNABEL, 2013, et voir figure 5.5). La logmap permet de retrouver les vélocités du plan $\mathcal{T}_x\mathcal{M}$ permettant de reconstituer les autres points de la variété depuis x. C'est la généralisation de l'addition dans un espace euclidien à une variété riemannienne.

Enfin, pour pouvoir arriver à généraliser la SGD et ses améliorations (avec momentum, Adam, etc.) à ces objets, il faut être capable d'additionner des vecteurs, mais aussi, de les transporter sur la variété afin qu'il puisse garder leur sens identique au travers du déplacement (déplacement du momentum à l'emplacement du paramètre mis à jour). On appelle cela le transport parallèle, qui est transparent dans un espace euclidien car sa courbure est nulle (BECIGNEUL et GANEA, 2019). Sur la figure 5.6, si le vecteur rouge m_t est conservé entre θ_t et θ_{t+1}, on perd la direction dans laquelle il aurait fallu se déplacer en prenant en compte l'historique des gradients, puisque m_t devient orthogonal au plan tangent de θ_{t+1}. Le transport parallèle $\varphi^{\mathcal{M}}_{\theta_t \to \theta_{t+1}}$ permet de préserver cette information en déplaçant m_t collé à la variété.

Comme les variétés sont des espaces courbes, on ne peut pas faire l'optimisation de paramètres appartenant à la variété riemannienne comme si on était dans

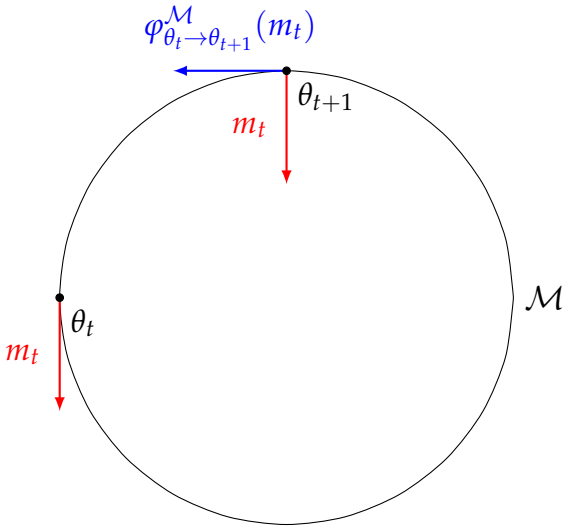


FIGURE 5.6 – Illustration du transport parallèle $\varphi^{\mathcal{M}}_{\theta_t \to \theta_{t+1}}$ entre deux points θ_t et θ_{t+1}.

un espace euclidien. C'est pourquoi ces opérateurs existent et nous permettent de faire une généralisation. À ce stade, il est aussi important de remarquer que les variétés sont des espaces à $n-1$ degrés de liberté existant dans un espace euclidien de dimension n. On appelle parfois l'espace euclidien accueillant la variété, l'espace euclidien ambiant (NICKEL et KIELA, 2018 ; WILSON et LEIMEISTER, 2018).

Considérons un ensemble de paramètres $\Theta = (\theta_i) \in \mathcal{M}$ et un critère d'optimisation $\mathcal{L}$ différentiable. On peut calculer le gradient euclidien de $\mathcal{L}$ en θ_i via l'approximation localement euclidienne de la variété riemannienne, on le note $\nabla_E \mathcal{L}(\theta_i)$. AMARI, 1998 et NICKEL et KIELA, 2018 donnent le moyen d'obtenir le gradient riemannien $\nabla_R \mathcal{L}(\theta_i)$ à partir du gradient euclidien et du tenseur métrique g en θ_i noté g_{θ_i} :

$$\nabla_R \mathcal{L}(\theta_i) = g^{-1}_{\theta_i} \nabla_E \mathcal{L}(\theta_i), \tag{5.1}$$

$\nabla_R \mathcal{L}(\theta_i)$ pointant vers une direction plus pertinente que $\nabla_E \mathcal{L}(\theta_i)$ sur la variété. Si nécessaire, $\nabla_R \mathcal{L}(\theta_i)$ est projeté dans $\mathcal{T}_{\theta_i}\mathcal{M}$ avant d'être appliqué comme suit :

$$\theta_i \leftarrow \exp\text{map}^{\mathcal{M}}_{\theta_i}(-\alpha \cdot \nabla_R \mathcal{L}(\theta_i)) \tag{5.2}$$

où α est le taux d'apprentissage. Pour la conservation des momentums, il faut utiliser le transport parallèle pour chaque θ_i, comme introduit par BECIGNEUL et GANEA, 2019.

En principe, on utilise le disque de Poincaré $\mathbb{D}^n$ ou l'espace de Lorentz $\mathcal{L}^n$ (voir section 1.2.2.2) car ils disposent de formules fermées pour l'expmap, la logmap, la distance et le transport parallèle, ce qui facilite grandement les étapes de la descente de gradient stochastique riemannienne. Le logiciel geoopt [1] (KOCHUROV, KARIMOV et KOZLUKOV, 2020) est une implémentation de ce type d'algorithme. Dans la suite,

1. https://github.com/geoopt/geoopt

on fera l'abus d'assimiler la variété riemannienne $(\mathcal{M}, g)$ à la variété différentiable $\mathcal{M}$, car on ne se servira pas de g.

5.2.3.3 Optimisation dans un espace hyperbolique avec un paramétrage euclidien

Il est plus compliqué de passer par des paramètres riemanniens, cela requiert plus de précautions lors de l'application des gradients et peut-être aussi davantage de calculs. Or, si on dispose d'une paramétrisation continue et différentiable de la variété dans un espace euclidien, il est possible de se passer de la géométrie riemannienne. C'est ce qui est proposé par LAW et al., 2019. On projette des paramètres euclidiens sur la variété pour calculer le critère d'optimisation. Le théorème de dérivation des fonctions composées nous permet de retrouver les gradients de nos paramètres euclidiens, et d'appliquer une SGD standard. LAW et al., 2019 démontre une convergence un peu moins efficace que la méthode précédente, le gain est surtout au niveau de la simplification théorique.

5.2.3.4 Plongements hyperboliques stochastiques

Les méthodes déterministes précédemment évoquées ont pour principale faiblesse leur incapacité à s'adapter à des critères d'optimisation multiples, comme elles sont façonnées pour optimiser essentiellement la distortion. C'est pourquoi les algorithmes stochastiques sont utiles ici, pour rendre plus flexible la projection et intégrer n'importe quel critère continu.

Des propositions ont été faites pour généraliser l'approche Word2Vec (voir chapitre 1) aux graphes quelconques exploitant la SGD et le NS, comme par exemple Node2Vec (GROVER et LESKOVEC, 2016). NICKEL et KIELA, 2017 a présenté une méthode similaire, usuellement nommé PoincaréModel, adapté aux hiérarchies car elle repose sur l'espace de Poincaré $\mathbb{D}^n$. Considérons une hiérarchie $\mathcal{G} = (V, E)$, un plongement $\Theta : V \to \mathbb{D}^n$ et $(u, v) \in E$, l'idée est de rapprocher $\Theta(u)$ de $\Theta(v)$ par rapport à des nœuds échantillonnés aléatoirement. Comme pour Word2Vec, il s'agit de pouvoir classifier les liens positifs par rapport à du bruit en se basant sur leur distance. À terme, un nœud aura comme plus proches voisins les nœuds avec lesquels il est en lien. On peut noter la fonction de coût (NICKEL et KIELA, 2017)

$$\mathcal{L}(\Theta) = \sum_{(u,v)\in E} \log \frac{e^{-d_{\mathbb{D}^n}(\Theta(u),\Theta(v))}}{\sum_{v'\sim\mathcal{N}(u)} e^{-d_{\mathbb{D}^n}(\Theta(u),\Theta(v'))}} \tag{5.3}$$

avec $\mathcal{N}(u) = \{v \in V | (u, v) \notin E\} \cup \{u\}$. Dans le cas de hiérarchies, $\mathcal{D} = \{(u, v) | u \text{ est un ancêtre de } v \text{ dans } \mathcal{G}\}$ est l'ensemble des liens d'hyperonymie (fermeture transitive). Ils peuvent être plus importants que les liens directs ($= E$) puisqu'ils permettent d'incorporer des relations hiérarchiques plutôt que structurelles. En pratique on incorpore donc $\mathcal{D}$ plutôt que E. Il faut remarquer que par conséquent, un ancêtre sera plus proche de ses successeurs qu'un de ses frères, ce qui est

un choix de représentation. Dans certaines situations il vaut mieux l'inverse, notamment pour l'extraction des liens directs, donc de la hiérarchie, nous pensons qu'il est préférable de pouvoir identifier clairement le parent direct plutôt que ses ancêtres.

On calcule les gradients de la fonction de coût de l'Eq. 5.3 et on optimise les paramètres du plongement Θ via la SGD riemannienne de la section 5.2.3.2. Ces plongements donnent des résultats en reconstruction largement supérieurs à leurs équivalents euclidiens et montrent une propriété particulière : la spécificité d'un concept semble s'encoder naturellement dans la norme du vecteur. Un concept spécifique aura alors une norme plus grande qu'un concept plus général.

Le fonctionnement de cette méthode a été amélioré par les améliorations successives, à savoir HypCones (GANEA, BECIGNEUL et HOFMANN, 2018a) et le travail de LAW et al., 2019, appelé Law19a par la suite. Ces deux améliorations ont pour objectif de renforcer l'encodage de la spécificité dans la norme en l'imposant explicitement et en élaborant des mesures favorisant ce comportement. Pour HypCones, il s'agit de définir une fonction d'énergie, minimisée lorsque les hyponymes d'un nœud se situent dans son cône. La distance de l'Eq. 5.3 est remplacée par cette fonction d'énergie. Nous avons remarqué pendant l'usage de cette méthode que la distance de l'espace n'avait plus de sens du fait de son inexistence dans le critère d'optimisation d'HypCones, ce qui ruine la distortion. En revanche, la fonction d'énergie peut faciliter d'autres tâches de reconstruction, pourvu qu'on l'étalonne sur un jeu d'entraînement.

Law19a utilise un autre espace hyperbolique nommé Hyperboloïde de Lorentz utilisant la variété $\mathcal{H}^{n,\beta} = \{x = (x_0, ..., x_n) \in \mathbb{R}^{n+1}, \langle x, x\rangle_{\mathcal{L}} = -\beta, x_0 > 0\}$, qui est la généralisation de l'espace de Lorentz $\mathcal{L}^n$ de la section 1.2.2.2 avec d'autres courbures. La pseudo-distance *squared lorentzian*

$$d_{\mathcal{H}^{n,\beta}}(x, y) = -2\beta - 2\langle x, y\rangle_{\mathcal{L}} \tag{5.4}$$

est donnée pour deux points x, $y \in \mathcal{H}^{n,\beta}$, et a l'avantage de disposer d'une forme close pour calculer le barycentre d'un ensemble de points (ce qui n'est pas le cas de $d_{\mathbb{D}^n}$). L'idée étant que l'hyperonyme se doit idéalement d'être le barycentre de ses hyponymes, la forme close permet alors de faciliter la reconstruction. Un autre atout de cette pseudo-distance est que le barycentre est de norme contrôlable en fonction de β, pouvant rendre sa norme inférieure à ses composantes, et donc, facilitant l'interprétation de l'espace.

L'optimisation pour Law19a se fait en utilisant l'optimisation de la section 5.2.3.3. On prend des vecteurs de $\mathbb{R}^n$ pour paramétrer $\mathcal{H}^{n,\beta}$. Soit $x = (x_1, ..., x_n) \in \mathbb{R}^n$, alors on transforme x via g_β,

$$g_\beta(x) = (\sqrt{\|x\|^2 + \beta}, x_1, ..., x_n) \in \mathcal{H}^{n,\beta}, \tag{5.5}$$

ce qui permet de procéder au calcul des distances de l'espace hyperbolique, le calcul des gradients par le théorème de dérivation des fonctions composées pour l'Eq 5.3,

et l'application de la SGD. Les résultats de LAW et al., 2019 confirment l'intérêt d'utiliser cette pseudo-distance au lieu de $d_{\mathbb{D}^n}$. L'inconvénient reste le fait qu'il ne s'agit pas d'une distance à proprement parler (l'inégalité triangulaire est insatisfaite), ce qui peut poser des problèmes pour certaines reconstructions.

5.3 Conclusion

Les plongements de hiérarchies reposent sur deux mécanismes différents, l'un est rigide et démontré mathématiquement, l'autre offre plus de flexibilité mais quelques lacunes théoriques. En effet, rien ne garantit une distortion minimale pour les méthodes stochastiques. Le gain en flexibilité est pertinent puisqu'il permet l'introduction de nouveaux critères prenant en compte d'autres aspects de la hiérarchie, comme la sémantique ou les attributs des nœuds, qui peuvent être incorporé à l'objectif d'optimisation.

Globalement, l'espace hyperbolique apparaît comme étant le plus adapté à la représentation de données sous forme de graphes, l'espace euclidien étant limité théoriquement et empiriquement (mauvais résultats de reconstruction pour les méthodes stochastiques, voir NICKEL et KIELA, 2017). La géométrie hyperbolique offre une réelle opportunité d'intégrer ces nouvelles informations (mieux préservées) dans des modèles.

L'intégration peut se faire via des réseaux de neurones hyperboliques, ou en fusion directe avec des réseaux de neurones standards. Il apparaît aujourd'hui difficile de donner une méthode d'exploitation de ces plongements hyperboliques, leur usage restant encore limité bien que leurs résultats soient prometteurs. Pour appuyer ce point, de récents travaux ont tenté d'exploiter avec succès un mélange entre plongements hyperboliques et réseaux euclidiens (LIU et al., 2020 ; B. CHEN, FU et al., 2021).

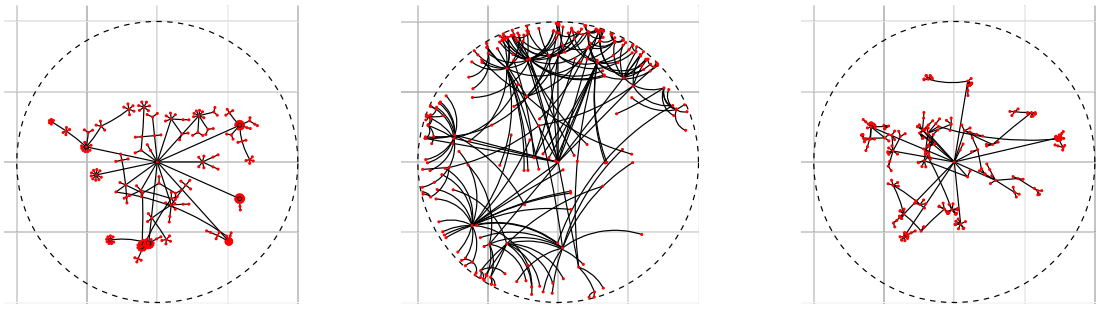
Dans les chapitres suivants, nous tentons d'exploiter la géométrie hyperbolique pour mieux représenter des informations hiérarchiques avec des informations sémantiques associés directement à la hiérarchie (chapitre 6), pour extraire des informations hiérarchiques à partir de sources de similarité ou de textes (voir chapitre C), et enfin pour combiner une représentation de la hiérarchie avec une représentation classique de document décrit par cette hiérarchie (voir chapitre 7). Maintenant que nous sommes équipés des connaissances sur la géométrie hyperbolique, nous pouvons commencer à traiter ces sujets. L'objectif final étant de pouvoir découvrir, représenter et combiner des informations hiérarchiques pour enrichir la représentation des mots.

Chapitre 6

Paramétrisation continue d'arbres en variété riemannienne

Sommaire

6.1 Variantes algorithmiques 106
6.1.1 HyperE Rayon Variable (VR) 107
6.1.2 HyperE Initialisation par Barycentre (BI) 107
6.2 ASTRE 108
6.2.1 Modèle de base 108
6.2.2 Généralisation aux autres variétés riemanniennes 110
6.2.3 Optimisation 112
6.3 Expériences 113
6.3.1 Expériences de reconstruction 113
6.3.2 Expériences de fusion 120
6.4 Conclusion 125



(A) VR (B) BI (C) VR + BI

FIGURE 6.1 – Trois variantes d'HyperE utilisant VR ou BI ou les deux (VR+BI) sur la taxonomie environnement de Semeval-2016 task 13 (BORDEA, LEFEVER et BUITELAAR, 2016) avec $\tau = 2$.

Les approches stochastiques et déterministes du chapitre 5 ont des avantages complémentaires : la flexibilité et la solidité théorique. On propose dans ce chapitre de combiner ces deux approches, en rendant paramétrable la méthode déterministe de SARKAR, 2011, ce qui permet l'optimisation des paramètres de la méthode au travers d'une SGD classique. On bénéficie ainsi d'une approche unifiée, tenant compte des contraintes topologiques de la hiérarchie, mais pouvant prendre en compte d'autres critères d'optimisation.

L'idée de cette contribution vient du manque d'optimisation de l'espace de la construction de SARKAR, 2011, soulevé à la fin de la section 5.2.2. Ce défaut vient, à notre sens, du fait que l'algorithme doit prévoir le pire cas de projection, forçant l'allocation homogène de portions du volume à des branches hétérogènes. La flexibilité des algorithmes stochastiques, et leur capacité à former le plongement d'une hiérarchie sur-mesure, est un atout dans ce cas. Elle permettrait à la construction de SARKAR, 2011 d'être plus fine sur le plongement, tout en préservant les aspects positifs d'une construction déterministe, par exemple la minimisation de la distortion. Nous essayons de tirer parti du meilleur des deux approches pour améliorer la représentation des hiérarchies.

On se contraint aux hiérarchies dans lesquelles les nœuds ont au plus un parent. Cette approximation est due à la construction de SARKAR, 2011, qui n'est conçue que dans ce cas précis (des arbres dirigés). On pourrait étendre la solution aux autres hiérarchies en adaptant le modèle de base, ou en trouvant un arbre couvrant minimisant la distortion avec la hiérarchie, comme le fait HyperE (SALA et al., 2018).

6.1 Variantes algorithmiques

Avant de commencer à présenter la méthode finale, nous souhaitons montrer les chemins par lesquelles nous sommes passés, car ils permettent de mieux comprendre l'intérêt de rendre adaptable les méthodes déterministes. En particulier, nous proposons des variantes d'HyperE (l'extension de la construction de SARKAR, 2011 aux dimensions supérieures par SALA et al., 2018).

6.1.1 HyperE Rayon Variable (VR)

Soit $T = (V, E)$ un arbre dirigé non pondéré, on peut produire un arbre dirigé pondéré T_w où chaque lien (u, v) de E est pondéré par

$$w_{u,v} = \frac{\text{degree}(v)}{\max\limits_{v' \in V} \text{degree}(v')} \in [0, 1]. \tag{6.1}$$

On peut désormais utiliser T_w avec HyperE. Quand on fournit un arbre pondéré à HyperE, il adapte le rayon τ pour chaque lien en se basant sur son poids. Plus spécifiquement, il le fixe à $\tau \cdot w_{u,v}$. L'intuition derrière cette pondération est que les nœuds avec un grand degré nécessitent un volume plus grand pour positionner leurs enfants, et donc, ils ont besoin d'être placés plus loin que les nœuds avec un degré plus faible (voir Figure 6.1a). Sur cette figure, les nœuds successeurs sans descendance forment une couronne autour de leur parent, et les nœuds avec descendance (sur la droite de la figure, par exemple) se placent à une distance plus lointaine.

6.1.2 HyperE Initialisation par Barycentre (BI)

On remarque que du volume est perdu entre les branches, à cause du placement des successeurs sur les sommets d'un hypercube. De plus, l'allocation des sommets ne doit pas être laissée au hasard, puisque deux branches fortement peuplées peuvent se retrouver adjacentes, augmentant le risque d'intersection entre les successeurs. On propose de réduire ce risque en écartant au maximum les grosses branches les unes des autres.

Au lieu d'affecter les nœuds aux sommets aléatoirement (au cours de l'Algorithme 5.1), on autorise les noeuds à se placer continûment sur l'hypersphère, ce qui permet une allocation inégale mais adaptée : les petites branches auront un volume plus restreint que les grosses.

Ce problème est formulé en utilisant le barycentre. Considérant un nœud x, ses successeurs $(s_i)_{[\![1,N]\!]}$, et son parent y, on positionne x au centre d'un cercle. Dans le cours de l'Algorithme 5.1, y est placé avant x, et donc, y occupe une position donnée sur l'hypersphère centré sur x. L'idée est de répartir les successeurs sur le reste de l'hypersphère, en respectant un équilibre défini par la taille des sous-branches des successeurs (d'où l'usage du barycentre). On pondère chaque successeur s_i par la taille de sa sous-branche w_i, et on calcule le barycentre de la famille composée des successeurs et du parent y. Pour y, on choisit un poids égal au poids moyen des successeurs $\bar{w} = \sum_i w_i / N$. Le barycentre pour un point x s'écrit

$$B(x) = \frac{\bar{w} \cdot \text{reflect}_{f(x)\to 0}(f(y)) + \sum_i w_i \cdot \text{reflect}_{f(x)\to 0}(f(s_i))}{(N+1) \cdot \bar{w}} \tag{6.2}$$

où f est le plongement des nœuds dans l'espace hyperbolique et $\text{reflect}_{a\to b}$ permet le déplacement réversible des points de a à b en préservant les distances entre les

éléments déplacés. On utilise l'inversion du cercle pour procéder à ce déplacement dans $\mathbb{D}^n$, voir les travaux de (SALA et al., 2018).

On considère que le barycentre est équilibré lorsqu'il est proche du centre de l'hypersphère. En deux dimensions, si on prend un polygone régulier pour lesquelles les sous-branches sont identiques (de même poids), alors le barycentre se situe au centre du cercle, et c'est pour cela qu'on prend l'hypothèse que rapprocher le barycentre du centre permet l'équilibre des branches dans l'espace. On minimise le coût

$$\mathcal{L}(f) = \sum_{x \in V} \|B(x)\|_2, \tag{6.3}$$

en veillant à ce que les plongements des vecteurs successeurs restent sur l'hypersphère centré sur x lors de l'optimisation.

On peut aussi rajouter un coût forçant les successeurs à être écartés au maximum les uns des autres, de façon à renforcer l'équilibre et l'écart entre les branches. On ne le fait pas ici car on est plutôt intéressé d'avoir une tendance pour initialiser les plongements pour la méthode ASTRE (de la section suivante), par conséquent, c'est surtout la direction que l'on cherche à obtenir.

La Figure 6.1c (combinant VR et BI) semble contenir moins d'intersections, sans que cela soit très convaincant. Nous voulons porter les deux idées plus loin en autorisant l'optimisation de ces deux paramètres (l'angle sur l'hypersphère et le rayon) par rapport à un critère quelconque. Cependant, pour que cela soit possible il faut être capable de calculer la représentation d'un nœud de façon globale et non relative à son parent. C'est ce que nous proposons avec ASTRE.

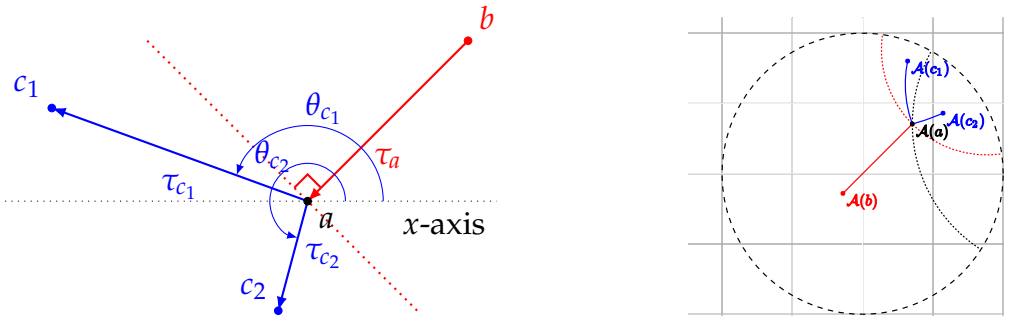
6.2 ASTRE

ASTRE signifie *Adaptive Sarkar TRee Embedding* ou plongements d'arbres de Sarkar adaptatifs. Ce modèle que nous proposons ne peut calculer des plongements que sur des arbres dirigés ; il convient de donner une façon de transformer les hiérarchies, utilisées pour l'évaluation, en arbre. Pour une hiérarchie $\mathcal{G} = (V, E)$, on la transforme en un arbre $T_{\mathcal{G}}$ via l'opération suivante : pour chaque nœud x avec de multiples parents dans $\mathcal{G}$, on préserve seulement le parent qui est sur le plus court chemin entre la racine r_0 de la hiérarchie et x.

Après cette transformation, les nœuds avec des parents multiples ne préservent qu'un seul parent, ce qui engendre la perte d'information hyperonymique. On espère limiter la perte de cette information au travers du critère d'optimisation, qui lui, est calculé sur l'ensemble de la hiérarchie. La simplification ne sert que pour le calcul des vecteurs des nœuds.

6.2.1 Modèle de base

La construction de SARKAR, 2011 a donc deux aspects variables : la distance à son parent (soit τ), et sa position sur l'hypersphère centré sur ce dernier. Ces



(A) Vue de l'espace des paramètres de ASTRE en $\theta_a(*)$. (B) Vue des paramètres projetés en $\mathcal{A}(a)$.

FIGURE 6.2 – Transformation de l'espace des paramètres de ASTRE (à gauche) vers $\mathbb{D}^2$ (à droite) pour un nœud a avec pour parent b et successeurs (c_1, c_2). Sur la Figure de droite, la distance entre deux nœuds est telle que $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(b)) = \tau_a$, $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(c_1)) = \tau_{c_1}$ et $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(c_2)) = \tau_{c_2}$, et les angles θ_{c_1}, θ_{c_2} sont préservés entre les tangentes en $\mathcal{A}(a)$.

deux aspects sont tournés en paramètres du modèle en utilisant les coordonnées n-sphériques (BLUMENSON, 1960). Formellement, les paramètres relatifs d'un nœud c à son parent a (voir Figure 6.2a), notés $\Theta_a(c)$ sont :

$$\Theta_a(c) = (\tau_c, \theta_{c,1}, ..., \theta_{c,n-1}) \in \mathbb{R}^n, \tag{6.4}$$

composés d'une partie radiale τ_c et d'une partie angulaire $(\theta_{c,1}, ..., \theta_{c,n-1})$ avec $\theta_{c,n-1} \in [0, 2\pi]$ et $(\theta_{c,1}, ..., \theta_{c,n-2}) \in [0, \pi]^{n-2}$. En particulier, dans $\mathbb{R}^2$ (resp. $\mathbb{R}^3$), on retrouve les coordonnées polaires (resp. sphériques). On écrit cartesian$(\theta_{c,1}, ..., \theta_{c,n-1})$ la fonction permettant le passage des coordonnées sphériques aux coordonnées cartésiennes.

Le calcul de la représentation d'un nœud dépend seulement de ses ancêtres, bien que l'Algorithme 5.1 ne mette pas en valeur cet aspect. On peut donc définir récursivement le calcul du vecteur du nœud c, noté $\mathcal{A}(c) \in \mathbb{D}^n$ (voir Figure 6.2) :

$$\mathcal{A}(c) = \begin{cases} \vec{s} & \text{si } c = r_0 \\ \text{reflect}_{0 \to \mathcal{A}(a)}(\text{proj}_{\mathbb{D}^n}(\Theta_a(c))) & \text{sinon } (a \text{ étant le parent de } c) \end{cases} \tag{6.5}$$

avec $\vec{s}$ le vecteur utilisé pour ancrer la représentation dans $\mathbb{D}^n$, qu'on appelle l'ancre, et $\text{proj}_{\mathbb{D}^n}$ est la fonction permettant la projection des paramètres dans $\mathbb{D}^n$:

$$\text{proj}_{\mathbb{D}^n}(\Theta_a(c)) = \frac{e^{\tau_c} - 1}{e^{\tau_c} + 1} \cdot \text{cartesian}(\theta_{c,1}, ..., \theta_{c,n-1}). \tag{6.6}$$

Un facteur d'échelle $\frac{e^{\tau_c}-1}{e^{\tau_c}+1}$ est employé pour avoir $d_{\mathbb{D}^n}(a, c) = \tau_c$, comme fait par SALA et al., 2018.

On peut remarquer sur la Figure 6.2a qu'on projette récursivement des graphes étoiles, d'où le nom de la méthode. Le nom est aussi métaphorique, signifiant l'écart

entre la théorie (distortion optimale) et la pratique (problèmes de complexité numérique), tout comme les astres que l'on peut observer exister sans jamais les atteindre.

Pour différencier plus facilement les successeurs d'un nœud de son parent, il convient d'assurer que leur norme euclidienne est plus grande que celle du parent. Pour cela on définit un plan orthogonal au vecteur du parent dans l'espace des paramètres (en pointillé rouge sur la Figure 6.2a), et forçons les éléments à rester dans la partie de l'espace qui ne contient pas b. Pour avoir un paramétrage continue de l'espace, on utilise une réflexion, qui revient à plier l'espace sur la partie souhaitée suivant le plan orthogonal.

L'initialisation des paramètres est faite soit aléatoirement en échantillonnant $[0, 0.01] \times [0, \pi]^{n-2} \times [0, 2\pi]$, ou en utilisant l'initialisation par barycentre (voir section 6.1.2). L'ancre est fixée au centre de $\mathbb{D}^n$ soit $\vec{s} = \vec{0}$.

6.2.2 Généralisation aux autres variétés riemanniennes

On peut assez facilement généraliser la méthode ASTRE sur $\mathbb{D}^n$ à d'autres variétés riemanniennes. Il faut pour cela changer les fonctions $\text{proj}_{\mathbb{D}^n}$ et $\text{reflect}_{a\to b}$, l'espace des paramètres reste identique. On note ces deux opérateurs $\text{proj}_{\mathcal{M}}$ et $\text{reflect}^{\mathcal{M}}_{a\to b}$, pour une variété riemannienne $\mathcal{M}$.

Plusieurs manières sont possibles afin de permettre la généralisation. Le cœur de la méthode est $\text{reflect}^{\mathcal{M}}_{a\to b}$ qui permet d'appliquer les propriétés géométriques de l'espace de représentation. Jusqu'à présent, nous avons parlé d'inversion du cercle sans formuler mathématiquement l'expression, qui n'est pas fournie dans le travail de SALA et al., 2018, il faut aller voir dans le code [1]. Nous n'avons pas réussi à généraliser l'inversion du cercle directement aux autres variétés. Au lieu de cela nous proposons une autre interprétation de ce que fait $\text{reflect}^{\mathcal{M}}_{a\to b}$, afin de formuler une expression générale utilisable pour toutes les variétés riemanniennes.

Considérons un point x et un ensemble de points $(c_i)_{i\in[\![1,N]\!]}$ appartenant à une variété riemannienne $\mathcal{M}$. On veut déplacer x, en une position y de $\mathcal{M}$, en emportant le reste des points $(c_i)_{i\in[\![1,N]\!]}$ de sorte que les distances restent inchangées. La préservation des distances entre x et c_i n'est pas suffisante, il faut que les angles entre ces points soient inchangés aussi, un peu à la manière d'une translation.

On propose de passer par les plans tangents $\mathcal{T}_x\mathcal{M}$ et $\mathcal{T}_y\mathcal{M}$. En effet, le plan $\mathcal{T}_x\mathcal{M}$ est une vision plane de $\mathcal{M}$ centrée sur x. Cette vision plane contient les informations de distances et d'orientations des points sur $\mathcal{M}$. Il suffit donc de déplacer cette vision sur une autre position y et de l'appliquer. On peut utiliser l'expmap, la logmap et le transport parallèle pour cela, et on écrit

$$x, y, z \in \mathcal{M},\ \text{reflect}^{\mathcal{M}}_{x\to y}(z) = \text{expmap}^{\mathcal{M}}_{y}(\varphi^{\mathcal{M}}_{x\to y}(\text{logmap}^{\mathcal{M}}_{x}(z))) \tag{6.7}$$

1. https://github.com/HazyResearch/hyperbolics

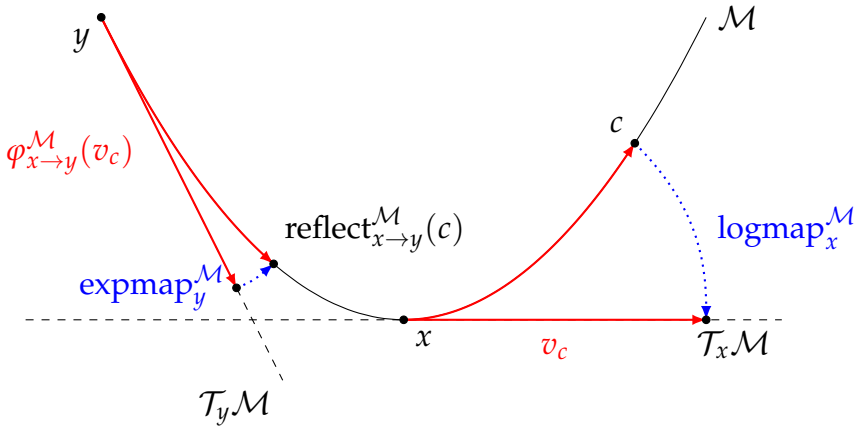


FIGURE 6.3 – Illustration de l'application de $\text{reflect}_{x\to y}$ en un point x sur une variété $\mathcal{M}$. Les vecteurs rouges ont tous la même longueur.

qui permet de réaliser les opérations décrites. L'obtention de la vision plane de la variété via $\text{logmap}^{\mathcal{M}}_x(z)$, son transport à une localisation différente via $\varphi^{\mathcal{M}}_{x\to y}$ et l'application de cette vision via $\text{expmap}^{\mathcal{M}}_y$ (voir Figure 6.3).

