Graphes de connaissances pour la représentation du contexte en traitement automatique du langage naturel

Edward-Kennedy Giamphy

La Rochelle Université

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Edward-Kennedy Giamphy

soutenue le : **4 mars 2024**
pour obtenir le grade de : **Docteur en Informatique**
Discipline : **Informatique et Applications**

Graphes de connaissances pour la représentation du contexte en traitement automatique du langage naturel

---

Rapporteurs	Thomas Bonald	Professeur des universités	Télécom Paris Tech
	Sébastien Fournier	Professeur des universités	Université Aix-Marseille
Examinateurs	Karell Bertet	Professeure des universités	La Rochelle Université
	Christine Largeron	Professeure des universités	Université Jean Monnet
Direction	Gohar Dashyan	Docteur	Preligens
	Antoine Doucet	Professeur des universités	La Rochelle Université
	Jean-Loup Guillaume	Professeur des universités	La Rochelle Université

Thèse réalisée au Laboratoire L3i
La Rochelle Université - Institut LUDI
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel : +33 5 46 45 82 62
Web : `http://l3i.univ-larochelle.fr`

En collaboration avec Preligens
Département AI Research
51 rue de Provence
75009 Paris

Web : `https://www.preligens.com/fr`

Sous la direction de

Gohar Dashyan	Manager de recherche
Antoine Doucet	Professeur des universités
Jean-Loup Guillaume	Professeur des universités
Kevin Sanchis	Ingénieur de recherche

Financement CIFRE Preligens

Résumé

Le langage et sa transcription sous forme écrite sont des piliers fondamentaux de la culture et de la transmission du savoir humain. Le recours de plus en plus commun aux documents numérisés comme moyens de communication en a fait des vecteurs d'information privilégiés. De même, la numérisation grandissante de nos sociétés participe à la prolifération de textes numérisés, et l'accès à l'information, de plus en plus abondante, se complexifie. Le traitement automatique du langage naturel devient alors indispensable pour gérer le nombre croissant de documents numérisés.

En particulier, le domaine de la recherche d'information permet de faciliter l'accessibilité aux contenus textuels recherchés par un utilisateur. Ce dernier usage constitue une application industrielle naturelle, en particulier dans le domaine du renseignement, auquel Preligens, une société spécialisée dans la surveillance de sites stratégiques et de zones de conflit, contribue. L'explosion du nombre de documents numérisés rend manuellement impossible le traitement exhaustif, par un analyste du renseignement, des documents nécessaires à la collecte d'informations d'intérêt pour une mission. Ce travail de thèse s'inscrit dans l'ambition d'appuyer les experts en géopolitique de Preligens, en plus de contribuer de manière générale au domaine de la recherche d'information.

Le langage naturel se caractérise par une combinaison complexe de structures linguistiques, de règles grammaticales et d'expressions culturelles qui est propre à chaque société. Bien qu'une variété de domaines distincts ait été développée pour analyser le langage naturel, les approches traitant les textes écrits reposent sur un découpage au préalable des textes en unités linguistiques, le mot étant l'exemple d'unité le plus évident. Même s'il s'agit d'une approche naturelle et universelle pour représenter un texte, l'utilisation d'unités linguistiques peut être questionnée : le raisonnement consistant à décomposer le texte en unités linguistiques peut-il s'avérer limitant ? Est-ce qu'il existe des méthodes plus aptes à capturer la proximité sémantique et contextuelle des textes en fonction de la tâche linguistique considérée ? Ce travail apporte des premières réponses à ces questions et propose des perspectives pour poursuivre leur exploration.

Des approches récentes prenant en compte le contexte autour des mots des textes se sont démarquées par rapport aux approches traditionnelles. Dans cette optique, les graphes, ou grands réseaux, constituent des objets commodes pour modéliser des phénomènes aussi complexes que variés et sont utilisés dans de nombreuses disciplines allant de l'informatique aux sciences sociales. La diversité des interactions qu'ils peuvent représenter en fait un outil particulièrement adapté à la nature protéiforme du langage naturel. C'est pourquoi cette thèse a pour objectif général d'étudier l'apport des graphes pour le traitement automatique du langage naturel et notamment leur capacité à représenter de manière structurée les relations complexes et variées entre les concepts contextuels des textes.

Étant données les difficultés inhérentes aux graphes et aux grands réseaux, notamment leur taille, nous nous intéressons aux plongements de graphes, une technique permettant de créer une représentation compressée d'un graphe pouvant ensuite être utilisée pour des tâches, normalement inextricables, sur de telles structures. Nous étudions ensuite le potentiel des graphes pour la reconnaissance d'entités nommées, une méthode permettant de caractériser contextuellement un texte, et nous analysons l'impact du bruit sur la recherche d'information. Enfin, nos travaux nous amènent à proposer, en perspective de notre travail, une méthode de recherche d'information reposant sur l'utilisation d'un graphe de connaissances et d'une représentation contextuelle des unités linguistiques.

Mots-clés : Traitement automatique du langage naturel ; recherche d'information ; graphes ; apprentissage de représentation ; apprentissage machine.

Knowledge Graphs for context representation in Natural Language Processing

Abstract

Language and its transcription into written form are fundamental pillars of the transmission of human knowledge and culture. The increasingly common use of digitized documents as a means of communication has made them privileged vectors of information. Similarly, the growing digitization of our societies is contributing to the proliferation of digitized texts, and access to the ever-increasing abundance of information is becoming ever more complex. Natural language processing (NLP) is becoming indispensable for managing the growing number of digitized documents.

In particular, the field of information retrieval makes it easier for users to access the textual content they are looking for. The latter is a natural industrial application, particularly in the field of intelligence, to which Preligens, a company specializing in the surveillance of strategic sites and conflict zones, contributes. The explosion in the number of digitized documents makes it manually impossible for an intelligence analyst to process exhaustively the documents needed to gather information of interest for a mission. The aim of this thesis is to support the geopolitical experts at Preligens, as well as making a general contribution to the field of information retrieval.

Natural language is characterized by a complex combination of linguistic structures, grammatical rules and cultural expressions that is specific to each society. Although a variety of distinct fields have been developed to analyze natural language, approaches to dealing with written texts rely on pre-cutting texts into linguistic units, the word being the most obvious example of an unit. Although this is a natural and universal approach to represent texts, the use of linguistic units can be questioned : can the reasoning behind breaking down texts into linguistic units prove limiting ? Are there more appropriate methods for capturing the semantic and contextual proximity of texts, depending on the linguistic task under consideration ? This work provides some initial answers to these questions, as well as perspectives for further exploration.

Recent approaches that take into account the context surrounding words in texts have set themselves apart from traditional approaches. From this perspective, graphs, or large networks, are convenient objects for modeling phenomena as complex as they are varied. They are used in many disciplines ranging from computer science to social sciences. The diversity of interactions they can represent makes them particularly well suited to the protean nature of natural language. The general aim of this thesis is therefore to study the contribution of graphs to natural language processing, and in particular their ability to represent in a structured way the complex and varied relationships between contextual concepts in texts.

Due to the inherent difficulties related to graphs and large networks, particularly their size, we focus on graph embedding. This technique is used to create a compressed representation of a graph that can be used for tasks, normally intractable, on such structures. We then study the potential of graphs for named entity recognition, a method to contextually characterize text, and analyze the impact of noise on information retrieval. Finally, our work leads us to suggest an information retrieval method based on the use of a knowledge graph and a contextual representation of linguistic units.

Keywords : Natural Language Processing ; Information Retrieval ; Graphs ; Representation Learning ; Machine Learning.

Remerciements

Je voudrais prendre un moment pour exprimer ma gratitude envers toutes les personnes qui ont rendu possible la réalisation de cette thèse CIFRE en informatique. Cette thèse est l'aboutissement d'un projet qui s'est dessiné il y a près de 5 ans à l'issue de mon passage au master MVA de l'ENS Paris Saclay. D'abord réticent à l'idée de faire un doctorat, les enseignements d'apprentissage machine auxquels j'ai assistés m'ont convaincu d'approfondir ce sujet en contexte industriel. Aujourd'hui je suis fier et comblé d'avoir pu mener ce projet qui n'aurait pas pu aboutir sans l'engagement des personnes ayant pris part à cette aventure académique et professionnelle.

Je souhaite d'abord remercier mes directeurs de thèse, Jean-Loup Guillaume et Antoine Doucet, pour leur soutien continu et leur expertise dans leur domaine scientifique. Leur collaboration a été essentielle pour façonner mes idées, élargir mes horizons et assurer la qualité scientifique de cette thèse. Je salue leur adaptabilité dans le déroulement de ma thèse quant aux changements induits par le contexte industriel.

Je tiens également à remercier Preligens, mon partenaire industriel, pour avoir cru en ce projet de recherche. Merci à Renaud Allioux et Arnaud Guérin, fondateurs de Preligens (ex-Earthcube), pour m'avoir offert un environnement stimulant et propice à mon développement. En particulier, je remercie chaleureusement Kevin Sanchis et Gohar Dashyan, mes encadrants chez Preligens avec qui j'ai appris énormément. Merci beaucoup pour votre accompagnement, vos précieux conseils et votre soutien. Bien-sûr, je n'oublie pas Lilian Sanselme qui a suivi ma thèse depuis ses prémices avec toute sa bienveillance et Tugdual Ceillier pour son implication et ses retours sur mes travaux. Je remercie Matthieu Limbert et Marie-Caroline Corbineau qui m'ont accompagné dans la construction et la gestion de cette thèse et sans qui celle-ci ne serait pas ce qu'elle est. Enfin, les figures de mon manuscrit de thèse ne seraient pas ce qu'elles sont sans l'expertise avisée de Vincent Vidal sur LaTeX, merci à toi pour ton soutien et ton recul sur l'aventure du doctorat.

Merci aux membres de mon jury qui ont pris le temps d'évaluer mes travaux de recherche. Merci à Thomas Bonald et Sébastien Fournier pour leurs rapports et à Karell Bertet, ainsi que Christine Largeron pour avoir examiné cette thèse. Je profite des ultimes modifications de ce manuscrit pour les remercier pour leurs questions et les discussions lors de ma soutenance de thèse. Cette journée charnière restera dans ma mémoire.

Je souhaite exprimer ma reconnaissance envers toute l'équipe du laboratoire L3i, ainsi que mes collègues et camarades de thèse, pour leur accueil, leurs échanges enrichissants. Leur collaboration a contribué de manière significative à l'avancement de mes recherches. En particulier Ibrahim Souleiman Mahmoud, Thomas Chambon et Guillaume Bourgeois pour leur soutien mutuel notamment lors de mes expériences de thèse. Merci à Mélanie Malinaud pour son aide dans les démarches administratives. Je remercie spécialement Carlos-Emiliano Gonzàlez-Gallardo pour nos discussions enrichissantes sur le TALN, mais surtout pour nos échanges de bonnes recettes culinaires et la découverte des bars de La Rochelle.

Je souhaite également remercier mes partenaires académiques de l'IIT Bhilai, Soumajit Pramanik et Tarun Singh pour leur implication dans notre projet de recherche commun. J'ai une pensée pour Tarun avec qui j'ai passé tout l'été 2023. Même si nous ne sommes pas allés à la plage, ces mois de travail ont été particulièrement stimulant et challengeant.

Sur un plan plus personnel, je tiens à exprimer ma profonde gratitude envers ma famille et mes amis pour leur soutien dans mon parcours. Plus particulièrement, je tiens à remercier mon frère Bob pour sa gestion maîtrisée du voyage de soutenance de thèse. Un grand merci à mes amis - Kelu, Doks, Jeje, Bybouz, Lim, Totit, Melhack, Jado, Pix, Bzor, Anne-So, Kevin, Pierre et Martin - qui ont constitué une « cellule de décompressement » tout au long de la thèse. Merci également à tous les autres que je ne saurai citer sans me risquer d'en oublier.

Enfin, merci Elodie, pour m'avoir soutenu, compris et encouragé dans cette aventure. Merci d'avoir partagé les hauts et les bas, les périodes de stress et les moments joyeux. J'ai eu la chance de bénéficier de ton regard objectif à chaque fois que l'occasion se présentait et j'en suis fort reconnaissant. Cette thèse est un peu la tienne et j'espère que tu en es fière.

En résumé, je suis reconnaissant envers chacune des personnes et organisations mentionnées pour leur contribution précieuse à cette réussite académique et professionnelle de thèse.

A mes parents, Mathieu et Naïma.

Table des matières

1 Introduction **11**
- 1.1 Le Traitement Automatique du Langage Naturel et les Graphes 11
- 1.2 Contexte 13
  - 1.2.1 Une thèse industrielle dans le renseignement 13
  - 1.2.2 Les cas d'usage du traitement automatique du langage naturel dans le renseignement 14
  - 1.2.3 Contexte académique 15
    - 1.2.3.1 Le classement de texte 15
    - 1.2.3.2 Taxonomie succincte du classement de textes 18
  - 1.2.4 Problématique 19
    - 1.2.4.1 Questions soulevées dans la thèse 19
    - 1.2.4.2 Axes de travaux et lien avec l'application industrielle d'extraction de documents pour Preligens 20
- 1.3 Organisation de la thèse 21

2 État de l'art de la recherche d'information **23**
- 2.1 Concepts primordiaux de la recherche d'information 25
  - 2.1.1 Classement de textes, besoin d'information et pertinence 25
  - 2.1.2 Évaluation du classement de textes 27
  - 2.1.3 Description d'un jeu de données de référence 31
- 2.2 Présentation chronologique des approches de classement de textes 33
  - 2.2.1 Les prémices du classement de textes 33
  - 2.2.2 Les défis de la correspondance exacte 35
  - 2.2.3 L'émergence de l'apprentissage de classements 37
  - 2.2.4 L'apparition de l'apprentissage profond 39
  - 2.2.5 L'irruption du modèle BERT 42

3 Une revue des plongements de graphes bipartis **49**
- 3.1 Contexte et vue d'ensemble 51
- 3.2 Notations et formulation du problème 52
  - 3.2.1 Notations 52
  - 3.2.2 Définitions 52
- 3.3 Taxonomie 55

3.3.1 Approches de préservation de la proximité . . . 56
3.3.2 Approches basées sur le passage de messages . . . 59
3.3.3 Discussion . . . 63
3.4 Outils d'évaluation et d'expérimentation . . . 64
3.4.1 Tâches d'évaluation . . . 64
3.4.1.1 Classification de nœuds . . . 65
3.4.1.2 Classification de graphes . . . 65
3.4.1.3 Prédiction de liens . . . 65
3.4.1.4 Reconstruction de graphes . . . 66
3.4.2 Jeux de données de référence . . . 68
3.4.3 Bibliothèques en source ouverte . . . 68
3.5 Conclusion et Directions Futures . . . 69

4 Injection de connaissances temporelles pour le NER 73
4.1 Introduction . . . 75
4.2 Travaux connexes . . . 76
4.2.1 Reconnaissance d'entités nommées dans des données historiques . . 76
4.2.2 Reconnaissance d'entités nommées avec une base de connaissances 77
4.2.3 Temporalité dans les graphes de connaissances . . . 78
4.3 Jeux de données . . . 79
4.4 Contextes temporels de connaissances pour la reconnaissance d'entités nommées . . . 80
4.4.1 Intégration de l'information temporelle . . . 81
4.4.2 Extraction des contextes . . . 81
4.4.3 Architecture de la reconnaissance d'entités nommées . . . 82
4.5 Configuration expérimentale . . . 84
4.5.1 Résultats . . . 85
4.5.2 Impact des intervalles de temps . . . 86
4.5.3 Impact des erreurs de numérisation . . . 87
4.5.4 Limitations . . . 87
4.6 Conclusions et travaux futurs . . . 87

5 Impact du bruit sur la recherche d'information 89
5.1 Introduction . . . 91
5.2 État de l'art . . . 92
5.3 Méthodologie . . . 93
5.3.1 Jeu de données . . . 94
5.3.2 Injection de bruit . . . 94
5.3.3 Outil d'injection de bruit . . . 97
5.3.4 Modèles de classement . . . 98
5.3.5 Évaluation en environnement bruité . . . 99
5.4 Résultats et discussion . . . 99
5.5 Conclusion . . . 102

6 Représentation spécifique à la requête **103**
6.1 Introduction . . . 105
6.2 Travaux connexes . . . 106
6.3 Méthode de travail . . . 108
6.4 Construction de sous-graphes spécifiques aux requêtes . . . 111
6.5 Apprentissage de plongements spécifiques aux requêtes . . . 113
6.5.1 Conditionnement à la requête . . . 113
6.5.2 Mise à jour hétérogène des entités . . . 114
6.5.3 Mise à jour hétérogène des documents . . . 115
6.5.4 Apprentissage de classements sur les plongements de documents . . . 115
6.5.5 Inférence . . . 117
6.6 Conclusion . . . 117

7 Conclusion **121**
7.1 Problématique traitée dans cette thèse . . . 121
7.2 Contributions . . . 122
7.3 Perspectives . . . 124

Table des figures

3.1 Exemple de graphe biparti avec à droite la matrice d'adjacence des deux domaines U et V. Les positions dans la matrice d'adjacence représentent les poids dans le graphe biparti. Le poids $w_{ij} = 1$ si les nœuds u_i et v_j sont connectés et 0 sinon. Dans certains graphes bipartis, les poids peuvent être un scalaire non négatif, décrivant ainsi la force de la relation entre les nœuds. 53
3.2 Illustration des attributs des nœuds des deux domaines U et V. 54
3.3 Illustration du processus de plongement pour les réseaux bipartis. L'opération de plongement vise à cartographier les nœuds dans un espace de plongement de faible dimension. L'une des difficultés réside dans le fait que les distances dans l'espace de plongement (« Embedding space ») doivent refléter celles entre les nœuds dans le graphe (« Original graph »). Par exemple, les nœuds u_2 et v_1 sont connectés dans le graphe, leurs représentations doivent donc être proches dans l'espace de plongement. Il en va de même pour les nœuds v_3 et u_4. 55
3.4 Illustration d'une relation à plusieurs ordres dans un graphe homogène. L'image de gauche met en évidence le nœud auquel nous nous intéressons (« focus node »). L'image au centre montre ses voisins directs ou voisins de distance 1 (« 1-step neighbors »), tandis que l'image de droite montre ses voisins de distance 2 (« 2-step neighbors »). 56
3.5 Exemple de considération structurelle au sein des réseaux. La figure représente différents motifs de connexion entre les nœuds. Dans la partie gauche de la figure, les nœuds A et B ont trois voisins communs, tandis que dans la partie droite de la figure, les nœuds C et D ont un voisin commun. Il existe différentes méthodes pour évaluer laquelle des paires A-B ou C-D a la relation la plus forte. On pourrait affirmer que les nœuds A et B partagent plus de voisins que les nœuds C et D, et qu'ils devraient donc être considérés comme la connexion la plus forte. Mais d'un autre côté, on pourrait affirmer que les nœuds C et D ont exactement les mêmes voisins, et qu'ils devraient donc être considérés comme la connexion la plus forte. . 57

3.6 La figure de gauche représente un graphe dans lequel le nœud A possède un voisin à distance 1 (« 1-step neighbor ») et quatre voisins à distance 2 (« 2-step neighbor » tandis que la figure de droite représente un graphe où le nœud A possède cinq voisins à distance 1. Si les voisins à distance 1 et 2 sont traités de la même manière, donc que la distance n'est pas prise en compte, alors ces deux graphes structurellement différents sont perçus de la même manière par le nœud A. Ceci est contre-intuitif et montre comment le traitement des voisins à distance k peut avoir un impact sur les considérations structurelles du graphe. 58
3.7 Illustration du processus de passage de messages (« Message-Passing »). L'exemple montre un processus de passage de messages en deux étapes pour le nœud 1. En partant du nœud 1, la première couche (« layer 1 ») contient tous les voisins de distance 1 (« 1-step neighbor's ») du nœud 1, à savoir les nœuds 3, 4 et 2. La même opération est ensuite répétée pour chaque nœud de la première couche. Par exemple, tous les voisins de distance 1 du nœud 3 (dans la première couche) sont placés dans la seconde couche, ce qui permet de voir les nœuds 1 et 4 dans la partie supérieure de la seconde couche (« layer 2 »). Pour obtenir les autres nœuds de la seconde couche, le même principe est répété pour les autres nœuds de la première couche. Les flèches convergeant vers un nœud représentent sa mise à jour et la transmission des informations de ses voisins. Cela se fait en deux opérations distinctes. Tout d'abord, tous les attributs des voisins sont agrégés par une fonction d'agrégation. Ensuite, l'état du nœud est mis à jour par une fonction de mise à jour qui tient compte à la fois de l'agrégation des attributs et de l'état précédent du nœud. 60

4.1 Un exemple tiré du jeu de données `hipe-2020`. 79
4.2 Un exemple tiré du jeu de données `ajmc`. 79
4.3 Architecture du modèle de NER avec les contextes temporels  (*jokers de contexte*). 83

5.1 Distribution du bruit dans les Batch 1 et 2 décrivant la proportion de WER en fonction du CER des jeux de données bruités. Chaque point de la figure correspond à un jeu de données bruitées, à l'exception du point 0 (CER et WER) qui correspond au corpus original. 97
5.2 Schéma montrant le CER moyen obtenu pour le corpus bruité par rapport aux valeurs de CER initialement visées. Les intervalles de tolérance sont représentés par des bandes grises et blanches. 98
5.3 Performance de BM25 et DistilBERT avec une quantité croissante de CER injecté dans le jeu de données de test MS MARCO Passage Ranking. Les différentes méthodes de classement sont représentées par deux types de marqueurs et deux styles de lignes différents, les différentes mesures étant représentées par des couleurs différentes. 100

6.1 Représentation schématique de la limite des plongements statiques d'entités lors de l'utilisation d'informations provenant d'un graphe de connaissances. Ici, les deux requêtes A et B mentionnent l'entité « Espagne » (« Spain ») pour laquelle le graphe de connaissances contient une description unique. La description d'« Espagne » est utile pour répondre à la requête A, car elle donne des informations sur la géographie de l'« Espagne » et donc sur une éventuelle activité agricole dans ce pays. En revanche, la même description d'« Espagne » n'est pas informative pour la requête B, qui concerne les médailles olympiques obtenues par l'Espagne. 106

6.2 Résultat du processus de construction du sous-graphe. Le sous-graphe spécifique à une requête G_q pour une requête q contient 2 types de nœuds : des entités (par exemple des entités du graphe de connaissances global G) et des documents textuels (par exemple des requêtes et des passages du corpus D). Ces deux types de nœuds sont sélectionnés à l'aide d'une méthodologie améliorant la pertinence par rapport à la requête des entités et des documents textuels composant le sous-graphe. On peut constater que les entités du sous-graphe spécifique à la requête sont de trois types : elles peuvent provenir de la requête initiale, des documents jugés pertinents pour la requête (extraits à partir d'un classement initial établi via une méthode parcimonieuse telle que BM25) ou elles peuvent être des entités provenant de chemins du graphe de connaissances G, reliées à des entités dans les documents pertinents pour la requête. 109

6.3 Processus de construction du sous-graphe. (en haut à gauche) le graphe de connaissances initial ne contenant que des entités ; (en haut au milieu) le graphe de connaissances hétérogène obtenu en ajoutant des sommets de type « document » qui peuvent correspondre soit à la requête soit à des documents du corpus, ainsi que les connexions vers les entités qu'ils contiennent ; (en haut à droite) le sous-graphe hétérogène spécifique à la requête dans lequel seule une partie des documents et des entités est conservée. Les documents conservés sont ceux jugés pertinents par un modèle préliminaire de recherche comme BM25. Les entités de ce sous-graphe spécifique sont de trois types : celles provenant de la requête initiale, celles provenant des documents pertinents pour la requête, et celles situées sur des chemins dans le graphe entre les entités de la requête et celles des documents pertinents. La figure est illustrative et le graphe ne correspond pas aux passages et à la requête. 110

6.4 Processus d'entraînement du modèle QuSpec. Le sous-graphe spécifique à la requête construit dans la Figure 6.2 est à la base du processus d'entraînement. À chaque étape de l'entraînement, les entités et les documents du sous-graphe sont mis à jour. Ensuite, les plongements des documents sont utilisés pour prédire leur pertinence vis-à-vis de la requête. Les prédictions ainsi obtenues sont utilisées pour calculer la perte d'apprentissage du classement décrite dans 6.5.4. Ensuite, la fonction de perte est minimisée pour l'étape courante et les paramètres résultants sont rétro-propagés dans le sous-graphe pour l'étape d'entraînement suivante. 111
6.5 Mécanisme de mise à jour des plongements d'entité. L'entité « Apple » est mise à jour avec les quatre composantes impliquées dans l'équation 6.3 qui sont données en entrée d'un réseau neuronal feed-forward (FFN). À côté de l'entité « Apple », on peut observer une boucle prenant en compte le plongement de l'entité « Apple » de l'étape précédente. Sur le côté gauche de la figure, les informations provenant des entités « Pear » et « Nashi », les voisines de l'entité « Apple » dans le graphe de connaissances, sont véhiculées. En haut, le plongement de la requête de l'étape précédente est inclus dans le processus. Enfin, dans la partie droite de la figure, les plongements des documents du sous-graphe spécifique à la requête dans lesquels l'entité « Apple » est mentionnée sont transmis au mécanisme de mise à jour. 116

Liste des tableaux

3.1 Tâches d'évaluation utilisées par les modèles de plongement de graphes bipartis 66
3.2 Informations sur les jeux de données, y compris leur référence, le type de réseau et les tâches d'évaluation. La colonne Tâche d'évaluation suit la légende : A-Classification de nœuds, B-Prédiction de liens. Le tableau indique également quel modèle de plongement de graphes bipartis a utilisé quel jeu de données. Les jeux de données non cochés ont été utilisés dans [CWTY19]. 67
3.3 Code source des modèles de plongement de graphes bipartis. 68
3.4 Ce tableau résume les statistiques ainsi que les éléments descriptifs sur la nature des jeux de données de graphes bipartis. Il fournit des informations utiles sur la taille des graphes, leur densité et leurs applications. 70

4.1 Aperçu des jeux de données `hipe-2020` et `ajmc`. LOC = Localisation, ORG = Organisation, PERS = Personne, PROD = Produit, DATE = Date, METIER = Métier, OBJET = Objet, et MISSION = Mission. Pour chaque langue, le jeu de données est divisé en trois sous ensembles : un d'entraînement (train), un de développement (dev) et un de test (test). 80
4.2 Résultats en français, allemand et anglais pour les jeux de données `hipe-2020` et `ajmc`. 85
4.3 Nombre de contextes introduits selon l'intervalle de temps. 87

5.1 Exemple d'injection de bruit avec un CER ($\simeq 10-15\%$) pour les Batch 1 et 2 96

Chapitre 1

Introduction

Sommaire

1.1 Le Traitement Automatique du Langage Naturel et les Graphes 11
1.2 Contexte 13
1.2.1 Une thèse industrielle dans le renseignement 13
1.2.2 Les cas d'usage du traitement automatique du langage naturel dans le renseignement 14
1.2.3 Contexte académique 15
1.2.4 Problématique 19
1.3 Organisation de la thèse 21

Le langage représente une caractéristique singulière de l'être humain. Il est utilisé depuis des millénaires et sous-tend nos sociétés. De nombreuses disciplines scientifiques ont été développées afin de l'étudier. Ces dernières années ont vu l'émergence de techniques automatisées qui traitent le langage et qui impactent nos sociétés. Cette thèse a pour objectif général d'étudier l'apport des graphes pour le traitement automatique du langage naturel et notamment leur capacité à représenter de manière structurée les relations complexes et variées entre les concepts contextuels des textes.