Avec une telle définition, $\text{reflect}^{\mathbb{R}^n}_{x\to y}$ ($\mathbb{R}^n$ étant l'espace euclidien) s'identifie à une simple translation de x à y :

$$x, y, z \in \mathbb{R}^n, \ \text{reflect}^{\mathbb{R}^n}_{x\to y}(z) = y + (z - x), \tag{6.8}$$

conférant une cohérence à la formulation. Comme projecteur, on utilise

$$(y, x) \in E, \ \text{proj}_{\mathbb{R}^n}(\Theta_y(x)) = \tau_x \cdot \text{cartesian}(\theta_{x,1}, ..., \theta_{x,n-1}). \tag{6.9}$$

Pour la variété $\mathcal{H}^{n,\beta}$, on utilise les formulations d'expmap, de logmap et de transport parallèle données par le projet geoopt (KOCHUROV, KARIMOV et KOZLUKOV, 2020). Le projecteur des paramètres vers la variété est :

$$(y, x) \in E, \ \text{proj}_{\mathcal{H}^{n,\beta}}(\Theta_y(x)) = g_\beta\left(\sqrt{\cosh^2(\tau_x) - 1} \cdot \text{cartesian}(\theta_{x,1}, ..., \theta_{x,n-1})\right). \tag{6.10}$$

On peut aussi éviter le passage par le plan tangent, dans $\mathcal{H}^{n,\beta}$, en utilisant de façon latente l'espace euclidien ou de Poincaré. Pour cela on utilise les fonctions $\text{proj}_{\mathcal{M}}$ et $\text{reflect}^{\mathcal{M}}_{a\to b}$ propres à la géométrie sous-jacente souhaitée, et on projette les

points de $\mathcal{M}$ dans $\mathcal{H}^{n,\beta}$ avec

$$x \in \mathbb{R}^n,\ g_\beta(x) = \left(\sqrt{\|x\|_2^2 + \beta}, x_1, ..., x_n\right) \in \mathcal{H}^{n,\beta}, \tag{6.11}$$

$$x \in \mathbb{D}^n,\ h(x) = g_\beta\left(\frac{2x}{1 - \|x\|_2^2}\right) \in \mathcal{H}^{n,\beta}, \tag{6.12}$$

selon qu'on utilise un espace euclidien ou de Poincaré (LAW et al., 2019). Passer par une autre géométrie permet de simplifier les calculs (passage par l'inversion du cercle plutôt que par le plan tangent), mais aussi de régler des contraintes de complexité numérique que nous détaillerons dans la section 6.3. Il est aussi important de noter que pour les Eq. 6.10, 6.11 et 6.12, la distance $d_{\mathcal{H}^{n,\beta}}$ de l'Eq. 5.4 n'est plus égale à τ_x pour des $\beta \neq 1$. On décide cependant de ne pas changer le facteur d'échelle des projections, car cela augmente beaucoup la complexité numérique rendant impossible l'optimisation. Cependant, cela empêche l'exploitation complète du modèle, en le bridant à des distances inférieurs à ce qu'elles devraient être. Cela est particulièrement inquiétant lorsque la valeur de τ est limitée à une valeur maximale. C'est un point à investiguer davantage pour savoir si c'est un problème réel.

6.2.3 Optimisation

Pour projeter les nœuds d'une hiérarchie $\mathcal{G} = (V, E)$ avec un plongement ASTRE $\mathcal{A}$, on réutilise la stratégie de NS présenté par NICKEL et KIELA, 2017 et dans la section 5.2.3.4. Les plongements ASTRE sont calculés en transformant $\mathcal{G}$ en arbre (comme expliqué précédemment). On échantillonne des liens négatifs pour un nœud $u : \mathcal{N}(u) = \{y \in V, (u, y) \notin E\} \cup \{u\}$. On utilise la fonction de perte suivante :

$$L_E(u, v, \mathcal{G}, \mathcal{A}) = -\log \frac{e^{-d(\mathcal{A}(u), \mathcal{A}(v))}}{\sum_{v' \in \mathcal{N}(u) \cup \{v\}} e^{-d(\mathcal{A}(u), \mathcal{A}(v'))}}, \tag{6.13}$$

où d est la distance dans l'espace de projection. Contrairement aux études précédentes projetant la fermeture transitive , nous utilisons ici seulement les liens directs (E) puisque nous voulons préserver l'information de distance entre les nœuds de la hiérarchie plutôt que l'hyperonymie. Les méthodes de l'état de l'art seront entraînées aussi avec E, puisqu'elles sont plus performantes sur nos métriques qui évaluent l'aspect des distances plutôt que l'hyperonymie.

En plus de l'Eq. 6.13, on ajoute un autre critère d'optimisation afin d'assurer que les successeurs soient plus proches de leur parent qu'entre eux. Cela peut survenir lorsque des petites valeurs de τ sont utilisées, puisque la distance hyperbolique croît de façon exponentielle. Quand τ est petit, il est difficile d'exploiter cette propriété car les enfants ne sont pas suffisamment placés loin du parent comparé à leurs frères. Considérons un nœud v et l'ensemble de ses successeurs $\mathcal{S}(v)$, on écrit le critère

$$L_S(v, \mathcal{G}, \mathcal{A}) = \sum_{s \in \mathcal{S}(v)} -\log \frac{e^{-d(\mathcal{A}(v), \mathcal{A}(s))}}{\sum_{s' \in \mathcal{S}(v)} e^{-d(\mathcal{A}(s), \mathcal{A}(s'))}}. \tag{6.14}$$

qui maximise la probabilité que v, le parent, soit le plus proche voisin d'un nœud s parmi $\{v\} \cup \mathcal{S}(v) \backslash \{s\}$.

Finalement, on minimise les deux critères via

$$L_{\text{taxo}}(\mathcal{G}, \mathcal{A}) = \sum_{u \in V} \sum_{v \in \mathcal{P}(u)} L_E(u, v, \mathcal{G}, \mathcal{A}) + L_S(v, \mathcal{G}, \mathcal{A}), \tag{6.15}$$

où $\mathcal{P}(u) = \{v \in V, (u, v) \in E\}$. Pendant la phase d'entraînement, une quantité fixe de nœuds est échantillonnée de l'espace négatif $\mathcal{N}(u)$. Les paramètres d'ASTRE étant dans un espace euclidien, on utilise le second schéma d'optimisation, présenté dans la section 5.2.3.3.

6.3 Expériences

On analyse la performance d'ASTRE en deux sections, traduisant deux schémas d'évaluation différents. D'abord nous montrons sa capacité à préserver les informations de la hiérarchie *via* l'évaluation de la reconstruction. Ensuite on analyse sa flexibilité en mélangeant des sources d'informations distinctes, puisque c'était l'intérêt de notre démarche de mélanger stochastique et déterministe.

6.3.1 Expériences de reconstruction

6.3.1.1 Protocole

On utilise les deux jeux de données suivants :

— SemEval-2016 task 13 (BORDEA, LEFEVER et BUITELAAR, 2016), un jeu de données multilingue contenant trois taxonomies sur trois sujets différents (environnement, alimentation et science) dans quatre langues (anglais, français, italien et néerlandais), que nous utilisons dans ce cadre de plongements de hiérarchies pour ses tailles petites et graduelles. Ces données sont originellement utilisées pour faire de l'induction de taxonomies. Ici, nous considérons seulement les taxonomies anglaises, puisque les autres taxonomies sont simplement des traductions.

— WordNet (G. A. MILLER, 1995) est une taxonomie générale sur les mots anglais, déjà utilisée dans ce domaine de plongement de hiérarchies. On utilise ici la partie contenant les nœuds transitivement connectés à *entity.n.01*, donnant lieu à une hiérarchie de 74 374 nœuds pour 75 384 liens directs.

Ces données sont des hiérarchies, c'est-à-dire que certains nœuds possèdent plusieurs parents. Le nombre de ces nœuds est limité (voir le tableau 6.1), puisque nous avons $|V| \simeq |E|$ et une seule composante connexe. C'est pourquoi, on considère que ces hiérarchies sont utilisables avec ASTRE, bien qu'il simplifie la hiérarchie en arbre. S'il y avait plus de liens comme ceux-ci, il y aurait pu y avoir une problématique autour de la préservation de ces derniers. L'évaluation se fait sur la hiérarchie complète.

TABLEAU 6.1 – Statistiques de WordNet (G. A. MILLER, 1995) et SemEval-2016 13 BORDEA, LEFEVER et BUITELAAR, 2016.

Jeu de données		$\|V\|$	$\|E\|$	profondeur
SemEval	environnement	261	261	5
	science	452	465	4
	alimentationn	1 556	1 587	5
WordNet (débutant à *entity.n.01*)		74 374	75 834	19

À des fins de comparaison, l'évaluation est conduite sur cinq méthodes de l'état de l'art. Quatre méthodes sont stochastiques : PoincaréModel (NICKEL et KIELA, 2017), LorentzModel (NICKEL et KIELA, 2018, l'adaptation de PoincaréModel à la variété riemannienne de Lorentz $\mathcal{L}$, voir section 1.2.2.2), HypCones (GANEA, BECIGNEUL et HOFMANN, 2018a) et Law19a (LAW et al., 2019) ; pour lesquelles on réutilise les mêmes hyper-paramètres que ceux utilisés à l'origine. On utilise une méthode déterministe : HyperE (SALA et al., 2018). Pour HyperE, on travaille avec une précision de 64 bits et on évalue plusieurs valeurs de $\tau \in \{0.5, 1, 1.5, 2, 2.5, 3\}$. Pour toutes ces méthodes, on utilise les implémentations officielles. La seule différence est que nous projetons le graphe direct plutôt que sa fermeture transitive, car cela donne de meilleurs résultats sur nos métriques.

Pour ASTRE, on propose d'utiliser trois variantes différentes :

— FreeASTRE qui consiste à l'optimisation libre de tous les paramètres du modèle,

— RestrictedASTRE qui limite les paramètres τ des nœuds dans un intervalle $[0, \tau_{\max}]$,

— FixedASTRE qui fixe le paramètre τ à une valeur donnée $\tau_{\max}$.

Aussi, on augmente graduellement les contraintes sur le modèle pour observer leurs effets. C'est aussi pour rendre la comparaison entre ASTRE et HyperE équitable, puisqu'HyperE est limité par la valeur τ alors que FreeASTRE ne l'est pas.

On entraîne les plongements ASTRE pendant 100 époques, prenant des lots (*batch*) de 20 nœuds et 10 exemples négatifs par lien. On utilise l'optimisation Adam avec un taux d'apprentissage de 0.02. Pour les 10 premières époques, on réduit le taux d'apprentissage à 0.002, comme fait par NICKEL et KIELA, 2017. Cette phase initiale permet de trouver les bons paramètres angulaires pour nos plongements. Après cela, le taux d'apprentissage est progressivement réduit en repartant de 0.02. L'initialisation avec les barycentres (voir section 6.1.2) est aussi utilisée pour cette raison.

On utilise trois combinaisons de variétés riemanniennes et de distances :

— **Euclidean** : $(\mathbb{R}^n, d_{\mathbb{R}^n} = \|\cdot\|_2)$, l'espace euclidien standard avec la distance L2.

— **Poincaré** : $(\mathbb{D}^n, d_{\mathbb{D}^n})$, l'espace de Poincaré avec la distance de Poincaré (utilisée aussi par GANEA, BECIGNEUL et HOFMANN, 2018a ; NICKEL et KIELA, 2017).

— **Squared Lorentzian** : $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$, est une combinaison particulière de variété et de distance. On utilise cette combinaison parce que nous remarquons qu'elle

est très pratique, en comparaison de celle utilisant $\mathcal{H}^{n,\beta}$ directement. Sur de gros jeux de données, on a été confronté à des gradients énormes, à cause de la profondeur. On a trouvé plus facile de contrôler l'explosion des gradients en limitant artificiellement les vecteurs dans une boule de rayon $1-\varepsilon$ et d'utiliser $\mathbb{D}^n$. Enfin, cette combinaison est aussi très adaptée à la visualisation des résultats, puisqu'on obtient directement des vecteurs dans $\mathbb{D}^n$ (NICKEL et KIELA, 2018). Les vecteurs de $\mathbb{D}^n$ sont projetés dans $\mathcal{H}^{n,\beta}$ (voir Eq. 6.12), afin de calculer $d_{\mathcal{H}^{n,\beta}}$. β est fixé à 0.01.

6.3.1.2 Mesures d'évaluation

On utilise des métriques similaires aux études précédentes : la *Mean Average Precision* (MAP) sur les liens directs (vu chez NICKEL et KIELA, 2017 ; GANEA, BECIGNEUL et HOFMANN, 2018a). Il est important de noter qu'ici on calcule la MAP des liens directs au lieu de la MAP des liens transitifs, puisque nous sommes plutôt intéressés par la préservation des distances de la hiérarchie.

De plus, on propose deux autres métriques pour faire ressortir les autres aspects des plongements ASTRE. En effet, la MAP indique la qualité local du voisinage, et on cherche à avoir une mesure plus globale. Pour cela, on propose d'utiliser une mesure basée sur l'extraction d'arbre dans des espaces hyperboliques, comme proposée par MONATH et al., 2019. Cela consiste à retrouver les liens directs entre les nœuds en se basant sur des heuristiques des distances hyperboliques. Ainsi, on considère comme l'ancêtre d'un nœud, son plus proche voisin de norme inférieure. On peut reconstruire un arbre à partir de l'espace en utilisant ce principe et le comparer à l'arbre original (ici une hiérarchie). Le F1-score, entre l'arbre reconstruit et la hiérarchie, qu'on appelle *F1-score de l'arbre* (faisant référence à l'arbre reconstruit) permet de mesurer à la fois la qualité des liens trouvés (faux positifs) et la quantité de liens retrouvés ou manquants (faux négatifs).

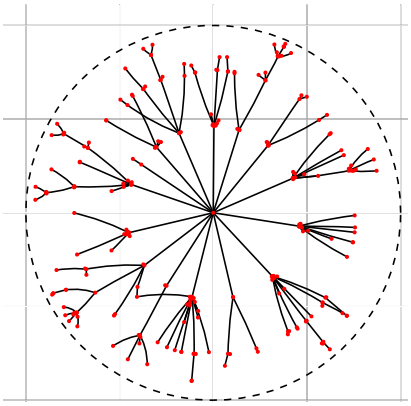
La seconde métrique vise à mesurer la distortion de la hiérarchie au plongement. On propose de mesurer la distance moyenne de la hiérarchie dans le voisinage d'un nœud. Soit k-nn(x) l'ensemble des k plus proches voisins d'un nœud x dans le plongement, et $d_{\mathcal{G}}$ la distance dans la hiérarchie, on écrit

$$d@k = \frac{1}{k|V|} \sum_{x \in V} \sum_{y \in k\text{-nn}(x)} d_T(x, y), \tag{6.16}$$

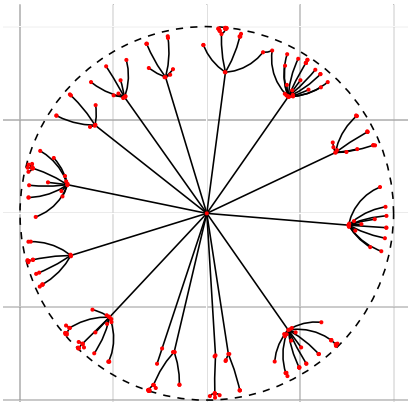
où V est l'ensemble des nœuds. Quand $d@k$ est bas, cela signifie que les voisins d'un nœud dans le plongement sont aussi proches dans la hiérarchie. Cela mesure donc la concordance entre les distances d'un espace métrique à l'autre.

6.3.1.3 Résultats

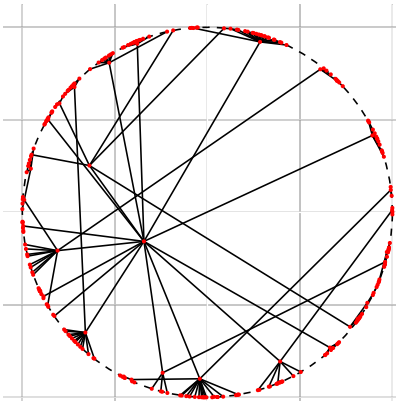
Nous proposons d'abord une analyse qualitative des plongements ASTRE (plus particulièrement RestrictedASTRE). La Figure 6.4 illustre dans $\mathbb{D}^2$ des plongements réalisés avec RestrictedASTRE, PoincaréModel, HypCones et Law19a. On remarque



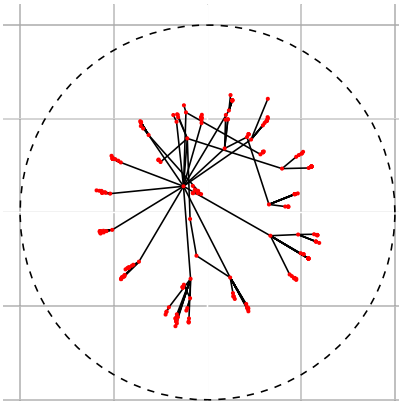
(A) RestrictedASTRE avec $\tau_{\max} = 1$.



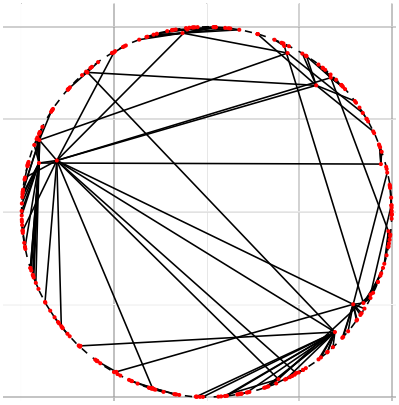
(B) RestrictedASTRE avec $\tau_{\max} = 2$.



(C) PoincaréModel (NICKEL et KIELA, 2017).



(D) HypCones (GANEA, BECIGNEUL et HOFMANN, 2018a).



(E) Law19a LAW et al., 2019, projeté dans $\mathbb{D}^2$.

FIGURE 6.4 – Méthodes de plongements de hiérarchies sur la taxonomie environnement de la tâche 13 de SemEval-2016 (BORDEA, LEFEVER et BUITELAAR, 2016), dans $\mathbb{D}^2$.

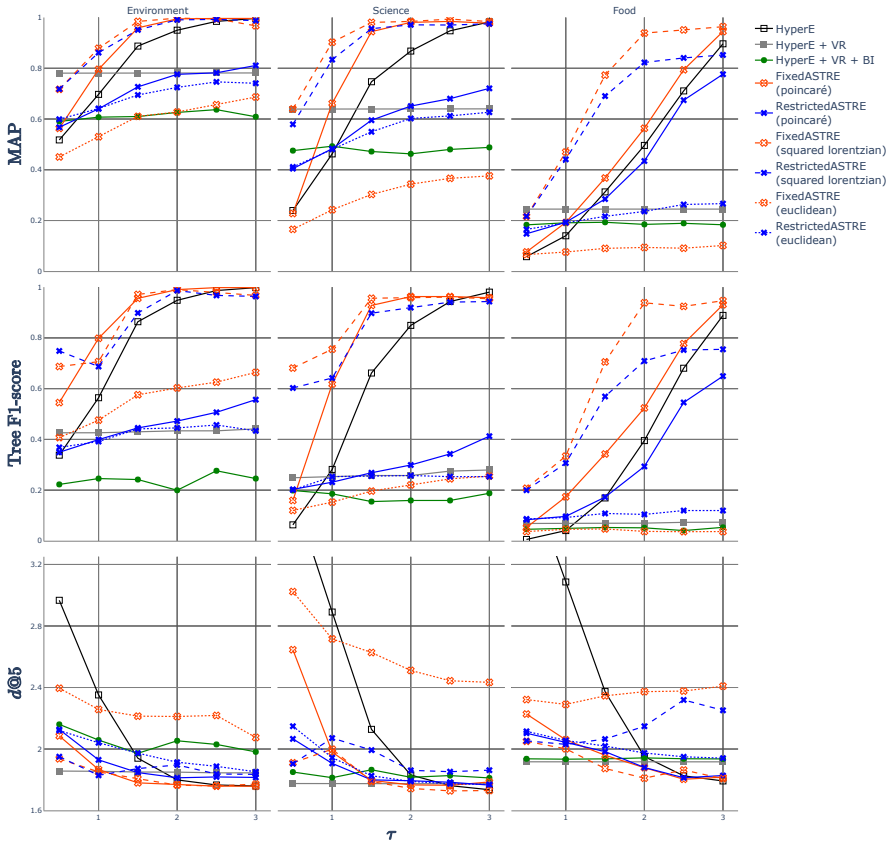


FIGURE 6.5 – MAP, F1-score de l'arbre, et d@5 de plongements ASTRE et d'HyperE (et variantes) sur les taxonomies en anglais de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016). La dimension des plongement vaut 5.

en premier pour les Figures 6.4c et 6.4e, à la fois que les nœuds sont poussés au bord du disque de Poincaré, et qu'il y a énormément d'intersections entre les branches de la hiérarchie, pouvant détériorer la distortion. Sur la Figure 6.4d, on peut observer que les cônes hyperboliques réduisent l'intersection entre les branches, cependant, le volume semble inexploité.

En comparaison, les plongements générés avec RestrictedASTRE sont plus simples à visualiser. En particulier, on détecte les branches individuellement ce qui tend à montrer une meilleure préservation de la structure hiérarchique. Aussi pour ASTRE, on peut contrôler le volume d'occupation du plongement avec le paramètre $\tau_{\max}$, pouvant aider à la visualisation des hiérarchies, tout en permettant aussi de calculer des représentations avec une précision numérique faible. Enfin, en comparaison des plongements HyperE de la Figure 5.4b avec la même valeur de τ, notre plongement semble beaucoup moins enchevêtré.

Maintenant nous regardons les résultats quantitatifs d'ASTRE par rapport aux autres méthodes de l'état de l'art.

SemEval 2016 - Tâche 13 (BORDEA, LEFEVER et BUITELAAR, 2016) : d'abord, ASTRE est comparé avec des approches déterministes (HyperE et variantes) sur les trois taxonomies de SemEval. Les résultats sont reportés sur la figure 6.5. Sur cette figure, on remarque la dépendance entre τ et la performance de la méthode.

Notre première variante HyperE+VR est plus efficace qu'HyperE quand τ est petit, comme attendu, puisqu'on alloue différemment le volume $\mathbb{D}^n$. Quand τ augmente, le volume accessible à HyperE augmente exponentiellement, ce qui rend la stratégie d'allocation moins pertinente.

Notre seconde variante HyperE+VR+BI a la même attitude avec une efficacité moins bonne que la première variante. L'écart de performance est principalement dû au fait que les branches n'ont pas une forte interaction entre elles, et peuvent occuper le même volume (voir la figure 6.1c). Cette variante a l'air tout de même être une bonne manière d'initialiser les paramètres pour ASTRE.

RestrictedASTRE et FixedASTRE sont entraînés sur trois combinaisons différentes de variétés riemanniennes et de distance. On observe que la combinaison euclidienne est la moins efficace, surtout lorsque τ augmente. Cela est dû au fait que les espaces hyperboliques bénéficient d'un plus grand τ, puisque le volume est exponentiellement croissant, là où les espaces euclidiens croissent linéairement. À l'opposé, la combinaison *squared lorentzian* est la plus efficace, elle atteint une efficacité plateau avant les autres méthodes en terme de τ.

Il est important de noter que la différence entre HyperE et ASTRE s'intensifie alors que la hiérarchie s'agrandit, ce qui est expliqué par l'optimisation du volume occupé par ASTRE, excepté pour la combinaison Poincaré. Cette attitude spécifique est potentiellement expliquée par le processus d'optimisation. En effet, dans notre protocole, on place les nœuds sans descendance très proches de leur parent (voir la Figure 6.4a), ce qui peut créer de la confusion sur la MAP. L_S (Eq. 6.14) est ajouté pour pallier cet effet.

Finalement, quand τ s'agrandit, HyperE renvoie les meilleurs résultats. Cependant, quand la hiérarchie s'agrandit, la valeur de τ pour laquelle HyperE obtient de meilleurs résultats qu'ASTRE augmente (voir Figure 6.5). Soit τ_0 la valeur de τ à partir de laquelle HyperE dépasse les performances d'ASTRE. Sur de grosses hiérarchies, τ_0 sera nécessairement très grand, requérant une haute précision numérique pour construire les vecteurs. Sur WordNet par exemple, il est impossible de calculer une représentation à partir $\tau = 4.5$ (environ) avec 64 bits de précision, et les performances stagnent ou décroissent peu avant cette valeur. En principe, τ_0 reste hors d'atteinte, car la précision numérique requise est irréalisable. Une autre solution est alors d'augmenter la dimension de l'espace, réduisant alors τ_0 pour ces dimensions (voir annexe A), mais augmentant la complexité. De plus, les différences de performances entre ASTRE et HyperE sont faibles à ce point, peu importe la mesure, ce qui fait d'ASTRE le meilleur compromis entre efficacité et complexité. De plus, ASTRE ajoute de la flexibilité par rapport à HyperE, car on peut l'utiliser avec n'importe quel critère d'optimisation (voir section 6.3.2).

TABLEAU 6.2 – MAP, F1-score de l'arbre et d@5 d'ASTRE, HyperE (et variantes) et approches stochastiques sur les hiérarchies Science (Sci.) et Alimentation (Al.) de SemEval 2016 tâche 13 (BORDEA, LEFEVER et BUITELAAR, 2016). τ est choisi via les résultats sur la hiérarchie *environnement*, pour les méthodes requérant une valeur (mises en valeur avec *). La dimension vaut 5. Les deux meilleurs résultats sont en gras.

	Méthodes	MAP		F1-score de l'arbre		d@5	
		Sci.	Al.	Sci.	Al.	Sci.	Al.
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.9938**	**0.9638**	0.9607	**0.9465**	**1.7279**	**1.8073**
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.9751	0.8524	0.9432	0.7549	1.8624	2.2517
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.9731	0.8496	0.9323	0.7594	1.9712	2.3459
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9846	0.9448	0.9629	0.9306	1.7642	1.8171
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7208	0.7765	0.4127	0.6493	1.7655	1.8276
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6138	0.3841	0.2751	0.1292	1.7664	1.8458
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.3758	0.1016	0.2555	0.0369	2.4336	2.4087
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.6268	0.2666	0.2533	0.1197	1.7695	1.9407
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.6099	0.2691	0.2402	0.1018	1.7695	1.9208
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9825	0.8964	**0.9803**	0.8886	**1.7354**	**1.7924**
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6400	0.2450	0.2795	0.0738	1.7743	1.9167
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4928	0.1931	0.1856	0.0528	1.8137	1.9360
Approches Stochastiques	PoincaréModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9602	**0.9800**	0.9640	**0.9605**	2.1051	2.4010
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.9342	0.9341	0.8603	0.8638	1.7934	1.8519
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1982	0.0697	0.1354	0.0433	1.9150	1.9968
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	**0.9923**	0.9217	**0.9651**	0.8275	1.8916	2.6427

Dans le tableau 6.5, on retourne les résultats par rapport aux approches stochastiques. Pour les méthodes nécessitant un τ (FreeASTRE, RestrictedASTRE, HyperE et variantes), on fixe la valeur en se basant sur les meilleurs résultats sur la MAP pour la taxonomie environnement. Dans ce tableau, on voit clairement qu'HyperE et ASTRE (avec la combinaison Squared Lorentzian) retourne des résultats similaires. Le retrait des restrictions sur le paramètre τ semble corrélé avec d@5, ce qui s'explique par le fait qu'on s'éloigne de la construction de SARKAR, 2011. On remarque aussi que les approches stochastiques peuvent donner des résultats compétitifs sur la MAP et le F1-score de l'arbre, mais de mauvais résultats sur d@5, potentiellement expliqués par les mêmes raisons. On explique les mauvais résultats d'HypCones par le fait que la méthode optimise la fonction d'énergie plutôt que la distance (utilisée ici pour l'évaluation). Globalement sur SemEval, nos propositions sont comparables sur la MAP et le F1-score de l'arbre, et augmente la qualité du d@5.

WordNet (G. A. MILLER, 1995) est une hiérarchie plus grande (voir tableau 6.1) utilisée pour révéler les différences de passage à l'échelle entre l'ensemble des méthodes. Pour cette hiérarchie, d@5 est calculé sur 1 000 nœuds aléatoires. Dans la Figure 6.5, on observe de faibles différences en terme de MAP et de F1-score de l'arbre entre les approches ASTRE et l'algorithme HyperE original. Quand τ est petit, une plus grande différence est observable pour d@5. Ceci peut s'expliquer par l'optimisation du volume procuré par ASTRE, on obtient une meilleure performance quand la précision numérique requise est moins grande. Aussi, pour les mêmes raisons,

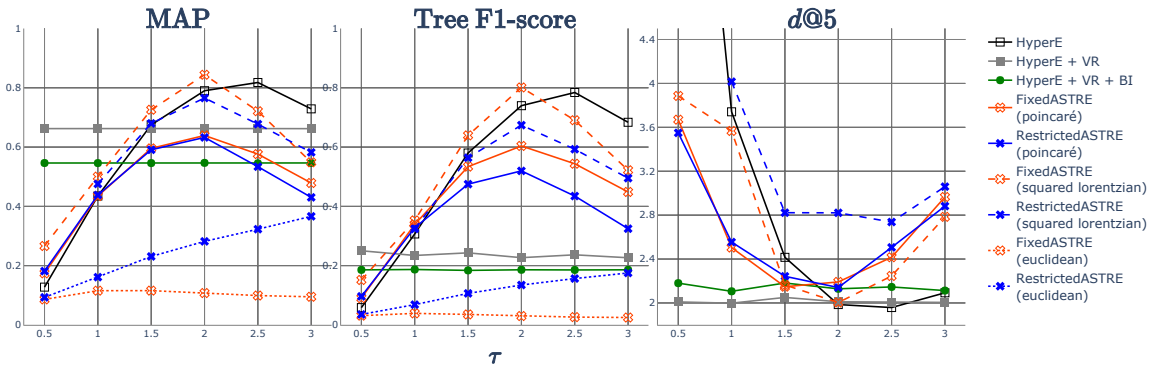


FIGURE 6.6 – MAP, F1-score de l'arbre, et d@5 de plongements ASTRE et d'HyperE (et variantes) sur WordNet (G. A. MILLER, 1995). La dimension des plongement vaut 5.

il est important de remarquer que la performance maximale pour ASTRE, avec la combinaison Squared Lorentzian, dépasse toutes les autres méthodes.

On compare dans le tableau 6.3, l'ensemble des méthodes, en particulier, on inclut les approches stochastiques. Pour les méthodes requérant une valeur τ, on la choisit, comme précédemment, avec les résultats de la MAP sur la taxonomie *environnement* de SemEval. Nous observons que les versions d'ASTRE exploitant des distances hyperboliques ($d_{\mathcal{H}^{n,\beta}}$ et $d_{\mathbb{D}^n}$) dépassent les approches stochastiques de l'état de l'art. En terme de MAP, dans les meilleures conditions, ASTRE double quasiment leur efficacité. On remarque aussi que pour la combinaison de Poincaré, nous avons des d@5 bien plus faibles que les approches stochastiques, évoquant une meilleure préservation de la structure de la hiérarchie. On donne les résultats pour d'autres dimensions dans l'annexe B.

6.3.2 Expériences de fusion

Nous avons précédemment mentionné la flexibilité d'ASTRE pour combiner des sources d'information différentes, sans réellement justifier ce point parce qu'il paraît naturel. En effet, ASTRE est contraint par la construction de SARKAR, 2011, mais, il donne la possibilité d'optimiser des paramètres de la construction pour adapter les plongements à la hiérarchie. L'impact direct de cette paramétrisation est que l'on peut incorporer d'autres sources d'information que la structure de la hiérarchie, en l'ajoutant comme un nouveau critère de la fonction d'optimisation, contrairement à HyperE. Les attributs des nœuds, la similarité entre ces nœuds pourraient être intégrés dans les plongements tout en préservant la structure *via* les contraintes de la construction.

Les autres méthodes stochastiques peuvent aussi être utilisées pour combiner des sources de données. On propose donc ici de regarder les différences entre ces algorithmes et ASTRE lorsqu'on cherche à fusionner une source d'information hiérarchique avec des informations sur les nœuds de cette hiérarchie (comme leur similarité sémantique).

TABLEAU 6.3 – MAP, F1-score de l'arbre et d@5 d'ASTRE, HyperE (et variantes) et approches stochastiques sur WordNet (G. A. MILLER, 1995). τ est choisi via les résultats sur la hiérarchie *environnement*, pour les méthodes requérant une valeur (mises en valeur avec *). La dimension vaut 5. Les deux meilleurs résultats sont en gras.

	Méthodes	MAP	F1-score de l'arbre	d@5
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.8440**	**0.8012**	**2.0038**
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.7655	0.6741	2.8212
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.4956	0.3562	4.8002
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6391	0.6038	2.1914
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6325	0.5197	2.1406
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6094	0.3820	2.2672
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.1156	0.0354	5.1372
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.3663	0.1753	4.7988
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.5214	0.2318	2.5000
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.8179**	**0.7841**	**1.9582**
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6625	0.2269	2.0108
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5467	0.1860	2.1288
Approches Stochastiques	PoincaréModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4342	0.1714	2.7078
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.4177	0.1582	2.6764
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1207	0.0663	2.6836
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.4647	0.1747	3.9186

6.3.2.1 Protocole

Pour réaliser la comparaison, on propose d'utiliser deux modèles différents :

— FreeASTRE, étant le modèle ASTRE débridé sur ses paramètres,

— Free$\mathcal{M}$, un modèle de plongement complètement libre sur une variété riemannienne $\mathcal{M}$ optimisé avec la SGD riemannienne de la section 5.2.3.2,

pour faire la fusion d'une hiérarchie et de données sémantiques. On entend par modèle complètement libre, un plongement où les vecteurs sont des éléments de $\mathcal{M}$ et sont directement optimisés sans contrainte (de la même manière que PoincaréModel et LorentzModel). En comparaison FreeASTRE possède des paramètres euclidiens pour calculer les vecteurs de $\mathcal{M}$, qui sont contraints par la construction de SARKAR, 2011. On utilise les trois variétés suivantes : $\mathbb{R}^n$ (euclidienne), $\mathbb{D}^n$ (poincaré) et $\mathcal{H}^{n,\beta}$ (hyperboloïde avec distance Squared Lorentzian).