1.1 Le Traitement Automatique du Langage Naturel et les Graphes

Le langage est un attribut fondamental de l'humanité. Il est défini comme le mode de communication primaire utilisé par les êtres humains pour exprimer des pensées, des émotions, des idées et des informations. Le langage repose sur une combinaison complexe de structures linguistiques, de règles grammaticales et d'expressions culturelles propres à chaque société. La transcription écrite du langage joue un rôle sociétal crucial en tant que vecteur essentiel de la transmission du savoir et de la préservation de la culture. Cette

transcription permet de consigner de manière permanente les connaissances, les histoires et les expériences humaines, permettant ainsi leur partage à travers le temps et l'espace.

D'un point de vue scientifique, le langage se caractérise par sa nature dynamique et sa capacité à refléter l'évolution de la pensée humaine. Il offre une fenêtre unique sur les mécanismes cognitifs et culturels qui sous-tendent les interactions humaines, et sa complexité va au-delà des simples schémas syntaxiques et sémantiques.

Le volume de données textuelles générées chaque jour par ou pour l'Homme est très élevé - il a par exemple plus de 500 millions de publications envoyées chaque jour sur le réseau X (ex-Twitter) - et il est impossible de traiter et d'analyser manuellement ces données de manière efficace et exhaustive. Le traitement automatique du langage naturel devient alors essentiel pour gérer ce nombre croissant de documents écrits. Il permet en effet d'automatiser et d'accélérer le traitement de ces données en utilisant des techniques d'analyse linguistique et sémantique. Le traitement automatique du langage naturel permet également d'extraire des informations pour caractériser rapidement des documents textuels notamment en identifiant automatiquement des personnes/lieux/dates et autres entités dans des textes, mais sert également pour résumer ou traduire des textes, analyser des opinions ou détecter des tendances. Les algorithmes de traitement automatique du langage naturel permettent de mettre en relation des concepts et des contextes au sein des textes, aidant ainsi à découvrir des liens qui pourraient autrement passer inaperçus du fait de volumes trop imposants à traiter. En facilitant la recherche et l'analyse d'informations précises dans d'énormes ensembles de données textuelles, le traitement automatique du langage naturel est essentiel pour la résolution de problèmes complexes et peut être utilisé dans de nombreux domaines tels que la recherche scientifique, l'analyse de marché, la veille stratégique et la prise de décision et dans de nombreuses applications industrielles comme les moteurs de recherche web ou la recommandation de contenus personnalisés.

Le langage est donc un attribut riche et complexe, et l'on comprend pourquoi une variété d'approches bien distinctes ont été développées pour l'analyser. Cette thèse s'applique à étudier une approche qui est particulièrement adaptée à la nature protéiforme du langage. En effet, nos travaux étudient comment une structure bien singulière, celle de graphe, peut améliorer l'analyse du langage. Les graphes, ou grands réseaux, qui constituent une partie importante de cette thèse et que nous aborderons en détails dans la suite, ont fait l'objet d'un engouement récent car ils permettent de modéliser des phénomènes aussi complexes que variés tels que le fonctionnement du web ou les interactions entre molécules pour la création de nouveaux médicaments. Ils sont utilisés dans de nombreuses disciplines allant de l'informatique aux sciences sociales et la diversité des interactions et des noeuds qu'ils peuvent représenter en font un outil adapté pour étudier le langage naturel.

En somme, le langage, sous forme de transcription écrite, est un pilier fondamental de la culture et de la transmission du savoir humain. Son traitement automatique est incontournable pour gérer la quantité croissante de documents écrits et automatiser l'analyse linguistique des contenus textuels, ce qui facilite l'accessibilité et l'extraction d'informations pertinentes. Cette thèse a pour objectif général d'étudier l'apport des graphes pour

le traitement automatique du langage naturel et nous reviendrons plus précisément sur cette problématique dans la suite.

1.2 Contexte

Ce projet de recherche est une convention industrielle de formation par la recherche (thèse CIFRE) menée avec Preligens, une société française qui développe des technologies d'intelligence artificielle venant en appui aux experts de la défense et du renseignement. L'encadrement scientifique a été effectué conjointement avec le Laboratoire Informatique, Image et Interaction (L3i) de La Rochelle Université qui a apporté des compétences sur le traitement automatique du langage naturel et l'analyse de grands réseaux.

1.2.1 Une thèse industrielle dans le renseignement

Le recours de plus en plus commun aux documents numérisés comme moyen de communication en a fait des supports d'information idéaux pour la détection d'événements ou la recherche d'informations spécifiques en temps de crise ou de conflit. François Bernard Huyghe va jusqu'à théoriser l'usage des réseaux d'information comme une nouvelle composante de la Défense d'un État. Selon lui, dans les nouveaux moyens de mener une guerre, un État peut développer « une stratégie d'espace-information où il serait important d'agir aussi vite et précisément que dans l'espace physique » [FB01].

Preligens s'est spécialisée dans la surveillance de sites stratégiques et de zones de conflit par l'analyse automatique de différents flux de données : imagerie satellitaire très haute résolution, infrarouge ou nocturne, données structurées AIS (système d'échanges automatisé entre navires) et ADS-B (suivi de positionnement par satellite pour le contrôle du trafic aérien), bases de documents (traitement automatique de documents textuels numériques), etc. Les méthodes et outils liés à l'analyse automatique de documents textuels numériques étant en cours de développement, l'objectif de la thèse est de contribuer au développement de cet axe en s'intéressant à des problématiques industrielles de long terme ne pouvant être étudiées par les équipes opérationnelles. Cette thèse vise à apporter les fondements méthodologiques à la mise en place de solutions, et les travaux réalisés dans ce cadre sont associés à un cas d'usage métier précis, détaillé dans la suite. Cependant, les méthodes scientifiques développées peuvent être généralisées à d'autres types d'applications (catastrophe naturelle, crise sanitaire, etc.).

Cette thèse a été effectuée dans l'équipe Recherche du département d'Intelligence Artificielle de Preligens dont la mission est notamment d'intégrer les avancées de l'état de l'art qui sont pertinentes pour les produits de surveillance de conflits et de sites stratégiques. Ce travail vise à appuyer les missions d'analyses tactiques et stratégiques de l'équipe d'experts en géopolitique, permettant l'apport d'une approche computationnelle en rupture et en complément des méthodes traditionnellement utilisées.

1.2.2 Les cas d'usage du traitement automatique du langage naturel dans le renseignement

Le traitement automatique du langage naturel peut être utilisé pour des besoins de renseignement afin d'analyser des informations abondantes et fournir un outil pour des cas d'applications d'ordre sécuritaire. En effet, le traitement automatique du langage naturel possède un potentiel considérable pour traiter des cas d'usages qui nécessitent de fournir rapidement des informations précises. En automatisant certaines tâches d'analyse linguistique et en permettant aux analystes du renseignement de se concentrer sur des tâches plus complexes et stratégiques, le développement d'outils dotés d'algorithmes de traitement automatique du langage naturel améliore l'efficacité opérationnelle. Dans le renseignement, le traitement automatique du langage naturel peut être utilisé dans différents contextes :

- **Collecte et tri des données** : dans un cycle conventionnel de renseignement, d'énormes volumes de données sont collectés et peuvent provenir de sources diverses telles que les médias, les communications interceptées, les réseaux sociaux et les rapports de terrain. Le traitement automatique du langage naturel peut être utilisé pour filtrer et trier ces données en identifiant rapidement les informations pertinentes et en éliminant les éléments résiduels considérés comme un bruit informationnel. Cela permet aux analystes de se concentrer sur les données les plus importantes et d'économiser un temps précieux.
- **Extraction d'informations** : les systèmes de traitement automatique du langage naturel peuvent extraire des informations spécifiques à partir de textes, comme des chiffres, des dates, des adresses, etc. Cela permet de regrouper rapidement des données pertinentes pour la prise de décision.
- **Analyse des communications** : les documents textuels de type conversationnel et les communications numériques sont riches en informations potentielles pour les analystes du renseignement. Le traitement automatique du langage naturel peut être utilisé pour effectuer une analyse linguistique des communications, détecter les sujets clés ou extraire des relations et des schémas.
- **Identification d'entités et de relations** : les systèmes de traitement automatique du langage naturel peuvent extraire automatiquement des noms de personnes, d'organisations, de lieux, ainsi que les dates et les événements mentionnés dans les textes. Cela permet de créer des liens entre différentes sources d'informations et entre des acteurs et des événements.
- **Surveillance des médias sociaux** : le traitement automatique du langage naturel peut être utilisé pour surveiller les médias sociaux en temps réel et détecter les tendances, les discussions et les opinions qui émergent sur des sujets spécifiques. Cette surveillance automatisée permet aux analystes de détecter rapidement les changements d'opinion, les mouvements sociaux ou les menaces émergentes.
- **Analyse de sentiment et d'émotion** : le traitement automatique du langage naturel peut déterminer le ton émotionnel d'un texte, qu'il s'agisse d'un rapport, d'une conversation ou d'un message en ligne. Cette analyse peut aider à évaluer les réactions publiques à des événements particuliers et à anticiper les comportements

futurs.

L'utilisation du traitement automatique du langage naturel dans le domaine du renseignement permet ainsi d'accélérer le processus d'analyse, de réduire la charge de travail manuel des analystes et d'améliorer la qualité globale des renseignements fournis.

1.2.3 Contexte académique

Cette thèse s'intéresse au second et au quatrième cas d'usage du renseignement cités ci-dessus. En particulier nous nous intéressons à la tâche de recherche d'information du domaine de la fouille de textes qui étudie la manière de retrouver des informations dans un corpus et s'apparente au cas d'usage d'extraction d'informations, décrit précédemment. Nous étudions également la tâche de reconnaissance d'entités nommées (ou NER pour « named entity recognition ») qui s'apparente au cas d'usage d'identification d'entités et de relations, et consiste à rechercher des objets textuels associés à des catégories prédéfinies telles que des noms de personnes, des dates ou des noms de lieux.

La recherche d'information offre un cadre pour extraire et analyser des documents textuels. Ces dernières années, ce domaine de recherche a reçu beaucoup d'attention, et de nombreux travaux ont été publiés avec l'émergence d'une multitude de catégories de méthodes. L'évolution des moteurs de recherche ces vingts dernières années est un exemple parlant des progrès réalisés dans le domaine.

En outre, une problématique académique et industrielle récurrente est la présence de bruit dans les textes traités. Le bruit peut correspondre à une variété de phénomènes dans le texte des documents ou des requêtes utilisateur, mais nous pouvons retenir pour le moment qu'il se manifeste par des erreurs au niveau des caractères des mots des textes. Cette notion, que nous aborderons en détails dans la suite, a une place importante dans nos travaux car elle est omniprésente dans le traitement automatique du langage naturel.

Le classement de textes est l'un des problèmes fondamentaux de la recherche d'information, la discipline scientifique et technique qui sous-tend les moteurs de recherche.

Dans la suite de cette section, nous définissons le classement de textes, la tâche la plus emblématique de la recherche d'information qui est un terme englobant, et précisons quelques tâches qui lui sont intrinsèquement liées. Cette tâche est omniprésente dans la recherche d'information et constitue un fil rouge de la thèse. Nous profitons de l'opportunité pour également définir la tâche de reconnaissance d'entités nommées. Cette dernière tâche est connexe à la recherche d'information et a une place particulière dans le contexte de cette thèse, qui nous le verrons, donne un rôle central au concept d'entité et à son apport pour les modèles de langage naturel.

1.2.3.1 Le classement de texte

L'objectif du classement de textes est de générer une liste ordonnée de textes extraits d'un corpus en réponse à une requête pour une tâche particulière. La formulation la plus courante du classement de textes est la recherche par requête, où un moteur de recherche (également appelé système d'extraction) produit une liste classée de textes (pages web, articles scientifiques, articles de presse, tweets, etc.) ordonnés selon leur

pertinence à la requête d'un utilisateur. Dans ce contexte, les textes pertinents sont ceux qui « traitent » du sujet de la requête de l'utilisateur et répondent à son besoin d'information. Les chercheurs en recherche d'information appellent cela le problème de la recherche ad hoc (« ad hoc retrieval »). Dans la recherche par mots-clés, également appelée interrogation par mots-clés (par exemple, sur le web), l'utilisateur entre en général quelques termes d'interrogation dans un emplacement dédié de recherche (par exemple, un navigateur) et obtient en retour des résultats contenant des textes classés. Ces résultats sont appelés listes classées, listes de résultats, résultats ou pages de résultats des moteurs de recherche (SERP pour « Search Engine Results Pages »). Les représentations des textes classés comprennent généralement le titre, les métadonnées associées, des extraits des textes retournés (par exemple, un résumé par mot-clé du contexte où les termes de la requête de l'utilisateur sont mis en évidence), ou des liens vers les sources d'origine. Bien qu'il existe de nombreux exemples de problèmes de classement de textes, ce scénario particulier est omniprésent et sans aucun doute familier à tous les lecteurs.

Ce travail de thèse se concentre sur la tâche de recherche ad hoc mais le classement de textes peut exister sous d'autres formes similaires à cette tâche. Nous jugeons utile à la connaissance du lecteur d'aborder quelques unes des autres formes du classement de textes :

— **Réponse aux questions**
  Bien qu'il existe de nombreuses formes de réponses aux questions, la plus connue des utilisateurs aujourd'hui apparaît dans les moteurs de recherche sous la forme de « boite d'information » qui apparaît au-dessus (ou parfois à droite) des principaux résultats d'une recherche dans un navigateur web. Dans le contexte d'un agent intelligent à capacité vocale tel que Siri ou Alexa, les réponses aux questions de l'utilisateur sont directement synthétisées. L'objectif du système est d'identifier (ou d'extraire) une portion de texte qui répond directement à la question de l'utilisateur, au lieu de renvoyer une liste de documents qu'il ou elle doit ensuite consulter manuellement. Pour les réponses aux questions factuelles, les systèmes se concentrent principalement sur les questions auxquelles il est possible de répondre par des phrases courtes ou des entités nommées telles que des dates, des lieux, des organisations, etc.
  Bien que l'histoire des systèmes de réponse aux questions remonte aux années 1960 et aux travaux de [Sim65], les approches extractives modernes (c'est-à-dire les techniques axées sur l'extraction de portions de texte à partir de documents) remontent au début des années 2000 [Voo02]. La plupart des architectures qui adoptent une approche extractive décomposent la tâche de réponse aux questions en deux étapes : dans un premier temps, des passages de texte susceptibles de contenir des réponses dans un corpus potentiellement large sont sélectionnés et, dans un second temps, des techniques d'extraction sont utilisées pour identifier les réponses elles-mêmes.
— **Comparaison de similarités sémantiques**
  La comparaison de similarités sémantiques est une tâche dont l'objectif est de calculer un score de similarité (distances, similarité cosinus, fréquence des termes,

etc.) entre deux documents donnés. Déterminer si deux textes ont la « même signification » est un problème fondamental dans le traitement du langage naturel et est étroitement lié à la question de savoir si un texte est pertinent par rapport à une requête. Dans le cadre du classement de textes, l'évaluation des similarités entre documents peut être utilisée pour identifier d'autres documents pertinents par rapport au besoin d'information exprimé dans la requête. En effet, un texte similaire à un texte pertinent pour la requête est susceptible d'être pertinent. Bien qu'il existe des différences évidentes, les chercheurs ont exploré des approches similaires et ont d'ailleurs souvent adopté les mêmes modèles pour traiter ces deux questions. Dans le contexte des représentations denses apprises pour le classement, décrites plus tard dans la thèse, les liens entre ces deux problèmes sont encore plus étroits et rapprochent les communautés du traitement automatique du langage naturel et de la recherche d'information en effaçant davantage les frontières entre le classement de textes, la réponse aux questions, la détection de paraphrases et de nombreux problèmes connexes.

— **Recommandation de textes**
Lorsqu'un système de recherche affiche un résultat, il peut suggérer d'autres résultats susceptibles d'intéresser l'utilisateur, par exemple pour l'aider dans sa navigation web [SA06]. Cette situation est fréquente sur les sites d'information, où des articles connexes intéressants sont suggérés pour offrir des connaissances additionnelles sur un sujet [SHH18]. Dans la recherche d'information spécifique à la littérature scientifique, le système de recherche peut suggérer des articles dont le contenu est similaire à un article recherché : un exemple de cette fonctionnalité existe dans le moteur de recherche PubMed, qui donne accès à la littérature scientifique dans le domaine des sciences de la vie [LW07]. La recommandation de citations [RLY+14] est un autre exemple de recommandation de texte dans le contexte académique. Tous ces défis impliquent le classement de textes.

Pour conclure, nous définissons la tâche de reconnaissance d'entités nommées qui est fortement liée à la recherche d'information. La reconnaissance d'entités nommées (NER pour « named entity recognition ») est une tâche d'extraction d'information dédiée à l'identification d'entités d'intérêt dans les textes, généralement de type personne, organisation et lieu. Ces entités agissent comme des ancrages référentiels qui sous-tendent la sémantique des textes et guident leur interprétation. De cette manière, le NER caractérise rapidement le contenu d'un texte, apporte des détails contextuels et permet ainsi de faciliter le classement de texte. Cette tâche est étroitement liée aux travaux menés dans la thèse et le Chapitre 4 est notamment dédié à son étude.

Comme nous le voyons, la recherche d'information est sûrement l'exemple le plus connu du problème du classement de textes, mais des manifestations de cette tâche se retrouvent partout, non seulement dans la recherche d'information, mais aussi dans le traitement automatique du langage naturel. Notre énumération de tâches semblables explique également la raison pour laquelle nous utilisons intentionnellement le terme « classement de textes » par opposition au terme plus populaire de « classement de docu-

ments ». Dans de nombreuses applications, l'« unité atomique » de texte à classer n'est pas un document, mais plutôt une phrase, un paragraphe, ou même un tweet (qui peut être considéré comme un document avec une limite en nombre de caractères).

Dans la suite de ce chapitre, nous allons décrire succinctement les différentes catégories de méthodes utilisées pour le classement de textes. Ces catégories seront étudiées plus en détails dans le Chapitre 2 d'état de l'art sur la recherche d'information.

1.2.3.2 Taxonomie succincte du classement de textes

Dans cette section nous donnons une vue d'ensemble des techniques les plus avancées et influentes pour organiser et catégoriser de vastes ensembles de documents textuels. Les principales approches incluent :

— **les approches traditionnelles**
Cette famille d'approches est fondée sur l'idée que des termes extraits automatiquement d'un document puissent servir de descripteurs ou de termes d'index pour décrire le contenu du document. Ce mode d'accès à l'information est basé sur le classement de représentations de documents et de requêtes construites automatiquement. Ces méthodes incluent le modèle de l'espace vectoriel, où les documents et les requêtes sont représentés sous forme de vecteurs dans un espace multidimensionnel, évaluant la similarité entre eux. Parce qu'il a rassemblé toutes les idées majeures de ce pan de méthodes, Salton et al. [SWY75] est fréquemment cité pour la proposition du modèle de l'espace vectoriel et du modèle emblématique BM25 [RJ76]. Dans ce modèle, les documents et les requêtes sont représentés comme des « sacs de mots » (« bag-of-words ») à l'aide de vecteurs parcimonieux en fonction d'un système de pondération des termes, TF-IDF [Rob04] dans ce cas (« Term Frequency-Inverse Document Frequency »), qui pondère les termes en fonction de leur fréquence dans un document et de leur rareté dans l'ensemble des documents. La similarité document-requête est ensuite calculée en termes de similarité cosinus (ou, plus généralement, de produits intérieurs).

— **l'apprentissage de classements**
L'apprentissage de classements (« learning to rank ») se réfère aux approches utilisant des techniques d'apprentissage automatique pour construire le modèle de classement. Il s'agit d'une évolution majeure dans le classement de textes qui se différencie en utilisant des techniques d'apprentissage automatique supervisé pour apprendre des modèles de classement. Les premiers exemples comprennent [Fuh89, WCY93, Gey94a]. Les données d'apprentissage consistent en des listes d'éléments avec un ordre partiel spécifié entre les éléments de chaque liste. Cette approche fait largement appel à des caractéristiques élaborées manuellement, basées principalement sur les propriétés statistiques des termes contenus dans les textes. Les propriétés statistiques des termes comprennent des fonctions des fréquences des termes, des fréquences des documents, des longueurs des documents, etc., les mêmes éléments qui apparaissent dans une fonction de notation telle que BM25. En plus de ces propriétés statistiques, des propriétés intrinsèques

aux textes, discutées plus en détails dans le Chapitre 2 sont prises en compte dans ces approches.

— **l'apprentissage profond**
Dans le contexte du classement des textes, l'apprentissage profond est apparu après l'apprentissage de classements, à la suite de nombreuses avancées des communautés de la vision par ordinateur et du traitement automatique du langage naturel pour cette technologie. Dans le cadre de la recherche d'information, les approches d'apprentissage profond montrent de nombreux atouts : d'abord, les représentations vectorielles continues, abordées dans le Chapitre 2, lèvent les limites de la correspondance exacte des termes. Ensuite, les réseaux neuronaux sont une alternative intéressante pour éliminer le besoin de caractéristiques laborieusement créées à la main et répondent à une difficulté majeure dans la construction de systèmes d'apprentissage de classements.

— **le modèle BERT**
Dans la niche des approches d'apprentissage profond pour le classement de textes, il est important de distinguer les modèles basés sur BERT [DCLT19] (et en général, sur les modèles transformeurs). Le modèle BERT, et plus globalement les modèles de langage neuronaux pré-entraînés, a eu un fort impact sur le domaine. BERT a permis une progression nette et rarement observée des performances de classement de textes, ce qui a provoqué un engouement immédiat de la communauté scientifique. Les modèles pré-entraînés sont à l'origine d'un changement de paradigme dans les domaines du traitement automatique du langage naturel et de la recherche d'information. L'exploration de variantes fondées sur les idées originales du modèle BERT constitue une part importante des publications du domaine aujourd'hui et il s'agit plus largement d'un pan incontournable du traitement automatique du langage naturel moderne. Le modèle est abordé en détail dans le Chapitre 2.

1.2.4 Problématique

Nous développons dans la suite la problématique étudiée dans cette thèse et les directions engagées par nos travaux.

1.2.4.1 Questions soulevées dans la thèse

La recherche d'information a longtemps été dominée par la modélisation de l'espace vectoriel, reposant sur la correspondance exacte des termes des textes considérés. Bien que les méthodes traditionnelles, telles que BM25, ont prouvé leur efficacité, l'avènement des approches d'apprentissage profond, et en particulier celles basées sur BERT, remet en question l'intérêt des méthodes traditionnelles en introduisant des représentations continues des mots et en éliminant la dépendance stricte à la correspondance exacte. Malgré leurs gains de performance indiscutables, ces modèles reposent sur un découpage au préalable des textes en unités linguistiques (les fameux tokens pour BERT) qui rappelle certains aspects de la correspondance de mots. Ce découpage en unités linguistiques

statiques peut être questionné, même s'il est une approche naturelle pour représenter le texte et peut être considéré comme un héritage tant il est universel dans tous les domaines traitant le langage. Par ailleurs, les modèles récents prenant en compte le contexte autour des mots des textes se sont démarqués par rapport aux modèles basés sur l'occurrence de termes, et une clé du succès de BERT et consorts peut justement être l'approche contextuelle qu'ils utilisent. Dans ce cadre, il est intéressant de se questionner : le raisonnement consistant à décomposer le texte en unités linguistiques peut-il s'avérer limitant ? Est-ce qu'il existe des méthodes plus aptes à capturer la proximité sémantique et contextuelle des textes en fonction de la tâche linguistique considérée ?

Dans ce contexte, les graphes, ou grands réseaux, apparaissent comme un moyen pertinent pour traiter le langage en raison de leur capacité à représenter de manière structurée les relations complexes et variées entre les concepts contextuels des textes. En particulier, un graphe peut être utilisé pour représenter un texte de langage naturel comme une séquence d'entités inter-connectées, les entités étant des concepts possédant une signification contextuelle allant au-delà du simple mot. Ces entités correspondent à un ou plusieurs mots et sont reliées par des arêtes qui capturent les relations sémantiques et contextuelles entre elles. Même si représenter un texte comme une séquence d'entités revient à le découper en unités linguistiques (ici les entités), cette représentation donne un découpage mettant plus en avant le contexte du texte. En effet, chaque entité dans le texte peut être enrichie d'informations contextuelles provenant de ses voisins directs et indirects dans le graphe. Cette approche permet d'exploiter efficacement l'information contextuelle et sémantique du texte, offrant ainsi une représentation plus riche et nuancée.

1.2.4.2 Axes de travaux et lien avec l'application industrielle d'extraction de documents pour Preligens

L'extraction de documents dans une base de données est une application industrielle populaire. Elle est particulièrement utile aux experts du renseignement et s'apparente au classement de texte défini précédemment. Cette application vise à extraire des informations d'intérêt à partir de vastes ensembles de documents textuels. Dans le cadre des activités de Preligens, cette tâche implique la recherche, la récupération et la mise en perspective de documents par rapport au besoin informationnel des analystes de renseignement.

Dans ce contexte, il est intéressant d'analyser les liens entre les documents et les entités spécifiques qu'ils mentionnent. En effet, les entités peuvent être des personnes, des lieux, des organisations, des concepts, etc., dont la présence au sein des documents peut fournir des informations importantes sur les thématiques abordées et les relations sémantiques entre les documents. De manière symétrique, on peut se demander si une entité est caractérisée par les documents dans lesquelles elle est mentionnée. Afin d'explorer cet axe de réflexion, les graphes bipartis apparaissent comme un choix naturel car ils possèdent une structure adaptée pour étudier les relations entre documents et entités. Ces graphes ont deux ensembles distincts de nœuds (ici un ensemble représente les documents et l'autre les entités) et leurs arêtes relient exclusivement les nœuds des

deux ensembles, c'est-à-dire qu'une arête relie un document à une entité si cette entité est mentionnée dans le document.

Étant donné les difficultés inhérentes aux graphes et aux grands réseaux, notamment leur taille, nous nous intéressons dans le Chapitre 3 aux plongements de graphes bipartis, qui permettent de créer une représentation compressée d'un graphe biparti pouvant ensuite être utilisée pour des tâches, normalement inextricables, sur les graphes issus de scénarios réels. Les graphes bipartis sont une modélisation pertinente mais moins complète que les graphes de connaissances, qui sont des graphes hétérogènes, c'est-à-dire possédant plusieurs types de noeuds et relations, et dont nous étudions le potentiel dans le Chapitre 4 dédié à la tâche de reconnaissance d'entité nommée (NER). Pour répondre à la problématique omniprésente du bruit, qui peut se manifester de différentes manières dans les textes (erreurs typographiques, fautes d'orthographes, etc.), nous analysons l'impact du bruit sur le classement de textes dans le Chapitre 5. Enfin, les conclusions des Chapitres 4 et 5 nous amènent à proposer, en perspective de notre travail, un modèle de classement de textes utilisant des graphes de connaissances, qui est l'objet du Chapitre 6.

1.3 Organisation de la thèse

Ce manuscrit de thèse est organisé en sept chapitres qui incluent la présente introduction et une conclusion.

Chapitre 2 : Dans ce chapitre nous présentons un état de l'art de la recherche d'information. Puisque ce domaine constitue le fil rouge de cette thèse, nous décrivons les concepts sur lesquels il repose et les outils couramment utilisés pour l'aborder. Ensuite, nous retraçons chronologiquement les méthodes développées pour traiter le classement de textes, et donnons un aperçu des méthodes plus récentes.

Chapitre 3 : Ce chapitre met en avant une revue détaillée des approches de plongement de graphes bipartis. Comme expliqué dans la section précédente, les graphes bipartis offrent une modélisation pertinente pour étudier notre problématique. Après avoir expliqué l'intérêt des plongements pour les graphes en général, nous introduisons des concepts propres au domaine puis proposons une taxonomie des approches de plongement de graphes bipartis. Nous explicitons ensuite les forces et faiblesses des familles d'approches présentées.