Pour obtenir des informations sémantiques sur les nœuds, on fait intervenir des vecteurs FastText [2], introduit par BOJANOWSKI et al., 2017. La similarité cosinus (Eq. 2.5) permet de calculer la similarité sémantique entre les nœuds. On obtient la représentation vectorielle d'un nœud dans FastText en prenant la moyenne des mots composant son étiquette.

À partir d'une similarité $s(x, y)$ entre deux nœuds x et y, NICKEL et KIELA, 2018 proposent une approche pour incorporer les similarités d'un élément dans son vecteur, qu'on réutilise ici :

$$L_{\text{sim}}(s, V, m) = \sum_{x \in V} -\log \frac{e^{-d(m(x), m(y_{\max}))}}{\sum\limits_{y' \in \text{CH}(x)} e^{-d(m(x), m(y'))}}, \tag{6.17}$$

avec

$$y_{\max} = \underset{y \in \text{CH}(x)}{\text{argmax}}\ s(x, y), \tag{6.18}$$

V l'ensemble des nœuds, $\text{CH}(x)$ est l'ensemble des co-hyponymes de x, et m est le modèle de plongement (FreeASTRE ou Free$\mathcal{M}$). Comme le nombre de co-hyponymes est différents pour chaque nœud, on échantillonne aléatoirement deux nœuds dans $\text{CH}(x)$. Un plus grand nombre d'échantillons pourrait potentiellement améliorer les résultats, mais ce n'est pas notre but ici, parce qu'on souhaite seulement évaluer la fusion de données.

L_{sim} est supposée ordonner les co-hyponymes de façon à ce que le co-hyponyme le plus similaire pour un nœud x, selon s, soit aussi le co-hyponyme le plus proche parmi l'ensemble des co-hyponymes, en terme de distance dans le plongement. On combine ce critère avec le critère précédent de l'Eq. 6.13 imposant la structure de la hiérarchie $\mathcal{G} = (V, E)$:

$$L(s, \mathcal{G}, m) = \alpha L_{\text{sim}}(s, V, m) + (1 - \alpha) \sum_{u \in V} \sum_{v \in \mathcal{P}(u)} L_E(u, v, \mathcal{G}, m), \tag{6.19}$$

2. https://fasttext.cc/docs/en/english-vectors.html

avec α un paramètre équilibrant l'intégration de l'un ou l'autre des critères. On étudie trois valeurs d'α : $\alpha = 0$ qui consiste à plonger seulement la source hiérarchique, $\alpha = 1$ qui revient à considérer seulement la source sémantique, et $\alpha = 0.5$ qui intègre les deux équitablement. On fixe la dimension des vecteurs à 5, on entraîne pendant 100 époques, et on utilise la même stratégie d'apprentissage que dans la section 6.3.1. On enlève la partie L_S, de l'Eq. 6.14, du critère global de l'Eq. 6.15, car L_S est directement opposée à L_{sim}. On utilise 19 sous-hiérarchies indépendantes de WordNet contenant entre 1 000 et 1 500 nœuds.

6.3.2.2 Mesures d'évaluation

On ajoute deux métriques pour mesurer la qualité d'ordonnancement des co-hyponymes : le rang moyen (*Mean Rank*, MR) du co-hyponyme le plus similaire dans la liste des co-hyponymes triée par la distance de l'espace ; et la MAP@k calculée sur la liste des co-hyoponyme triée, en prenant les k plus similaires comme positifs (seulement quand la liste contient strictement plus de k éléments). Pour continuer à mesurer la préservation de la hiérarchie, on préserve deux métriques concernant cet aspect : la MAP des liens directs, et le F1-score de l'arbre.

6.3.2.3 Résultats

On affiche sur la Figure 6.7 le résultat moyen de chaque métrique sur les 19 sous-hiérarchies. Les axes à droite et en haut sont attachés à la mesure de la qualité de l'incorporation des informations hiérarchiques, là où les axes à gauche et en bas correspondent à la qualité d'incorporation des informations sémantiques. Il faut d'abord noter que comme WordNet et FastText ne sont pas des sources totalement indépendantes, les résultats sur MAP@5 sont non nuls même quand $\alpha = 0$.

Quand $\alpha = 0$, seule la hiérarchie est plongée, et on observe une différence limitée entre les deux modèles pour $\mathbb{R}^n$ et $\mathbb{D}^n$. Pour $\mathcal{H}^{n,\beta}$, la différence est plus grande, en faveur d'ASTRE. Comme $\mathcal{H}^{n,\beta}$ impose plus facilement l'infériorité de la norme des parents que les deux autres, il est possible que les liens directs sont plus facilement identifiables. De plus, comme L_S est enlevée du critère d'optimisation, on aura plus tendance à confondre les co-hyponymes avec son parent.

Quand $\alpha = 1$, seule la source sémantique est utilisée pour calculer le critère. Comme ASTRE calcule les vecteurs des nœuds *via* la hiérarchie, il obtient une meilleure performance sur les aspects hiérarchiques que le modèle libre même quand aucune information sur la hiérarchie n'est imposée lors de l'optimisation. En comparaison, le modèle libre a une performance nulle sur ces aspects puisqu'aucune information n'est donnée. Cela montre que même lorsque la hiérarchie n'est pas fournie directement, ASTRE parvient (via les contraintes sur ses vecteurs) à la modéliser, illustrant un point fort d'ASTRE. Concernant les mesures de similarité, le modèle libre produit des résultats légèrement meilleurs. On explique cela par le fait que ce modèle n'a aucune contrainte.

Quand $\alpha = 0.5$, on essaie d'intégrer également les aspects sémantiques et hiérarchiques. Pour les deux types de modèles, on observe une baisse de performance

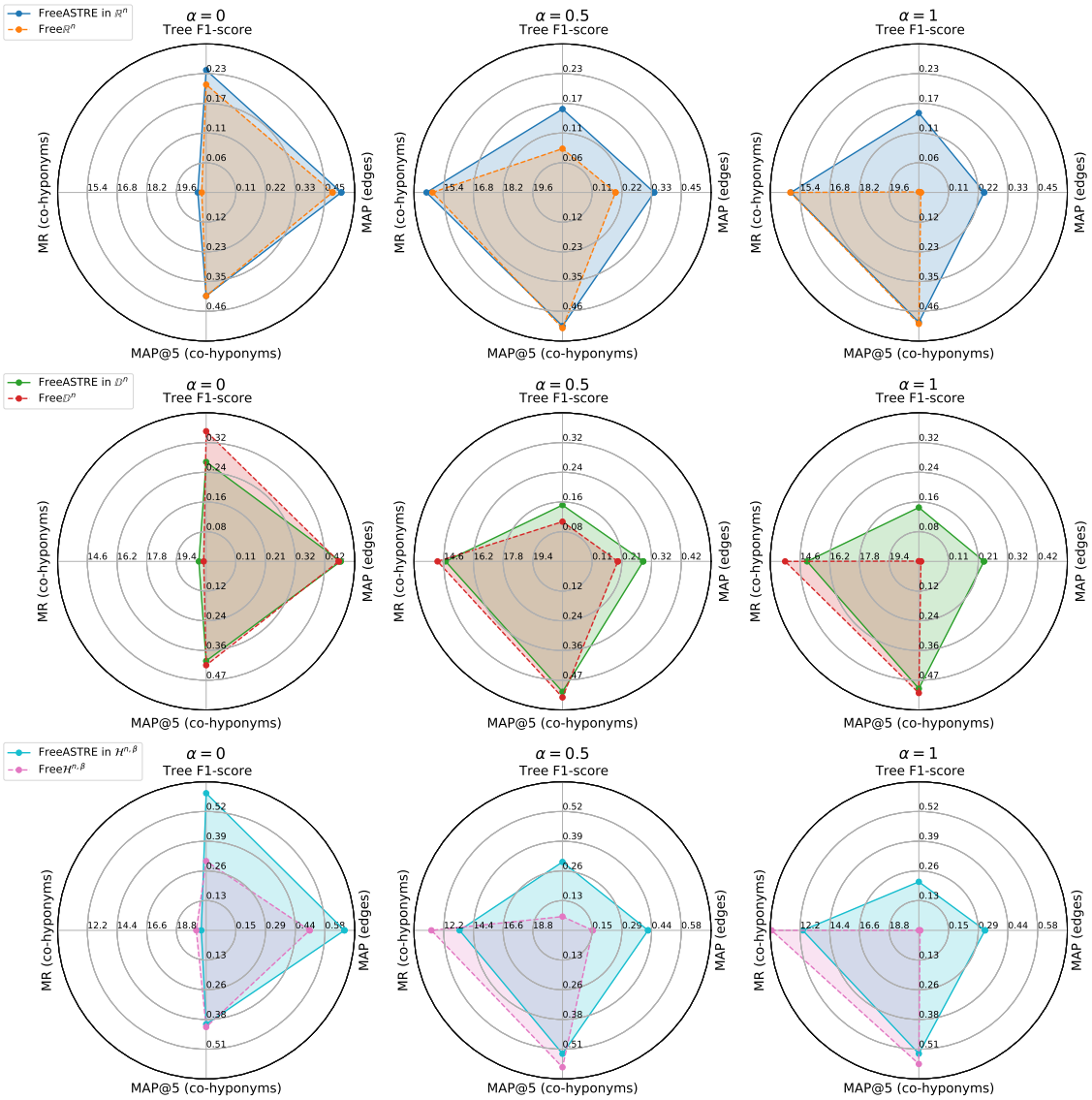


FIGURE 6.7 – Comparaison entre FreeASTRE et Free$\mathcal{M}$ sur 19 sous-hiérarchies de WordNet (on prend le score moyen sur chaque axe), et 3 variétés différentes (de haut en bas : $\mathbb{R}^n$, $\mathbb{D}^n$, and $\mathcal{H}^{n,\beta}$), dans la tâche de fusion sémantique et hiérarchique. L'incorporation de l'information sémantique augmente avec α (de gauche à droite), pendant que l'information hiérarchique décroît.

sur les mesures hiérarchiques, par rapport au cas $\alpha = 0$ (où aucune information sémantique n'est utilisée). Au contraire, quasi aucune différence n'est visible pour les mesures sémantiques par rapport au cas $\alpha = 1$. On remarque cependant que les modèles FreeASTRE sont plus polyvalents que les modèles libres. Même si FreeASTRE est légèrement moins performant sur les mesures sémantiques, de l'autre côté, la perte de qualité des informations hiérarchiques est bien plus grande pour le modèle libre, et ne semble pas compenser cette faible supériorité sémantique.

Globalement, les modèles ASTRE semblent être mieux adaptés à combiner des flux d'informations différents, particulièrement lorsqu'un est hiérarchique, soulignant l'aspect flexible et polyvalent d'ASTRE.

6.4 Conclusion

Nous avons proposé une nouvelle méthode de représentation nommé ASTRE, combinant les forces des algorithmes déterministes et stochastiques, qui permet le calcul de plongements hyperboliques plus polyvalents. Aussi, lorsque la précision numérique est basse, ASTRE permet de produire des plongements de bien meilleure qualité.

Les pistes ouvertes par ASTRE sont nombreuses. D'abord, la généralisation de cette méthode à des graphes quelconques. Ensuite le mélange entre connaissances hiérarchiques partielles et données déstructurées pour tenter d'étendre des hiérarchies. Enfin l'usage de ces nouvelles représentations pour des tâches hiérarchiques au lieu des plongements actuels.

La partie la plus prometteuse est peut-être la seconde, concernant le mix entre connaissances hiérarchiques et autres données. On peut posséder des connaissances sous forme d'une hiérarchie et des entités quelconques. Ces dernières ne forment pas une structure particulière, mais peuvent être comparées à des nœuds de la hiérarchie grâce à une similarité fournie. En exploitant cette similarité, on peut définir des vecteurs libres dans l'espace de Poincaré, et chercher à optimiser leur position par rapport à la hiérarchie modélisée avec ASTRE. Cela permettrait de positionner ces entités dans la hiérarchie et de l'enrichir, puis d'itérer.

La découverte et l'enrichissement de connaissances sont des enjeux des chapitres suivants. Dans lesquelles nous essayons d'extraire des hiérarchies d'espaces hyperboliques. Ces derniers pouvant idéalement être construits à partir de caractéristiques sur les nœuds de la hiérarchie à construire.

Chapitre 7

Extraction d'arbre dans des plongements hyperboliques

Sommaire

7.1 Besoins 128
7.2 Etat de l'art 129
7.2.1 Historique des méthodes 129
7.2.2 Composants principaux 131
7.3 Méthodologie proposée 134
7.3.1 Combinaison de *features* syntaxiques et lexicales 134
7.3.2 Plongement des termes 136
7.3.3 Extraction d'arbre 137
7.3.4 Raffinage de la taxonomie 138
7.4 Expériences 138
7.4.1 Protocole 138
7.4.2 Evaluation 139
7.4.3 Reconstruction de taxonomies 141
7.4.4 Induction de taxonomie 142
7.5 Conclusion 145

Avec ASTRE, nous avons évalué la qualité de projection de données hiérarchiques et sémantiques. On a alors produit des plongements uniquement sémantiques dans des espaces hyperboliques, en exploitant une fonction de similarité entre les éléments. Il est possible d'étendre cette méthode à grande échelle (plusieurs dizaines de milliers de mots), et nous le faisons avec notre logiciel HierarX (voir C.2). Ce logiciel applique la méthode de plongements à partir de similarité de NICKEL et KIELA, 2018, et a été élaboré pour compenser le manque d'une implémentation efficace de la méthode.

Il est alors possible de produire des plongements hyperboliques révélant la hiérarchie sous-jacente aux éléments considérés, uniquement à partir d'une similarité, pourvu qu'elle dispose des bonnes propriétés. Par exemple, l'ensemble de données historique sur les langues, fourni par BOUCKAERT et al., 2012, permet de définir une similarité entre les langues en se basant sur les mots d'origine commune. On peut ainsi produire un plongement hyperbolique dont la structure évoque un arbre d'évolution des langues (au centre, on trouve les langues les plus anciennes). Par conséquent, il paraît possible de proposer une telle similarité pour n'importe quel type de données, et ainsi d'obtenir de tels plongements, reflétant une hiérarchie cachée dans les données. Il reste ensuite une phase d'interprétation, que nous abordons ici, qui consiste à retrouver la hiérarchie de façon discrète, en spécifiant les liens entre les éléments. De cette façon, on pourrait construire des hiérarchies à partir de n'importe quelle source de données.

Dans ce chapitre, on parle de taxonomie au lieu de hiérarchie, mais il s'agit du même objet. C'est simplement le terme utilisé dans le domaine plus restreint de l'induction de taxonomie, où le but est de créer des hiérarchies à partir de données textuelles.

7.1 Besoins

L'avantage des données structurées, ou hiérarchisées, formalisées sous forme d'un KG ou d'une hiérarchie, est qu'elles peuvent s'intégrer dans de nombreux systèmes d'Intelligence Artificielle (IA), et améliorer leur pertinence et performance. Le chapitre 3 donne un exemple d'usage de données sans structures particulières (hormis leur type) dans un système de *slot filling*, et, on peut facilement imaginer l'extension à des données structurées, anticipant un gain potentiel sur la performance. Il est connu que d'autres tâches, comme la recherche d'information, la réponse à des questions et l'extension de requêtes (CLEUZIOU et al., 2015 ; MAO, REN et al., 2018 ; C. WANG, HE et ZHOU, 2017), profitent de l'inclusion de connaissances hiérarchiques car elles fournissent une capacité de généralisation aux solutions.

La création de taxonomies est essentielle pour la résolution et l'augmentation de la pertinence d'une vaste gamme de tâches, mais ces taxonomies restent souvent confectionnées à la main par des experts, conditionnant ces avantages à des coûts financiers ou temporels. L'intervention manuelle est principalement justifiée par le caractère *sui generis* des taxonomies requises, faisant des océans de données libres,

tels que WikiData, des sources inadaptées (pas assez spécifiques, ni couvrant le domaine souhaité) aux échelles plus fines de certaines problématiques industrielles (C. WANG, HE et ZHOU, 2017).

L'automatisation de la création de connaissances permettrait la fabrication surmesure de taxonomies, réduisant l'écart contextuel avec l'application. De telles méthodes reposent originellement sur des textes écrits en langage naturel (HEARST, 1992). On fournit un ensemble de termes V, étant les nœuds de la taxonomie, et l'objectif et de retrouver l'ensemble des liens E à partir de textes fournis (ou non), idéalement recouvrant le domaine d'usage de la taxonomie. Si les textes ne sont pas fournis, il existe des manières de se constituer un corpus, nous expliquons ce point dans les sections suivantes.

Quand le corpus est assez grand (PANCHENKO et al., 2016, $\simeq$ 10 GB), on peut créer un ensemble d'attributs (*features*) lexicaux, syntaxiques ou sémantiques, facilitant la détection ou la classification des relations entre les termes, pour enfin construire la taxonomie (PANCHENKO et al., 2016 ; ALY et al., 2019 ; CLEUZIOU et al., 2015 ; MAO, REN et al., 2018). Cette taxonomie est par la suite testée sur des applications réelles, comme des applications biomédicales ou d'e-commerce, pour évaluer le processus d'induction (C. WANG, HE et ZHOU, 2017).

Usuellement, les *features* collectées interviennent pour produire des plongements faits manuellement ou appris sur les termes. Ces plongements servent comme entrée à des heuristiques ou à des systèmes de classification, aidant à prédire les liens de la taxonomie. Nous proposons ici d'éliminer la phase de classification et de directement interpréter le plongement comme un arbre, et par la suite, comme une taxonomie. Ceci est réalisé en essayant de trouver l'arbre minimisant la distortion avec le plongement. De la même façon que, précédemment, nous créions un plongement par rapport à une hiérarchie, ici, nous voulons faire l'exact inverse. Procéder comme cela évite l'entraînement de modèles de classification, venant en supplément d'un processus avec beaucoup de couches, et tente de limiter la perte d'information. Pour résumer, la tâche à laquelle on s'intéresse est d'extraire une taxonomie à partir de plongements d'objets.

7.2 Etat de l'art

7.2.1 Historique des méthodes

L'induction de taxonomie est, rappelons-le, un problème dans lequel il faut trouver un ensemble de liens pour organiser un ensemble de termes. En général, on dispose d'un corpus de texte utilisé pour calculer les *features* des termes (ALY et al., 2019 ; BORDEA, LEFEVER et BUITELAAR, 2016), et déterminer la plausibilité des liens. On peut étendre ce corpus en inspectant des pages en ligne reliées au sujet de la taxonomie, et ramener ainsi des connaissances spécifiques. Plusieurs tâches partagées fonctionnent sur ce principe, notamment SemEval-2016 Task 13 (BORDEA, LEFEVER et BUITELAAR, 2016). Parmi les participants, TAXI (PANCHENKO et al., 2016) est la méthode la plus performante en exploitant des schémas lexico-syntaxiques

sur un corpus général (Wikipedia) et un corpus spécifique (obtenu avec le logiciel BootCat de BARONI et BERNARDINI, 2004). D'autres systèmes participèrent, comme USAAR (TAN, BOND et GENABITH, 2016) qui utilise l'enocentricité des hyponymes (voir section suivante) et JUNLP (MAITRA et DAS, 2016), une solution reposant sur BabelNet [1] et des règles morphologiques.

Les premières approches exploitèrent les *features* morphologiques des chaînes de caractères composant les termes de la taxonomie, et leur co-occurrences sur des patrons syntaxiques, connus sous le nom des patrons de Hearst (PANCHENKO et al., 2016 ; TAN, BOND et GENABITH, 2016 ; HEARST, 1992 ; C. WANG, HE et ZHOU, 2017). Les relations extraites sont ensuite filtrées avec des règles ou des systèmes entraînés de façon supervisé. Par exemple, pour TAXI, il y a deux modes, soit les relations sont sélectionnées grâce à une machine à vecteur de support (*Support Vector Machine*, SVM) entraînée sur un ensemble de relations hyperonymiques (mode supervisé), soit on fixe des seuils sur certaines *features* (mode non supervisé). Pour USAAR et JUNLP, ils utilisent principalement des règles morphologiques, et sont donc, totalement non supervisées. Un autre système nommé SubSeq (Amit GUPTA et al., 2017) se repose aussi sur des heuristiques, mais introduit la notion de sous-séquence d'hyperonymes (c'est-à-dire la modélisation des chemins entre un terme et un autre), retournant de meilleurs résultats que TAXI.

Aujourd'hui, il est rare de trouver l'utilisation de *features* distributionnelles, et elles restent donc sous-exploitées (C. WANG, HE et ZHOU, 2017 ; BORDEA, LEFEVER et BUITELAAR, 2016). En effet, elles interviennent plutôt en fin de traitement, dans l'objectif d'affiner la taxonomie construite (ALY et al., 2019). Les systèmes bout-en-bout, comme TaxoRL (MAO, REN et al., 2018), commencent à les intégrer en entrée.

TaxoRL parvient à faire l'induction de taxonomie de façon bout-en-bout grâce à un cadre d'apprentissage par renforcement. Cela lui permet d'intégrer de nombreuses *features* sur les liens et les termes, et de procéder successivement à l'ajout de liens validés par le système dans la taxonomie (commençant avec une taxonomie vide). Cependant, ce système nécessite un ensemble de taxonomies d'entraînement, souvent des sous-taxonomies WordNet (G. A. MILLER, 1995), ce que nous évitons dans ce chapitre.

Graph2Taxo (SHANG et al., 2020) est une autre approche qui prédit la matrice d'adjacence de la taxonomie, et utilise directement l'information de taxonomies externes. En d'autres termes, ils utilisent des taxonomies *a priori* connues pour créer des *features* aidant à la prédiction des liens pour les taxonomies inconnues. Ces deux contributions (TaxoRL, Graph2Taxo) ont établi un nouvel état de l'art pour l'induction de taxonomie sur la tâche 13 de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016).

Plus récemment, une proposition (CTP) a été faite pour affiner les plongements contextuels (BERT) pour de l'induction de taxonomie (C. CHEN, K. LIN et KLEIN, 2021). CTP explore l'usage des patrons syntaxiques dans des modèles linguistiques

1. https://babelnet.org/

contextuels pré-entraînés et tente de les affiner avec des sous-taxonomies de WordNet. Bien qu'il montre une amélioration sur de petites taxonomies (10 à 50 termes), il ne reste pas plus efficace sur des taxonomies moyennes ou larges (100 à 1000 termes).

La principale faiblesse des systèmes actuels est le besoin en supervision. TAXI, TaxoRL, Graph2Taxo et CTP requièrent des données d'entraînement pour fonctionner. À la place, nous proposons un système d'induction non supervisé.

Notre approche est basée sur les HyperCones (M. LE et al., 2019), qui sont des plongements hyperboliques des termes non supervisés (utilisant la méthode HypCones, voir section 5.2.3.4), se basant sur les comptes de co-occurrences des patrons de Hearst, et qui sont finalement utilisés pour la détection d'hyperonymes. La géométrie hyperbolique intervient ici car elle se montre plus efficace que la géométrie euclidienne pour représenter et préserver des informations hiérarchiques (NICKEL et KIELA, 2017 ; GANEA, BECIGNEUL et HOFMANN, 2018a, et chapitre 5, 6 et C). On construit notre approche à partir de ces idées, en les poussant plus loin, notamment parce que nous voulons produire des taxonomies complètes à partir de ces plongements plutôt que de faire simplement de la détection d'hyperonymes. En effet, la détection est une tâche plus facile car elle n'a pas à gérer la nature dirigée, acyclique et connexe des taxonomies.

Dans ce travail, on se compare essentiellement à TAXI (PANCHENKO et al., 2016), USAAR (TAN, BOND et GENABITH, 2016), JUNLP (MAITRA et DAS, 2016), et TAXI raffiné (ALY et al., 2019), une version améliorée de TAXI. Malheureusement, on ne se compare pas à SubSeq (Amit GUPTA et al., 2017), à cause de problèmes de reproductibilité (pas d'implémentation fournie). À notre connaissance, TAXI raffiné est le meilleur système d'induction de taxonomie pour SemEval 2016, qui n'utilise pas d'information sur d'autres taxonomies. Comme nous aussi, nous n'utilisons pas ce type d'information, nous ne nous comparerons pas à des systèmes comme Graph2Taxo, TaxoRL, et CTP, qui eux en utilisent.

Nous proposons dans la suite une méthode pour l'induction de taxonomie à partir d'un plongements de termes. Ces plongements sont obtenus en projetant, *grosso modo*, les comptes de co-occurrences. On emploie ensuite des algorithmes d'extraction d'arbre à partir de plongements pour produire une taxonomie, pouvant être enrichie ou non.

7.2.2 Composants principaux

On fournit dans ce qui suit les composants principaux des méthodes précédentes, permettant la compréhension de notre proposition et la résolution de la tâche.

7.2.2.1 *Feature Engineering*

L'extraction de *features* de données textuelles ou des chaînes de caractères est une étape cruciale pour l'induction de taxonomie, puisqu'elles forment un socle

commun à l'ensemble des méthodes, sur lesquelles reposent les couches successives d'interprétation et d'extraction des liens de la taxonomie.

Features lexicales. Elles comparent simplement les deux chaînes, et ne nécessitent donc aucun corpus pour être calculées, ce qui ne les empêche pas d'être particulièrement précises. Ces deux raisons font qu'elles sont largement utilisées (MAO, REN et al., 2018 ; PANCHENKO et al., 2016 ; ALY et al., 2019 ; TAN, BOND et GENABITH, 2016). En effet, l'apparence d'un terme dans un autre est un bon indicateur d'une relation *est_un* entre ces termes. Par exemple, *biologie* et *biologie moléculaire* partagent un mot commun soulignant la relation. Cet aspect est connu sous le nom d'*endocentricité des hyponymes*, et décrit par TAN, BOND et GENABITH, 2016. Généralement, un ensemble de *features* lexicales est généré à partir de correspondances entre les préfixes, les suffixes ou la taille du morceau commun le plus long. Bien que les *features* lexicales soient très précises, elles manquent de rappel puisqu'elles n'aident pas à détecter les relations pour les paires sans morphologie commune, comme pour *algèbre* et *mathématique*.

Features syntaxiques. Elles sont produites avec les patrons syntaxiques de Hearst (HEARST, 1992), qui sont passés sur un corpus pour collecter des co-occurrences entre les termes de la taxonomie et sont sensés identifier les phrases exprimant des relations génériques/spécifiques. Un ensemble de patrons est défini, comme *"x est un y"* ou *"x tel que y"*, et on compte le nombre de fois que les patrons se déclenchent pour une paire x et y. Ces *features* sont un ingrédient majeur des processus d'induction d'aujourd'hui PANCHENKO et al., 2016 ; ALY et al., 2019 ; MAO, REN et al., 2018 ; ROLLER, KIELA et NICKEL, 2018 ; M. LE et al., 2019. Cette technique est simple, explicable, passe à l'échelle et elle est efficace au regard des méthodes distributionnelles, comme rapporté par M. LE et al., 2019. Les fréquences entre les paires sont combinées en de nouvelles *features* pour essayer de normaliser et d'enlever le bruit, dans le but d'augmenter la performance. *WebIsA* (SEITNER et al., 2016) est une ressource en ligne gratuite rassemblant ces comptes sur un corpus très vaste de documents, utilisé par PANCHENKO et al., 2016 ; ALY et al., 2019, et dans notre travail, pour enrichir les *features* des termes.

Features distributionnelles. Elles sont finalement ajoutées soit avant la modélisation (MAO, REN et al., 2018), soit après pour détecter les co-hyponymes (ALY et al., 2019). Les *features* distributionnelles peut aussi être utilisées pour détecter des relations hyperonymiques, grâce la *Distributional Inclusion Hypothesis* (DIH), affirmant que le contexte d'un terme est inclus dans celui de ses hyperonymes.

7.2.2.2 Plongements hyperboliques

Récemment, ALY et al., 2019 et M. LE et al., 2019 ont proposé de projeter le graphe d'Hearst dans un espace hyperbolique, en suivant la méthode de plongement de hiérarchies proposée par NICKEL et KIELA, 2017 et présentée dans le chapitre 5. Le graphe d'Hearst est une représentation formelle des comptes de fréquence obtenu en passant les patrons syntaxiques sur le corpus de texte. En d'autres termes, les

nœuds du graphe sont les termes, et un lien entre un terme x et y est pondéré par le nombre de fois que x et y remplissent un patron dans le corpus. Ce graphe est bruité et épars, car, les patrons sont parfois trop sensibles, ce qui crée des cycles, ou trop restrictifs, ce qui cache alors des relations. Une solution à ces problème est d'élaguer le graphe (PANCHENKO et al., 2016), de normaliser les comptes avec l'information mutuelle (M. LE et al., 2019), et/ou de le projeter dans un espace hyperbolique, permettant le filtrage des liens moins pertinents (ALY et al., 2019). La distance hyperbolique et des heuristiques particulières à la géométrie hyperbolique permettent l'interprétation du plongement comme un graphe.

ALY et al., 2019 utilisent les plongements conjointement avec les méthodes traditionnelles pour raffiner les taxonomies prédites, et pas, comme nous le proposons ici, directement employés pour trouver la taxonomie minimisant la distortion des distances du plongement. Le problème avec le principe de raffinage est l'addition d'une couche supplémentaire à un processus déjà en mille-feuille (extraction des *features*, *feature engineering* puis classification), augmentant le risque de la propagation d'erreur et réduisant la capacité à bénéficier d'amélioration des couches précédentes. Notre travail est aussi différent de la détection d'hyperonymes à partir d'espaces hyperboliques, proposée par M. LE et al., 2019, parce que nous visons à construire la structure d'une taxonomie entière.

7.2.2.3 Minimisation de la distortion

L'originalité de notre approche est qu'elle repose sur la distortion (voir section 5.2.1). SARKAR, 2011 l'utilise pour prouver que sa construction est capable de projeter un arbre dans un espace hyperbolique avec une déformation des distances arbitrairement minime. L'induction de taxonomie est un problème exactement inverse, dans lequel un plongement de termes est donné (ou plutôt construit avec les *features*), et où on doit déterminer l'arbre (vue comme le squelette de la taxonomie) préservant au mieux les distances. Autrement dit, on veut une méthode capable d'extraire un arbre avec une distortion optimale.

Un plongement peut être considéré comme un graphe complètement connecté dans lequel le poids d'un lien (x, y) est donné par la distance entre les vecteurs de x et de y. Il s'agit par la suite de déterminer un arbre couvrant ayant une distortion optimale avec ce graphe (AHMED et al., 2020). Ce problème est connu sous le nom de arbre couvrant d'expansion-maximale minimale (*Minimum Max-strech Spanning Tree*, MMST) et est dans le cas général NP-difficile. En particulier, quand il existe un arbre couvrant qui préserve les distances du graphe parfaitement (distortion = 1), alors cet arbre coïncide avec le arbre couvrant de poids minimal (*Minimum Spanning Tree*, MST) (NAYYERI et RAICHEL, 2019 ; AHMED et al., 2020). En général, la distortion atteignable n'est pas parfaite, ce qui requiert une autre stratégie pour construire le MMST. À cause de la complexité du problème, on implémente un algorithme de l'état de l'art présenté par DHAMDHERE, Anupam GUPTA et RÄCKE, 2006, une solution naïve et une solution basée sur une heuristique.

7.3 Méthodologie proposée

7.3.1 Combinaison de *features* syntaxiques et lexicales

7.3.1.1 Généralités.

De façon similaire à M. LE et al., 2019 ; ROLLER, KIELA et NICKEL, 2018 ; PANCHENKO et al., 2016, des fréquences obtenues *via* des patrons syntaxiques sont transformées en utilisant l'information mutuelle. Nous proposons aussi d'utiliser l'information mutuelle sur les *features* lexicales, avant de les combiner en les moyennant avec les *features* syntaxiques. La combinaison des informations morphologiques et syntaxiques forment une nouvelle *feature* f_{agg}, détaillée ci-après, pondérant les liens entre les termes. Ce graphe (analogue au graphe de Hearst) est raffiné, pour retirer le bruit, en utilisant une SVD et en conservant les plus proches voisins seulement. Nous espérons ainsi obtenir un graphe moins ambigu que celui de Hearst.

7.3.1.2 Corpus.

Pour des raisons de comparaisons, on utilise le corpus de comptes des patrons syntaxiques fourni par ALY et al., 2019 [2]. Ce corpus est construit à partir de deux sources (une générale et l'autre plus spécifique). Le corpus général est composé de textes venant de ressources en ligne comme Wikipedia. Le corpus spécifique provient de textes collectés sur internet et corrélés au sujet de la taxonomie. En particulier, un moteur de recherche est requêté avec des combinaisons aléatoires des termes et garantit que les pages web ont un thème relié à la taxonomie. Le déclenchement des patrons syntaxiques entre les termes de la taxonomies est en principe obtenu avec des outils comme *PattaMaika* [3]. On récupère aussi les comptes de *WebIsA* (SEITNER et al., 2016).

7.3.1.3 *Features* syntaxiques.

On dispose de trois catégories de comptes : généraux, spécifiques et ceux de *WebIsA*. Tous sont indépendamment transformés dans l'espace de l'information mutuelle positive par point (*Positive Pointwise Mutual Information*, PPMI), suivant ROLLER, KIELA et NICKEL, 2018. De la même manière, on note $w_d(x, y)$ le compte de co-occurrences asymétriques entre x et y pour le jeu de données d. La probabilité

2. https://github.com/uhh-lt/Taxonomy_Refinement_Embeddings
3. http://ltmaggie.informatik.uni-hamburg.de/jobimtext/components/pattamaika/

jointe $p_d(x, y)$ sur les paires des termes, la probabilité marginale d'être un hyponyme $p_d^-(x)$ ou un hyperonyme $p_d^+(x)$, sont approximées pour d par :

$$p_d(x, y) = \frac{w_d(x, y)}{\sum_{x', y'} w_d(x', y')}, \tag{7.1}$$

$$p_d^-(x) = \frac{\sum_{y'} w_d(x, y')}{\sum_{x', y'} w_d(x', y')}, \tag{7.2}$$

$$p_d^+(x) = \frac{\sum_{y'} w_d(y', x)}{\sum_{x', y'} w_d(x', y')}. \tag{7.3}$$

Les scores PPMI sont calculés pour chaque paire de termes et pour chaque jeu de données d avec

$$\text{ppmi}_d(x, y) = \max\left(0, \log \frac{p_d(x, y)}{p_d^-(x) p_d^+(y)}\right). \tag{7.4}$$

7.3.1.4 *Feature* lexicale.

On utilise une *feature* lexicale, notée $\sigma(x, y)$, entre deux termes x et y. Elle utilise un score basé sur l'inclusion d'un terme dans l'autre

$$\sigma(x, y) = \begin{cases} \frac{\text{length}(x)}{\text{length}(y)} & \text{si } x \text{ est contenu dans } y \\ 0 & \text{sinon} \end{cases}. \tag{7.5}$$

Avant d'être combinée, cette *feature* est transformée dans l'espace PPMI comme pour les *features* syntaxiques (en remplaçant $w_d(x, y)$ par $\sigma(x, y)$)

7.3.1.5 *Feature* aggrégée.

On propose une nouvelle *feature* d'agrégation f_{agg}, obtenue en assemblant chaque *feature* comme suit :

$$f_{\text{agg}}(x, y) = \text{ppmi}_\sigma(x, y) + \frac{1}{3} \sum_{d \in \left\{\begin{smallmatrix}\text{General}\\ \text{Specific}\\ \text{WebIsA}\end{smallmatrix}\right\}} \text{ppmi}_d(x, y). \tag{7.6}$$

Cette *feature* pondère la force d'une relation entre x et y dans les trois corpus syntaxiques, tout en considérant leur similarité morphologique. Les méthodes de l'état de l'art exploitent généralement un classificateur pour combiner les *features*, ou utilisent exclusivement la source syntaxique. Nous proposons f_{agg} qui fusionne précocement les deux catégories, intégrant toutes les informations avant le plongement des termes.

Pour justifier cette nouvelle *feature*, on montre son efficacité à trier le voisinage des termes par rapport à ses composantes, dans le tableau 7.1. Les données utilisées font partie de la tâche 13 SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016),

exploitées ultérieurement dans ce chapitre pour l'induction de taxonomies. On remarque que prendre les composantes individuellement est moins efficace pour organiser le voisinage, ce qui défend l'usage de f_{agg}.

On crée finalement une matrice basée sur f_{agg} (contenant les poids des liens terme à terme), qui est lissée en enlevant quelques-unes de ses valeurs singulières, comme fait par ROLLER, KIELA et NICKEL, 2018, et les valeurs en dessous de la moyenne. La matrice est élaguée davantage en ne conservant que les 5 plus proches voisins. Cette valeur a été arbitrairement choisie et doit être investiguée en profondeur. La matrice est vue comme un graphe, ce qui nous permet d'appliquer les algorithmes de projection de graphe dans un espace hyperbolique ou même euclidien, comme vu dans le chapitre 5.

7.3.2 Plongement des termes

Pour plus de simplicité, on note le graphe construit dans la section précédente $\mathcal{G} = (V, M)$, V étant l'ensemble des termes et M étant la matrice asymétrique du poids des liens entre chaque termes. On note aussi pour $x, y \in V$, le poids de la relation (x, est_un, y) par $M(x, y)$.

Soit $\mathcal{D}^N = \{(x, y) | M(x, y) > 0, (x, y) \in V^2\}$ l'ensemble des liens de poids strictement positifs issus de la matrice M. On paramètre des vecteurs des termes dans une dimension n, et on note $\Theta = (\theta_x)_{x \in V}$. On procède à l'optimisation de Θ avec trois distances différentes (les mêmes que précédemment, voir chapitre 6) :

— la distance euclidienne $d_{\mathbb{R}^n}$,

— la distance de Poincaré $d_{\mathbb{D}^n}$,

— la distance *squared lorentzian* $d_{\mathcal{H}^{n,\beta}}$.

TABLEAU 7.1 – Étude d'ablation de f_{agg} par rapport à ses composantes, sur la taxonomie environnement de SemEval (BORDEA, LEFEVER et BUITELAAR, 2016). La *feature* est utilisée pour trier le voisinage d'un terme, et on calcule la MAP par rapport aux liens direct de la taxonomies. * indique une p-valeur inférieure à 0.01 pour un test d'hypothèse nulle $f_{\text{agg}} = \text{Others}$.

Features	MAP
$\text{ppmi}_{\text{Specific}}$	0.46
$\text{ppmi}_{\text{General}}$	0.50
$\text{ppmi}_{\text{WebIsA}}$	0.37
ppmi_{σ}	0.51
$\text{avg}(\text{ppmi}_{\text{Specific}}, \text{ppmi}_{\text{General}}, \text{ppmi}_{\text{WebIsA}})$	0.59
f_{agg}^{*}	0.71

On optimise ensuite les plongements des termes avec la distance choisie d de façon à incorporer les informations de $\mathcal{D}^N$. La fonction de coût est analogue à 5.3,

$$\mathcal{L}(\Theta) = \sum_{(x,y)\in\mathcal{D}^N} \log \frac{e^{-d(x,y)}}{\sum_{y'\sim\mathcal{N}(x)} e^{-d(x,y')}} \tag{7.7}$$

avec $\mathcal{N}(x) = \{y' | (x,y') \notin \mathcal{D}^N\}$ l'ensemble des termes non liés à x dans $\mathcal{D}^N$, appelés liens négatifs. Pour chaque batch, on échantillonne 10 liens négatifs et on utilise l'optimisation riemannienne de la section 5.2.3.2 pour les distances hyperboliques, et une optimisation standard pour la distance euclidienne. L'implémentation repose sur le code des HypCones [4] (GANEA, BECIGNEUL et HOFMANN, 2018a) et gensim [5].

7.3.3 Extraction d'arbre

Cette partie du travail est la plus innovante car elle applique des algorithmes tentant de minimiser la distortion entre le plongement et l'arbre extrait. Les autres propositions font généralement appel à des classificateurs ou à des modélisations plus complexes (MAO, REN et al., 2018 ; PANCHENKO et al., 2016 ; ALY et al., 2019).

Comme mentionné précédemment, les taxonomies sont des DAG connexes. Alors que nous cherchons à minimiser la distortion, l'aspect dirigé de la taxonomie est encombrant, puisque cela empêche les chemins entre des branches distinctes. Pour cette raison, on travaille sur des graphes non dirigés acycliques connexes, plus généralement nommés des arbres. La direction des liens est retrouvée en enracinant l'arbre au centroïde du plongement. On obtient des arbres dirigés interprétés comme des taxonomies. Par exemple, si nous avons des liens non dirigés $E = \{(\text{vélo}, \text{véhicule}), (\text{véhicule}, \text{voiture})\}$, et si le centroïde est véhicule, la taxonomie est $(\text{vélo}, \textit{est_un}, \text{véhicule})$ et $(\text{voiture}, \textit{est_un}, \text{véhicule})$.

La principale faille de cette simplification des taxonomies en arbres enracinés est que les termes sont contraints d'avoir un seul parent. La contrepartie est la possibilité d'exploiter des résultats de la théorie des graphes. Nous proposons d'explorer trois différentes solutions d'extraction (visant à minimiser la distortion), d'autres solutions pourraient être investiguées. L'extracteur d'arbre est appliqué sur le plongement, produit dans la section précédente.

7.3.3.1 Extraction Naïve (NE).

Cette technique est très similaire à celle de MONATH et al., 2019, sauf que l'arbre non dirigé extrait est ensuite enraciné au centroïde du plongement. L'extraction repose sur des heuristiques de la géométrie hyperbolique : la distance encode la puissance d'une relation *est_un* et la direction de celle-ci est déterminée par la différence des normes euclidiennes. Pour un terme x,

4. https://github.com/dalab/hyperbolic_cones
5. https://github.com/RaRe-Technologies/gensim

$$y^* = \arg\min_{y \in V} d_E(\theta_y, \theta_x) \qquad \text{with} \qquad \|\theta_y\|_2 \leq \|\theta_x\|_2 \tag{7.8}$$

est considéré relié à x et le lien (x, y^*) est ajouté à l'ensemble des liens de l'arbre. Lorsque l'arbre est construit ainsi, il n'a qu'une seule composante et aucun cycle.

7.3.3.2 MST Extraction (MSTE).

Comme mentionné précédemment, le MST correspond à la solution lorsqu'il existe un arbre préservant parfaitement la distortion. Pour cette raison, on regarde ce cas, car il peut retourner parfois un bon arbre couvrant.

7.3.3.3 Low Average Stretch Extraction (LASE).

Cet algorithme stochastique a été proposé par DHAMDHERE, Anupam GUPTA et RÄCKE, 2006. Nous l'adaptons pour qu'il puisse fonctionner sur des arbres pondérés. Cet algorithme décompose récursivement le plongement en utilisant des sphères avec un rayon spécifique. Un paramètre contrôle le rayon et a été choisi sur la sous-taxonomie *mammal.n.01*.

7.3.4 Raffinage de la taxonomie

On investigue aussi le raffinage de la taxonomie proposé par ALY et al., 2019 qui consiste à ajouter ou à supprimer des liens aberrants, par rapport aux *features* distributionnelles. Avec la même idée, on considère l'addition de liens sur la taxonomie extraite, avec des plongements de Spacy pour chaque langue (HONNIBAL et al., 2020). Ceux préservant la structure de la taxonomie (sans ajout de cycle) sont rajoutés à l'arbre. Seuls les termes déconnectés sont rajoutés ainsi.

7.4 Expériences

On présente ici deux séries d'expériences : la première évalue la performance d'extraction d'un plongement directement issu de taxonomies, la seconde s'attaque à l'induction de taxonomie.

7.4.1 Protocole

Pour chaque type de distance, on calcule des plongements de dimensions différentes $\{2, 5, 10, 20, 50, 100\}$ pendant 1 500 époques. On trouve la meilleure dimension sur un ensemble de validation. Le reste des hyper-paramètres est choisi en fonction des études précédentes, notamment celle de NICKEL et KIELA, 2017, en prenant 10 liens négatifs par lien positif.

7.4.2 Evaluation

En comparaison avec les cadres d'évaluation proposés dans l'état de l'art, principalement mesurant le F1-score des liens directs (F1-Score direct), on propose un cadre plus complet, car il fournit des aperçus sur la structure de la taxonomie. On utilise pour cela les projets open-source GMatch4py [6] et aproximated_ged [7]. Les mesures transitives ou directes (voir ci-dessous) quantifient la qualité des taxonomies à pouvoir généraliser, et nous proposons d'enrichir ces mesures via une comparaison géométrique des taxonomies. Nous détaillons les scores d'évaluation utilisés pour comparer la taxonomie prédite (avec des liens E^p) avec la taxonomie cible (avec des liens E^*).

F1-score direct est la combinaison de la précision $p = \frac{|E^* \cap E^p|}{|E^p|}$ et du rappel $r = \frac{|E^* \cap E^p|}{|E^*|}$, et mesure la qualité à découvrir des liens pertinents.

F1-score transitif est l'extension du F1-score direct (trouvé dans le travail de GANEA, BECIGNEUL et HOFMANN, 2018a), et mesure la concordance entre les ancêtres des deux taxonomies pour un terme. Il s'agit en réalité d'un F1-score direct sur la fermeture transitive des taxonomies.

Nous proposons d'ajouter les trois métriques suivantes, qui ne sont pas habituellement employées pour l'évaluation de l'extraction de taxonomies,

Score du Kernel Weisfeiler Lehman (WL Kernel, SHERVASHIDZE et al., 2011) est un score de comparaison de graphe à base de noyau, les noyaux capturent les schémas des sous-arbres. Ce score quantifie donc la similarité des sous-arbres des deux graphes.

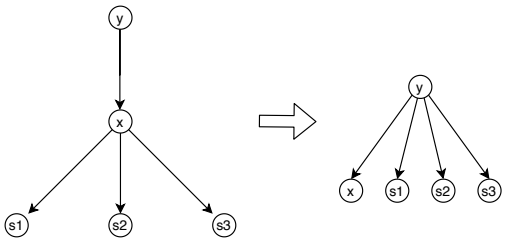
Score du Kernel du plus court chemin (SP Kernel) [8] est aussi un score de comparaison de graphe à base de noyaux, ceux-ci se focalisent sur les plus courts chemins entre les nœuds.

Distance d'édition de Hausdorff (HED, FISCHER et al., 2015) est l'approximation d'une distance d'édition entre les graphes, similaire à la distance de Levenshtein. On compte le nombre minimum d'ajouts ou de retraits de liens ou de nœuds pour passer d'un graphe à l'autre.

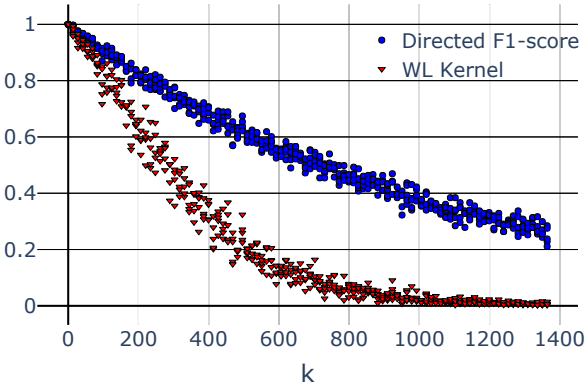
On propose d'étudier la différence entre les scores proposés, en particulier le F1-score direct et WL Kernel. Considérons une famille d'arbres n-aires de profondeur h, notée $\mathcal{T}_h^n$. On note $\widetilde{\mathcal{T}}_{h,k}^n$, la famille d'arbres corrompus comme suit :

1. Soit $t \in \mathcal{T}_h^n$, on note t' l'arbre modifié.

6. https://github.com/Jacobe2169/GMatch4py
7. https://github.com/priba/aproximated_ged
8. https://github.com/Jacobe2169/GMatch4py



(A) Corruption d'une branche ternaire.



(B) F1-score direct et WL Kernel en fonction du nombre de corruption k.

FIGURE 7.1 – Etude de la corruption des arbres sur l'évaluation.

2. Répéter k fois (le nombre de corruption) :
   - Sélectionner un nœud x dans t. Si y est un parent direct de x dans t', continuer, sinon passer les étapes suivantes.
   - Retirer les liens de x à ses enfants.
   - Connecter les nœuds orphelins à y.

Ce processus introduit volontairement des erreurs dans les arbres qui sont très proches des erreurs faites par les systèmes d'induction de taxonomies : c'est équivalent à faire la confusion entre hyperonyme et hyponyme. Sur la figure 7.1, nous comparons les arbres $\widetilde{\mathcal{T}}^4_{5,k}$ (k variant) avec leur version originale de $\mathcal{T}^4_5$, et on rapporte le F1-score direct et WL Kernel. On observe que le F1-score direct décroit linéairement (coefficient de Pearson ≥ 0.98), et que le WL Kernel a décroissance plus forte. Autour de 600 corruptions, on remarque que le F1-score direct reste assez élevé et que le WL Kernel est bien plus bas. On suppose qu'avec ce nombre de corruptions la structure des sous-arbres de la taxonomie est pourtant largement endommagée, ce qui n'est pas reflété par le F1-score direct. Pour cette raison, on suggère d'utiliser les métriques à base de noyaux, pour obtenir une description complète des taxonomies prédites.

7.4.3 Reconstruction de taxonomies

Avant de s'occuper de l'induction de taxonomie dans le cadre standard, on propose d'abord d'évaluer nos extracteurs sur des plongements de sous-taxonomies WordNet (G. A. MILLER, 1995). Au lieu d'utiliser le graphe bruité issue des *features* syntaxiques et lexicales de la section 7.3, nous utilisons de vraies taxonomies, et les plongeons en utilisant la méthode de la section 5.2.3.4 (i.e. on minimise la distance entre les liens positifs par rapport à des liens négatifs). On travaille sur des sous-taxonomies car la méthode d'extraction LASE a des difficultés pour passer à l'échelle.

Prenant les 16 plus grosses sous-taxonomies de WordNet, on évalue d'abord la corrélation entre la performance et la distortion entre la taxonomie extraite et le plongement de la sous-taxonomie. Ainsi, les corrélations de Spearman entre le F1-score direct et la distortion pour les distances $d_{\mathbb{R}^n}$, $d_{\mathbb{D}^n}$ et $d_{\mathcal{H}^{n,\beta}}$ sont respectivement : -0.79 ($p < 0.01$), -0.89 ($p < 0.01$) et 0.24 ($p < 0.17$). On voit que pour $d_{\mathbb{R}^n}$ et $d_{\mathbb{D}^n}$, les meilleurs F1-score directs sont obtenus quand la distortion est basse, confirmant notre approche qui essaye de trouver des taxonomies minimisant la distortion. Pour $d_{\mathcal{H}^{n,\beta}}$, nous n'observons pas de corrélation particulière, peut-être à cause du fait qu'il s'agisse d'une pseudo-distance.

Les résultats pour deux sous-taxonomies (les tendances sont identiques sur d'autres) sont présentés sur le tableau 7.2, en omettant volontairement SP Kernel et HED (pour des raisons de complexités temporelles). Pour chaque paire composée d'un extracteur et d'une distance, on choisit la dimension correspondant au meilleur F1-score direct sur la sous-taxonomie *communication.n.02*.

Pour toutes les expériences, les plongements utilisant $d_{\mathbb{R}^n}$ sont significativement moins performants que ceux utilisant $d_{\mathbb{D}^n}$ ou $d_{\mathcal{H}^{n,\beta}}$. Ceci confirme encore le fait que les espaces hyperboliques sont mieux capables de préserver des informations hiérarchiques. On remarque aussi que les extracteurs ont des résultats différents en fonction de la distance utilisée. En particulier, MSTE est meilleur avec $d_{\mathbb{R}^n}$, NE avec $d_{\mathbb{D}^n}$, et LASE avec $d_{\mathcal{H}^{n,\beta}}$. On peut expliquer ces différences par le fait qu'elles exploitent des heuristiques qui ne sont pas satisfaites dans toutes les géométries. Ce résultat est crucial, car il souligne l'importance d'adapter l'algorithme d'extraction au plongement.

La distance $d_{\mathcal{H}^{n,\beta}}$ est la plus performante sur toutes les métriques excepté le F1-score transitif sur *plant.n.02*. Cela peut être expliqué par la politique d'enracinement, fixant la racine de l'arbre au centroïde. Quand le centroïde ne coïncide pas avec la racine, le F1-score transitif est très affecté. L'entité *vascular_plant.n.01* est mieux que *herb.n.01* car elle est plus générale dans la taxonomie *plant.n.02*. C'est pourquoi il est aussi important de regarder le WL Kernel, étant moins affecté par ce genre d'erreur, et reflètent mieux les différences de structures.

Nous ajoutons à cette analyse quantitative une visualisation qualitative avec la figure 7.2. D'abord la figure 7.2a représente les plongements des termes et l'arbre

TABLEAU 7.2 – Résultats de reconstruction de deux sous-taxonomies de WordNet (G. A. MILLER, 1995). Les dimensions sont choisies sur la sous-taxonomie *communication.n.02* avec le F1-score direct. Les meilleurs résultats sont en gras.

Sous-Taxonomie ($\|V\|$, $\|E\|$)	Extracteur d'arbre	Distance	Dimension	Centroïde	F1-score direct	F1-score transitif	WL Kernel
cognition.n.01 (3999, 4033)	NE	$d_{\mathcal{E}}$	5	basic_cognitive_process.n.01	0.2146	0.1183	0.0477
		$d_{\mathcal{P}}$	100	cognition.n.01	0.391	0.6129	0.2494
		$d_{\mathcal{L}}$	100	cognition.n.01	**0.5466**	**0.7615**	**0.7341**
	MSTE	$d_{\mathcal{E}}$	100	cognition.n.01	0.249	0.2754	0.0402
		$d_{\mathcal{P}}$	100	cognition.n.01	0.373	0.6048	0.2436
		$d_{\mathcal{L}}$	100	cognition.n.01	**0.6074**	**0.9098**	**0.7263**
	LASE	$d_{\mathcal{E}}$	5	basic_cognitive_process.n.01	0.0853	0.0596	0.0658
		$d_{\mathcal{P}}$	10	concept.n.01	0.3073	0.2037	0.1593
		$d_{\mathcal{L}}$	50	cognition.n.01	**0.7805**	**0.5961**	**0.8954**
plant.n.02 (4487, 4493)	NE	$d_{\mathcal{E}}$	5	vascular_plant.n.01	0.1568	0.1397	0.0147
		$d_{\mathcal{P}}$	100	vascular_plant.n.01	0.5439	**0.5908**	0.8565
		$d_{\mathcal{L}}$	100	herb.n.01	**0.6243**	0.4102	**0.9256**
	MSTE	$d_{\mathcal{E}}$	100	vascular_plant.n.01	0.1795	0.2467	0.0153
		$d_{\mathcal{P}}$	100	vascular_plant.n.01	0.517	**0.5751**	0.852
		$d_{\mathcal{L}}$	100	herb.n.01	**0.6341**	0.5387	**0.924**
	LASE	$d_{\mathcal{E}}$	5	vascular_plant.n.01	0.09	0.231	0.0345
		$d_{\mathcal{P}}$	10	vascular_plant.n.01	0.4633	**0.4984**	0.4603
		$d_{\mathcal{L}}$	50	herb.n.01	**0.9032**	0.349	**0.9789**

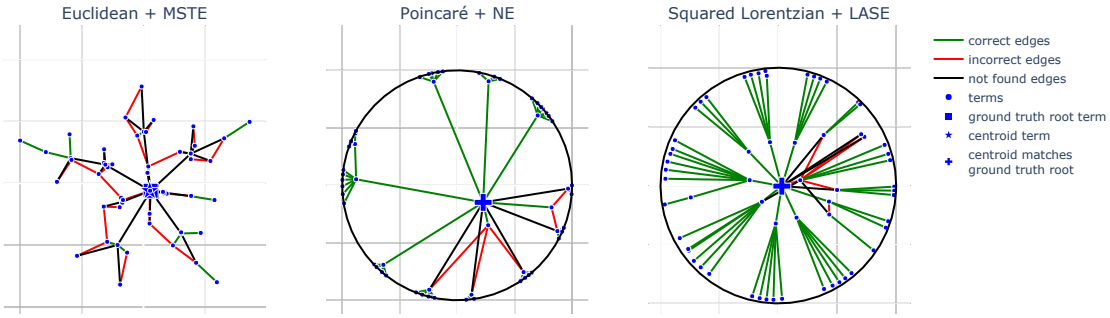
extrait avec la meilleure méthode d'extraction. Il est visible que les distances hyperboliques projettent mieux les données hiérarchiques, utilisant le centre et les frontières pour organiser les données spécifiques par rapport aux données génériques. La figure 7.2b montre les différences entre les extracteurs pour un même plongement obtenu avec $d_{\mathcal{H}^{n,\beta}}$. L'extracteur LASE connecte les termes au barycentre là où les autres connectent aux plus proches voisins, entraînant la formation d'une chaîne qui ne reflètent pas la hiérarchie originale.

7.4.4 Induction de taxonomie

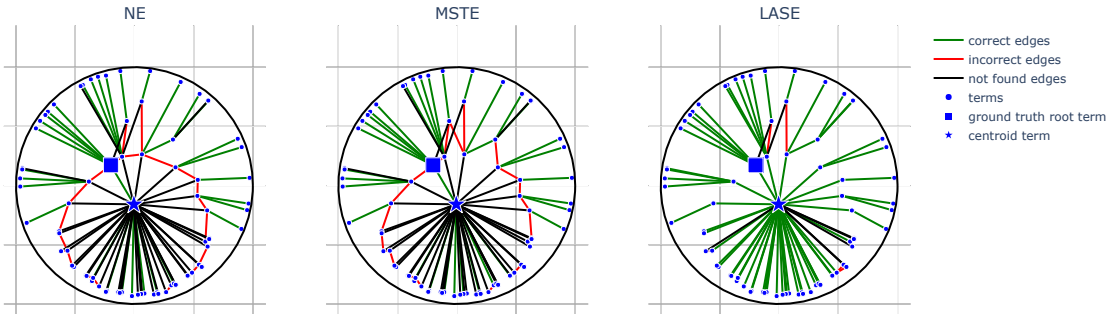
On démontre l'applicabilité de notre approche sur la tâche d'induction de taxonomie : la tâche 13 de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016). Cette tâche est composée de taxonomies multilingues : 4 langues (anglais (EN), français (FR), italien (IT) et néerlandais (NL)) et 3 domaines (science (Sc.), alimentaton (Al.) et environnement (En.)), pour un total de 12 taxonomies, une pour chaque paire possible. Pour ce problème, on se compare avec TAXI, USAAR, JUNLP et TAXI raffiné. Le code source est réutilisé pour calculer les taxonomie sur TAXI et TAXI raffiné, et les taxonomies de USAAR et JUNLP sont récupérées sur le site officiel de la tâche [9]. Comme les résultats de USAAR et JUNLP ne sont pas convaincants sur les taxonomies anglaises, on ne calcule pas leurs résultats pour d'autres langues. D'autres systèmes récents (MAO, REN et al., 2018 ; SHANG et al., 2020, TaxoRL, Graph2Taxo) ne sont pas évalués ici, car comme mentionné précédemment, ils reposent sur des connaissances externes (autres taxonomies)

Pour cette tâche, on utilise les comptes des patrons syntaxiques fournis par (ALY et al., 2019), et les comptes de WebIsA (SEITNER et al., 2016), qui sont transformés

9. `https://alt.qcri.org/semeval2016/task13/index.php?id=evaluation#subm`



(A) Meilleure combinaison de distance et d'extracteur. La sous-taxonomie de WordNet (G. A. MILLER, 1995) *food_fish.n.01* est utilisée pour les plongements.



(B) NE, MSTE, et LASE. La sous-taxonomie de WordNet (G. A. MILLER, 1995) *medical_science.n.01* avec la distance $d_{\mathcal{H}^{n,\beta}}$ est utilisée pour les plongements.

FIGURE 7.2 – Plongement à deux dimensions sur des sous-taxonomies de WordNet (G. A. MILLER, 1995) et leur arbre extrait.

TABLEAU 7.3 – Induction de taxonomie sur la tâche 13 de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016). P-valeurs des expériences * : p < 0.1, ** : p < 0.05, *** : p < 0.01, pour un test de l'hypothèse nulle TAXI (raf.) = Méthode. Les meilleurs résultats sont en gras.

Score	Méthode	EN		FR			IT			NL		
		Sc.	Al.	Sc.	Al.	En.	Sc.	Al.	En.	Sc.	Al.	En.
F1-score direct	Nous**	0.396	0.342	**0.348**	0.252	0.261	0.338	0.195	0.24	0.322	0.215	0.183
	Nous (raf.)***	0.40	0.337	0.319	0.25	0.249	0.329	**0.197**	0.225	0.293	0.211	0.173
	TAXI	0.369	0.307	0.321	0.232	0.244	0.323	0.172	0.31	0.298	0.197	0.285
	TAXI (raf.)	**0.417**	**0.374**	0.334	**0.295**	**0.29**	**0.346**	0.185	**0.309**	0.305	**0.218**	**0.266**
	JUNLP	0.3232	0.253	0.324	0.231	-	0.336	0.187	-	**0.325**	0.225	-
	USAAR	0.373	0.268	-	-	-	-	-	-	-	-	-
F1-score transitif	Nous	0.489	0.408	0.407	0.335	0.507	0.316	0.088	0.38	0.364	0.267	0.399
	Nous (raf.)**	0.497	0.42	**0.453**	**0.354**	**0.547**	0.335	**0.271**	**0.434**	**0.396**	**0.279**	**0.434**
	TAXI	0.501	0.367	0.376	0.218	0.316	0.367	0.214	0.288	0.365	0.226	0.305
	TAXI (raf.)	**0.559**	**0.452**	0.391	0.301	0.375	**0.383**	0.233	0.304	0.379	0.258	0.304
	JUNLP	0.357	0.253	0.269	0.12	-	0.181	0.093	-	0.20	0.114	-
	USAAR	0.191	0.115	-	-	-	-	-	-	-	-	-
WL Kernel	Nous***	0.646	**0.397**	0.517	0.293	0.423	0.484	0.271	**0.598**	0.473	0.303	**0.518**
	Nous (raf.)*	**0.647**	0.393	**0.583**	**0.306**	0.309	**0.559**	**0.326**	0.405	**0.597**	**0.309**	0.393
	TAXI	0.396	0.205	0.362	0.14	0.366	0.42	0.144	0.41	0.366	0.122	0.349
	TAXI (raf.)	0.528	0.348	0.419	0.26	**0.503**	0.465	0.150	0.4849	0.393	0.166	0.436
	JUNLP	0.439	0.173	0.261	0.16	-	0.415	0.169	-	0.398	0.203	-
	USAAR	0.470	0.239	-	-	-	-	-	-	-	-	-
SP Kernel	Nous***	0.762	0.777	0.881	0.986	**0.741**	**0.911**	0.952	**0.895**	**0.809**	**0.981**	**0.957**
	Nous (raf.)***	**0.787**	**0.810**	**0.901**	**0.992**	0.732	0.91	**0.962**	0.87	0.778	0.968	0.952
	TAXI	0.466	0.501	0.569	0.895	0.455	0.383	0.509	0.656	0.405	0.573	0.719
	TAXI (raf.)	0.526	0.643	0.573	0.982	0.592	0.418	0.557	0.664	0.406	0.572	0.922
	JUNLP	0.714	0.779	0.587	0.849	-	0.506	0.743	-	0.309	0.782	-
	USAAR	0.06	0.116	-	-	-	-	-	-	-	-	-
HED	Nous	**11**	69	**6**	79	**11**	7	72	**10**	**5**	87	6
	Nous (raf.)	**11**	68	11	**76**	22	8	**55**	23	7	**72**	15
	TAXI	17	67	8	87	16	9	92	7	18	85	7
	TAXI (raf.)	20	**57**	11	82	15	8	78	11	21	83	**4**
	JUNLP	17	123	22	83	-	**6**	101	-	10	85	-
	USAAR	12	111	-	-	-	-	-	-	-	-	-
Profondeur centroïde	Nous	0	0	0	1	0	2	2	1	0	3	3

en de nouvelles *features* comme expliqué dans la section 7.3. En particulier, pour les comptes de WebIsA, on utilise un système de traduction pour enrichir les comptes pour les autres langues que l'anglais (cette base étant uniquement en anglais). Il faut aussi noter que les racines des taxonomies doivent être données pour les autres méthodes. En comparaison, notre méthode est capable de fournir une racine candidate (étant le centroïde) qui peut se montrer proche de la racine réelle (voir tableau 7.3). Cependant, pour les autres mesures d'évaluation et pour plus d'équité, on utilise la racine réelle pour enraciner les arbres dans notre méthode.

La taxonomie anglaise sur l'environnement est utilisée pour ajuster les hyperparamètres : la distance, la méthode d'extraction et la dimension. On trouve que la meilleure combinaison est $d_{\mathbb{D}^n}$ avec NE pour 100 dimensions. Les résultats sont reportés sur le tableau 7.3. Quand le nom de la méthode est suivi de (raf.), cela indique que le système utilise le raffinage de la taxonomie prédite.

On observe que nos propositions (raffinées ou pas) sont légèrement, mais significativement, moins efficace que TAXI (raf.), pour prédire les liens directs de la taxonomie. En effet, notre méthode a un rappel moins grand mais est plus précise que TAXI (raf.), ceci est potentiellement dû à l'approximation de la taxonomie par un arbre, empêchant les termes d'avoir de multiples parents. Notre proposition est plus efficace sur les autres métriques. En particulier sur WL Kernel et SP Kernel, où nos propositions ont un score largement supérieur. HED n'est pas significatif mais montre une tendance identique. Pour le F1-score transitif, mesurant la capacité de généralisation, notre proposition avec raffinage retourne de meilleurs résultats que TAXI (raf.), et sans raffinage, nous avons une performance comparable.

Globalement, nos propositions, avec ou sans raffinage, prédisent des taxonomies avec une structure plus proche de la cible, sans réellement dégrader la performance du F1-score direct. On donne deux clés d'interprétation de ce comportement : d'abord la nouvelle *feature* f_{agg} est plus efficace que celles utilisées dans TAXI, et ensuite, la façon de TAXI prédit la taxonomie est locale et brutale, comme chaque lien est individuellement prédit puis potentiellement élagué après, là où nous proposons une extraction globale de la taxonomie avec la distortion.

7.5 Conclusion

Nous avons proposé une méthodologie pour permettre l'interprétation d'espaces continues en taxonomie. Appliqué à l'induction de taxonomie, notre méthode produit des taxonomies ayant une structure plus proche de l'attendu, cela pouvant en partie s'expliquer par la diminution du nombre d'étapes entre les *features* et la taxonomie. Il reste encore à intégrer plus fortement les *features* distributionnelles ou encore des connaissances provenant d'autres taxonomies.

Au delà de l'aspect pratique, l'extraction ou l'interprétation d'espaces continus est cruciale pour permettre l'usage de ces plongements avec d'autres applications, n'ayant pas nécessairement de composant de machine learning. Cette méthodologie ouvre de nouveaux horizons concernant l'enrichissement ou la restructuration des

données. Il est souvent possible de produire des plongements avec n'importe quelle source de données via des auto-encodeurs, une similarité (voir annexe C) ou encore du texte, mais, il est difficile de les faire communiquer avec d'autres composants plus traditionnels ou même de les comprendre (voir chapitre 2). La minimisation de la distortion assure la préservation des informations, et donc, s'il est possible de préserver les distances du plongement dans une structure interprétable, on se donne les moyens de comprendre la représentation et les données qui la produisent.

On souhaite combiner les connaissances extraites avec des plongements de mots classiques. Ces nouvelles connaissances devant idéalement apporter redondance et complémentarité aux vecteurs traditionnels. Pour procéder à cette combinaison, on s'intéresse à la tâche de classification hiérarchique dans le chapitre suivant.