Chapitre 4 : Dans ce chapitre, nous explorons l'intérêt des graphes de connaissances pour la tâche de reconnaissance d'entités nommées dans des collections historiques multilingues. En raison de leurs spécificités, les documents historiques peuvent être affectés par les changements et l'évolution du langage naturel. Pour répondre à ce défi, nous explorons l'inclusion de la temporalité dans la tâche de reconnaissance d'entités nommées en exploitant des graphes de connaissances temporelles.

Chapitre 5 : Ce chapitre apporte un éclairage sur l'impact que peut causer la présence de bruit sur la tâche de classement de textes. Après avoir explicité les types de bruit que nous considérons, nous proposons un protocole expérimental exhaustif pour traiter cette problématique qui est peu explorée par la littérature. Cette étude a conduit à la construction et la mise en source ouverte d'une collection de jeux de données bruitées afin que la communauté scientifique puisse explorer plus facilement cette problématique à l'avenir.

Chapitre 6 : Dans ce chapitre d'ouverture, nous proposons une approche afin d'utiliser des graphes de connaissances pour la tâche de classement de textes. Le chapitre 4 ayant confirmé l'intérêt des graphes de connaissances pour une tâche de langage, nous décrivons comment l'information issue de tels graphes peut être utilisée pour le classement de textes.

Chapitre 7 : Enfin, ce chapitre de conclusion revient sur les travaux menés durant la thèse et aborde les réponses que nos différentes contributions apportent à la problématique étudiée. Il suggère également des perspectives et des directions futures afin de donner suite aux travaux initiés par cette thèse.

Chapitre 2

État de l'art de la recherche d'information

Sommaire

2.1 Concepts primordiaux de la recherche d'information 25
2.1.1 Classement de textes, besoin d'information et pertinence 25
2.1.2 Évaluation du classement de textes 27
2.1.3 Description d'un jeu de données de référence 31
2.2 Présentation chronologique des approches de classement de textes 33
2.2.1 Les prémices du classement de textes 33
2.2.2 Les défis de la correspondance exacte 35
2.2.3 L'émergence de l'apprentissage de classements 37
2.2.4 L'apparition de l'apprentissage profond 39
2.2.5 L'irruption du modèle BERT 42

Dans ce chapitre, après avoir défini des concepts communs aux travaux menés dans cette thèse, nous allons décrire les différentes catégories de méthodes utilisées pour le classement de textes, avec une attention particulière pour les méthodes de classement neuronal (neural ranking). Ces dernières méthodes sont fortement influencées par une famille de modèles de réseaux de neurones appelée transformeur (« Transformer ») dont BERT [DCLT19] est l'exemple le plus connu. Ces modèles sont à l'origine d'un changement de paradigme dans les domaines du traitement automatique du langage naturel et de la recherche d'information. Peu d'activités impliquant le traitement automatique du langage naturel sont exemptes des progrès amenés par BERT. Dans le contexte du classement de textes, BERT apporte des résultats dont la qualité est indiscutablement supérieure à celle des précédentes approches explorées. Cette observation a été largement reproduite de manière empirique dans de nombreuses tâches de classement de textes [MRS+21]. Les

éléments de ce chapitre sont inspirés des travaux exhaustifs de [LNY21] qui dressent un tour d'horizon de la recherche d'information.

2.1 Concepts primordiaux de la recherche d'information

Nous abordons dans la suite quelques concepts fondamentaux de la recherche d'information qui seront rencontrés dans les autres chapitres de la thèse.

2.1.1 Classement de textes, besoin d'information et pertinence

La formulation du classement de textes suppose l'existence d'une collection de textes ou d'un corpus $C = \{d_i\}_{i=1}^{l}$ contenant l documents textuels en langage naturel. Notre collection C peut être arbitrairement grande (mais finie) - dans le cas du web, plusieurs milliards de pages. Cela signifie que les questions liées à l'efficacité informatique, par exemple la latence et le débit du classement de textes, sont des considérations importantes, en particulier dans les systèmes de production. La longueur des textes peut varier, allant de quelques phrases à des livres entiers, bien que l'organisation des textes sources, la manière dont ils sont traités et la granularité finale du classement puissent être indépendantes. Une alternative peut être de segmenter le texte des documents en paragraphes et de considérer chaque paragraphe comme l'unité de recherche (le système renvoie alors une liste de paragraphes). Dans cette thèse, nous ne nous intéressons pas à la structure interne des textes, cela étant nous appliquons les traitements les plus génériques aux textes.

Maintenant que nous avons suffisamment caractérisé le corpus, nous nous intéressons aux requêtes. Dans le cadre du web, les courtes requêtes par mots-clés qu'un utilisateur tape dans une barre de recherche ne sont que les manifestations externes d'un besoin d'information. Ce même besoin peut donner lieu à différentes manifestations en fonction du système utilisé : par exemple, dans la barre de recherche d'un moteur de recherche sur le web, quelques mots-clés sont entrés par l'utilisateur, mais pour un assistant vocal, l'utilisateur doit formuler et énoncer une question en langage naturel fluide. Dans cette thèse, nous ne nous intéressons pas aux processus cognitifs qui sous-tendent la recherche d'information, mais nous nous concentrons sur le fonctionnement des modèles de classement de textes après qu'ils aient reçu un signal tangible à traiter. Par conséquent, nous abusons quelque peu de la terminologie et désignons la requête comme « la chose » pour laquelle le classement est demandé (c'est-à-dire l'entrée du modèle de classement), et l'utilisons comme métonymie pour le besoin d'information sous-jacent. En d'autres termes, bien que la requête ne soit pas la même chose que le besoin d'information, nous ne nous intéressons qu'à ce qui alimente le modèle de classement et nous ne prenons en compte que les requêtes exprimées sous forme de texte.

Après avoir décrit plus précisément les entrées des modèles de classement, nous pouvons maintenant définir formellement le problème du classement de textes : étant donné un besoin d'information exprimé sous la forme d'une requête q, la tâche de classement de textes consiste à renvoyer une liste classée de k textes $(d_1, d_2, ..., d_k)$ à partir de $C = \{d_i\}_{i=1}^{l}$ qui maximise une métrique d'intérêt. Les métriques courantes d'évaluation sont présentées dans la suite, mais elles visent toutes à mesurer la « qualité » des résultats par rapport au besoin d'information. En règle générale, les textes classés obtiennent des scores $(s_1, s_2, ..., s_k)$, et les résultats d'un modèle de classement peuvent donc être ca-

ractérisés plus explicitement comme $((d_1, s_1), (d_2, s_2), ..., (d_k, s_k))$ avec la contrainte que $s_1 \geq s_2 \geq ... \geq s_k$. Une distinction mérite d'être introduite ici : le classement se réfère généralement à la tâche de construction d'une liste classée de textes sélectionnés dans le corpus C. Comme nous le verrons dans la suite, il n'est pas computationnellement efficace d'appliquer des modèles basés sur des transformeurs pour classer directement tous les textes d'un corpus (potentiellement grand) et de n'extraire que les k premiers. On fonctionne souvent en deux étapes, une première pour extraire k textes pertinents non classés $R = \{d_1, d_2, ..., d_k\}$ via un modèle de classement computationnellement léger, puis on effectue un reclassement (« reranking ») avec un modèle généralement plus précis que le premier, qui prend R en entrée et ordonne ces k textes en fonction de leur score pour obtenir $R' = (d'_1, d'_2, ..., d'_k)$. Le classement devient conceptuellement équivalent au reclassement si l'on alimente un reclasseur (« reranker ») avec l'ensemble du corpus, mais dans la pratique, les deux impliquent des techniques très différentes.

Un dernier concept est nécessaire pour lier la requête, en tant qu'expression du besoin d'information, à la « qualité » des textes classés. Le fondement de toutes les métriques de classement repose sur la notion de pertinence, qui est une relation entre un texte et un besoin d'information particulier. Un texte est dit pertinent s'il répond au besoin d'information, sinon il n'est pas pertinent. Toutefois, ce traitement binaire de la pertinence est une simplification, car il est plus précis, par exemple, de caractériser la pertinence à l'aide d'échelles ordinales à dimensions multiples [SG01]. Les discussions et les débats sur la nature de la pertinence sont presque aussi anciens que la recherche d'information elle-même. Nous renvoyons le lecteur vers [Sar16] pour une étude approfondie sur le sujet de la pertinence [Sar75].

Formellement, les jugements de pertinence, également appelés *qrels*, comprennent un ensemble de triplets (q, d, r), où le jugement de pertinence r (pour « relevance » en anglais) est une annotation (fournie par un moyen humain) sur les paires requête - document (q, d). Les jugements de pertinence sont également appelés étiquettes de pertinence ou jugements humains. En pratique, ils sont contenus dans des fichiers texte qui peuvent être téléchargés avec une collection de tests (ils sont alors considérés comme une « vérité terrain »). Dans le cas le plus simple, r est une variable binaire : soit le document d est pertinent pour la requête q, soit il ne l'est pas. Une échelle à trois niveaux (non pertinent, pertinent et très pertinent) est une alternative courante et, dans le domaine de la recherche sur le web, une échelle à cinq niveaux est souvent utilisée (parfait, excellent, bon, passable et mauvais). Les jugements de pertinence non binaires sont appelés jugements de pertinence gradués. Les jugements de pertinence ont deux objectifs : ils peuvent être utilisés pour entraîner des modèles de classement dans un cadre supervisé et ils peuvent également être utilisés pour évaluer ces modèles. Pour un chercheur en apprentissage automatique moderne, cette distinction peut sembler étrange, puisqu'il s'agit simplement des rôles des jeux de données d'entraînement, de développement et de test. Mais historiquement, les collections de recherche d'information n'étaient pas suffisamment conséquentes pour évaluer des modèles de classement de manière significative. Cependant, avec l'apparition des jeux de données MS MARCO, que nous présentons dans la suite, la communauté scientifique a eu accès à une collection suffisamment conséquente

de jugements de pertinence pour l'entraînement de modèles dans un cadre supervisé. Nous profitons de cette parenthèse pour souligner l'importance des conférences TRECs (Text Retrieval Conferences) organisées par NIST (« U.S. National Institute for Standards and Technology ») qui rassemblent la communauté scientifique autour d'événements dédiés à l'évaluation des méthodes de Fouille de données. Elles apportent une structure organisationnelle et des ressources incontournables au domaine. Elles sont à l'origine de beaucoup de conventions (comme les *qrels*) actuellement utilisées par les chercheurs en recherche d'information et ont contribué à la création de nombreuses collections réutilisables pour l'évaluation de système du domaine (dont MS MARCO).

Lorsque les jugements de pertinence sont binaires (c'est-à-dire lorsque r est pertinent ou non pertinent), les chercheurs conceptualisent souvent les données d'entraînement comme un ensemble de paires (requête, document pertinent). Dans certains articles, les données d'apprentissage sont décrites comme des triplets (requête, document pertinent, document non pertinent), mais il s'agit simplement d'une organisation différente des triplets (q, d, r) décrits précédemment. Il est important de noter que les documents non pertinents sont souvent qualitativement différents des documents pertinents. Les documents pertinents sont presque toujours jugés comme tels par un évaluateur humain. Les documents non pertinents, en revanche, peuvent soit provenir de jugements humains explicites, soit être construits de manière heuristique. Par exemple, dans la collection de tests de classement de passages MS MARCO, les documents non pertinents sont échantillonnés à partir des résultats de BM25 [RJ76]. Par conséquent, il se peut que les documents non pertinents contiennent des documents qui sont en fait pertinents.

2.1.2 Évaluation du classement de textes

Les métriques de classement quantifient la qualité d'un classement de textes et sont calculées à partir des jugements de pertinence (qrels), décrits dans la section précédente. Les listes de classement (ou liste classées) produites par un système pour un ensemble de requêtes sont appelées « listes de classement ». Les listes de classement produites par un système pour un ensemble de requêtes sont des « soumissions » (appelées « run », ou parfois « submission »).

Nous décrivons les métriques communes à la recherche d'information. Nos notations et conventions sont largement inspirées de [MC18]. Soit une liste classée $R = ((d_i, s_i))_{i=1}^{l}$ de longueur l et la même liste $((i, d_i))_{i=1}^{l}$, en ne conservant que le rang i induit par le score s_i. En effet, la majorité (sinon la totalité) des métriques utilisent le rang plutôt que le score lui-même pour évaluer la qualité d'un résultat, la question est en effet de savoir si les résultats sont présentés dans le bon ordre. Plusieurs métriques sont calculées pour un rang particulier de textes pertinents, ce qui signifie que la liste de classement R est tronquée à une certaine longueur k, $((d_i, s_i))_{i=1}^{k}$, où $k \leq l$: c'est ce qu'on désigne par « Metric@k », où *Metric* désigne la métrique d'intérêt. Cette troncature correspond au comportement des utilisateurs qui s'arrêtent souvent de regarder les documents après les k premiers résultats. La principale différence entre l et k est que le système décide de l (c'est-à-dire du nombre de résultats à renvoyer), alors que k est une propriété de la métrique d'évaluation, généralement fixée par les organisateurs d'une évaluation ou les

auteurs d'un article. Parfois, l et k ne sont pas spécifiés, auquel cas $l = k = 1000$. Dans la plupart des configurations d'évaluation, les modèles renvoient jusqu'à 1000 résultats par thème, et les mesures évaluent la totalité des listes classées (à moins que ce ne soit spécifié autrement).

À partir d'une liste classée R, nous pouvons calculer les mesures suivantes :
Précision. La précision est définie comme la fraction des documents de la liste classée R qui sont pertinents, ou :

$$Precision(R, q) = \frac{\sum_{(i,d_i)\in R} rel(q, d_i)}{|R|}$$

où $rel(q, d)$ indique si le document d est pertinent pour la requête q (en supposant une pertinence binaire). Souvent, la précision est évaluée pour un rang k, noté « Precision@k » ou abrégé « P@k ». Les jugements de pertinence gradués sont binarisés avec un certain seuil de pertinence, par exemple, dans une échelle à trois niveaux, nous pourrions fixer $rel(q, d) = 1$ pour les jugements « pertinents » et « très pertinents ».