Chapitre 8

Difficultés de la classification hiérarchique

Sommaire

8.1 Classification hiérarchique de textes 148
8.1.1 Description de la tâche 148
8.1.2 État de l'art 148
8.2 Élaboration d'un corpus sur-mesure 150
8.3 Méthodologie 151
8.3.1 Représentation des documents 151
8.3.2 Passage de vecteurs euclidiens vers des vecteurs hyperboliques 151
8.3.3 HMCN préfixé d'une projection hyperbolique (HYP-HMCN) 152
8.3.4 *Baseline* BERT 153
8.4 Expériences 153
8.4.1 Métriques d'évaluation 154
8.4.2 Résultats 154
8.5 Améliorations effectuées 155
8.5.1 Plongement des classes 155
8.5.2 Conversion séquentielle locale des vecteurs euclidiens . . . 155
8.5.3 Résultats 157
8.6 Conclusion 158

Ce dernier chapitre constitue une approche succincte de l'utilisation pratique des plongements de hiérarchies vus dans les chapitres précédents, en traitant le problème de la classification hiérarchique. Cette tâche consiste à ranger des objets dans une hiérarchie de classes. Nous souhaitons analyser comment la représentation hyperbolique de la hiérarchie des classes peut améliorer la tâche, en combinaison avec des plongements contextuels (comme BERT). Traditionnellement, cette tâche est résolue en utilisant soit uniquement des plongements hyperboliques, soit seulement des plongements euclidiens, chacun avec leurs avantages. Ici, nous tentons de cumuler des représentations contextuelles euclidiennes avec des plongements hyperboliques. Cette tâche s'inscrit aussi dans le cadre de Solocal, qui a besoin de classer des établissements et des professionnels dans une hiérarchie d'activités ou de rubriques. De manière générale, la classification hiérarchique bénéficie à de nombreuses applications en bio-informatique, vision par ordinateur, publicité et recherche d'information (MAO, TIAN et al., 2019 ; WEHRMANN, CERRI et BARROS, 2018 ; MENG et al., 2019). Pour ce travail exploratoire et encore en cours, nous présentons dans ce chapitre quelques premiers résultats sans prétendre à l'exhaustivité.

8.1 Classification hiérarchique de textes

8.1.1 Description de la tâche

La tâche de classification hiérarchique consiste à étiqueter un ensemble de documents avec des classes $\mathcal{Y} = (y_i)_{[\![1,C]\!]}$ organisées dans une hiérarchie $\mathcal{H} = (\mathcal{Y}, E)$. Cette hiérarchie divise successivement les niveaux de classification du document. Par exemple, la classe *Restaurant* est la racine de la sous-hiérarchie contenant des classes comme : *Restaurant Italien*, *Restaurant Japonais*... Eux-mêmes racines de sous-hiérarchies. En principe, la classification hiérarchique doit être capable de gérer plusieurs chemins de prédiction (WEHRMANN, CERRI et BARROS, 2018 ; MAO, TIAN et al., 2019). Par exemple, certains établissements peuvent avoir plusieurs activités et se retrouver dans les catégories *Restaurant*, *Bar* et *Hôtel*. Ces classes empruntent différents chemins dans la hiérarchie, car elles appartiennent à des classes parentes différentes. Dans ce chapitre, on fera l'hypothèse que les documents n'ont qu'un seul chemin de prédiction, pour simplifier le problème.

8.1.2 État de l'art

La classification hiérarchique peut être menée sur des textes ou des images (DHALL et al., 2020 ; LIU et al., 2020). Dans ce chapitre on traite uniquement des documents textes.

On distingue en général deux catégories d'approche pour la classification hiérarchique de textes : globale ou locale. Les approches locales sont plus gourmandes en paramètres et modélisation car elles associent un modèle de classification spécifique à chaque niveau de la hiérarchie ou à chaque sous-arbre. Le processus de classification est *top-down*, ce qui consiste à enchaîner les différents classifieurs : on prédit la classe du premier niveau, puis on choisit le sous-classifieur associé à la

classe prédite pour classer le niveau suivant, et ainsi de suite. Bien que cela permette de prendre finement en compte les particularités de chaque niveau, on leur reproche des coûts de passage à l'échelle pour des hiérarchies profondes et la propagation d'erreur (WEHRMANN, CERRI et BARROS, 2018 ; SINHA et al., 2018). Pour éviter la propagation d'erreur, on peut avoir un processus de classification *bottom-up* en parallèle (RIVAS ROJAS et al., 2020).

Les méthodes globales sont une alternative et solutionnent ces inconvénients en proposant un classifieur unique pour l'ensemble de la hiérarchie, mais elles souffrent d'une faiblesse de modélisation des spécificités des catégories de la hiérarchie (WEHRMANN, CERRI et BARROS, 2018).

Un modèle incontournable (B. CHEN, HUANG et al., 2020 ; MAO, TIAN et al., 2019) dans le domaine de la classification hiérarchique de texte est HMCN, proposé par WEHRMANN, CERRI et BARROS, 2018, dont l'objectif est d'unifier les approches locales et globales. La fonction d'objectif d'un classifieur global est combinée avec celle d'un classifieur local, les deux classifieurs partageant des couches intermédiaires. Un coefficient permet de contrôler l'intensité d'intégration de l'un ou l'autre des modèles. Les auteurs démontrent une supériorité de leur modèle par rapport à plusieurs autres propositions, cependant, le problème de passage à l'échelle reste présent pour HMCN, puisqu'il repose en partie sur des modèles locaux pour faire sa prédiction. Pour contrecarrer cet effet, on peut remplacer les couches successives de classification par des RNN, en contrepartie d'une légère perte de performance.

Une autre alternative aux approches globales et locales est HiLAP (MAO, TIAN et al., 2019). Il s'agit d'une méthode fondée sur l'apprentissage par renforcement. On définit un ensemble de classes accessibles (vu comme des actions) par rapport à l'état d'avancement de la prédiction. Le modèle apprend à déterminer la classe à rajouter à cet état, et on enrichit les classes accessibles par les successeurs de la nouvelle prédiction. Au départ, on part de la racine, et on prédit progressivement les classes à partir du document, de l'état de prédiction et des classes accessibles. L'avantage de procéder ainsi et de permettre plus facilement la modélisation de plusieurs chemins de classification.

Le problème des approches précédentes est leur incapacité à modéliser la hiérarchie de façon directe. Les méthodes se focalisent sur des distributions de probabilités sur les classes, et n'ont pas une représentation directe de la structure. La hiérarchie est, au final, comprise de façon indirecte par les fonctions de perte (DHALL et al., 2020). Modéliser la hiérarchie de façon conjointe avec les documents permet d'étendre les usages du modèle à d'autres tâches que celle qui sert à l'apprentissage, comme par exemple le *zero-shot learning* (découverte de classes inconnues). Dans le cas des images, la classification hiérarchique avec des espaces hyperboliques améliore la prédiction en mode *zero-shot*.

Pour le texte, B. CHEN, HUANG et al., 2020 ont proposé une méthode pour faire interagir les classes avec les documents dans un espace commun. L'utilisation d'un plongement conjoint semble augmenter la performance de classification hiérarchique, car elle permet de mieux modéliser les interactions entre les documents et les

classes. Cependant, les auteurs n'évaluent le modèle que dans le cadre de la classification hiérarchique, et n'analysent pas la qualité du plongement conjoint. De plus, ils n'imposent pas directement des contraintes sur les distances entre les classes et les documents. Enfin, ils utilisent un modèle de langue hyperbolique contextuel surmesure, et ne considèrent que des documents de taille fixe.

Nous voulons dans la suite améliorer certains aspects du modèle, en utilisant des modèles de langue pré-entraînés, comme BERT, car nous supposons qu'il apporte une meilleure représentation du document qu'un modèle sur-mesure plus petit et n'ayant pas vu autant de documents. Nous savons aussi que la géométrie hyperbolique est plus adaptée à la représentation d'une hiérarchie. C'est pourquoi, plutôt que d'employer exclusivement la géométrie euclidienne ou hyperbolique, nous souhaitons combiner les deux, chacune sur des domaines où elles excellent, c'est-à-dire, la géométrie euclidienne pour la représentation des textes, et la géométrie hyperbolique pour la représentation de la hiérarchie des classes. De plus, nous sommes particulièrement intéressés par la qualité du plongement conjoint et pas uniquement par sa capacité à faire de la classification hiérarchique, pour pouvoir étendre plus facilement la solution à d'autres tâches.

8.2 Élaboration d'un corpus sur-mesure

Bien que la tâche possède de réelles applications pratiques, il apparaît assez difficile de trouver un corpus possédant les propriétés suivantes : accès direct au texte brut, libre d'utilisation dans un cadre de recherche académique et enfin dont la hiérarchie possède une certaine profondeur. Pour ces raisons, on établit notre propre corpus à partir des dépôts (*dumps*) d'OpenLibrary [1].

Ces *dumps* contiennent des résumés et titres de livres ainsi que leur catégorie via la catégorisation décimale de Dewey. La classification de Dewey sert à répertorier les différentes catégories de littérature : philosophie, sciences sociales, histoire... On utilise les chiffres de 0 à 9 pour distinguer les dix catégories primaires, puis chaque catégorie est subdivisée en dix nouvelles catégories et ainsi de suite. On note par exemple 006.3 la catégorie de l'intelligence artificielle [2], où chaque chiffre successif précise la sous-catégorie de la catégorie du chiffre précédent. 006.3 correspond donc à la quatrième catégorie de 006 (applications informatiques), lui même septième catégorie de 00 (généralités sur l'informatique, l'information et ouvrages généraux) et première catégorie de 0 (informatique, information et ouvrages généraux).

On extrait les résumés et titres, ainsi que les classes correspondantes pour obtenir un jeu de données de documents classés hiérarchiquement. Après filtrage des entrées contenant un code Dewey conforme et un résumé non vide, on obtient un ensemble d'approximativement 130 000 documents classés. Les classes sont regroupées de sorte qu'elles contiennent au minimum 5 documents. Par exemple si 006.1 et 006.2 contiennent respectivement 2 et 3 documents, on supprime 006.1 et 006.2 et

1. `https://openlibrary.org/developers/dumps`
2. `https://catalogue.bnf.fr/ark:/12148/cb13632144m`

TABLEAU 8.1 – Quantité de documents par niveau de profondeur dans le jeu de données extrait des dumps d'OpenLibrary.

Profondeur	Quantité
0	486
1	6784
2	49288
3	50631
4	23142

TABLEAU 8.2 – Quantité de documents par catégorie principale dans le jeu de données extrait des dumps d'OpenLibrary.

Catégorie	000	100	200	300	400	500	600	700	800	900
Quantité	3483	2735	5418	19679	1838	14196	17219	17907	33329	14527

on ajoute leurs documents à 006. On donne sur les tableaux 8.1 et 8.2 des statistiques sur la distribution des documents par niveau et par catégorie.

8.3 Méthodologie

Nous expliquons ici les différents éléments permettant la construction conjointe d'un plongement hyperbolique de la hiérarchie et d'un plongement euclidien de documents.

8.3.1 Représentation des documents

BERT est utilisé pour calculer la représentation vectorielle des documents. On concatène le titre avec le résumé du livre, et on utilise le modèle *bert-base-uncased* en tronquant les chaînes de caractères trop longue. On prend la moyenne de la représentation des mots en sortie comme vecteur représentant le document entier.

8.3.2 Passage de vecteurs euclidiens vers des vecteurs hyperboliques

Comme nous utilisons BERT pour produire des représentations continues des documents, et que nous voulons faire interagir ces documents avec une représentation hyperbolique de la hiérarchie des classes, il nous faut une méthode permettant le transport de la géométrie euclidienne vers la géométrie hyperbolique. La première idée a été de réutiliser HierarX (voir annexe C) pour produire des vecteurs hyperboliques à partir de la similarité cosinus entre les documents. Cependant, cette solution n'a pas retourné un espace satisfaisant, probablement car la similarité ne possède pas les bonnes propriétés (voir annexe C.2). Aussi, HierarX voit les vecteurs de BERT sous forme compressée, *via* leur similarité (cosinus) et ne peut pas tirer parti des différentes composantes du vecteur. Pour ces raisons, nous nous sommes orientés vers des méthodes plus traditionnelles.

Dans l'état de l'art, le problème de conversion de caractéristiques euclidiennes en caractéristiques hyperboliques est résolu souvent de la même manière (B. CHEN, FU et al., 2021 ; CHAMI et al., 2019). On considère les caractéristiques euclidiennes comme composant le plan tangent en 0 de la boule de Poincaré. Grâce à l'expmap, on plie ce plan sur la variété et on obtient des représentations hyperboliques. On peut éventuellement rajouter des couches linéaires avant le pliage, pour affiner les vecteurs. Cette méthode n'est pas satisfaisante de notre point de vue, car le plan tangent est une estimation locale de la variété, et en ce sens, elle ne permet pas de profiter de l'ensemble des propriétés de l'espace hyperbolique.

On propose un autre mode de projection, s'inspirant du principe des bases d'espace vectoriel euclidien. Soit une famille de vecteurs $B = (b_i)_{i \in [\![1,k]\!]} \in \mathcal{M}$ dans une variété riemannienne, et $v \in \mathbb{R}^n$, la représentation euclidienne d'un document. On a une fonction $f : \mathbb{R}^n \to \mathbb{R}^k$, permettant l'adaptation du nombre de dimensions à la taille de la famille B. Notons $f(v) = (\lambda_1, ..., \lambda_k)$, $\text{sum}^{\mathcal{M}}$ l'opérateur de somme sur $\mathcal{M}$ et $\times^{\mathcal{M}}$ l'opérateur de multiplication (tous deux dérivés de l'expmap), on peut écrire

$$p_{\mathcal{M}}(f(v)) = \underset{i \in [\![1,k]\!]}{\text{sum}^{\mathcal{M}}} \left(\lambda_i \times^{\mathcal{M}} b_i \right) \tag{8.1}$$

la projection du vecteur euclidien $f(v)$ vers $\mathcal{M}$. Ces projections interviennent ensuite dans un critère d'optimisation, et durant l'optimisation, on considère f et B comme des paramètres du modèle. On espère que cette méthode profite mieux des caractéristiques de l'espace hyperbolique, parce qu'elle tente d'exprimer la projection directement à partir d'éléments de l'espace hyperbolique. On appelle cette méthode *Base Poincaré*. Le principal désavantage de cette approche est son coût de calcul, car l'addition n'est pas toujours un processus qui se parallélise facilement en espace hyperbolique (car non associatif).

8.3.3 HMCN préfixé d'une projection hyperbolique (HYP-HMCN)

HMCN étant une méthode répandue, on propose de l'adapter à des espaces hyperboliques. Plutôt que d'utiliser directement la représentation BERT du document dans HMCN, on rajoute une étape antérieure à HMCN, consistant en la projection du vecteur dans espace hyperbolique commun avec les classes. Autrement dit, on calcule une représentation hyperbolique de la hiérarchie des classes avec PoincaréModel ou HyperE, et on cherche à projeter dans cet espace les vecteurs des documents. On fait cela pour modéliser de façon directe la hiérarchie dans HMCN, qui n'en est pas capable. Ensuite, on calcule les distances du vecteur du document projeté à des hyperplans dans l'espace commun, pour se donner des vecteurs euclidiens servant en entrée à HMCN. Pour l'optimisation, on régularise les distances de l'espace conjoint, en réduisant la distance entre les documents et leur classe la plus fine, et on calcule la fonction de coût d'HMCN. La figure 8.1 illustre l'ensemble du modèle.

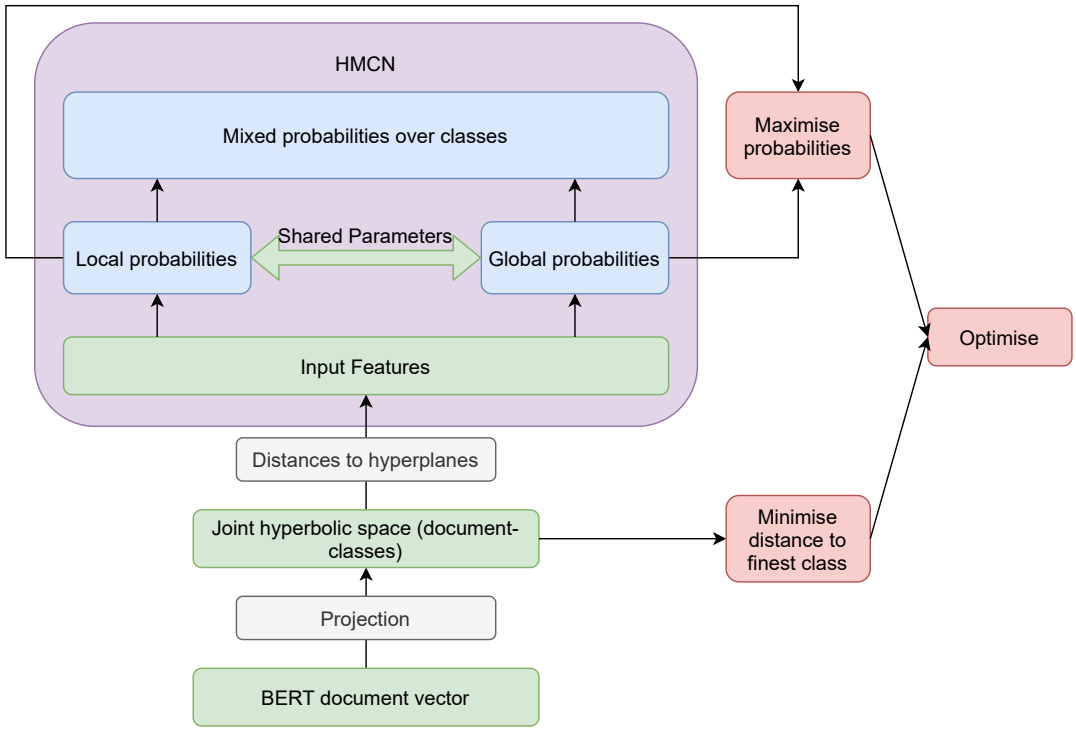


FIGURE 8.1 – HMCN avec une étape de projection hyperbolique antérieure.

8.3.4 *Baseline* BERT

Pour comparer la représentation jointe des documents et des classes, on ne dispose pas de référence pour HMCN. On propose donc d'utiliser les vecteurs de BERT, en étiquetant un document avec la classe du document le plus proche (1 plus proche voisin). On appelle cette méthode BERT 1-NN. La similarité cosinus (Eq. 2.5) sert à l'ordonnancement du voisinage. Cela nous permet de comparer la répartition des classes dans le voisinage d'un document avec celle de l'espace conjoint.

8.4 Expériences

Avant de procéder à l'évaluation sur l'ensemble du corpus d'OpenLibrary, on se restreint à la verticale 500 (sciences naturelles et mathématiques). Cela nous a permis de voir si notre proposition produit un plongement conjoint de bonne qualité, et s'il aide à l'étiquetage hiérarchique. On divise ce corpus en trois parts, une pour l'entraînement, une autre pour la validation, et la dernière pour le test.

On évalue HMCN avec en entrée les vecteurs de BERT, et HMCN préfixé avec le plongement conjoint, afin de mesurer l'apport de la modélisation hiérarchique. Pour l'espace hyperbolique conjoint, on utilise $\mathbb{D}^n$. On étudie deux modes de projections, le mode de projection dit standard (consistant au pliage de l'espace tangent), et notre mode de projection se basant sur une base de vecteur de $\mathbb{D}^n$.

TABLEAU 8.3 – Evaluation de la classification des modèles HMCN sur la catégorie 500 d'OpenLibrary

Modèle		AUC
Type	Projection	
HMCN	N/A	0.4212
HYP-HMCN (HyperE)	Standard	0.3792
	Base Poincaré	0.3936
HYP-HMCN (PoincaréModel)	Standard	0.1998
	Base Poincaré	0.2022

Pour chaque méthode on entraîne avec les mêmes hyper-paramètres et pour le même nombre d'époques, en s'arrêtant si la fonction de coût sur l'ensemble de validation augmente de façon notable. Le modèle obtenant le coût sur la validation le moins élevé est utilisé pour l'évaluation sur le test.

8.4.1 Métriques d'évaluation

Pour l'évaluation du modèle de classification, on emploie l'aire sous la courbe (*Area Under the Curve*, AUC) macro, qui nous évite d'avoir à définir un seuil pour les probabilités des classes en sortie de HMCN. Concernant l'espace hyperbolique conjoint, on utilise la MAP, le MR de la classe la plus fine et $d@k$ (Eq. 6.16), dans le but de mesurer la qualité du voisinage (MAP et MR), et l'écart entre la classe prédite (la plus proche) et la classe réelle dans la hiérarchie ($d@k$).

8.4.2 Résultats

Dans le tableau 8.3, on reporte l'AUC pour les modèles HMCN. On utilise deux méthodes pour représenter les classes, PoincaréModel ou HyperE. On remarque d'abord que la modélisation de la hiérarchie avant le passage au modèle HMCN, n'augmente significativement la performance du modèle. Parmi les deux méthodes de plongement de hiérarchies, HyperE apparaît plus performante avec HMCN. Les deux projections proposées ne sont pas très différentes, même si *Base Poincaré* apparaît légèrement au-dessus dans les deux versions d'HYP-HMCN.

Globalement, nos propositions ne semblent pas améliorer HMCN pour la classification hiérarchique. Même si on ajoute le vecteur du document en entrée d'HMCN concaténé avec le vecteur issu du plongement conjoint, aucune différence n'est observée. On en conclut qu'on ne parvient pas réellement à modéliser la hiérarchie dans HMCN, ou que cette hiérarchie n'apporte pas d'information supplémentaire par rapport à la modélisation indirecte.

Cependant, avec HYP-HMCN, on obtient un plongement conjoint document-classe, qui permet d'effectuer d'autres tâches. La MAP de ces plongements reste pourtant inférieure à la *baseline* BERT, et on observe cette tendance sur l'ensemble des métriques. Pour ces raisons, on décide de s'orienter sur l'amélioration du plongement conjoint.

8.5 Améliorations effectuées

Pour améliorer le plongement conjoint on décide de travailler sur les deux axes possibles, les documents et la hiérarchie. Nous présentons nos propositions et nos résultats ci-après.

8.5.1 Plongement des classes

Pour la représentation de la hiérarchie des classes, on remarque que les plongements issus PoincaréModel sont moins bons en classification mais obtiennent parfois de meilleurs résultats sur la structure de l'espace conjoint. Par conséquent, nous avons souhaité combiner PoincaréModel avec HyperE, donnant naissance à la méthode ASTRE (voir chapitre 6). Cette nouvelle méthode peut en particulier s'intégrer dans la classification hiérarchique, mais a des répercussions plus larges sur le mélange de différentes sources d'information (hiérarchique ou sémantique), la visualisation de hiérarchies, ou encore le calcul de représentation en faible précision. À noter que c'est pour cette raison qu'on ne trouve pas ASTRE dans les méthodes de plongement de l'expérience du tableau 8.3, puisque ASTRE ayant été développé après.

8.5.2 Conversion séquentielle locale des vecteurs euclidiens

Le deuxième axe d'amélioration est la projection des documents. Jusqu'à présent, on minimise seulement la distance du document à sa classe la plus fine, et on ne prend pas réellement en compte la structure hiérarchique lors de la projection. Pour remédier à cela, on propose de réutiliser le principe des classifieurs locaux, en proposant de projeter niveau par niveau le document. On souhaite recycler le principe des modèles de classification locale, avec un modèle par niveau. Chaque modèle se basant sur la prédiction précédente pour prédire la classe suivante.

Au départ, on calcule le plan tangent en 0 et on le replie sur la variété riemannienne. Pour le niveau suivant, on récupère la position du niveau précédent, on transporte le plan tangent en ce point, et on l'applique. Formellement, considérons une hiérarchie de l niveaux, sa représentation hyperbolique dans $\mathbb{D}^d$, un document x ayant pour représentation $v_x \in \mathbb{R}^n$, un ensemble de modèle par niveau $(m_i : \mathbb{R}^{n+k} \to \mathbb{R}^k)_{i\in[\![1,l]\!]}$, et un modèle de calcul du plan tangent par niveau $(t_i : \mathbb{R}^k \to \mathcal{T}_0\mathbb{D}^d)_{i\in[\![1,l]\!]}$. On peut calculer la représentation d'un document du niveau i dans le plan tangent en 0, notée p_i, dépendante de v_x et de l'état des projections antérieurs par

$$p_i(x) = \begin{cases} t_0(m_0([v_x, \vec{0}])) & \text{si } i = 0 \\ t_i(m_i([v_x, m_{i-1}(v_x)])) & \text{sinon} \end{cases} \tag{8.2}$$

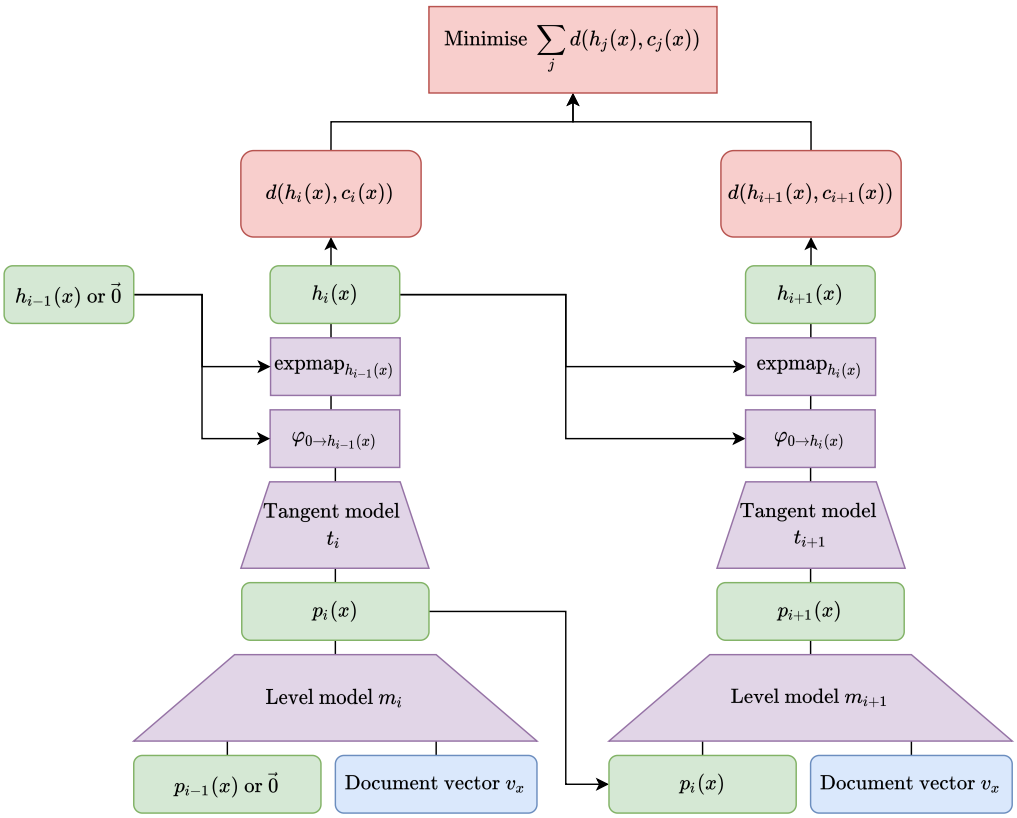


FIGURE 8.2 – Modèle de projection séquentiel.

où $[\cdot, \cdot]$ désigne la concaténation. Par la suite, on peut transporter ce vecteur à l'endroit d'application (la position précédente) et l'appliquer comme suit

$$h_i(x) = \begin{cases} \text{expmap}_0^{\mathbb{D}^d}(p_0(x)) & \text{si } i = 0 \\ \text{expmap}_{h_{i-1}(x)}^{\mathbb{D}^d}(\varphi_{0 \to h_{i-1}(x)}^{\mathbb{D}^d}(p_i(x))) & \text{sinon} \end{cases} \tag{8.3}$$

L'intérêt de passer par le transport parallèle et l'expmap est d'imposer les courbure de la variété à la représentation calculée. Comme on procède étape par étape, on peut considérer que l'approximation euclidienne du plan tangent est valable, car à chaque fois on se décale d'une quantité limité à partir du point précédent. Dans la projection globale (ou standard), on considère un unique plan tangent qui paramètre l'ensemble de la variété, ce qui rend l'approximation euclidienne moins pertinente (car elle doit rester locale). On illustre sur la figure 8.2, le processus de création des vecteurs *via* ce modèle. Si on note $c_i(x) \in \mathbb{D}^d$, la représentation hyperbolique de la classe de niveau i du document x, on peut calculer les distances $d(h_i(x), c_i(x))$, niveau par niveau et les minimiser.

À l'entraînement, on corrige les prédictions $h_i(x)$ par $c_i(x)$ de façon à guider l'apprentissage. Lors de l'inférence, on décode de façon *greedy*, en prenant à chaque niveau i la classe la plus proche de $h_i(x)$ (sans remplacer dans le modèle).

TABLEAU 8.4 – Evaluation des plongements conjoints sur le corpus extrait d'OpenLibrary (en entier).

Méthode		MAP	MR	Profondeur Moyenne	d@1	d@2	d@5	d@10
BERT	N/A	0.2228	71.04	4.7872	4.7708	4.9011	5.3562	5.6468
Standard	HyperE	0.4816	543.87	2.1071	3.8677	4.0052	4.4695	4.6217
	PoincaréModel	0.4291	625.67	2.3287	3.9805	4.2161	4.7823	5.0767
	ASTRE	0.2984	484.06	2.2126	3.9577	4.2128	4.6661	4.8583
Projection séquentielle	HyperE	0.3387	422.54	2.9618	3.5160	3.9298	4.2423	4.3640
	PoincaréModel	0.4131	568.51	3.1642	3.5873	3.9769	4.5916	4.9123
	ASTRE	0.2550	409.91	3.0670	3.5507	3.9305	4.3690	4.5427

8.5.3 Résultats

Dans le tableau 8.4, on reporte les métriques d'évaluation des plongements conjoints, par rapport à la *baseline* BERT et aux méthodes de projection standards. Les modèles standards et séquentiels sont entraînés pendant 25 époques, et on sélectionne le modèle donnant de coût minimal sur la totalité de l'entraînement.

On remarque en premier que le MR de la classe la plus fine explose par rapport à la *baseline* BERT. On peut comprendre cela en regardant les d@k et la profondeur moyenne, qui sont beaucoup plus élevées pour BERT que pour les autres méthodes. La *baseline* a tendance à donner des classes trop profondes, et lorsqu'elle se trompe, elle donne une classe éloignée dans la hiérarchie. Les autres méthodes donnent des classes moins profonde mais plus proche dans la hiérarchie. Elles n'ont pas nécessairement une bonne représentation des classes les plus fines, mais parviennent à classer les documents correctement avec une classe de plus haut niveau.

Le deuxième point est la MAP qui est améliorée par les méthodes de projection, notamment parce que les documents sont plongées conjointement avec une représentation hyperbolique des classes. Autrement dit, les méthodes de projection exploitent les liens de la hiérarchie des classes améliorant la structure du voisinage. Enfin par rapport à la méthode standard, la projection séquentielle prédit des classes plus profondes (de profondeur moyenne quasiment égale à la profondeur moyenne du jeu de donnée, valant 3.67), et plus proches dans la hiérarchie.

Concernant les différentes méthodes de représentation des classes (HyperE, PoincaréModel, ASTRE), HyperE ou PoincaréModel semblent les plus adaptées. Vu la taille des hiérarchies de classes (peu profondes), il se peut qu'ASTRE soit moins intéressant dans ce cas. Cependant, il existe plusieurs hyper-paramètres à la méthode ASTRE ; ici on n'utilise que les hyper-paramètres RestrictedASTRE avec $\tau_{\max} = 3$, mais il se peut que d'autres soient plus avantageux.

Pour conclure, la projection séquentielle semble créer des espaces mieux organisées que la projection standard, en produisant des prédictions plus fines et plus proche de l'objectif dans la hiérarchie. Utiliser une méthode de projection permet d'intégrer des connaissances hiérarchiques à l'espace conjoint, aboutissant à un espace mieux structuré. Il faut ensuite réutiliser ces plongements pour faire de classification hiérarchique. L'avantage de la projection séquentielle est aussi sa capacité à classer chaque niveau de la hiérarchie, contrairement à la projection standard.

8.6 Conclusion

La classification hiérarchique est une tâche difficile à cause du manque de données, et de la complexité des différentes approches existantes. Nous avons mis en valeur une source libre de données adaptée au problème et nous avons proposé, plutôt que de classer les documents, de les plonger dans un espace conjoint avec les classes, de façon à étendre la représentation obtenue à d'autres usages.

Nous avons évalué la qualité d'une première approche basique se reposant sur HMCN, HYP-HMCN, et remarqué aucune incidence de la nouvelle modélisation. Pour tenter d'améliorer la représentation conjointe, nous avons développé deux nouvelles approches pour la projection des classes (ASTRE, voir chapitre 6) et la projection séquentielle. Cette dernière doit encore être évaluée plus finement pour confirmer la tendance des résultats préliminaires positifs. En particulier, en évaluant la classification par niveau ou en évaluant la classification totale d'un document par un décodage en *beam seach* plutôt que *greedy*, pour exploiter tout le potentiel de cette modélisation, par rapport à la projection standard.

Il pourrait être intéressant par la suite d'intégrer la représentation et les principes de projection dans un modèle de classification comme HYP-HMCN. Une idée pertinente est d'apprendre la représentation des classes de façon simultané à la projection des documents, pour permettre de rapprocher les branches de la hiérarchie qui sont similaires vis-à-vis des documents, et ainsi, potentiellement, améliorer la représentation conjointe. ASTRE est idéal pour cela, puisqu'il permet de faire la représentation continue et flexible de la hiérarchie.

La littérature met en garde sur l'augmentation de l'efficacité de classification (DHALL et al., 2020 ; LIU et al., 2020). Plutôt que de chercher une amélioration, la production d'un espace conjoint cherche plutôt à résoudre d'autres types de problèmes comme le *zero-shot* ou la recherche d'information. Il faudrait prendre en compte ces nouvelles tâches en même temps que d'évaluer la classification hiérarchique. L'objectif étant, au final, de fabriquer des plongements conjoints recyclables dans un ensemble de tâches variées attestant alors de sa qualité (comme pour les plongements de mots), plutôt que de produire des méthodes à usage unique, certes efficaces, mais moins polyvalentes et requérant de coûteuses phases d'entraînement.

Conclusion générale

Contributions

L'hétérogénéité des plongements est l'axe principal de travail de cette thèse. Nous défendons l'usage de plongements hétérogènes (en terme d'espace) pour favoriser la découverte d'information et/ou leur restructuration sans perdre en efficacité. À chaque type de données son plongement adapté. Pour représenter des graphes, on choisira l'espace hyperbolique, et pour le langage, l'espace euclidien.

Nous avons cherché, en premier, à améliorer les plongements de mots en formulant les liens entre les métriques d'évaluation, en important et en créant des mesures globales, censées refléter des qualités structurelles. Principalement, il s'agit de quantifier continûment l'orthogonalité entre les vecteurs, et nous avons observé un lien entre la performance finale du plongement et l'orthogonalité. Par la suite, nous avons tenté d'imposer l'orthogonalité aux vecteurs à l'apprentissage, de façon à maximiser les performances sur les autres métriques. Cependant, cette régularisation des paramètres est assez difficile à réaliser, principalement pour des raisons de complexité temporelle, mais aussi parce qu'il est nécessaire de maintenir un équilibre entre l'orthogonalité et la tâche principale d'incorporation, sans quoi, les vecteurs semblent perdre en efficacité. C'est en tout cas, ce que suggère les travaux récents de LUKA, SOULIER et PICARD, 2021 et nos tentatives. Ces travaux d'évaluation furent publiés à la conférence LREC 2020 (TORREGROSSA, CLAVEAU et al., 2020). L'ensemble des connaissances sur les plongements de l'état de l'art, en particulier des plongements de mots et leur évaluation, ont été synthétisée dans un *survey* publié dans le journal international IJDSA en 2021 (TORREGROSSA, ALLESIARDO, CLAVEAU, KOOLI et al., 2021).

Les processus d'évaluation et les connaissances autour des tâches extrinsèques sont aussi exploités pour la tâche de *slot filling* alimentant le moteur de recherche PagesJaunes. Ceci a permis de proposer le vecteur de connaissances et de soutenir l'intérêt de mélanger les sources d'informations différentes pour la résolution d'une tâche réelle, en particulier quand ces sources sont le fruit d'une expertise du métier ciblé. Les mots d'une requête utilisateur sont alors enrichis par un nouveau vecteur indiquant sa présence dans les bases de données PagesJaunes et sa participation dans des expressions s'étendant sur plusieurs mots. Le vecteur du mot standard (FastText) et ce nouveau vecteur de connaissances sont fusionnés dans une couche linéaire, dont les paramètres sont entraînés par l'objectif final, qui permet la projection conjointe des informations. Cette réalisation a été publiée dans la conférence ICONIP 2019 (TORREGROSSA, KOOLI et al., 2019).

Dans l'objectif d'obtenir davantage de connaissances structurées, pour potentiellement proposer des vecteurs de connaissances plus riches, mais aussi pour réutiliser des métriques d'évaluation des plongement, nous avons proposé un nouveau critère d'optimisation pour l'alignement de graphes de connaissances par plongement. Ce critère est plus global, sans faire d'hypothèse sur les entités sans alignement connu, et offre la possibilité de réaliser un alignement non supervisé en amorçant l'apprentissage avec des méthodes d'alignement plus traditionnelles (exploitant les chaînes de caractères des entités). Cette nouvelle méthodologie a été publié à la conférence ICTAI 2020 (VAIGH et al., 2020), et donne la capacité d'enrichir un graphe de connaissance avec un autre, dans le cas d'une fusion de connaissances d'entreprise, ou simplement, pour l'enrichissement de données par des données publiques gratuites.

Un autre enjeu de la thèse a été de rechercher des moyens d'améliorer la représentation des graphes (en particulier des hiérarchies) ou de découvrir des connaissances hiérarchiques à partir de données non structurées. D'un point de vue théorique et pratique, les espaces les plus efficaces pour représenter de l'information hiérarchique sont les espaces hyperboliques.

D'abord, nous présentons une nouvelle méthode de projection dans des espaces hyperboliques en fusionnant deux catégories de méthodes (déterministes et stochastiques). Au départ, cette méthode est élaborée pour répondre au besoin d'améliorer la hiérarchie des classes pour la classification hiérarchique. Au final, elle apparaît assez naturelle et s'étend à des usages plus vastes, notamment ceux reposant sur la mixité informationnelle. Nous avons montré une meilleure polyvalence des plongements ASTRE, et une meilleure capacité à représenter les hiérarchies par rapport aux méthodes stochastiques actuelles, en particulier, lorsque les conditions en précision et dimension sont tendues.

Les connaissances sur les espaces hyperboliques sont aussi employés dans l'implémentation du logiciel HierarX, présenté en démonstration à EGC 2020 (TORREGROSSA, GRAVIER et al., 2020, voir annexe C). Celui-ci montre la possibilité, dans certains cas, de produire des représentations qui, visuellement, semblent interprétables comme des hiérarchies. Poussant ce principe au bout, on s'est intéressé à l'induction de taxonomie, qui consiste à l'inférence d'une hiérarchie à partir d'un corpus de texte et d'un ensemble de termes d'intérêt. Nous proposons une approche qui repose sur l'interprétation des plongements en taxonomie, grâce à la distortion, et par l'inversion du principe de plongement.

Finalement, nous étudions le cas de la classification hiérarchique, dans le but d'exploiter les représentations hyperboliques et de les fusionner avec des plongements traditionnels. L'objectif ultime étant de mélanger des informations hiérarchiques acquises à partir de données déstructurées pour augmenter le potentiel des représentations classiques. Nous avons proposé un nouveau modèle de classification hiérarchique HYP-HMCN, tentant de modéliser la hiérarchie des classes de façon directe dans le modèle HMCN, sans succès. Ceci nous a poussé à travailler les

façons existantes pour projeter des plongements euclidiens dans des espaces hyperboliques, ainsi que d'améliorer la qualité du plongement des classes. Les résultats préliminaires tendent à pousser en faveur du projecteur séquentiel proposé.

Dans cette thèse, nous sommes partis de l'évaluation des plongements de mots pour trouver des métriques globales utiles à l'amélioration de tâches extrinsèques, mais aussi, pour mettre en évidence des relations entre ces tâches (chapitre 2). La tâche de *slot filling*, pour Solocal, résolue par l'enrichissement des vecteurs par des connaissances métiers, nous encourage à la recherche de nouvelles connaissances (voir chapitre 3), ainsi qu'à l'amélioration de leur représentation et leur intégration dans des plongements de mots euclidiens. L'enrichissement des connaissances (chapitre 4) peut alors se réaliser comme une tâche d'alignement guidée par un critère d'optimisation issue d'une mesure globale. Ce nouveau critère initie l'alignement de graphes de connaissances de façon non supervisée. Les espaces hyperboliques viennent ensuite pour représenter ces connaissances, en particulier lorsqu'il s'agit de hiérarchies. Ils nous permettent de représenter, découvrir et éventuellement fusionner des connaissances hiérarchiques avec des plongements euclidiens, en ayant une perte d'information minimale.

Nos contributions sur ces différents sujets ont de nombreuses formes. D'abord des logiciels pour calculer des représentations hyperboliques à partir de plongements de mots (HierarX), de hiérarchies (ASTRE) ou encore se basant sur les *features* de l'induction de taxonomie. Les contributions sont aussi théoriques, comme avec ASTRE, AlignD et la proposition de métriques d'évaluations (*powered effective rank* ou l'auto-corrélation dimensionnelle). Elles sont aussi pratiques avec la participation à la mise en place d'un système industriel de compréhension de requêtes utilisateurs (chapitre 3.3), l'induction de taxonomies et la classification hiérarchique. Parfois, nous avons dû mettre en place des jeux de données (basés sur OpenLibrary, chapitre 8) et des protocoles d'évaluation (proposition de *d@k*, utilisation de métriques de comparaison de graphes). À chaque fois, nous avons tenté de réfléchir aux enjeux théoriques des thèmes abordés, sans abandonner le pragmatisme lié à la résolution de problèmes réels.

Perspectives

Une première extension des travaux de cette thèse concerne l'évaluation des plongements. Tous les modèles d'apprentissage profonds ont pour but de créer des plongements intermédiaires possédant les caractéristiques utiles à la résolution d'une tâche finale. Qu'en est-il de la qualité de ces plongements ? Il est sûrement possible d'évaluer la dimension empirique ou le rang effectif des ces représentations cachées, et de mettre en évidence un lien entre leurs propriétés et l'efficacité finale de la tâche. Par exemple, certains travaux tentent de forcer l'orthogonalité des plongements finaux entre les classes d'une tâche de classification (LEZAMA et al., 2018), augmentant la séparation des classes et une meilleure efficacité de classification. Pouvons-nous exporter les métriques globales et intrinsèques des plongements de mots à des plongements cachés de modèles profonds ? Ceci permettrait d'extraire

des propriétés à imposer à l'apprentissage, et mieux comprendre le fonctionnement de ces couches.

Ensuite, l'alignement de graphes de connaissances par plongements se fait en utilisant des représentations euclidiennes de ces graphes de connassances. Si on considère l'alignement de hiérarchies, il serait préférable d'utiliser des représentations hyperboliques, capables de préserver leur structure. Appliquer l'alignement entre ces plongements hyperboliques nécessitent la traduction du principe d'alignement (basé sur QVEC) en espace hyperbolique. On pourrait aussi simplement minimiser la distance hyperbolique de paires de nœuds alignés, mais on tomberait dans les mêmes désavantages que BootEA et RDGCN. D'un point de vue plus fondamental, il s'agit de généraliser la corrélation de Pearson entre les dimensions dans un espace hyperbolique. Par la suite, il reste à améliorer la génération de la *seed* avec des méthodes traditionnelles plus complexes, pour étendre la proposition à des cas plus réels. Dans le cas de DBP15k, on traite avec deux graphes de connaissances qui sont quasiment des traductions l'un de l'autre, et donc reposer uniquement sur les chaînes de caractères après traduction peut résoudre le problème. Dans un cas plus général, on ne sait pas si un alignement existe et dans quelles proportions, il faut apprendre à traiter l'alignement dans ces conditions.

ASTRE, la méthode de paramétrisation continue des arbres pourrait s'appliquer à la résolution de problèmes NP-difficile sur les arbres, pourvu qu'on puisse formuler le problème comme un critère d'optimisation. En particulier pour l'induction de taxonomies, l'extraction du MMST est difficile dans le cas général. Dans un premier temps, si on connaît la taxonomie, est-il possible de l'aligner avec le plongement des termes obtenus à partir des *features* ? Ensuite, un peu à la manière des algorithmes génétiques, il faudrait explorer l'espace des arbres en commençant par des arbres aléatoires et conserver ceux qui correspondent au mieux à au plongement des termes (*via* l'alignement), pour ensuite en produire de nouveaux en les recombinant, dans l'espoir d'obtenir l'arbre le plus proche possible du plongement. En réalité, il faudrait être capable d'avoir un arbre et sa représentation en évolution conjointe. Lorsqu'un élément est modifié sur l'arbre, que la modification soit répercutée sur la représentation hyperbolique et inversement. Lorsqu'on modifie la position d'un nœud dans la représentation, que l'arbre puisse être modifié en même temps. Au début, tous les nœuds sont connectés aléatoirement pour former un arbre, et lorsqu'on fait évoluer la représentation suivant le critère, l'arbre doit se constituer progressivement.

Grâce à ASTRE, le mélange des connaissances sous forme d'arbre et des connaissances sémantiques est facilité. Dans la même idée, si on dispose de connaissances hiérarchiques partielles sur des termes, on pourrait tenter de positionner les autres termes, grâce aux *features*, par rapport au plongement ASTRE de cette hiérarchie. On garantit ainsi que les termes, pour lesquelles des relations sont connus, respectent les connaissances a priori. De la même manière que certaines méthodes d'induction de taxonomies utilisent des taxonomies externes pour enrichir les *features*, on pourrait exploiter des taxonomies externes pour inférer des liens entre les termes et

les imposer lors de la construction du plongement. Cela ouvre la voie à des plongements mixtes entre les entités d'une hiérarchie et des entités externes d'un autre type trouvant leur place dans cette hiérarchie. Comment évaluer ce type de plongements conjoints ? Est-il possible d'intégrer ces entités externes à la hiérarchie partielle ?

La production de plongements conjoints de connaissances et de mots reste difficile à réaliser. Pourtant, posséder de tels plongements serait bénéfique à de nombreuses tâches, en particulier celles liées à la classification ou la catégorisation. En effet, l'ajout de connaissances hiérarchiques aux plongements de mots offre des caractéristiques stéréotypées, c'est-à-dire donnant des indications sur la parenté ou l'affiliation d'un concept. Par exemple, les modèles de langues actuels sont peu capables de discriminer des concepts hyponymes, à savoir qu'un *trombone* est plus proche d'une *trompette* ou d'un *cuivre* que d'un *violon* ou d'un *instrument à cordes*. C'est ce qu'on peut apercevoir en prenant la perplexité des plongements contextuels sur des phrases comme «une trompette est un cuivre» ou «une trompette est un instrument à cordes». Injecter des connaissances hiérarchiques dans la représentation résoudrait cette problématique. La multi-modalité langage-connaissances semble loin d'être nativement prise en considération par les modèles actuels, et comprendre pourquoi et comment intégrer ou extraire ces connaissances hiérarchiques dans les représentations paraît primordial. L'induction de taxonomie serait grandement facilitée en prenant les mots les plus probables remplissant les patrons de Hearst, par exemple «_ est une science». Le *clustering* ou classification hiérarchique de documents, aussi, en produisant des représentations de documents contenant des informations catégorielles et sémantiques.

Listes des publications de la thèse

— TORREGROSSA, François, Nihel KOOLI, Robin ALLESIARDO et Erwan PIGNEUL (2019), « How We Achieved a Production Ready Slot Filling Deep Neural Network Without Initial Natural Language Data », in : *Neural Information Processing - 26th International Conference*, Springer, p. 247-255

— TORREGROSSA, François, Vincent CLAVEAU, Nihel KOOLI, Guillaume GRAVIER et Robin ALLESIARDO (2020), « On the Correlation of Word Embedding Evaluation Metrics », in : *Proceedings of The 12th Language Resources and Evaluation Conference*, ELRA, p. 4789-4797

— TORREGROSSA, François, Guillaume GRAVIER, Vincent CLAVEAU et Nihel KOOLI (2020), « HierarX : un outil pour la découverte de hiérarchies dans des espaces hyperboliques à partir de similarités », in : *Extraction et Gestion des Connaissances*, É, p. 491-498

— VAIGH, Cheikh Brahim El, François TORREGROSSA, Robin ALLESIARDO, Guillaume GRAVIER et Pascale SÉBILLOT (2020), « A correlation-based entity embedding approach for robust entity linking », in : *Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence*, IEEE, p. 949-954

— TORREGROSSA, François, Robin ALLESIARDO, Vincent CLAVEAU, Nihel KOOLI et Guillaume GRAVIER (2021), « A survey on training and evaluation of

word embeddings », in : *International Journal of Data Science Analytics* 2, p. 85-103

— TORREGROSSA, François, Robin ALLESIARDO, Vincent CLAVEAU et Guillaume GRAVIER (2021), « Unsupervised Tree Extraction in Embedding Spaces for Taxonomy Induction », in : *International Conference on Web Intelligence and Intelligent Agent Technology*

Table des matières

Remerciements **iii**

Introduction générale **11**

I Plongements de mots et gestion des connaissances **17**

1 État de l'art sur les plongements de mots **19**
1.1 Les plongements : construire un espace sémantique continu à partir des mots . . . 20
1.1.1 Contexte et besoins . . . 20
1.1.2 Formalisation . . . 21
1.2 Plongements statiques . . . 22
1.2.1 Plongements euclidiens statiques . . . 22
1.2.1.1 Word2Vec . . . 22
1.2.1.2 GloVe . . . 24
1.2.1.3 FastText . . . 26
1.2.1.4 Plongements enrichis par des connaissances . . . 27
1.2.2 Plongements non euclidiens statiques . . . 27
1.2.2.1 Plongements probabilistes . . . 28
1.2.2.2 Plongements hyperboliques . . . 31
1.3 Plongements contextuels . . . 33
1.3.1 Plongements autorégressifs . . . 34
1.3.2 Plongements autoencoding . . . 35
1.4 Caractérisation des plongements . . . 36

2 Evaluation des plongements de mots **39**
2.1 Evaluations, entre complexité et nécessité . . . 40
2.2 Les modes d'évaluation . . . 40
2.2.1 Evaluations globales . . . 41
2.2.1.1 Dimensionalité intrinsèque . . . 41
2.2.1.2 Rang effectif et dimension empirique . . . 41
2.2.1.3 Auto-alignement . . . 44
2.2.2 Evaluations intrinsèques . . . 45
2.2.2.1 Similarité . . . 45
2.2.2.2 Analogie . . . 46
2.2.2.3 Catégorisation . . . 47
2.2.3 Evaluations extrinsèques . . . 49

2.2.4 Discussion : l'évaluation parfaite 50
2.3 Corrélations entre les évaluations 51
2.3.1 Expériences et résultats 51
2.3.2 Perspectives 57

3 Ajout de connaissances a priori pour la détection et classification d'entités 59
3.1 Un moteur de recherche mono-champ 60
3.2 La tâche de *Slot filling* 61
3.2.1 Rappel de la tâche et adaptation pour PagesJaunes 61
3.2.2 Solutions disponibles 61
3.3 Solution sur mesure 62
3.3.1 Première version 62
3.3.1.1 Données d'apprentissage synthétiques 62
3.3.1.2 Modèle initial 63
3.3.1.3 Collecte de données réelles 63
3.3.2 Améliorations du modèle 63
3.3.2.1 Vecteurs de connaissances 64
3.3.2.2 Pseudo-Labeling 64
3.3.3 Expériences 66
3.4 Conclusion 68

4 Application d'une métrique d'évaluation à l'alignement de bases de connaissances par plongement 69
4.1 Contexte et détails techniques 70
4.1.1 Plongement de Graphes de Connaissances 71
4.1.2 Tâche d'alignement 72
4.1.3 Historique des méthodes d'alignement 72
4.1.3.1 Parameter Swapping 74
4.1.3.2 Objectif d'alignement 74
4.1.3.3 Alignement Itératif 76
4.2 AlignD : méthode d'alignement globale par plongements 76
4.2.1 Mesurer l'alignement 76
4.2.2 Objectif total 79
4.2.3 Génération automatique de *seed* 79
4.3 Expériences 80
4.3.1 Protocole 80
4.3.2 *Seed* réelle 81
4.3.3 Taille de la *seed* et performance 82
4.3.4 *Seed* générée automatiquement 84
4.4 Conclusion 85

II Les espaces hyperboliques pour la représentation de connaissances hiérarchiques 87

5 Représentation de hiérarchies dans des espaces hyperboliques 91
5.1 Enjeux de la représentation des données 92

5.2 Plongement de hiérarchies en espace hyperbolique 93
5.2.1 Sur le principe des plongements de hiérarchies 93
5.2.2 Méthodes déterministes 95
5.2.3 Méthodes stochastiques 98
5.2.3.1 Approche intuitive sur les variétés 98
5.2.3.2 Optimisation dans un espace hyperbolique 99
5.2.3.3 Optimisation dans un espace hyperbolique avec un paramétrage euclidien 101
5.2.3.4 Plongements hyperboliques stochastiques 101
5.3 Conclusion 103

6 Paramétrisation continue d'arbres en variété riemannienne 105
6.1 Variantes algorithmiques 106
6.1.1 HyperE Rayon Variable (VR) 107
6.1.2 HyperE Initialisation par Barycentre (BI) 107
6.2 ASTRE 108
6.2.1 Modèle de base 108
6.2.2 Généralisation aux autres variétés riemanniennes 110
6.2.3 Optimisation 112
6.3 Expériences 113
6.3.1 Expériences de reconstruction 113
6.3.1.1 Protocole 113
6.3.1.2 Mesures d'évaluation 115
6.3.1.3 Résultats 115
6.3.2 Expériences de fusion 120
6.3.2.1 Protocole 122
6.3.2.2 Mesures d'évaluation 123
6.3.2.3 Résultats 123
6.4 Conclusion 125

7 Extraction d'arbre dans des plongements hyperboliques 127
7.1 Besoins 128
7.2 Etat de l'art 129
7.2.1 Historique des méthodes 129
7.2.2 Composants principaux 131
7.2.2.1 *Feature Engineering* 131
7.2.2.2 Plongements hyperboliques 132
7.2.2.3 Minimisation de la distortion 133
7.3 Méthodologie proposée 134
7.3.1 Combinaison de *features* syntaxiques et lexicales 134
7.3.1.1 Généralités. 134
7.3.1.2 Corpus. 134
7.3.1.3 *Features* syntaxiques. 134
7.3.1.4 *Feature* lexicale. 135
7.3.1.5 *Feature* aggrégée. 135
7.3.2 Plongement des termes 136
7.3.3 Extraction d'arbre 137

7.3.3.1 Extraction Naïve (NE). 137
7.3.3.2 MST Extraction (MSTE). 138
7.3.3.3 Low Average Stretch Extraction (LASE). 138
7.3.4 Raffinage de la taxonomie 138
7.4 Expériences 138
7.4.1 Protocole 138
7.4.2 Evaluation 139
7.4.3 Reconstruction de taxonomies 141
7.4.4 Induction de taxonomie 142
7.5 Conclusion 145

8 Difficultés de la classification hiérarchique 147
8.1 Classification hiérarchique de textes 148
8.1.1 Description de la tâche 148
8.1.2 État de l'art 148
8.2 Élaboration d'un corpus sur-mesure 150
8.3 Méthodologie 151
8.3.1 Représentation des documents 151
8.3.2 Passage de vecteurs euclidiens vers des vecteurs hyperboliques 151
8.3.3 HMCN préfixé d'une projection hyperbolique (HYP-HMCN) 152
8.3.4 *Baseline* BERT 153
8.4 Expériences 153
8.4.1 Métriques d'évaluation 154
8.4.2 Résultats 154
8.5 Améliorations effectuées 155
8.5.1 Plongement des classes 155
8.5.2 Conversion séquentielle locale des vecteurs euclidiens 155
8.5.3 Résultats 157
8.6 Conclusion 158

Conclusion générale 159

Table des matières 165

Table des figures 171

Liste des tableaux 175

Liste des acronymes 177

Annexes 180

A Résultats d'ASTRE dans d'autres dimensions (SemEval) 183
A.1 2 dimensions 183
A.2 3 dimensions 185
A.3 10 dimensions 187

B Résultats d'ASTRE dans d'autres dimensions (WordNet) 189
B.1 2 dimensions 189
B.2 3 dimensions 191
B.3 10 dimensions 192

C Un logiciel pour la transformation de plongements euclidiens en plongements hyperboliques et la découverte de connaissances hiérarchiques. 193
C.1 Interface 194
C.2 Fonctionnement 195
C.3 Expériences 196
C.3.1 Découverte de hiérarchies 196
C.3.2 Réduction de dimensions 197
C.4 Limites 198
C.5 Conclusion 199

Bibliographie 203

Bibliographie 203

Résumé/Abstract 217

Table des figures

1.1 Vecteurs des capitales et leur pays issus d'un plongement FastText (BOJANOWSKI et al., 2017) dont la dimension est réduite à 2 dimensions par une PCA. 21
1.2 Fenêtre de taille k autour d'un mot cible w_i dans un corpus T. Les mots du contexte sont alors $\{w_{i-k}, w_{i-(k+1)}, ..., w_{i-1}, w_{i+1}, ..., w_{i+(k-1)}, w_{i+k}\}$. . 23
1.3 Les modèles SG et CBoW. À gauche, le modèle SG qui prédit des mots de la fenêtre du contexte connaissant le mot cible. À droite, le modèle CBoW utilisant les mots du contexte pour prédire le mot cible (adaptée de MIKOLOV, K. CHEN et al., 2013). 23
1.4 Performances de plongements FastText en fonction de leur dimension. 28
1.5 Example de contraintes de $\mathbb{R}^d$, se basant sur K. SUN et al., 2015. 29
1.6 Exemple d'une hiérarchie plongée dans un espace probabiliste, comme expliqué par VILNIS et MCCALLUM, 2015 30
1.7 Représentation des plus courts chemins (en pointillés) dans le disque de Poincaré $\mathbb{D}^2$. Chaque marque périphérique est à la même distance de la marque centrale, à noter que pour chacune le plus proche voisin est la marque centrale. 32
1.8 Espace de Lorentz $\mathcal{L}^2$, constitué de l'ensemble des points de la surface conique. 33
1.9 Modélisation autorégressive d'une séquence de tokens (t_1, t_2, t_3, t_4). Le modèle du dessus illustre une factorisation en avant et celui du dessous une factorisation en arrière. 34
1.10 Modélisation masquée du langage, utilisée pour entraîner des plongements autoencoding. Les tokens sont aléatoirement remplacé par un token `[MASK]`. À la fin, la représentation vectorielle de `[MASK]` est donnée à une couche de classification sur le vocabulaire. L'objectif est de prédire le token caché par `[MASK]`. 36

2.1 Exemple pour montrer le comportement de perank et edim. Une matrice contenant deux vecteurs à deux dimensions est construite en prenant les vecteurs de 2.1a. Les valeurs de perank et edim sont calculées en fonction de l'angle $\theta \in [0, \pi]$. 43
2.2 Valeur de la dimension j d'un plongement B en fonction de celle de la dimension i d'un plongement A, pour des mots aléatoires (marques rouges). 44
2.3 Tâches intrinsèques 48

2.4 Matrice de corrélation de Pearson entre les évaluations. Les trois premières colonnes sont des tâches extrinsèques, les suivantes sont intrinsèques, et la dernière est globale (ACD médiane). 52
2.5 Matrices de corrélation de perank (2.5a) et edim (2.5b) avec les autres évaluations. Chaque ligne correspond à une valeur de p différente au formatperank$-\{p\}$ et edim$-\{p\}$. 53
2.6 Performances des tâches extrinsèques et intrinsèques en fonction des évaluations globales. 54
2.7 Corrélations de Spearman des métriques globales avec la dimension du plongement (en fonction du paramètre p pour edim et perank). 55

3.1 Interface du moteur de recherche mono-champ de PagesJaunes 61
3.2 Exemple d'un patron syntaxique. 63
3.3 Décomposition d'une requête en n-grammes et création des vecteurs de connaissances à partir des bases de données de Solocal. 65
3.4 Performances des modèles sur l'ensemble de validation en fonction des améliorations proposées. 67

4.1 Calcul de l'objectif d'alignement des méthodes actuelles. 75
4.2 Processus de calcul de la matrice de corrélation A_D à partir d'une *seed* A'. .. 78
4.3 Processus de génération automatique de *seed*. 79
4.4 Influence de la taille de la *seed* (pourcentage de la totalité des alignements disponibles, en abscisse) sur la performance (Hits@1, en ordonné) pour BootEA et AlignD. 83

5.1 Plongement d'un arbre binaire dans la boule de Poincaré $\mathbb{D}^2$ (la limite est illustrée par la ligne en tiret noir). Le chemin le plus court de A à B dans l'arbre (en bleu plein) est montré en relation de la distance dans $\mathbb{D}^2$ entre A et B (en pointillés bleus). La distortion est le produit des rapports maximaux (des deux sens) entre les distances de chaque paire de nœuds. ... 94
5.2 Construction de SARKAR, 2011. Tous les cercles ont la même surface et sont de même rayon. 96
5.3 Déplacement d'un cercle centré sur $\vec{0}$ (en noir) vers deux autres emplacements (rouge et bleu) dans $\mathbb{D}^2$. Les distances sont préservées. 96
5.4 Plongement de la hiérarchie *environnement* de la tâche 14 de SemEval-2016 (BORDEA, LEFEVER et BUITELAAR, 2016) dans $\mathbb{D}^2$, avec plusieurs rayons (2 et 4). Les plongements des nœuds sont les points rouges et les liens de la hiérarchie sont représentés par des lignes pleines noires. 97
5.5 Illustration de l'application d'une expmap en un point x sur une variété $\mathcal{M}$. La longueur des segments rouges est identique. 99
5.6 Illustration du transport parallèle $\varphi^{\mathcal{M}}_{\theta_t \to \theta_{t+1}}$ entre deux points θ_t et θ_{t+1}. 100

6.1 Trois variantes d'HyperE utilisant VR ou BI ou les deux (VR+BI) sur la taxonomie environnement de Semeval-2016 task 13 (BORDEA, LEFEVER et BUITELAAR, 2016) avec $\tau = 2$. 106

6.2 Transformation de l'espace des paramètres de ASTRE (à gauche) vers $\mathbb{D}^2$ (à droite) pour un nœud a avec pour parent b et successeurs (c_1, c_2). Sur la Figure de droite, la distance entre deux nœuds est telle que $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(b)) = \tau_a$, $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(c_1)) = \tau_{c_1}$ et $d_{\mathbb{D}^2}(\mathcal{A}(a), \mathcal{A}(c_2)) = \tau_{c_2}$, et les angles θ_{c_1}, θ_{c_2} sont préservés entre les tangentes en $\mathcal{A}(a)$..... 109
6.3 Illustration de l'application de reflect$_{x \to y}$ en un point x sur une variété $\mathcal{M}$. Les vecteurs rouges ont tous la même longueur. 111
6.4 Méthodes de plongements de hiérarchies sur la taxonomie environnement de la tâche 13 de SemEval-2016 (BORDEA, LEFEVER et BUITELAAR, 2016), dans $\mathbb{D}^2$. 116
6.5 MAP, F1-score de l'arbre, et d@5 de plongements ASTRE et d'HyperE (et variantes) sur les taxonomies en anglais de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016). La dimension des plongement vaut 5..... 117
6.6 MAP, F1-score de l'arbre, et d@5 de plongements ASTRE et d'HyperE (et variantes) sur WordNet (G. A. MILLER, 1995). La dimension des plongement vaut 5. 120
6.7 Comparaison entre FreeASTRE et Free$\mathcal{M}$ sur 19 sous-hiérarchies de WordNet (on prend le score moyen sur chaque axe), et 3 variétés différentes (de haut en bas : $\mathbb{R}^n$, $\mathbb{D}^n$, and $\mathcal{H}^{n,\beta}$), dans la tâche de fusion sémantique et hiérarchique. L'incorporation de l'information sémantique augmente avec α (de gauche à droite), pendant que l'information hiérarchique décroît. 124

7.1 Etude de la corruption des arbres sur l'évaluation. 140
7.2 Plongement à deux dimensions sur des sous-taxonomies de WordNet (G. A. MILLER, 1995) et leur arbre extrait. 143

8.1 HMCN avec une étape de projection hyperbolique antérieure. 153
8.2 Modèle de projection séquentiel. 156

A.1 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (2 dimensions). 183
A.2 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (3 dimensions). 185
A.3 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (10 dimensions). 187

B.1 MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 2)..... 189
B.2 MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 3)..... 191
B.3 MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 10)..... 192

C.1 Exemples des deux formats possibles pour HierarX. 194

C.2 Projection dans l'espace de Poincaré (dimension 2) des concepts appartenant aux instruments de musique dans la source de données WordNet. Le résultat est obtenu en fournissant à HierarX une matrice de similarités où la similarité entre deux concepts se base sur le plus court chemin les reliant dans WordNet. 197
C.3 Projection dans l'espace de Poincaré (dimension 2) des concepts appartenant aux instruments de musique dans la source de données WordNet. Le résultat est obtenu en fournissant à HierarX un plongement FastText de 300 dimensions obtenu en ligne [4]. 198

4. `https://fasttext.cc/docs/en/english-vectors.html`

Liste des tableaux

1.1 Propriétés des algorithmes de plongement. 37

2.1 Jeux de données de similarité (BAKAROV, 2018). 46
2.2 Jeux de données d'Analogie (BAKAROV, 2018). 47
2.3 Jeux de données pour la catégorisation (BAKAROV, 2018). 47
2.4 Evaluations extrinsèques (Précision : P, Rappel : R, F1-score : F1). 50
2.5 Distributions des hyper-paramètres utilisées pour générer les plongements FastText. 52

3.1 Exemple de *slot filling* (mode IOB) sur une requête. 61
3.2 Performance des solutions commerciales sur la tâche de *slot filling* sur des requêtes manuellement étiquetés. La précision est la proportion de requêtes dans lesquelles les champs ont été correctement identifiés. 62
3.3 Performances de plusieurs algorithmes de *slot filling* évaluées dans le cadre de PagesJaunes (les détails expérimentaux sont dans la section 3.3.3). La précision est la proportion de requêtes dans lesquelles les champs ont été correctement identifiés. 64
3.4 Distribution des hyper-paramètres. 66
3.5 Résultats obtenus avec le RNN, avec le vecteur de connaissances (KV) et/ou avec le pseudo-labeling (PL). La précision est calculée sur les patrons syntaxiques, l'ensemble de validation et test. 67

4.1 Quantité d'éléments dans les KGs de DBP15K. 80
4.2 Hits@1, Hits@10 et MRR avec une *seed* réelle (représentant 30% de la taille total des alignements) sur DBP15K. La partie supérieure du tableau reprend les résultats publiés dans les études respectives. Les résultats des deux parties en-dessous sont produits via nos expériences. 82
4.3 Hits@1 pour des méthodes symboliques seules ou en combinaison avec AlignD. 84
4.4 Hits@1 pour des méthodes d'alignement par plongement utilisant une *seed* générée automatiquement par approche symbolique. 84

6.1 Statistiques de WordNet (G. A. MILLER, 1995) et SemEval-2016 13 BORDEA, LEFEVER et BUITELAAR, 2016. 114

6.2 MAP, F1-score de l'arbre et d@5 d'ASTRE, HyperE (et variantes) et approches stochastiques sur les hiérarchies Science (Sci.) et Alimentation (Al.) de SemEval 2016 tâche 13 (BORDEA, LEFEVER et BUITELAAR, 2016). τ est choisi via les résultats sur la hiérarchie *environnement*, pour les méthodes requérant une valeur (mises en valeur avec *). La dimension vaut 5. Les deux meilleurs résultats sont en gras. 119
6.3 MAP, F1-score de l'arbre et d@5 d'ASTRE, HyperE (et variantes) et approches stochastiques sur WordNet (G. A. MILLER, 1995). τ est choisi via les résultats sur la hiérarchie *environnement*, pour les méthodes requérant une valeur (mises en valeur avec *). La dimension vaut 5. Les deux meilleurs résultats sont en gras. 121