La précision a l'avantage d'être facile à interpréter : parmi les k premiers résultats, quelle en est la fraction pertinente ? Il y a deux inconvénients principaux : premièrement, la précision ne prend pas en compte les jugements de pertinence gradués et, par exemple, ne peut pas distinguer les résultats « pertinents » des résultats « très pertinents » puisque la distinction est effacée dans $rel(q, d)$. Deuxièmement, la précision ne mesure pas la qualité de l'ordonnancement des k premiers résultats puisqu'elle en mesure simplement la fraction pertinente.

Rappel. Le rappel est défini comme la fraction des documents pertinents pour q qui sont retrouvés dans la liste classée R, ou :

$$Recall(R, q) = \frac{\sum_{(i,d_i)\in R} rel(q, d_i)}{\sum_{d\in C} rel(q, d)}$$

où $rel(q, d)$ est défini comme précédemment et donc $\sum_{d\in C} rel(q, d)$ est le nombre de documents de C qui sont pertinents. À l'instar de la précision, le rappel est souvent évalué pour un rang k, noté « Rappel@k » (« Recall@k ») ou abrégé « R@k ». Cette mesure présente les mêmes avantages et inconvénients que la précision : elle est facile à interpréter, mais ne prend pas en compte les degrés de pertinence ou les positions de classement dans lesquelles les documents pertinents apparaissent.

Rang Réciproque. Le rang réciproque (RR) est défini comme suit :

$$RR(R, q) = \frac{1}{rang_i}$$

où $rang_i$ est le plus petit numéro de rang d'un document pertinent. En d'autres termes, si un document pertinent apparaît en première position, le rang réciproque vaudra 1, 1/2 s'il apparaît en deuxième position, 1/3 s'il apparaît en troisième position, etc. Si

un document pertinent n'apparaît pas dans les k premiers, la requête reçoit un score de zéro. Comme la précision et le rappel, le RR est calculé pour des jugements binaires. Bien que le RR ait une interprétation intuitive, il ne tient compte que de l'apparition du premier résultat pertinent. Pour les réponses aux questions ou les tâches dans lesquelles l'utilisateur peut se contenter d'une seule réponse, il peut s'agir d'une mesure appropriée, mais le rang réciproque est généralement un mauvais choix pour la recherche ad hoc, car les utilisateurs souhaitent généralement plus d'un document pertinent.

Précision Moyenne. La précision moyenne (ou AP pour « Average Precision ») est définie comme suit :

$$AP(R,q) = \frac{\sum_{(i,d_i)\in R} Precision@i(R,q) \cdot rel(q,d_i)}{\sum_{d\in C} rel(q,d)}$$

où toutes les notations utilisées ont déjà été définies. Intuitivement, la précision moyenne est la moyenne des scores de précision aux rangs correspondant à l'apparition de chaque document pertinent ; le numérateur ne comprend que les contributions des documents pertinents du fait du produit avec $rel(q,d_i)$ qui est binaire. Le dénominateur étant le nombre total de documents pertinents, les documents pertinents qui n'apparaissent pas du tout dans la liste classée ne contribuent pas à la moyenne. Une fois de plus, on suppose que la pertinence est binaire.

Généralement, la précision moyenne est mesurée sans rang explicite, sur l'ensemble de la liste classée ; étant donné que la longueur par défaut de l utilisée dans la plupart des évaluations est de 1000, l'effet pratique est que l'AP est calculée au rang 1000, bien qu'il ne soit presque jamais écrit AP@1000. Étant donné que la mesure tient compte de la recherche de tous les documents pertinents, un rang réduirait artificiellement le score (c'est-à-dire qu'il aurait pour effet d'inclure un grand nombre de zéros dans la moyenne pour les documents pertinents qui n'apparaissent pas dans la liste classée). Les évaluations utilisent la précision moyenne lorsque la tâche exige de prendre en compte le rappel, de sorte que l'imposition d'un rang n'a généralement pas de sens. Le rang implicite de 1000 est un compromis entre la précision de la mesure et l'aspect pratique : dans la pratique, la contribution des documents pertinents apparaissant au-delà du rang 1000 est négligeable dans le score final (qui est généralement rapporté avec quatre chiffres après la virgule), et le calcul du score pour 1000 documents reste computationnellement raisonnable.

La précision moyenne est plus difficile à interpréter, mais il s'agit d'une statistique récapitulative qui combine les informations de la précision et du rappel, tout en favorisant l'apparition des documents pertinents vers le haut de la liste classée. L'inconvénient de la précision moyenne est qu'elle ne fait pas de distinction entre les degrés de pertinence, c'est-à-dire que les documents « marginalement » pertinents et « hautement » pertinents contribuent de la même manière au score.

Gain Cumulatif Actualisé Normalisé. Le gain cumulatif actualisé normalisé (ou nDCG pour « Normalized Discounted Cumulative Gain ») est une mesure fréquemment utilisée pour évaluer la qualité des résultats de recherche sur le web. Contrairement aux

autres mesures ci-dessus, le nDCG a été spécifiquement conçu pour les jugements de pertinence gradués. Par exemple, si la pertinence est mesurée sur une échelle de cinq points, $rel(q, d)$ renvoie $r \in 0, 1, 2, 3, 4$. Nous définissons tout d'abord le gain cumulatif actualisé (DCG) :

$$DCG(R, q) = \sum_{(i,d_i)\in R} \frac{2^{rel(q,d_i)} - 1}{log_2(i + 1)}$$

Le terme « gain » est utilisé ici dans le sens de l'utilité, c'est-à-dire de la valeur qu'un utilisateur tire d'un résultat particulier. Deux facteurs entrent dans ce calcul : le degré de pertinence (les résultats très pertinents « valent » plus que les résultats pertinents) et le rang auquel le résultat apparaît (les résultats pertinents situés en haut de la liste de classement « valent » plus). L'actualisation fait référence à la diminution du gain (utilité) au fur et à mesure que l'utilisateur consomme des résultats de plus en plus bas dans la liste de classement. Enfin, nous introduisons la normalisation :

$$nDCG(R, q) = \frac{DCG(R, q)}{IDCG(R, q)}$$

où IDCG représente le DCG d'une liste classée « idéale » : il s'agit d'une liste classée qui commence par tous les documents ayant le degré de pertinence le plus élevé, puis les documents ayant le degré de pertinence suivant, etc. Ainsi, nDCG représente le DCG normalisé dans une plage de [0, 1] par rapport au meilleur classement possible. Généralement, nDCG est associé à un rang de classement (par exemple 10 ou 20). Étant donné que la plupart des moteurs de recherche commerciaux présentent souvent dix résultats par page, ces deux paramètres représentent le nDCG par rapport à la première ou aux deux premières pages de résultats. Pour des raisons similaires, les paramètres « nDCG@3 » ou « nDCG@5 » sont souvent utilisés dans le contexte de la recherche mobile, étant donné que les écrans des téléphones sont beaucoup plus petits. Cette mesure est populaire pour évaluer les résultats de la recherche sur le web pour un certain nombre de raisons : tout d'abord, le nDCG peut tirer parti des jugements de pertinence gradués, qui fournissent des distinctions plus fines sur la qualité des résultats. Deuxièmement, l'actualisation et la troncation représentent un modèle raisonnablement précis (bien que simplifié) du comportement réel des utilisateurs, comme le révèlent les études d'oculométrie [JGP+07]. Les utilisateurs ont tendance à parcourir les résultats de manière linéaire, avec une probabilité croissante d'« abandon » et de « perte d'intérêt » au fur et à mesure qu'ils avancent dans la liste de classement renvoyée. Ce phénomène est modélisé par l'actualisation, et il existe des variantes de nDCG qui appliquent différents schémas d'actualisation pour modéliser cet aspect du comportement de l'utilisateur. La troncation modélise un arrêt brutal lorsque les utilisateurs cessent de lire (c'est-à-dire qu'ils abandonnent). Par exemple, « nDCG@10 » quantifie la qualité des résultats de la première page de résultats de recherche dans un navigateur, en supposant que l'utilisateur ne clique jamais sur « page suivante » (ce qui est souvent le cas).

Toutes les mesures examinées ci-dessus quantifient la qualité d'une liste unique classée en fonction d'un thème spécifique (la requête). En règle générale, la moyenne arithmétique

de tous les sujets d'une collection de tests est utilisée comme statistique récapitulative pour indiquer la qualité d'une soumission. Nous insistons sur le fait qu'il est totalement dénué de sens de comparer les scores de pertinence de différentes collections de tests (puisque les scores n'incluent pas les différences inhérentes aux corpus, la difficulté des requêtes et de nombreux autres aspects).

Un détail souvent omis mais qui est essentiel à l'interprétation des métriques est le cas où la liste classée R contient un document pour lequel il n'existe pas de jugement de pertinence. C'est ce qu'on appelle un « document non jugé », et le traitement standard de ce cas particulier consiste à considérer que les documents non jugés ne sont pas pertinents. Les documents non jugés sont assez courants parce qu'il est souvent trop coûteux d'évaluer de manière exhaustive la pertinence de chaque document d'une collection par rapport à chaque besoin d'information.

2.1.3 Description d'un jeu de données de référence

Sur la base des discussions ci-dessus, nous pouvons énumérer les composants nécessaires à l'évaluation d'un modèle de classement de textes : un corpus ou une collection de textes à classer, un ensemble de besoins d'information (c'est-à-dire des sujets, ou requêtes) et des jugements de pertinence (qrels) pour ces besoins. Ensemble, ces éléments constituent ce que l'on appelle une collection de tests, que nous simplifierons par collection, pour la recherche d'information. Avec une collection, il devient facile de générer des classements avec un modèle, puis de calculer des mesures pour quantifier la qualité de ces classements, par exemple, en utilisant l'un des outils décrits dans la section précédente. Après avoir quantifié l'efficacité des résultats, il devient possible de mesurer les progrès dans l'amélioration des modèles de classement. Les techniques d'apprentissage automatique supervisé nécessitent des données annotées, et la communauté a accès à de nombreuses collections, constituées au fil des décennies, pour l'entraînement et l'évaluation des modèles de classement de textes. Dans cette section, nous décrivons « MS MARCO Passage Ranking » [BCC+18a], une collection couramment utilisée par les chercheurs aujourd'hui. Notre intention est de nous concentrer sur une ressource représentative qui a joué un rôle important dans le développement de modèles de classement basés sur des transformeurs.

Cet ensemble de données, publié à l'origine en 2016, mérite d'être salué pour avoir lancé la révolution de BERT, discuté plus tard, pour le classement de textes. L'ensemble de données MS MARCO a été publié à l'origine en 2016 pour permettre à la communauté universitaire d'explorer la recherche d'information pour des données volumineuses - en particulier, pour entraîner des modèles de réseaux neuronaux [CMY+21]. Initialement, le jeu de données a été conçu pour étudier la réponse aux questions sur des passages issus du web, mais il a ensuite été adapté à des tâches de classement ad hoc traditionnelles. Ici, nous nous concentrons uniquement sur la tâche de classement de passages [BCC+18a]. Le corpus comprend 8,8 millions d'extraits de différente longueur de pages web ; ces passages sont typiques des « réponses » que de nombreux moteurs de recherche affichent aujourd'hui en haut de leurs pages de résultats. Les besoins d'information sont des questions

anonymisées en langage naturel tirées des journaux de requêtes du moteur de recherche Bing, où les utilisateurs cherchaient spécifiquement une réponse ; les requêtes avec des intentions de navigation et autres ont été écartées. Comme ces questions ont été tirées de requêtes d'utilisateurs issues de scénarios réels, elles sont souvent ambiguës, mal formulées et peuvent même contenir des erreurs typographiques. Néanmoins, ces requêtes reflètent une distribution plus « naturelle » des besoins d'information que, par exemple, d'autres jeux de données de réponse aux questions tels que SQuAD [RZLL16].

Pour chaque requête, la collection contient en moyenne un passage pertinent (évalué par des annotateurs humains). Dans le jeu d'entraînement, il y a un total de 532,8K paires (requête, passage pertinent) sur 502,9K requêtes uniques. Le jeu de développement (validation) contient 7437 paires sur 6980 requêtes uniques. Le jeu de test (évaluation) contient 6837 requêtes, mais les jugements de pertinence ne sont pas accessibles au public ; les scores sur les requêtes de test ne peuvent être obtenus que par le biais d'une soumission au tableau de classement officiel de MS MARCO. La mesure d'évaluation officielle est MRR@10 (rang réciproque moyen @10).

Une caractéristique notable de cette ressource qui mérite d'être soulignée est la rareté des jugements - il y a beaucoup de requêtes, mais en moyenne, seulement un jugement pertinent par requête. Comme nous l'avons vu plus haut, ces jugements sont souvent qualifiés de « superficiels » ou « épars », et cette conception a une conséquence importante. En effet, l'apprentissage d'un modèle nécessite des exemples positifs et négatifs. Pour ce faire, les organisateurs de la tâche ont préparé des fichiers de triplets comprenant des triplets (requête, passage pertinent, passage non pertinent). Cependant, ces exemples négatifs sont des pseudo-étiquettes induites de manière heuristique : ils sont tirés des résultats de BM25 et n'ont pas été marqués comme non pertinents par des annotateurs humains. En d'autres termes, les exemples négatifs n'ont pas été explicitement vérifiés par des annotateurs humains comme étant définitivement non pertinents. L'absence d'étiquette positive ne signifie pas nécessairement que le passage n'est pas pertinent.

Malgré ces défauts, on ne peut négliger le rôle important que le jeu de données MS MARCO a joué dans l'avancement de la recherche d'information. Jamais auparavant un ensemble de données aussi vaste et réaliste n'avait été mis à la disposition de la communauté scientifique. Auparavant, des bases de données de cette ampleur n'étaient accessibles qu'à une niche privilégiée de chercheurs (au sein de sociétés privées, sociétés de moteurs de recherche ayant un nombre important d'utilisateurs etc.). Aujourd'hui, cet ensemble de données est utilisé par de nombreux chercheurs pour diverses tâches de recherche d'information et il est devenu un point de départ courant pour la construction de modèles de classement basés sur des transformeurs. Même pour le classement dans des domaines plus éloignés, par exemple la biomédecine, de nombreux modèles basés sur des transformeurs sont d'abord entraînés avec les données de MS MARCO avant d'être affinés sur des données spécifiques au domaine et à la tâche.