7.1 Étude d'ablation de f_{agg} par rapport à ses composantes, sur la taxonomie environnement de SemEval (BORDEA, LEFEVER et BUITELAAR, 2016). La *feature* est utilisée pour trier le voisinage d'un terme, et on calcule la MAP par rapport aux liens direct de la taxonomies. * indique une p-valeur inférieure à 0.01 pour un test d'hypothèse nulle $f_{\text{agg}} = \text{Others}$. 136
7.2 Résultats de reconstruction de deux sous-taxonomies de WordNet (G. A. MILLER, 1995). Les dimensions sont choisies sur la sous-taxonomie *communication.n.02* avec le F1-score direct. Les meilleurs résultats sont en gras. 142
7.3 Induction de taxonomie sur la tâche 13 de SemEval 2016 (BORDEA, LEFEVER et BUITELAAR, 2016). P-valeurs des expériences * : $p < 0.1$, ** : $p < 0.05$, *** : $p < 0.01$, pour un test de l'hypothèse nulle TAXI (raf.) = Méthode. Les meilleurs résultats sont en gras. 144

8.1 Quantité de documents par niveau de profondeur dans le jeu de données extrait des dumps d'OpenLibrary. 151
8.2 Quantité de documents par catégorie principale dans le jeu de données extrait des dumps d'OpenLibrary. 151
8.3 Evaluation de la classification des modèles HMCN sur la catégorie 500 d'OpenLibrary 154
8.4 Evaluation des plongements conjoints sur le corpus extrait d'OpenLibrary (en entier). 157

A.1 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (2 dimensions). 184
A.2 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (3 dimensions). 186
A.3 Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (10 dimensions). 188

B.1 Resultats sur WordNet (G. A. MILLER, 1995) (dimension 2). 190
B.2 Resultats sur WordNet (G. A. MILLER, 1995) (dimension 3). 191
B.3 Resultats sur WordNet (G. A. MILLER, 1995) (dimension 10). 192

C.1 Principaux arguments de HierarX. Les arguments dans la partie supérieure du tableau sont obligatoires. 195

C.2 Résultats avant/après la compression d'un plongement FastText de 300 dimensions dans un espace de Poincaré à 20 dimensions. Le coefficient de Spearman est reporté pour chaque corpus (les pvaleurs sont très faibles). Le vocabulaire est composé des 100000 mots les plus fréquents. Pour HyperLex, le score is_a défini par NICKEL et KIELA, 2017 est utilisé pour l'espace hyperbolique. Pour WordSim, la distance hyperbolique est choisie. Pour FastText, la similarité cosinus sert pour le calcul des similarités dans les deux cas. 198

Liste des acronymes

ACD *Auto-Corrélation Dimensionnelle*

AUC *aire sous la courbe* (Area Under the Curve)

CBoW *Continuous Bag-of-Words*

CNN *réseau de neurones de convolution* (Convolutional Neural Network)

CRF *champ aléatoire conditionnel* (Conditional Random Field)

DAG *graphe dirigé acyclique* (Directed Acyclic Graph)

DIH *Distributional Inclusion Hypothesis*

EL *Expected Likelihood*

GCN *Réseau Convolutif de Graphe* (Graph Convolutional Network)

GPT *Generative Pre-Training*

IA *Intelligence Artificielle*

KG *graphe de connaissances* (Knowledge Graph)

divergence K-L *divergence de Kullback-Leibler*

MAP *Mean Average Precision*

MMST *arbre couvrant d'expansion-maximale minimale* (Minimum Max-strech Spanning Tree)

MR *rang moyen* (Mean Rank)

MRR *Mean Reciprocal Rank*

MST *arbre couvrant de poids minimal* (Minimum Spanning Tree)

NER *reconnaissance d'entités nommées* (Named Entity Recognition)

NS *échantillonnage négatif* (Negative Sampling)

OOV *hors vocabulaire* (Out Of Vocabulary)

PPMI *information mutuelle positive par point* (Positive Pointwise Mutual Information)

RDF *Resource Description Framework*

RNN *réseau de neurones récurrent* (Recurrent Neural Network)

SA *analyse de sentiment* (Sentiment Analysis)

SG *Skip-Gram*

SGD *descente de gradient stochastique* (Stochastic Gradient Descent)

SST *Stanford Sentiment Treebank*

SVD *décomposition en valeurs singulières* (Singular Value Decomposition)

SVM *machine à vecteur de support* (Support Vector Machine)

TALN *Traitement Automatique du Langage Naturel*

TC *classification de textes* (Text Classification)

URI *Uniform Resource Identifier*

Annexes

Annexe A

Résultats d'ASTRE dans d'autres dimensions (SemEval)

A.1 2 dimensions

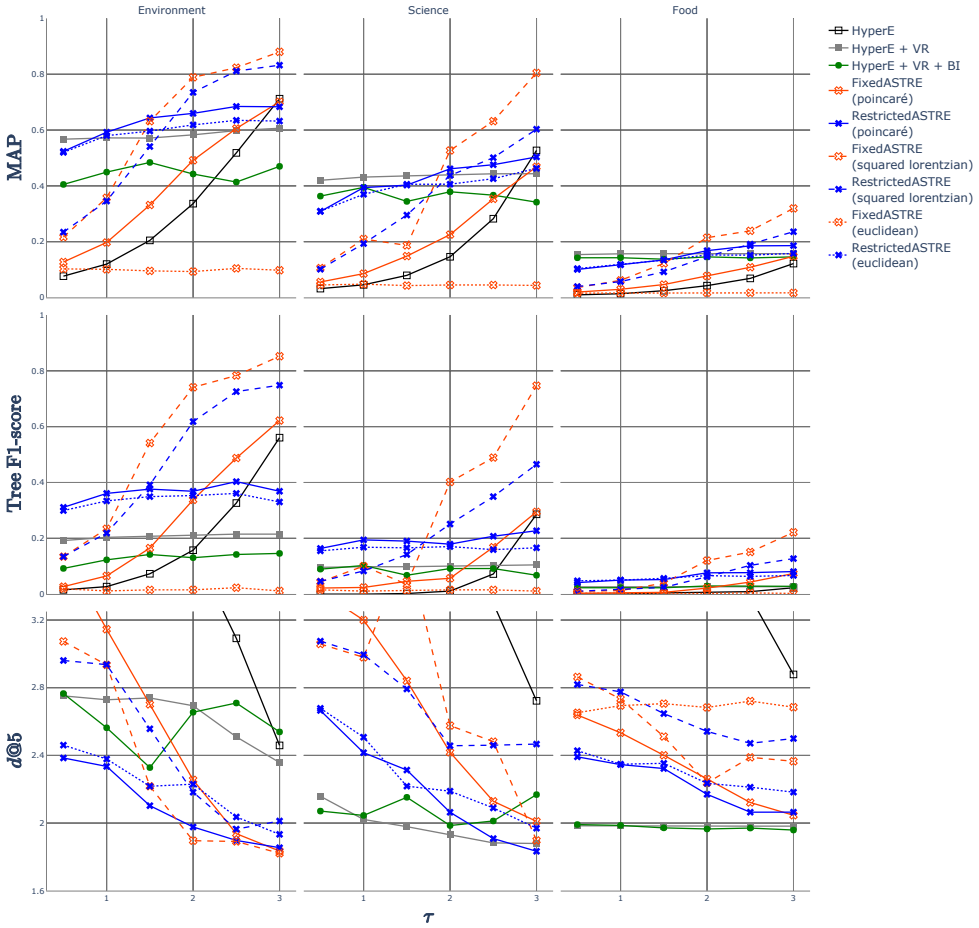


FIGURE A.1 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (2 dimensions).

TABLEAU A.1 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (2 dimensions).

	Méthodes	MAP		F1-score de l'arbre		d@5	
		Sci.	Al.	Sci.	Al.	Sci.	Al.
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.8047**	0.3194	**0.7467**	0.2215	1.8973	2.3652
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.6030	0.2358	0.4651	0.1273	2.4668	2.4997
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.5205	0.2397	0.3668	0.1509	2.6779	2.4559
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4683	0.1476	0.2948	0.0738	2.0102	2.0461
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5036	0.1860	0.2271	0.0808	**1.8332**	2.0645
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5053	0.2029	0.2052	0.0764	**1.8394**	1.9771
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.0479	0.0171	0.0109	0.0025	3.5575	2.6833
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.4627	0.1581	0.1659	0.0668	1.9695	2.1826
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.4797	0.2031	0.1441	0.0707	1.8903	**1.9564**
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5270	0.1217	0.2860	0.0229	2.7226	2.8785
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4435	0.1571	0.1048	0.0267	1.8792	1.9821
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.3946	0.1456	0.1026	0.0293	2.0451	**1.9650**
Approches Stochastiques	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.9279**	**0.7767**	**0.8832**	**0.7148**	2.3239	2.1287
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.6297	0.2654	0.3843	0.0840	2.0606	1.9680
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1585	0.0496	0.0917	0.0280	2.0190	2.0562
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.3460	0.0903	0.2162	0.0382	2.5000	2.8810

A.2 3 dimensions

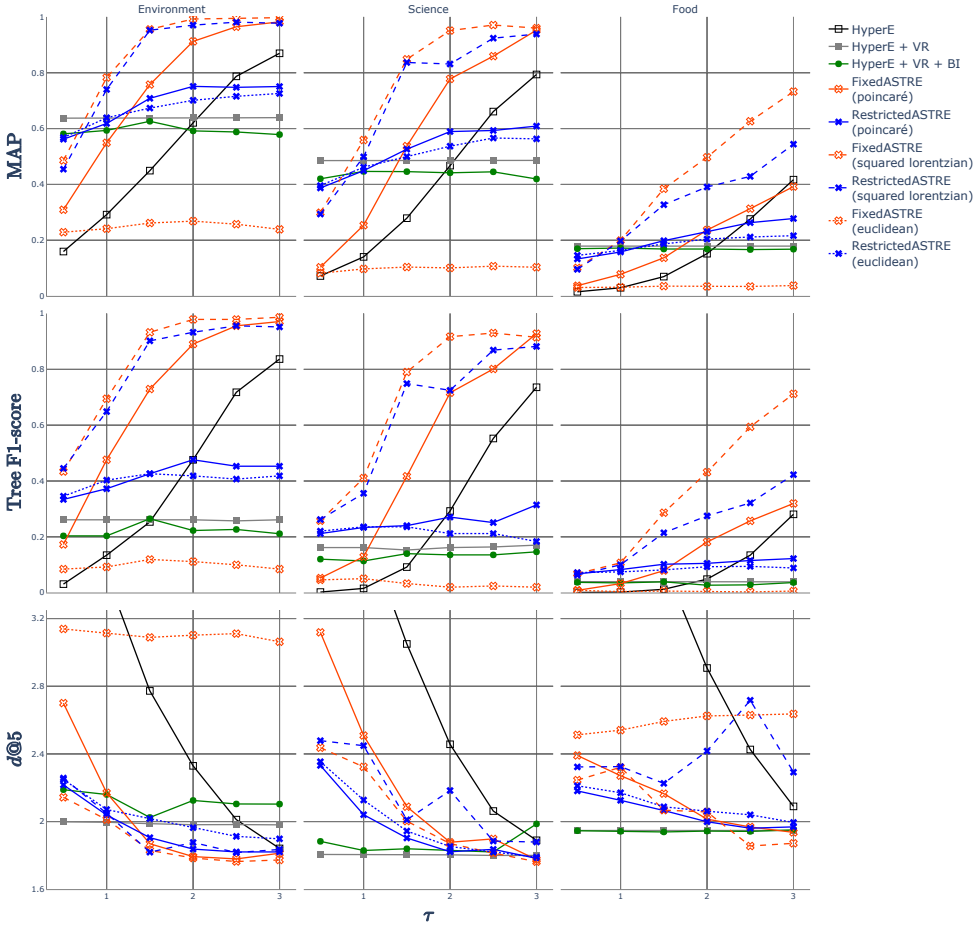


FIGURE A.2 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (3 dimensions).

TABLEAU A.2 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (3 dimensions).

Méthodes		MAP		F1-score de l'arbre		d@5	
		Sci.	Al.	Sci.	Al.	Sci.	Al.
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.9709**	0.7326	**0.9301**	**0.7123**	1.8142	**1.8717**
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.9385	0.5440	0.8821	0.4227	1.8792	2.2920
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.8475	0.5365	0.7576	0.4125	2.3575	2.3635
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.9526**	0.3921	0.9279	0.3195	**1.7752**	1.9362
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6089	0.2778	0.3144	0.1222	**1.7827**	1.9688
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5807	0.2586	0.2489	0.1082	1.8035	1.9314
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.1072	0.0377	0.0240	0.0057	3.4195	2.6365
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.5662	0.2163	0.2118	0.0885	1.8204	1.9950
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.5785	0.2425	0.1987	0.0840	1.7903	1.9283
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7938	0.4169	0.7358	0.2807	1.8894	2.0898
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4860	0.1790	0.1703	0.0388	1.8000	1.9476
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4465	0.1720	0.1135	0.0350	1.8292	1.9424
Stochastic approaches	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9520	**0.9729**	**0.9400**	**0.9491**	2.2681	2.3040
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.8830	**0.7427**	0.7686	0.5805	1.7898	**1.8055**
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1856	0.0618	0.1048	0.0363	1.9314	2.0127
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.8432	0.3836	0.7140	0.2667	2.2903	2.6667

A.3 10 dimensions

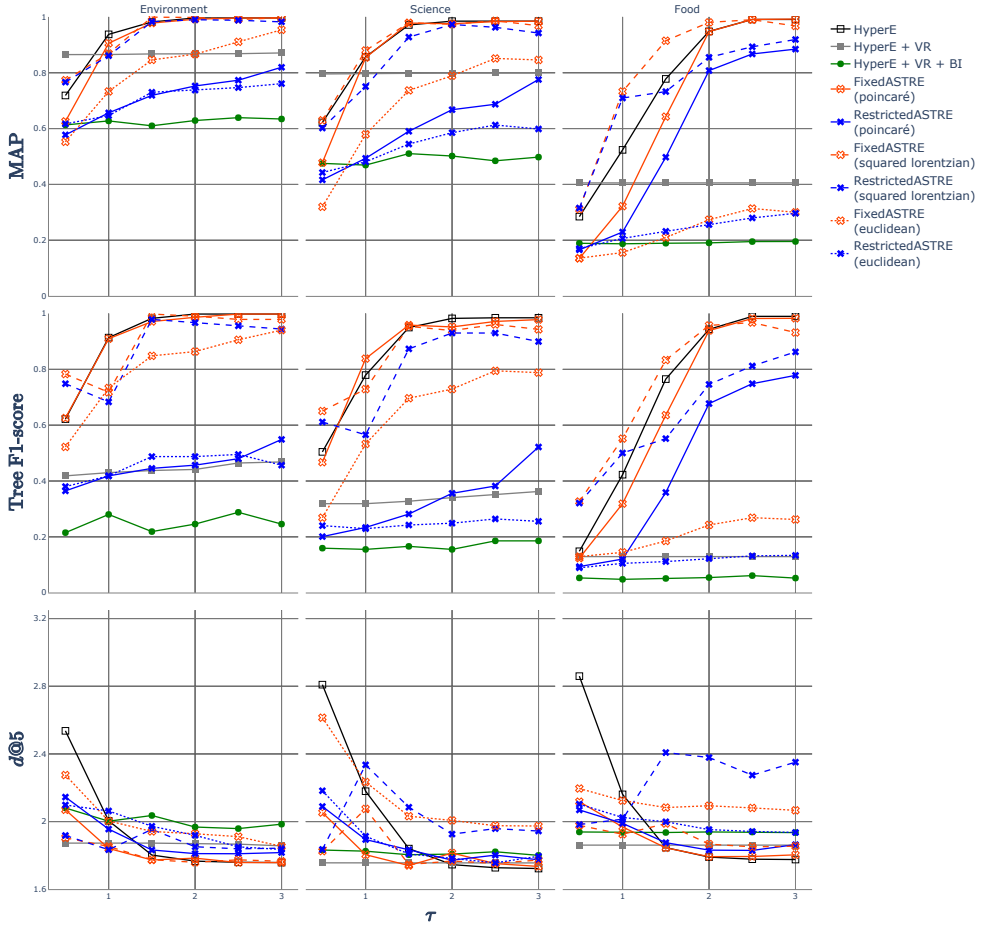


FIGURE A.3 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) avec les méthodes déterministes (10 dimensions).

TABLEAU A.3 – Résultats d'ASTRE sur SemEval (BORDEA, LEFEVER et BUITELAAR, 2016) (10 dimensions).

Méthodes		MAP		F1-score de l'arbre		d@5	
		Sci.	Al.	Sci.	Al.	Sci.	Al.
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.9863**	0.9900	0.9607	0.9675	1.7535	1.8508
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.9730	0.9190	0.9301	0.8625	1.9261	2.3513
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.9716	0.8780	0.9192	0.7842	1.9686	2.3713
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9847	**0.9923**	**0.9782**	**0.9828**	**1.7350**	**1.8035**
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7758	0.8846	0.5218	0.7785	1.7743	1.8639
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6037	0.4910	0.2838	0.1827	1.7518	1.8482
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.8513	0.3137	0.7948	0.2686	1.9757	2.0807
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.6127	0.2966	0.2642	0.1337	1.7562	1.9357
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.5904	0.2775	0.2052	0.0993	1.7690	1.9117
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9849	0.9909	**0.9847**	**0.9898**	**1.7226**	**1.7757**
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7994	0.4054	0.3624	0.1292	1.7558	1.8609
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5103	0.1955	0.1659	0.0522	1.8040	1.9350
Stochastic approaches	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.9620	0.9806	0.9705	0.9586	2.1420	2.4605
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.9528	0.9522	0.9039	0.8994	1.7699	1.8852
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.2041	0.0732	0.1441	0.0516	1.9146	1.9897
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	**0.9917**	**0.9967**	0.9672	0.9739	1.8695	2.2904

Annexe B

Résultats d'ASTRE dans d'autres dimensions (WordNet)

B.1 2 dimensions

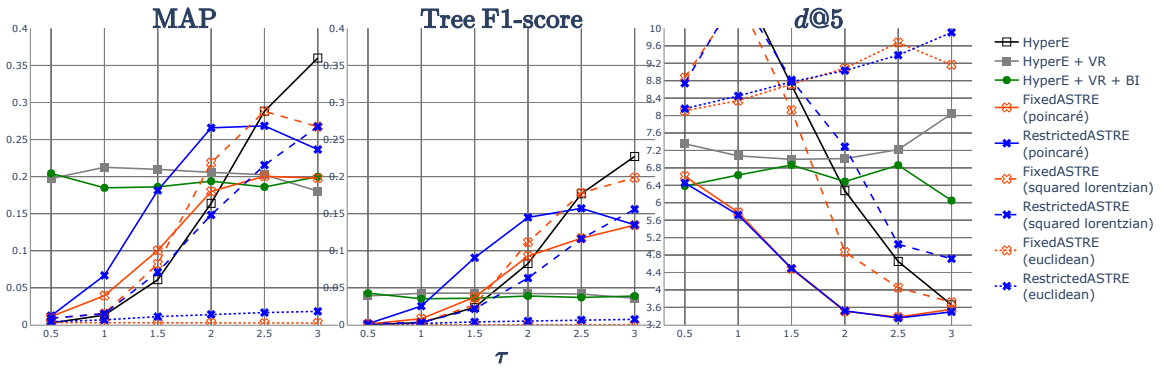


FIGURE B.1 – MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 2).

TABLEAU B.1 – Resultats sur WordNet (G. A. MILLER, 1995) (dimension 2).

	Méthodes	MAP	F1-score de l'arbre	d@5
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.2881	0.1780	4.0516
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.2675	0.1560	4.7150
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.0817	0.0344	8.2294
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.2003	0.1168	**3.3766**
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.2685	0.1571	**3.3558**
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.3546**	**0.1976**	3.4398
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.0029	0.0001	8.1086
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.0181	0.0073	9.9074
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.2093	0.0783	6.2692
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.3600**	**0.2272**	3.6564
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.2125	0.0425	7.0758
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.2044	0.0427	6.3842
Approches Stochastiques	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1594	0.0559	10.0938
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.3065	0.0974	6.3948
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.0119	0.0013	6.5874
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.0142	0.0019	12.6838

B.2 3 dimensions

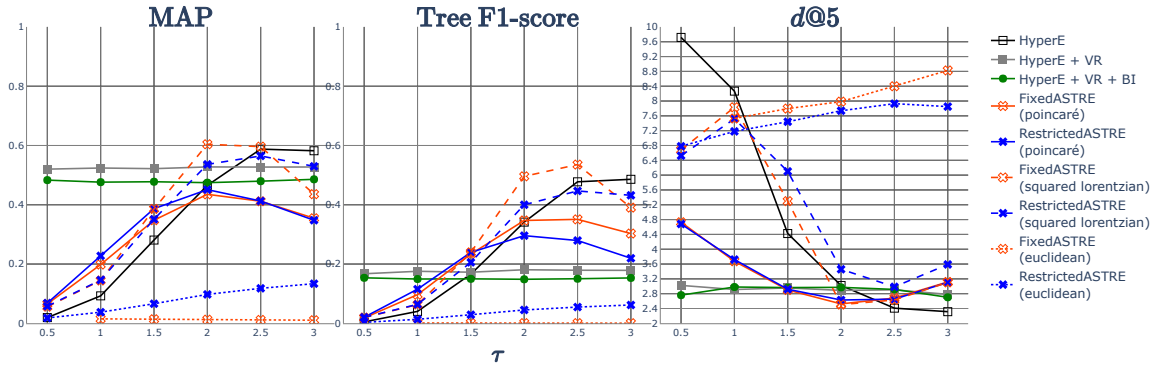


FIGURE B.2 – MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 3).

TABLEAU B.2 – Resultats sur WordNet (G. A. MILLER, 1995) (dimension 3).

	Méthodes	MAP	F1-score de l'arbre	d@5
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.6042**	**0.4959**	**2.5054**
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.5647	0.4466	2.9848
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.2343	0.1349	6.3478
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4352	0.3470	2.5246
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4510	0.2958	2.6292
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5162	0.3109	2.8060
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.0150	0.0011	7.5298
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.1337	0.0619	7.8498
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.4162	0.1757	3.8312
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.5877**	**0.4774**	**2.4090**
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5276	0.1808	2.8888
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4858	0.1531	2.7026
Approches Stochastiques	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.4922	0.2325	4.9206
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.4165	0.1549	4.1734
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.0662	0.0338	3.8146
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.2341	0.0796	7.5148

B.3 10 dimensions

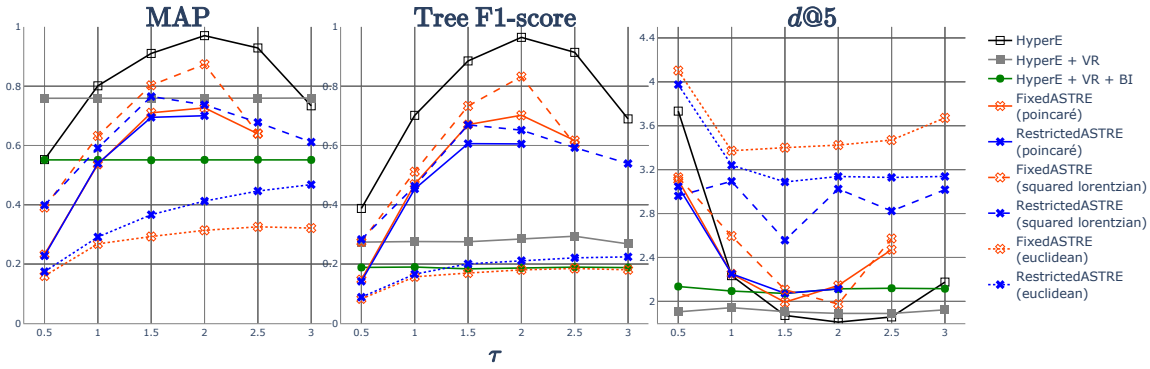


FIGURE B.3 – MAP, F1-score de l'arbre et d@5 d'ASTRE et HyperE (et variantes) sur WordNet G. A. MILLER, 1995 (dimension 10).

TABLEAU B.3 – Resultats sur WordNet (G. A. MILLER, 1995) (dimension 10).

	Méthodes	MAP	F1-score de l'arbre	d@5
ASTRE	Fixed* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	**0.8741**	**0.8324**	1.9750
	Restricted* $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.7652	0.6695	2.5568
	Free $(\mathbb{D}^n, d_{\mathcal{H}^{n,\beta}})$	0.4429	0.3257	4.2428
	Fixed* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7269	0.7019	2.1450
	Restricted* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7004	0.6053	2.1132
	Free $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6961	0.4834	2.2138
	Fixed* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.3255	0.1850	3.4694
	Restricted* $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.4680	0.2242	3.1382
	Free $(\mathbb{R}^n, d_{\mathbb{R}^n})$	0.5304	0.2277	2.2946
HyperE	Original* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	**0.9706**	**0.9648**	**1.8108**
	VR* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.7601	0.2679	**1.9228**
	VR + BI* $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.5521	0.1895	2.1194
Approches Stochastiques	PoincareModel $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.6285	0.4138	3.2078
	LorentzModel $(\mathcal{H}^n, d_{\mathcal{L}})$	0.4315	0.1704	2.3272
	HypCones $(\mathbb{D}^n, d_{\mathbb{D}^n})$	0.1465	0.0888	2.3946
	Law19a $(\mathbb{R}^n, d_{\mathcal{H}^{n,\beta}})$	0.6882	0.3369	2.8564

Annexe C

Un logiciel pour la transformation de plongements euclidiens en plongements hyperboliques et la découverte de connaissances hiérarchiques.

```
5 2
hierarx 0.37961 0.52684
learns 0.25774 0.52448
hyperbolic 0.09547 0.51548
word 0.093367 0.47123
representations 0.1313 1.0844
```

(A) Format GloVe (PENNINGTON, SOCHER et MANNING, 2014) pour l'entrée plongement.

```
Left,signals,3.395
coaches,Todd,4.502
Todd,aware,2.446
aware,signals,4.765
bridges,kilometers,7.450
intellectual,fundamental,7.773
```

(B) Format CSV pour l'entrée matrice de similarité.

FIGURE C.1 – Exemples des deux formats possibles pour HierarX.

Ici nous présentons le logiciel HierarX [1] permettant la transformation des plongements euclidiens, ou d'une fonction de similarité $s(x, y)$ entre éléments quelconques x et y, en un plongement hyperbolique. Ce logiciel a été élaboré pour procéder de façon efficace à cette transformation (avec des vocabulaires de plusieurs milliers d'éléments), et se base essentiellement sur les travaux de NICKEL et KIELA, 2018, et d'implémentations d'espaces hyperboliques comme geoopt (KOCHUROV, KARIMOV et KOZLUKOV, 2020). Le but est de construire ou de découvrir une structure hiérarchique dans la source de données plate (similarités), *via* la géométrie hyperbolique. On suppose que cette géométrie et l'incorporation de la similarité dans la distance entre les vecteurs hyperboliques des éléments fera en sorte que les éléments s'organisent continûment dans l'espace pour former une hiérarchie.

La force de l'idée est que, si cela est possible, la découverte de hiérarchie continue peut s'appliquer à n'importe quel type de données pourvu qu'on sache définir une similarité. Ensuite reste l'interprétation de l'espace hyperbolique en une hiérarchie discrète, c'est-à-dire, la production d'un graphe dirigé acyclique connexe minimisant la distortion des distances entre les vecteurs et les nœuds du graphe extrait. Nous étudions cette problématique d'extraction dans le chapitre 7.

C.1 Interface

Il y a deux formats possibles en entrée de HierarX correspondant aux deux entrées prises en charge : un plongement ou une matrice de similarités. Des exemples sont présentés en figure C.1. Pour le plongement, nous reprenons le format standard GloVe : la première ligne décrit le nombre de mots ainsi que la dimension, et les lignes suivantes contiennent un mot suivi de ses coordonnées dans la représentation. Pour la matrice, nous prenons un format csv de trois colonnes : mot1,mot2,s(mot1, mot2). Les deux premières colonnes détaillent la paire de mots et la dernière la similarité entre les mots précédents.

Un ensemble d'arguments contrôle le déroulement de l'entraînement ce qui produit des plongements hyperboliques avec différentes propriétés. Ils sont listés dans le tableau C.1.

1. https://github.com/pagesjaunes/HierarX

TABLEAU C.1 – Principaux arguments de HierarX. Les arguments dans la partie supérieure du tableau sont obligatoires.

Argument	Type (valeur par défaut)	Explication
-nvoc	int	nombre de mot dans le vocabulaire du plongement hyperbolique
-dim	int	dimension du plongement
-thread	int	nombre de thread optimisant simultanément
-input	string	chemin vers le plongement (format GloVe) ou la matrice de similarité (format CSV)
-expdir	string	répertoire d'expérience (où certains fichiers sont sauvegardés)
-lr	float (0.01)	facteur d'apprentissage
–similarity	flag	indique que l'entrée est une matrice de similarité
-momentum	float (0.9)	facteur du momentum
-bs	int (10)	taille du batch
-sampling	int (20)	nombre d'éléments par élément du batch (1 élément positif et le reste négatif)
-posthres	float (0.1)	proportion d'éléments proches introduit dans les négatifs
-plateau	float (0.1)	pourcentage d'époque avec un facteur d'apprentissage divisé par 10 (au début)
-hmode	string ('Poincare')	Nom du manifold parmi 'Poincare', 'PoincareStack', 'Lorentz'.

À la fin de la phase d'optimisation, il faudra convertir le plongement (sauvegardé en format binaire) dans un format GloVe (similaire à celui passé en entrée). Pour ce faire, HierarX propose un convertisseur intégré permettant d'effectuer l'opération. Davantage d'informations sont détaillées dans le readme du projet.

C.2 Fonctionnement

Le processus d'optimisation repose sur des similarités entre des paires de mots. Elles peuvent être fournies *via* un plongement de type FastText (au format GloVe) ou alors *via* une matrice de similarités. Dans le cas d'un plongement, la similarité est déduite du produit scalaire entre les vecteurs. Pour bénéficier de la similarité cosinus, les vecteurs doivent être normalisés.

Nous reprenons la méthode proposée par NICKEL et KIELA, 2018 pour incorporer des similarités dans des espaces hyperboliques. L'idée est d'incorporer la similarité de façon à ce qu'elle organise le voisinage de chaque point de l'espace. Faisons l'hypothèse d'une hiérarchie sous-jacente aux éléments, deux propriétés sont nécessaires sur la similarité pour voir émerger cette hiérarchie selon les auteurs. D'abord il faut que les éléments d'une même branche soient plus proches entre eux selon s par rapport aux éléments d'une autre branche. Ensuite, les hyperonymes doivent être similaires à davantage d'éléments que leur hyponymes. L'ordonnancement par la similarité du voisinage des éléments dans l'espace hyperbolique fera donc émerger cette hiérarchie.

Nous notons $s(e_1, e_2)$ la similarité entre deux éléments e_1 et e_2 d'un vocabulaire V. Chaque élément e est représenté par un vecteur v_e dans l'espace hyperbolique, et on note $\Theta = (v_e)_{e \in V}$ l'ensemble de ces vecteurs. Par la suite, un élément cible e_t est tiré aléatoirement dans le vocabulaire. Avec celui-ci, on tire m autres éléments $(e_i)_{i \in [\![1,m]\!]}$. Le but de la procédure est de faire en sorte que l'élément qui a la plus forte similarité avec e_t, parmi ces m, soit le plus proche de e_t dans la projection en terme de distance hyperbolique d. Notant $i_{\max} = \operatorname{argmax}_{i \in [\![1,m]\!]} s(e_t, e_i)$, l'indice de l'élément maximisant la similarité avec la cible dans le lot, cela revient à vouloir

maximiser, selon NICKEL et KIELA, 2018 :

$$P(\underset{i\in[\![1,m]\!]}{\operatorname{argmin}} d(v_{e_t}, v_{e_i}) = i_{\max}|\Theta) = \frac{e^{-d(v_{e_t}, v_{e_{i_{\max}}})}}{\sum_{i=1}^{m} e^{-d(v_{e_t}, v_{e_i})}}. \tag{C.1}$$

Pour plus de simplicité, nous préférons minimiser le problème équivalent suivant (s'inspirant de la fonction de perte de BOJANOWSKI et al., 2017) :

$$\mathcal{L}(\Theta) = \sum_{e_t\in V} \left(\log(1 + e^{d(v_{e_t}, v_{e_{i_{\max}}})}) + \sum_{i=1, i\neq i_{\max}}^{m} \log(1 + e^{-d(v_{e_t}, v_{e_i})}) \right). \tag{C.2}$$

Quand le vocabulaire est large, nous ajoutons artificiellement des éléments proches de la cible au sens de la similarité, ceci dans le but d'avoir un échantillonnage qui représente mieux le voisinage du mot cible.

Par la suite, la minimisation se fait par la méthode de la descente du gradient stochastique en espace riemannien (voir section 5.2.3.2) implémentée dans le logiciel, en se basant sur l'implémentation de geoopt (KOCHUROV, KARIMOV et KOZLUKOV, 2020) ainsi que plusieurs autres articles (NICKEL et KIELA, 2018 ; TIFREA, BECIGNEUL et GANEA, 2019 ; NICKEL et KIELA, 2017 ; BECIGNEUL et GANEA, 2019). Pour rappel, elle repose sur le calcul des gradients euclidiens, leur projection dans l'espace hyperbolique choisi puis l'application de ces dernières aux paramètres du modèle.

Les trois distances $d_{\mathbb{D}^n}$, $d_{\mathcal{L}}$ et $d_{\mathcal{H}^{n,\beta}}$ sont implémentées, et deux variétés riemanniennes le sont aussi, à savoir $\mathcal{D}^n$ et $\mathcal{L}$. Pour $d_{\mathcal{H}^{n,\beta}}$, on peut soit utiliser $\mathcal{D}^n$ et projeter dans $\mathcal{H}^{n,\beta}$ ou utiliser l'espace de Lorentz $\mathcal{L}$.

C.3 Expériences

Nous proposons de mettre en valeur les deux principaux cadres d'utilisation du logiciel, à savoir, la découverte de hiérarchies continues à partir de similarités à plat et la compression d'une représentation euclidienne dans une représentation hyperbolique.

C.3.1 Découverte de hiérarchies

Nous proposons de faire la découverte de hiérarchies continues à partir de similarités fournies par WordNet. Pour ce faire, une sous-famille de concepts est extraite du graphe et la similarité entre deux concepts se base sur le plus court chemin les reliant dans WordNet. Nous constituons alors une matrice de similarités entre tous les concepts de cette sous-famille. La figure C.2 est une projection obtenue en utilisant HierarX sur la matrice de similarités de la famille des instruments de musique. On peut remarquer que les concepts s'organisent dans l'espace en fonction de leur

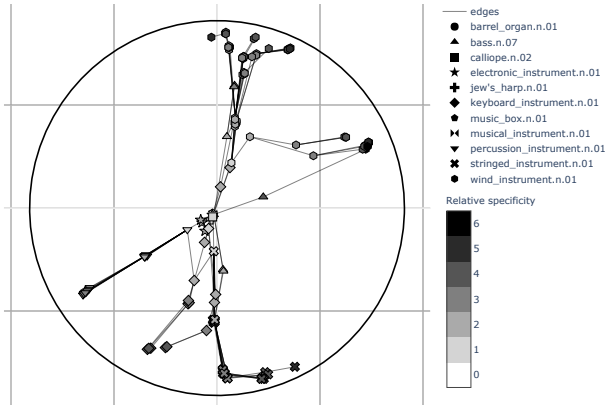


FIGURE C.2 – Projection dans l'espace de Poincaré (dimension 2) des concepts appartenant aux instruments de musique dans la source de données WordNet. Le résultat est obtenu en fournissant à HierarX une matrice de similarités où la similarité entre deux concepts se base sur le plus court chemin les reliant dans WordNet.

catégorie. Ainsi, les instruments à cordes, à vent et à percussions (ou clavier) se répartissent dans trois zones différentes de l'espace. Il est aussi important de noter l'encodage de la spécificité dans la norme des vecteurs. En effet, la corrélation de Spearman entre la norme et la profondeur du concept dans WordNet vaut 0.82. Cela signifie que les concepts les plus spécifiques vont se retrouver aux abords de la boule de Poincaré. À l'inverse, les plus génériques se situent au centre (comme le concept racine, par exemple).

C.3.2 Réduction de dimensions

La seconde utilisation de HierarX fait intervenir les plongements continus de mots comme FastText. Le but est d'obtenir des hiérarchies continues à partir des similarités entre les mots acquises par l'apprentissage de plongements sur du texte brut. Cela permet aussi de réduire la dimension des espaces euclidiens. Le tableau C.2 donne les performances du plongement et de sa compression sur diverses tâches : HyperLex (VULIĆ et al., 2017), et WordSim (FINKELSTEIN et al., 2002). Pour l'évaluation, on calcule le coefficient de Spearman entre la distance de Poincaré (plongement hyperbolique) ou la similarité cosinus (plongement euclidien) avec la similarité fournie par ces corpus. Le premier corpus évalue la capacité à déterminer si un concept est un hyponyme ou hyperonyme d'un autre, alors que le second évalue la similarité sémantique. On constate une perte sur la performance au profit d'une représentation moins lourde pour WordSim. En revanche, le score est légèrement plus haut pour la tâche avec HyperLex. Cela penche en faveur de la capacité des représentations hyperboliques à découvrir des hiérarchies. D'autres hyperparamètres peuvent donner des représentations qui sont meilleures sur WordSim et moins bonnes sur HyperLex. Il faut donc les sélectionner en fonction du besoin. C'est pourquoi ces résultats restent à prendre avec précaution, les plongements issus de HierarX sont souvent moins bons que les FastText originaux, mais sont aussi moins lourds.

TABLEAU C.2 – Résultats avant/après la compression d'un plongement FastText de 300 dimensions dans un espace de Poincaré à 20 dimensions. Le coefficient de Spearman est reporté pour chaque corpus (les pvaleurs sont très faibles). Le vocabulaire est composé des 100000 mots les plus fréquents. Pour HyperLex, le score is_a défini par NICKEL et KIELA, 2017 est utilisé pour l'espace hyperbolique. Pour WordSim, la distance hyperbolique est choisie. Pour FastText, la similarité cosinus sert pour le calcul des similarités dans les deux cas.

Plongement	HyperLex	WordSim
FastText (300 dimensions)	0.20	0.75
Poincaré (20 dimensions)	0.28	0.38

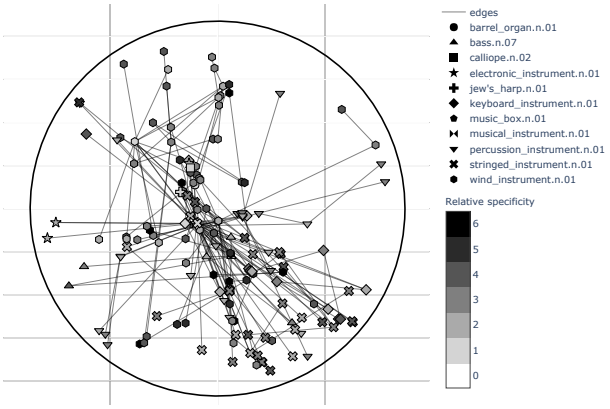


FIGURE C.3 – Projection dans l'espace de Poincaré (dimension 2) des concepts appartenant aux instruments de musique dans la source de données WordNet. Le résultat est obtenu en fournissant à HierarX un plongement FastText de 300 dimensions obtenu en ligne [2].

C.4 Limites

On souhaite pouvoir créer une hiérarchie à partir de n'importe quel type de similarité. Comme évoqué, la similarité est soumise à des contraintes rendant difficile l'apparition d'une hiérarchie sous forme visuelle. Là où la similarité issue des plus courts chemins de WordNet parvient à remplir les deux contraintes de NICKEL et KIELA, 2018, le produit scalaire entre deux vecteurs issus de FastText semble plutôt limité à la compression d'information que de découverte.

On propose de voir sur un cas les limites de l'approche. Prenons les noms des entités d'une hiérarchie de WordNet, par exemple *musical_instrument.n.01*, et calculons la similarité avec les vecteurs de FastText plutôt qu'en se basant sur la hiérarchie. Sur la figure C.3, on voit que les éléments ne s'organisent pas aussi bien que sur la figure C.2. Cela fait de la similarité utilisée par HierarX un élément crucial, qui, s'il ne respecte pas les critères évoqués précédemment, aboutit à un plongement de mauvaise qualité.

C.5 Conclusion

Ce logiciel dispose de propriétés utiles comme la compression ou la découverte de connaissances hiérarchiques lorsque la similarité est bien définie. Pour aller plus loin, il faudrait comprendre précisément et pouvoir évaluer les contraintes faisant de la similarité une bonne représentante de la hiérarchie sous-jacente cachée. Le logiciel a été publié en tant que démonstration à la conférence EGC 2020 (TORRE-GROSSA, GRAVIER et al., 2020).

Bibliographie

Bibliographie

AHMED, Reyan et al. (2020), « Graph spanners : A tutorial review », in : *Computer Science Review* 37.

AKBIK, Alan, Duncan BLYTHE et Roland VOLLGRAF (2018), « Contextual String Embeddings for Sequence Labeling », in : *Proceedings of the 27th International Conference on Computational Linguistics*, ACL, p. 1638-1649.

ALMUHAREB, Abdulrahman (2006), *Attributes in lexical acquisition*.

ALY, Rami et al. (2019), « Every Child Should Have Parents : A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings », in : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL, p. 4811-4817.

AMARI, Shun-ichi (1998), « Natural Gradient Works Efficiently in Learning », in : *Neural Computation* 10.2, p. 251-276.

AMSALEG, Laurent et al. (2015), « Estimating Local Intrinsic Dimensionality », in : *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining*, ACM, p. 29-38.

ARORA, Sanjeev et al. (2019), « Implicit Regularization in Deep Matrix Factorization », in : *Advances in Neural Information Processing Systems*, Curran Associates, Inc.

BAKAROV, Amir (2018), « A Survey of Word Embeddings Evaluation Methods », in : *CoRR* abs/1801.09536.

BANSAL, Nitin, Xiaohan CHEN et Zhangyang WANG (2018), « Can We Gain More from Orthogonality Regularizations in Training Deep CNNs ? », in : *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Curran Associates Inc., p. 4266-4276.

BARONI, Marco et Silvia BERNARDINI (2004), « BootCaT : Bootstrapping Corpora and Terms from the Web », in : *Proceedings of the 4th International Conference on Language Resources and Evaluation*, ELRA.

BARONI, Marco, Georgiana DINU et Germán KRUSZEWSKI (2014), « Don't count, predict ! A systematic comparison of context-counting vs. context-predicting semantic vectors », in : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, p. 238-247.

BARONI, Marco et Alessandro LENCI (2011), « How we BLESSed distributional semantic evaluation », in : *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, ACL, p. 1-10.

BARONI, Marco, Brian MURPHY et al. (2010), « Strudel : A Corpus-Based Semantic Model Based on Properties and Types », in : *Cognitive science* 34, p. 222-54.

BECIGNEUL, Gary et Octavian-Eugen GANEA (2019), « Riemannian Adaptive Optimization Methods », in : *International Conference on Learning Representations*.

BERGSTRA, James et Yoshua BENGIO (2012), « Random Search for Hyper-parameter Optimization », in : *Journal of Machine Learning Research* 13.

BEYER, Lucas et al. (2019), « S4L : Self-Supervised Semi-Supervised Learning », in : *International Conference on Computer Vision*, IEEE, p. 1476-1485.

BLUMENSON, L. E. (1960), « A Derivation of n-Dimensional Spherical Coordinates », in : *The American Mathematical Monthly* 67, p. 63-66.

BOJANOWSKI, Piotr et al. (2017), « Enriching Word Vectors with Subword Information », in : *Transactions of the Association for Computational Linguistics* 5, p. 135-146.

BONNABEL, Silvère (2013), « Stochastic Gradient Descent on Riemannian Manifolds », in : *IEEE Transactions on Automatic Control* 58, p. 2217-2229.

BORDEA, Georgeta, Els LEFEVER et Paul BUITELAAR (2016), « SemEval-2016 Task 13 : Taxonomy Extraction Evaluation (TExEval-2) », in : *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, ACL, p. 1081-1091.

BORDES, Antoine et al. (2013), « Translating embeddings for modeling multi-relational data », in : *Proceedings of the 27th Advances in Neural Information Processing Systems*, p. 2787-2795.

BOUCKAERT, R. et al. (2012), « Mapping the Origins and Expansion of the Indo-European Language Family », in : *Science* 337, p. 957-960.

BRILL, Eric et al. (2001), « Data-Intensive Question Answering », in : *Proceedings of the 10th Text REtrieval Conference*, Gaithersburg, Maryland, USA, p. 393-400.

BROWN, T. et al. (2020), « Language Models are Few-Shot Learners », in : *OpenAI publications*.

BRUNI, Elia, Nam Khanh TRAN et Marco BARONI (2014), « Multimodal Distributional Semantics », in : *Journal Artificial Intelligence Research* 49, p. 1-47.

CAO, Yixin et al. (2019), « Multi-Channel Graph Neural Network for Entity Alignment », in : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 1452-1461.

CHAMI, Ines et al. (2019), « Hyperbolic Graph Convolutional Neural Networks », in : *Advances in Neural Information Processing Systems*, Curran Associates, Inc.

CHEN, Boli, Yao FU et al. (2021), « Probing BERT in Hyperbolic Spaces », in : *International Conference on Learning Representations*.

CHEN, Boli, Xin HUANG et al. (2020), « Hyperbolic Interaction Model for Hierarchical Multi-Label Classification », in : *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 7496-7503.

CHEN, Catherine, Kevin LIN et Dan KLEIN (2021), « Constructing Taxonomies from Pretrained Language Models », in : *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, ACL, p. 4687-4700.

CHEN, Muhao et al. (2017), « Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment », in : *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, AAAI Press, p. 1511-1517.

CLAVEAU, Vincent (2018), « Indiscriminateness in representation spaces of terms and documents », in : *40th European Conference in Information Retrieval*, Springer, p. 251-262.

CLAVEAU, Vincent et Ewa KIJAK (2016), « Direct vs. indirect evaluation of distributional thesauri », in : *International Conference on Computational Linguistics*, COLING, p. 1837-1848.

CLEUZIOU, Guillaume et al. (2015), « QASSIT : A Pretopological Framework for the Automatic Construction of Lexical Taxonomies from Raw Texts », in : *SemEval*, ACL, p. 955-959.

COLLOBERT, Ronan et al. (2011), « Natural Language Processing (Almost) from Scratch », in : *Journal of Machine Learning Research*, p. 2493-2537.

DEVLIN, Jacob et al. (2019), « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding », in : *Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics*, ACL.

DHALL, Ankit et al. (2020), « Hierarchical Image Classification using Entailment Cone Embeddings », in : *CoRR* abs/2004.03459.

DHAMDHERE, Kedar, Anupam GUPTA et Harald RÄCKE (2006), « Improved Embeddings of Graph Metrics into Random Trees », in : *Symposium Of Discrete Algorithms*, SIAM, p. 61-69.

EL BOUKKOURI, Hicham et al. (2020), « CharacterBERT : Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters », in : *Proceedings of the 28th International Conference on Computational Linguistics*, International Committee on Computational Linguistics, p. 6903-6915.

FINKELSTEIN, Lev et al. (2002), « Placing Search in Context : The Concept Revisited », in : *ACM Transactions on Information Systems* 20, p. 116-131.

FISCHER, Andreas et al. (2015), « Approximation of graph edit distance based on Hausdorff matching », in : *Pattern Recognition* 48, p. 331-343.

GANEA, Octavian-Eugen, Gary BECIGNEUL et Thomas HOFMANN (2018a), « Hyperbolic Entailment Cones for Learning Hierarchical Embeddings », in : *Proceedings of the 35th International Conference on Machine Learning*, PMLR, p. 1646-1655.

— (2018b), « Hyperbolic Neural Networks », in : *Advances in Neural Information Processing Systems*, sous la dir. de S. BENGIO et al., t. 31, Curran Associates, Inc.

GERZ, Daniela et al. (2016), « SimVerb-3500 : A Large-Scale Evaluation Set of Verb Similarity », in : *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 2173-2182.

GOYAL, Palash et Emilio FERRARA (2018), « Graph embedding techniques, applications, and performance : A survey », in : *Knowledge-Based Systems* 151, p. 78-94.

GRANDVALET, Yves et Yoshua BENGIO (2005), « Semi-supervised Learning by Entropy Minimization », in : *Advances in Neural Information Processing Systems*, MIT Press.

GROVER, Aditya et Jure LESKOVEC (2016), « Node2vec : Scalable Feature Learning for Networks », in : *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, ACM, p. 855-864.

GUHA, Ramanathan, Rob MCCOOL et Eric MILLER (2003), « Semantic Search », in : *Proceedings of the 12th International World Wide Web Conference*, ACM, p. 700-709.

GUPTA, Amit et al. (2017), « Taxonomy Induction Using Hypernym Subsequences », in : *Proceedings of the 26th Conference on Information and Knowledge Management*, ACM, p. 1329-1338.

HARRIS, Zellig (1954), « Distributional Structure », in : *Word* 10, p. 146-162.

HEARST, Marti A. (1992), « Automatic Acquisition of Hyponyms from Large Text Corpora », in : *COLING*.

HILL, Felix, Roi REICHART et Anna KORHONEN (2015), « SimLex-999 : Evaluating Semantic Models With (Genuine) Similarity Estimation », in : *American Journal of Computational Linguistics* 41, p. 665-695.

HONNIBAL, Matthew et al. (2020), *spaCy : Industrial-strength Natural Language Processing in Python*.

HOULE, Michael E., Hisashi KASHIMA et Michael NETT (2012), « Generalized Expansion Dimension », in : *Proceedings - 12th IEEE International Conference on Data Mining Workshops*, p. 587-594.

IACOBACCI, Ignacio, Mohammad Taher PILEHVAR et Roberto NAVIGLI (2015), « SensEmbed : Learning Sense Embeddings for Word and Relational Similarity », in : *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL, p. 95-105.

JOST, Jürgen (2008), *Riemannian geometry and geometric analysis*, Springer.

JÓZEFOWICZ, Rafal et al. (2016), « Exploring the Limits of Language Modeling », in : *CoRR* abs/1602.02410.

KARYPIS, George (2003), « CLUTO : A clustering toolkit », in : *Technical Report 02-017, University of Minnesota (Department of Computer Science)*.

KIPF, Thomas N. et Max WELLING (2017), « Semi-Supervised Classification with Graph Convolutional Networks », in : *International Conference on Learning Representations*.

KLEINBERG, R. (2007), « Geographic Routing Using Hyperbolic Space », in : *Proceedings of the IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, IEEE, p. 1902-1909.

KOCHUROV, Max, Rasul KARIMOV et Serge KOZLUKOV (2020), *Geoopt : Riemannian Optimization in PyTorch*.

KRIOUKOV, Dmitri et al. (2009), « Greedy Forwarding in Scale-Free Networks Embedded in Hyperbolic Metric Spaces », in : *SIGMETRICS Performance Evaluation Review* 37, p. 15-17.

LAFFERTY, John D., Andrew MCCALLUM et Fernando C. N. PEREIRA (2001), « Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data », in : *International Conference on Machine Learning*.

LAW, Marc et al. (2019), « Lorentzian Distance Learning for Hyperbolic Representations », in : *Proceedings of the 36th International Conference on Machine Learning*, PMLR, p. 3672-3681.

LE, Hang et al. (2019), *FlauBERT : Unsupervised Language Model Pre-training for French*.

LE, Matthew et al. (2019), « Inferring Concept Hierarchies from Text Corpora via Hyperbolic Embeddings », in : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL, p. 3231-3241.

LEE, James R., Assaf NAOR et Yuval PERES (2009), « Trees and Markov Convexity », in : *Geometric and Functional Analysis* 18, p. 1609-1659.

LEIMEISTER, Matthias et Benjamin WILSON (2018), « Skip-gram word embeddings in hyperbolic space », in : *CoRR* abs/1809.01498.

LEVY, Omer et Yoav GOLDBERG (2014), « Neural Word Embedding as Implicit Matrix Factorization », in : *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., p. 2177-2185.

LEVY, Omer, Yoav GOLDBERG et Ido DAGAN (2015), « Improving Distributional Similarity with Lessons Learned from Word Embeddings », in : *Transactions of the Association for Computational Linguistics* 3, p. 211-225.

LEZAMA, José et al. (2018), « OLÉ : Orthogonal Low-Rank Embedding - A Plug and Play Geometric Loss for Deep Learning », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

LIN, Yankai et al. (2015), « Modeling Relation Paths for Representation Learning of Knowledge Bases », in : *Proceedings of the Twentieth Conference on Empirical Methods in Natural Language Processing*, p. 705-714.

LING, Wang et al. (2015), « Not All Contexts Are Created Equal : Better Word Representations with Variable Attention », in : *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 1367-1372.

LIU, Shaoteng et al. (2020), « Hyperbolic Visual Embedding Learning for Zero-Shot Recognition », in : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

LU, W. et al. (2020), « Concept Representation by Learning Explicit and Implicit Concept Couplings », in : *IEEE Intelligent Systems*.

LUKA, Thomas, Laure SOULIER et David PICARD (2021), « Apprentissage non supervisé de représentations de mots à l'aide de réseaux de convolution bilinéaires sur des caractères », in : *CORIA 2021*.

LUONG, Thang, Richard SOCHER et Christopher D. MANNING (2013), « Better Word Representations with Recursive Neural Networks for Morphology », in : *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, ACL, p. 104-113.

MA, X. et E. H. HOVY (2016), « End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. », in : *In proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

MAITRA, Promita et Dipankar DAS (2016), « JUNLP at SemEval-2016 Task 13 : A Language Independent Approach for Hypernym Identification », in : *SemEval*, ACL, p. 1310-1314.

MALLART, Cyrielle et al. (2020), « Relation, es-tu là ? Détection de relations par LSTM pour améliorer l'extraction de relations », in : *Actes de la 6e conférence conjointe Journées d'Études sur la Parole, Traitement Automatique des Langues Naturelles, Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, ATALA, p. 279-287.

MANNING, Christopher D., Prabhakar RAGHAVAN et Hinrich SCHÜTZE (2008), *Introduction to Information Retrieval*, Cambridge University Press.

MAO, Yuning, Xiang REN et al. (2018), « End-to-End Reinforcement Learning for Automatic Taxonomy Induction », in : *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL, p. 2462-2472.

MAO, Yuning, Jingjing TIAN et al. (2019), « Hierarchical Text Classification with Reinforced Label Assignment », in : *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China : ACL, p. 445-455.

McCann, Bryan et al. (2017), « Learned in Translation : Contextualized Word Vectors », in : *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., p. 6297-6308.

Meng, Yu et al. (2019), « Weakly-supervised hierarchical text classification », in : *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 6826-6833.

Mesnil, Grégoire et al. (2013), « Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding », in :

Mikolov, Tomas, Kai Chen et al. (2013), « Efficient Estimation of Word Representations in Vector Space », in : *International Conference on Learning Representations*.

Mikolov, Tomas, Ilya Sutskever et al. (2013), « Distributed Representations of Words and Phrases and their Compositionality », in : *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., p. 3111-3119.

Mikolov, Tomas, Wen-tau Yih et Geoffrey Zweig (2013), « Linguistic Regularities in Continuous Space Word Representations », in : *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, ACL, p. 746-751.

Miller, George A. (1995), « WordNet : A Lexical Database for English », in : *Commun. ACM* 38, p. 39-41.

Monath, Nick et al. (2019), « Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space », in : *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM.

Monma, Clyde et Subhash Suri (1992), « Transitions in geometric minimum spanning trees », in : *Discrete & Computational Geometry* 8, p. 265-293.

Nayyeri, Amir et Benjamin Raichel (2019), « Viewing the Rings of a Tree : Minimum Distortion Embeddings into Trees », in : *Symposium Of Discrete Algorithms*, SIAM, p. 2380-2399.

Nickel, Maximilian et Douwe Kiela (2017), « Poincaré Embeddings for Learning Hierarchical Representations », in : *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., p. 6338-6347.

— (2018), « Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry », in : *Proceedings of the International Conference on Machine Learning*.

Panchenko, Alexander et al. (2016), « TAXI at SemEval-2016 Task 13 : a Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling », in : *SemEval*, ACL, p. 1320-1327.

Paszke, Adam et al. (2019), « PyTorch : An Imperative Style, High-Performance Deep Learning Library », in : *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., p. 8024-8035.

Pennington, Jeffrey, Richard Socher et Christopher D. Manning (2014), « GloVe : Global Vectors for Word Representation », in : *Empirical Methods in Natural Language Processing*, p. 1532-1543.

Peters, Matthew E. et al. (2018), « Deep Contextualized Word Representations », in : *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, ACL, p. 2227-2237.

POLING, Bryan et Gilad LERMAN (2014), « A New approach to two-view motion segmentation using global dimension minimization », in : *International Journal of Computer Vision* 108, p. 165-185.

RADFORD, Alec (2018), « Improving Language Understanding by Generative Pre-Training », in : *OpenAI publications*, URL : `https://openai.com/blog/language-unsupervised/`.

RADFORD, Alec et al. (2019), « Language Models are Unsupervised Multitask Learners », in : *OpenAI publications*.

RIVAS ROJAS, Kervy et al. (2020), « Efficient Strategies for Hierarchical Text Classification : External Knowledge and Auxiliary Tasks », in : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL, p. 2252-2257.

ROLLER, Stephen, Douwe KIELA et Maximilian NICKEL (2018), « Hearst Patterns Revisited : Automatic Hypernym Detection from Large Text Corpora », in : *ACL*, ACL, p. 358-363.

ROY, Olivier et Martin VETTERLI (2007), « The effective rank : A measure of effective dimensionality », in : *Proceedings of the 15th European Signal Processing Conference*, p. 606-610.

RUBENSTEIN, Herbert et John B. GOODENOUGH (1965), « Contextual Correlates of Synonymy », in : *Commun. ACM* 8.10, p. 627-633.

SALA, Frederic et al. (2018), « Representation Tradeoffs for Hyperbolic Embeddings », in : *Proceedings of the 35th International Conference on Machine Learning*, t. 80, PMLR, p. 4460-4469.

SANG, Erik F. Tjong Kim et Fien De MEULDER (2003), « Introduction to the CoNLL-2003 Shared Task : Language-Independent Named Entity Recognition », in : *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

SARKAR, Rik (2011), « Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane », in : *Proceedings of the 19th International Conference on Graph Drawing*, Springer-Verlag, p. 355-366.

SCHNABEL, Tobias et al. (2015), « Evaluation methods for unsupervised word embeddings », in : *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 298-307.

SEDDIQUI, Md. Hanif et Masaki AONO (2009), « An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size », in : *Journal of Web Semantics* 7, Semantic Web challenge 2008, p. 344-356.

SEITNER, Julian et al. (2016), « A Large DataBase of Hypernymy Relations Extracted from the Web. », in : *Proceedings of the 10th International Conference on Language Resources and Evaluation*, ELRA.

SHANG, Chao et al. (2020), « Taxonomy Construction of Unseen Domains via Graph-based Cross-Domain Knowledge Transfer », in : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL, p. 2198-2208.

SHERVASHIDZE, Nino et al. (2011), « Weisfeiler-Lehman Graph Kernels », in : *Journal of Machine Learning Research* 12, p. 2539-2561.

SHVAIKO, Pavel et Jérôme EUZENAT (2013), « Ontology Matching : State of the Art and Future Challenges », in : *IEEE Transactions on Knowledge and Data Engineering* 25, p. 158-176.

SINHA, Koustuv et al. (2018), « A Hierarchical Neural Attention-based Text Classifier », in : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 817-823.

SOCHER, Richard et al. (2013), « Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank », in : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 1631-1642.

SUN, Ang et al. (2011), « New York University 2011 System for KBP Slot Filling. », in : *Proceedings of the 4th Text Analysis Conference*, NIST.

SUN, Chi et al. (2018), « Gaussian Word Embedding with a Wasserstein Distance Loss », in : *CoRR* abs/1808.07016.

SUN, Ke et al. (2015), « Space-Time Local Embeddings », in : *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., p. 100-108.

SUN, Zequn, Wei HU et Chengkai LI (2017), « Cross-lingual entity alignment via joint attribute-preserving embedding », in : *Proceedings of the 16th International Semantic Web Conference*, p. 628-644.

SUN, Zequn, Wei HU, Qingheng ZHANG et al. (2018), « Bootstrapping Entity Alignment with Knowledge Graph Embedding », in : *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, p. 4396-4402.

SUZUKI, Atsushi, Yosuke ENOKIDA et Kenji YAMANISHI (2019), *Riemannian TransE : Multi-relational Graph Embedding in Non-Euclidean Space*, URL : `https://openreview.net/forum?id=r1xRW3A9YX`.

TAN, Liling, Francis BOND et Josef van GENABITH (2016), « USAAR at SemEval-2016 Task 13 : Hyponym Endocentricity », in : *SemEval*, ACL, p. 1303-1309.

TIFREA, Alexandru, Gary BECIGNEUL et Octavian-Eugen GANEA (2019), « Poincaré GloVe : Hyperbolic Word Embeddings », in : *International Conference on Learning Representations*.

TORREGROSSA, François, Robin ALLESIARDO, Vincent CLAVEAU et Guillaume GRAVIER (2021), « Unsupervised Tree Extraction in Embedding Spaces for Taxonomy Induction », in : *International Conference on Web Intelligence and Intelligent Agent Technology*.

TORREGROSSA, François, Robin ALLESIARDO, Vincent CLAVEAU, Nihel KOOLI et al. (2021), « A survey on training and evaluation of word embeddings », in : *International Journal of Data Science Analytics* 2, p. 85-103.

TORREGROSSA, François, Vincent CLAVEAU et al. (2020), « On the Correlation of Word Embedding Evaluation Metrics », in : *Proceedings of The 12th Language Resources and Evaluation Conference*, ELRA, p. 4789-4797.

TORREGROSSA, François, Guillaume GRAVIER et al. (2020), « HierarX : un outil pour la découverte de hiérarchies dans des espaces hyperboliques à partir de similarités », in : *Extraction et Gestion des Connaissances*, É, p. 491-498.

TORREGROSSA, François, Nihel KOOLI et al. (2019), « How We Achieved a Production Ready Slot Filling Deep Neural Network Without Initial Natural Language Data », in : *Neural Information Processing - 26th International Conference*, Springer, p. 247-255.

TSVETKOV, Yulia et al. (2015), « Evaluation of Word Vector Representations by Subspace Alignment », in : *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 2049-2054.

VAIGH, Cheikh Brahim El et al. (2020), « A correlation-based entity embedding approach for robust entity linking », in : *Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence*, IEEE, p. 949-954.

VASWANI, Ashish et al. (2017), « Attention is All you Need », in : *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., p. 5998-6008.

VILNIS, Luke et Andrew MCCALLUM (2015), « Word Representations via Gaussian Embedding », in : *International Conference on Learning Representations*.

VULIĆ, Ivan et al. (2017), « HyperLex : A Large-Scale Evaluation of Graded Lexical Entailment », in : *American Journal of Computational Linguistics* 43, p. 781-835.

WANG, Alex, Yada PRUKSACHATKUN et al. (2019), « SuperGLUE : A Stickier Benchmark for General-Purpose Language Understanding Systems », in : *CoRR* abs/1905.00537.

WANG, Alex, Amanpreet SINGH et al. (2018), « GLUE : A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding », in : *Proceedings of the 2018 Empirical Methods in Natural Language Processing Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, ACL, p. 353-355.

WANG, Bin et al. (2019), « Evaluating word embedding models : methods and experimental results », in : *APSIPA Transactions on Signal and Information Processing*, e19.

WANG, Chengyu, Xiaofeng HE et Aoying ZHOU (2017), « A Short Survey on Taxonomy Learning from Text Corpora : Issues, Resources and Recent Advances », in : *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, ACL, p. 1190-1203.

WANG, Zhichun et al. (2018), « Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks », in : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 349-357.

WEHRMANN, Jonatas, Ricardo CERRI et Rodrigo BARROS (2018), « Hierarchical Multi-Label Classification Networks », in : *Proceedings of the 35th International Conference on Machine Learning*, PMLR, p. 5075-5084.

WILSON, Benjamin et Matthias LEIMEISTER (2018), *Gradient descent in hyperbolic space*.

WOLF, Thomas et al. (2019), « HuggingFace's Transformers : State-of-the-art Natural Language Processing », in : *CoRR* abs/1910.03771.

WU, Yuting et al. (2019), « Relation-Aware Entity Alignment for Heterogeneous Knowledge Graphs », in : *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, p. 5278-5284.

YANG, Zhilin et al. (2019), « XLNet : Generalized Autoregressive Pretraining for Language Understanding », in : *Advances in Neural Information Processing Systems*, sous la dir. de H. WALLACH et al., Curran Associates, Inc.

YE-YI WANG, Li Deng et Alex ACERO (2011), « Semantic Frame Based Spoken Language Understanding », in : *Chapter 3, Tur and De Mori (eds) Spoken Language Understanding : Systems for Extracting Semantic Information from Speech*, p. 35-80.

ZHANG, Jiong, Qi LEI et Inderjit DHILLON (2018), « Stabilizing Gradients for Deep Neural Networks via Efficient SVD Parameterization », in : *Proceedings of the 35th International Conference on Machine Learning*, PMLR, p. 5806-5814.

ZHU, Hao et al. (2017), « Iterative Entity Alignment via Joint Knowledge Embeddings », in : *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, p. 4258-4264.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Représentation des mots et des connaissances: construction, utilisation et évaluation des plongements euclidiens et hyperboliques

Mots clés : Traitement automatique du langage naturel, Représentation des connaissances, Classification, Apprentissage automatique, Ontologies

Résumé : Les données présentent des natures hétérogènes: textes, graphes et hiérarchies. Dans cette thèse, nous traitons l'ensemble des manières de représenter ces trois types en tirant profit d'espaces géométriques adaptés à chacun. Au départ, nous étudions les plongements de mots pour la représentation de textes et tentons d'évaluer leur qualité et de proposer des propriétés déterminant leur efficacité. L'enrichissement de ces plongements par des connaissances expertes nous permet de mieux traiter une tâche industrielle. À partir de ce constat, nous cherchons d'autres moyens pour obtenir des connaissances structurées (hiérarchies) à partir de sources disponibles gratuitement (*Linked Open Data*) ou à partir d'informations à plat contenus dans des textes (induction de taxonomies) ou d'autre types de bases de données. Ces hiérarchies de connaissances sont représentés au mieux par des plongements hyperboliques, et nous proposons une nouvelle méthode de plongement dans ces espaces. Le but est de limiter la perte d'information de la hiérarchie à l'espace continue, puis de déterminer une manière de combiner ces nouvelles connaissances dans les plongements de mots usuels.

Title: Word and knowledge representation: building, using and evaluating euclidean and hyperbolic embeddings

Keywords: Natural language processing, Knowledge representation, Classification, Machine learning, Ontologies

Abstract: Texts, graphs and hierarchies are heterogeneous form of data. In this thesis, we deal with these three categories by leveraging geometric spaces suiting each of them. Starting with the study of word embedding for representing texts, we try to evaluate their quality and provide properties highlighting effectiveness. Enriching those word embeddings with expert knowledge entails a better processing of an industrial task. Building from this fact, we develop new ways to accumulate structured knowledges (e.g. hierarchies) from free online sources (Linked Open Data), or, from information contained in flat contents such as texts (taxonomy induction) or other databases. These hierarchies are then represented using hyperbolic spaces, and we propose a new hyperbolic embedding method. The ultimate goal being to restrain information loss from the structure to the continuous space, and then to combine this new knowledge with traditional word embeddings.