En résumé, la collection de classement de passages MS MARCO a eu et a toujours un fort impact et nous tenons à saluer le travail des créateurs du jeu de données (et Microsoft) pour leur contribution à l'élargissement du domaine.

2.2 Présentation chronologique des approches de classement de textes

L'idée d'exploiter les machines informatiques pour accéder à l'information remonte à l'invention des machines informatiques elles-mêmes, et date de bien avant que l'informatique n'émerge en tant que discipline cohérente telle qu'on la connaît aujourd'hui. L'essai souvent cité de Vannevar Bush, « As We May Think » [Bus45] en 1945, décrit une machine hypothétique appelée « memex » qui effectue une indexation associative pour relier des éléments arbitraires de contenu stockés sur microfilm, comme moyen de capturer des idées et d'améliorer la mémoire des scientifiques. L'article décrit des technologies que nous pourrions reconnaître aujourd'hui comme reprenant certains aspects des ordinateurs personnels, des liens hypertextes, du web sémantique et des encyclopédies en ligne.

2.2.1 Les prémices du classement de textes

Bien que le besoin de machines pour améliorer l'accès à l'information ait été identifié dès le milieu des années 40, il est intéressant de noter que la conception du classement de textes n'est apparue qu'une décennie plus tard. Les bibliothèques existant depuis des millénaires, les premières formulations de la recherche ont été décrites comme l'automatisation de ce que les bibliothécaires faisaient depuis des siècles : l'appariement basé sur des descripteurs de contenu extraits par l'Homme et stockés sur des cartes perforées représentant physiquement les textes à rechercher (livres, articles scientifiques, etc.). Ces descripteurs (également appelés « termes d'indexation ») étaient généralement attribués par des experts et provenaient de répertoires regroupant un vocabulaire prédéfini. Ce processus était connu sous le nom d'« indexation »- le sens originel de cette activité impliquait des êtres humains et est assez étranger aux notions modernes qui impliquent un traitement automatisé. L'émergence de requêtes pour rechercher du contenu exigeait que les bibliothécaires traduisent le besoin d'information de l'usager cherchant une information dans ces mêmes descripteurs ; la recherche s'effectue en faisant correspondre ces descripteurs de manière booléenne (d'où l'absence de classement).

En proposant de tenir compte « des informations statistiques dérivées de la fréquence et de la distribution des mots [...] pour calculer une mesure relative de l'importance », Luhn [Luh58] s'est écarté de l'approche d'indexation humaine dominante de l'époque et a permis de créer des « auto-résumés ». Il a décrit un précurseur de ce que nous appellerions aujourd'hui la pondération TF-IDF (c'est-à-dire la pondération des termes basée sur la fréquence des termes et la fréquence inverse des documents). Toutefois, Luhn n'a ni mis en œuvre ni évalué les techniques qu'il proposait. Une articulation plus claire du classement de textes a été présentée par Maron et Kuhns [MK60], qui ont caractérisé le problème de la recherche d'information (bien qu'ils n'aient pas utilisé ces mots) comme la réception de demandes de l'utilisateur et « la fourniture en sortie d'une liste ordonnée des documents qui satisfont le plus probablement les besoins d'information de l'utilisateur ». Ils ont proposé que les termes de l'index (ou « tags ») soient pondérés en fonction

de la probabilité qu'un utilisateur désireux d'obtenir des informations contenues dans un document particulier utilise ce terme dans une requête. Aujourd'hui, nous pourrions appeler cela la probabilité d'une requête [PC98]. Le document décrit également l'idée d'un « numéro de pertinence » pour chaque document, « qui est une mesure de la probabilité que le document réponde à la requête donnée ». Aujourd'hui, nous appellerions cela des scores de recherche. Au-delà de la présentation de ces concepts fondamentaux, Maron et Kuhns ont décrit des expériences visant à tester leurs idées. L'idée que des termes extraits automatiquement d'un document puissent servir de descripteurs ou de termes d'index pour décrire le contenu de ces documents nous paraît aujourd'hui évidente, mais il s'agissait à l'époque d'un saut conceptuel important dans le développement de la recherche d'information.

Tout au long des années 1960 et 1970, les chercheurs et les praticiens ont débattu des mérites de l'« analyse automatique du contenu » (voir, par exemple, Salton [Sal68]) par rapport à l'indexation « traditionnelle » basée sur l'Homme. Salton [Sal72] a décrit une évaluation notable comparant le système de recherche SMART basé sur le modèle de l'espace vectoriel à l'indexation humaine dans le contexte de MEDLARS (« Medical Literature Analysis and Retrieval System »), qui était une version informatisée de l'Index Medicus, un index bibliographique imprimé complet d'articles médicaux que la National Library of Medicine (NLM) des États-Unis publiait depuis 1879. Il a été démontré que SMART produisait des résultats de meilleure qualité, et Salton a conclu « qu'il n'existe aucune justification technique au maintien d'une indexation manuelle contrôlée dans les environnements de recherche opérationnelle ». Cet axe de recherche a eu un impact significatif, puisque MEDLARS a évolué pour devenir MEDLINE (abréviation de MEDLARS onLINE). À l'ère d'internet, MEDLINE est devenu accessible au public via le moteur de recherche PubMed, qui reste aujourd'hui la base de données bibliographiques faisant autorité pour la littérature des sciences de la vie.

Le mode d'accès à l'information que nous tenons pour acquis aujourd'hui - basé sur le classement de représentations de documents et de requêtes construites automatiquement - s'est progressivement imposé, bien que l'histoire de la recherche d'information ait montré qu'il s'agissait d'une bataille difficile. Écrivant sur les débuts de l'histoire de la recherche d'information, Harman [Har19] va jusqu'à parler de « guerres d'indexation » : la bataille entre les termes d'indexation dérivés de l'Homme et ceux générés automatiquement. Cela rappelle quelque peu les « guerres » entre le traitement automatique du langage naturel basé sur des règles et celui basé sur les statistiques [Mar94] qui ont eu lieu à la fin des années 1980 et dans les années 1990, et montre que les changements d'idées fondamentaux rencontrent souvent une résistance initiale.

Parce qu'il a rassemblé toutes les idées majeures de ce revirement du domaine, Salton et al. [SWY75] est fréquemment cité pour la proposition du modèle de l'espace vectoriel, dans lequel les documents et les requêtes sont tous deux représentés comme des « sacs de mots » (« bag-of-words ») à l'aide de vecteurs clairsemés en fonction d'un système de pondération des termes (TF-IDF dans ce cas), où la similarité document-requête est calculée en termes de similarité cosinus défini plus tard (ou, plus généralement, de produits intérieurs). Toutefois, cette évolution ne s'est pas produite d'un seul coup, et plusieurs in-

novations se sont progressivement accumulées au cours des deux décennies qui précèdent son achèvement. Pour plus de détails sur les premiers développements historiques de la recherche d'information, nous renvoyons le lecteur intéressé vers Harman [Har19].

2.2.2 Les défis de la correspondance exacte

Pour établir un contraste clair avec les modèles de réseaux neuronaux, la caractéristique la plus marquante de toutes les approches jusqu'à ce stade de l'histoire est leur dépendance exclusive à l'égard de ce que nous appellerions aujourd'hui la « correspondance exacte des termes », c'est-à-dire que les termes des documents et les termes des requêtes doivent exactement se correspondre pour contribuer à un score de pertinence. Étant donné que les systèmes effectuent généralement une étape de racinisation ou désuffixation (« stemming » en anglais), c'est-à-dire l'élimination des suffixes, la correspondance intervient après que les termes aient été normalisés dans une certaine mesure (par exemple, la racinisation garantit que « chat » correspond à « chats »).

Néanmoins, avec des techniques basées sur la correspondance exacte des termes, une fonction de notation entre une requête q et un document d pourrait s'écrire comme suit :

$$S(q,d) = \sum_{t \in q \cap d} f(t)$$

où f est une fonction d'un terme et des statistiques associées, dont les trois plus importantes sont la fréquence des termes (combien de fois un terme apparaît dans un document), la fréquence des documents (le nombre de documents qui contiennent au moins une occurrence du terme) et la longueur du document (la longueur du document dans lequel le terme apparaît). C'est à partir des deux premières statistiques que nous dérivons l'omniprésente fonction de notation TF-IDF (Term Frequency-Inverse Document Frequency). Dans le modèle de l'espace vectoriel, la similarité cosinus a une composante de normalisation qui gère implicitement les problèmes liés à la longueur des documents.

L'exploration de différents schémas de pondération des termes dans le modèle de l'espace vectoriel [SB88], basés sur des statistiques faciles à calculer liées aux termes, telles que celles décrites ci-dessus, a constitué l'un des principaux axes de recherche dans les années 1980 et dans les années 1990. L'une des méthodes les plus réussies, Okapi BM25 [RZ09, KT, RWJ+94], constitue encore aujourd'hui le point de départ de nombreuses approches de classement de textes, tant dans la recherche universitaire que dans les solutions commercialisées.

Compte tenu de l'importance de BM25, la fonction de notation exacte mérite d'être répétée pour illustrer ce à quoi ressemble un modèle de classement basé sur la correspondance exacte des termes. Le score de pertinence d'un document d par rapport à une requête q est défini comme suit :

$$BM25(q,d) = \sum_{t \in q \cap d} log \frac{N - df(t) + 0.5}{df(t) + 0.5} \cdot \frac{tf(t,d) \cdot (k_1 + 1)}{tf(t,d) + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})}$$

BM25 étant basé sur la correspondance exacte des termes, le score est dérivé de la somme des contributions de chaque terme de la requête apparaissant dans le document. Plus précisément :

— le premier facteur du terme de la somme (le terme logarithmique) est la composante IDF (fréquence inverse du document) : N est le nombre total de documents dans le corpus, et $df(t)$ est le nombre de documents qui contiennent le terme t (c'est-à-dire sa fréquence de document).

— le deuxième facteur du terme de la somme, $tf(t, d)$ représente le nombre de fois où le terme t apparaît dans le document d (c'est-à-dire sa fréquence). L'expression du dénominateur est responsable de la normalisation de la longueur, étant donné que les collections ont généralement des documents de longueur différente : l_d est la longueur du document d tandis que L est la longueur moyenne du document pour tous les documents de la collection.

Enfin, k_1 et b sont des paramètres libres. Il convient de noter que la formulation originale de Robertson et al. [RWJ+94] comprend des composantes de notation supplémentaires avec les paramètres k_2 et k_3, mais elles sont rarement utilisées et sont souvent omises dans les implémentations modernes. Outre la fonction de notation originale décrite ci-dessus, il existe plusieurs variantes de BM25 qui ont été discutées dans la littérature [KdVBL20].

Si les systèmes de pondération des termes peuvent modéliser l'importance des termes (parfois appelée « saillance ») sur la base des propriétés statistiques des textes, les techniques de correspondance exacte sont fondamentalement impuissantes dans les cas où les termes des requêtes et des documents ne correspondent pas du tout. Cela se produit assez fréquemment, lorsque les utilisateurs utilisent des termes différents de ceux utilisés par les auteurs des documents pertinents pour décrire leurs besoins d'information. Une façon de concevoir la recherche d'information est de la voir comme un scénario où un utilisateur cherchant une information tente de deviner les termes (c'est-à-dire la requête) que les auteurs des textes pertinents auraient utilisés lorsqu'ils ont rédigé les textes. Il s'agit du « problème d'inadéquation du vocabulaire » [FLGD87], qui représente un défi fondamental dans la recherche d'information.

En prenant une vue de hauteur, les modèles de recherche traités jusqu'à maintenant s'opposent à l'appariement sémantique (aussi appelé « soft matching ») permis par les représentations continues dans les réseaux neuronaux, où les termes de la requête ne doivent pas nécessairement correspondre exactement aux termes du document afin de contribuer à la pertinence. L'appariement sémantique fait référence à des techniques et à des tentatives pour traiter une variété de phénomènes linguistiques, y compris la synonymie, la paraphrase, la variation des termes et différentes expressions d'intentions similaires, en particulier dans le contexte de l'accès à l'information [LX14]. Suivant cet usage, la « correspondance par pertinence » est souvent utilisée pour décrire les correspondances entre les requêtes et les textes qui expliquent qu'un texte soit pertinent pour une requête. Ainsi, la correspondance par pertinence est généralement considérée comme comprenant à la fois des éléments de correspondance exacte et de correspondance sémantique. Cependant, il existe une autre phase majeure dans le développement des techniques de classement avant d'aborder la correspondance sémantique et la manière dont les réseaux neuronaux

l'accomplissent. Il s'agit de l'apprentissage de classements.

2.2.3 L'émergence de l'apprentissage de classements

L'apprentissage de classements (« learning to rank ») se réfère à une approche utilisant des techniques d'apprentissage automatique pour construire automatiquement le modèle de classement. BM25 et les systèmes de pondération de termes sont généralement considérés comme non supervisés, bien qu'ils contiennent des paramètres libres (par exemple, k_1 et b) qui peuvent être réglés en fonction des données d'apprentissage. Une évolution majeure dans le classement de textes, qui a démarré à la fin des années 1980, est l'application de techniques d'apprentissage automatique supervisé pour apprendre des modèles de classement : les premiers exemples comprennent [Fuh89, WCY93, Gey94a]. Les données d'apprentissage consistent en des listes d'éléments avec un ordre partiel spécifié entre les éléments de chaque liste. Cette approche fait largement appel à des caractéristiques élaborées manuellement, basées principalement sur les propriétés statistiques des termes contenus dans les textes ainsi que sur les propriétés intrinsèques des textes :

— Les propriétés statistiques des termes comprennent des fonctions des fréquences des termes, des fréquences des documents, des longueurs des documents, etc., les mêmes éléments qui apparaissent dans une fonction de notation telle que BM25. En fait, les scores BM25 entre la requête et divers champs du document (comme le titre ou le corps du document) sont généralement inclus en tant que caractéristiques dans une configuration d'apprentissage de classements. Souvent, les caractéristiques intègrent des contraintes de proximité, telles que la fréquence de co-occurrence d'une paire de termes dans un intervalle de cinq positions. Les contraintes de proximité peuvent être localisées dans un champ spécifique du texte, par exemple la co-occurrence de termes dans le titre d'une page web.

— Les propriétés intrinsèques des textes, allant de statistiques très simples, telles que la quantité de code JavaScript sur une page web ou le rapport entre les balises HTML et le contenu, à des mesures plus sophistiquées, telles que la qualité éditoriale ou le score de spam déterminé par un classifieur. Dans le contexte du web, les caractéristiques du graphe des hyperliens, telles que le nombre de liens entrants et sortants et les scores de PageRank [PBMW99] (un algorithme qui mesure quantitativement la popularité d'une page web et à été adapté au classement de textes), sont également courantes.

Un moteur de recherche utilisé dans des scénarios réels peut comporter des centaines de caractéristiques (voire plus). Pour les systèmes disposant d'une base d'utilisateurs suffisamment importante, les caractéristiques basées sur le comportement des utilisateurs - par exemple, le nombre de fois où les utilisateurs ont émis une requête particulière ou ont cliqué sur un lien particulier (dans différents contextes) - sont des signaux de pertinence très précieux et sont parfaitement intégrés dans les méthodes d'apprentissage par classement.

L'essor de l'apprentissage de classements s'explique en grande partie par l'importance croissante des moteurs de recherche en tant qu'outils indispensables pour naviguer sur

le web, car les approches antérieures basées sur des répertoires gérés par un être humain (par exemple, Yahoo !) sont vite devenues impossible à maintenir face à l'explosion du contenu disponible. Un pan des modèles de classements utilise des informations issues des comportements des utilisateurs sur le web. En effet, les données de journal (« log data ») capturant les traces comportementales des utilisateurs (par exemple, les requêtes et les clics) peuvent être utilisées pour améliorer les modèles de classement appris par la machine. Parmi les innovations notables qui ont joué un rôle important dans cette croissance, citons le développement et l'amélioration des techniques d'interprétation des clics « de bruit » des utilisateurs et la conversion de ces clics en données de référence [Joa02, RJ05].

Les méthodes d'apprentissage de classements peuvent être divisées en trois grandes familles d'approches, basées sur la forme générale de leurs fonctions de perte (« loss function »). L'apprentissage revient à minimiser une fonction empirique de risque qui est définie comme la somme sur les éléments de la collection considérée, auxquels la fonction de perte est appliquée. La forme de la fonction de perte peut varier :

— **Une approche par points.** Dans cette famille d'approches la fonction de pertes est appliquée sur les documents individuellement, et transforme le problème de classement en problème de classification ou de régression.
— **Une approche par paire.** Cette famille d'approches considère les pertes sur des paires de documents et se concentre donc sur les préférences, c'est-à-dire la propriété selon laquelle A est plus pertinent que (ou préférable à) B.
— **Une approche par liste.** Cette famille d'approches prend en compte les pertes sur des listes entières de documents, par exemple en optimisant directement une mesure de classement telle que le gain cumulatif actualisé normalisé.

Le lecteur intéressé peut se référer aux travaux de [Liu09, Li11] pour un traitement exhaustif des modèles d'apprentissage de classements.

Comme cette catégorisation se concentre sur la forme de la fonction de perte, elle peut également être utilisée pour décrire les techniques de classement avec transformeurs.

L'apprentissage de classements a atteint son apogée au début des années 2010, à la veille de la révolution de l'apprentissage profond, avec le développement de modèles basés sur des ensembles d'arbres [Bur10]. À cette époque, un consensus émergeait sur le fait que les modèles à base d'arbres, et en particulier les arbres de décision boostés par gradient [GCL11], représentaient la solution la plus efficace pour l'apprentissage de classements. Durant cette période, les ensembles d'arbres avaient été déployés pour résoudre un large éventail de problèmes ; une réussite notable est le rôle important qu'ils ont joué dans l'obtention du prix Netflix, un concours très médiatisé qui vise à améliorer la qualité des recommandations de films.

Il convient de noter que l'expression « apprentissage de classements » ne doit pas être comprise comme étant synonyme d'« approches supervisées d'apprentissage automatique de classements ». L'apprentissage de classements fait plutôt référence à des techniques apparues au cours d'une période spécifique de l'histoire de la recherche d'information. Les transformeurs pour le classement de textes peuvent être caractérisés comme une approche d'apprentissage automatique supervisé, mais ne seraient généralement pas considérés comme une méthode d'apprentissage pour le classement. En particulier, il existe une

caractéristique clé qui distingue l'apprentissage de classements des approches d'apprentissage profond qui ont suivi. L'important n'est pas le modèle spécifique d'apprentissage automatique supervisé : en fait, les réseaux neuronaux sont utilisés depuis le début des années 1990 [WCY93], et RankNet [Bur10], l'un des modèles d'apprentissage de classements les plus influents et les plus connus, a adopté une architecture neuronale « feedforward » de base. En revanche, l'apprentissage du classement se caractérise par l'utilisation de nombreuses caractéristiques éparses, généralement créées à la main. Toutefois, pour brouiller un peu les pistes, l'expression « apprentissage profond pour le classement » a récemment émergé dans le discours pour décrire les approches d'apprentissage profond qui intègrent également des caractéristiques éparses [PBW+19].

2.2.4 L'apparition de l'apprentissage profond

Dans le contexte du classement de textes, l'apprentissage profond est apparu après l'apprentissage de classements, à la suite d'un fort enthousiasme des communautés de la vision par ordinateur et du traitement automatique du langage naturel pour cette technologie. Dans le contexte de la recherche d'information, les approches d'apprentissage profond sont intéressantes pour deux raisons : d'abord, les représentations vectorielles continues lèvent les limites de la correspondance exacte des termes abordées précédemment (nous verrons comment ci-dessous). Ensuite, les réseaux neuronaux sont prometteurs pour éliminer le besoin de caractéristiques laborieusement créées à la main (ce qui répond à une difficulté majeure dans la construction de systèmes d'apprentissage de classements).

Dans la niche des approches d'apprentissage profond pour le classement de textes, il est utile de distinguer davantage les modèles « pré-BERT » des modèles basés sur BERT (et plus généralement, des modèles transformeurs). Dans le Deep Learning Track de TREC 2019, la première évaluation à grande échelle des techniques de recherche d'information après l'introduction de BERT, son impact, et plus généralement, l'impact des modèles de langage neuronaux pré-entraînés, a été clairement démontré [CMY+20]. L'analyse des résultats montre que, considérés comme une famille de techniques, les modèles basés sur BERT sont nettement plus performants que les modèles pré-BERT. Les organisateurs de l'évaluation ont reconnu qu'il s'agissait d'une distinction significative qui séparait deux « époques » différentes dans le développement d'approches neuronales profondes pour le classement de textes.

Dans la suite, nous fournissons un aperçu non exhaustif des modèles pré-BERT. Nous pointons le lecteur intéressé vers une étude plus complète sur ce sujet [OZ18, MC18, XHL20] au besoin. Il est à noter que nous nous intéressons à des modèles spécifiques au classement de textes et n'abordons pas un vaste pan de la littérature, principalement dû à la communauté du traitement automatique du langage naturel, qui traite du problème très proche de comparaison de similarités sémantiques. Les modèles pour ces tâches distinctes partagent de nombreux aspects et sont à l'origine d'un rapprochement entre les communautés de la recherche d'information et du traitement automatique du langage naturel. Néanmoins, une différence importante entre ces deux problèmes scientifiques est

que les entrées des modèles de comparaison de similarités sémantiques sont symétriques (i.e. $R(s_1, s_2) = R(s_2, s_1)$) alors qu'une requête est clairement différente d'un texte pour le classement de textes.

Les modèles de classement neuronal pré-BERT sont généralement classés en deux catégories : les modèles basés sur la représentation et les modèles basés sur l'interaction. Les modèles basés sur la représentation se concentrent sur l'apprentissage indépendant de représentations vectorielles denses des requêtes et des documents qui peuvent être comparées pour calculer la pertinence au moyen d'une métrique simple telle que la similarité cosinus ou les produits intérieurs. Les modèles basés sur l'interaction comparent les représentations des termes de la requête avec celles des termes d'un document pour produire une matrice de similarité qui capture les interactions entre les termes. Cette matrice fait ensuite l'objet d'une analyse plus poussée afin d'obtenir un score de pertinence. Dans les deux cas, les modèles peuvent intégrer de nombreux composants neuronaux différents (par exemple, des réseaux neuronaux convolutionnels [ON15] et des réseaux neuronaux récurrents [Sch19]) pour extraire des signaux de pertinence.

Les modèles basés sur la représentation et ceux basés sur l'interaction sont généralement formés de bout en bout avec des jugements de pertinence (abordés ultérieurement), en utilisant uniquement les plongements (définis dans le Chapitre 3 des termes de la requête et du document comme données d'entrée. Notamment, des caractéristiques supplémentaires (créées à la main) ne sont généralement pas utilisées, ce qui constitue une différence majeure par rapport à l'apprentissage de classements. Nous fournissons ci-dessous plus de détails, avec des exemples illustratifs :

— **Les modèles à représentation.** Cette classe de modèles apprend les représentations vectorielles des requêtes et des documents pour calculer les scores de pertinence requête-document. Elle permet de calculer les représentations des documents de manière hors ligne (« offline »). L'un des premiers modèles de classement neuronal de l'ère de l'apprentissage profond, le Deep Structure Semantic Model (DSSM) [HHG$^+$13] construit des n-grammes (une sous-séquence de n éléments construite à partir d'une séquence donnée) de caractères à partir d'une entrée (c'est-à-dire une requête ou un document) et transmet les résultats à une série de couches neuronales pleinement connectées pour produire une représentation vectorielle. Au moment de l'extraction, les représentations de la requête et du document peuvent ensuite être comparées à l'aide de la similarité cosinus. [SHG$^+$14] ont amélioré DSSM en utilisant des CNNs (Convolutional Neural Network) pour capturer le contexte. Plutôt que d'apprendre les représentations de texte dans le cadre du modèle, Dual Embedding Space Model (DESM) [MNCC16, NMCC16] représente les textes à l'aide de plongements word2vec pré-entraînés [LM14] et calcule les scores de pertinence en agrégeant les similitudes cosinus entre toutes les paires de termes requête-document. Les modèles linguistiques basés sur les plongements de mots [GRMJ15] peuvent également être classés dans la catégorie des modèles basés sur la représentation. Il est intéressant de noter que nous assistons à un regain d'intérêt pour les approches basées sur la représentation, bien qu'elles utilisent des architectures de transformeurs.

— **Les modèles à interactions.** Cette catégorie de modèles capture explicitement les « interactions » entre les termes de la requête et les termes du document. Ces interactions sont généralement traitées à l'aide d'une matrice de similarité dont les lignes correspondent aux termes de la requête et les colonnes aux termes du document. Chaque entrée $M_{i,j}$ de la matrice représente généralement la similarité cosinus entre le plongement du i-ème terme de la requête et le plongement du j-ème terme du document. Ces modèles fonctionnent en deux étapes : l'extraction des caractéristiques et la notation de la pertinence. Lors de l'étape d'extraction des caractéristiques, le modèle extrait les signaux de pertinence de la matrice de similarité. En exploitant les représentations vectorielles continues des termes, ces modèles peuvent potentiellement surmonter le problème de l'inadéquation de vocabulaire. Les modèles unigrammes comme DRMM [GFAC16] et KNRM [XDC+17] agrègent les similarités entre chaque terme de la requête et chaque terme du document, qui peuvent être vues comme des histogrammes. DRMM crée des histogrammes explicites, tandis que KNRM utilise des noyaux gaussiens pour créer des « histogrammes progressif » différentiables qui permettent d'apprendre les plongements pendant l'entraînement. Les modèles sensibles à la position comme MatchPyramid [PLG+16], PACRR [HYBdM17], CoPACRR [HYBdM18], et ConvKNRM [DXCL18] utilisent des composants architecturaux supplémentaires pour identifier les correspondances entre les séquences de termes de la requête et du document. Dans l'étape de notation de la pertinence, les caractéristiques extraites ci-dessus sont combinées et traitées pour produire un score de pertinence entre la requête et le document. Cette étape consiste souvent à appliquer des opérations de mise en commun, à concaténer les caractéristiques extraites, puis à transmettre la représentation résultante à un réseau de type « feedforward » qui calcule le score de pertinence.
— **Les modèles hybrides.** Enfin, les approches basées sur la représentation et sur l'interaction ne s'excluent pas mutuellement. Un hybride bien connu est le modèle DUET [MDC16, MC19], qui augmente un composant d'apprentissage de représentation avec un composant basé sur l'interaction responsable de l'identification des correspondances exactes de termes.

En général, les études ont montré que les modèles à interactions pré-BERT sont plus efficaces mais plus lents que les modèles à représentations pré-BERT. Ces derniers réduisent le classement de textes à de simples comparaisons de similarité entre les vecteurs de requête et les vecteurs de documents précalculés, qui peuvent être effectuées rapidement sur de grands corpus à l'aide des techniques de recherche du plus proche voisin. En revanche, les modèles à interactions sont généralement déployés en tant que re-classeur (« reranker ») sur un ensemble de documents candidats récupérés via une recherche par mot-clé. Les modèles à interactions conservent également la capacité de capturer explicitement les signaux de correspondance exacte, qui restent importants dans l'identification de pertinence.

Il y a indéniablement eu tout au long des années 2010 une profusion de travaux explorant un large éventail d'architectures neuronales pour le classement de textes, mais

jusqu'où le domaine a-t-il concrètement progressé, en particulier depuis que les approches basées sur l'apprentissage profond nécessitent de grandes quantités de données d'apprentissage ? [Lin19] s'interroge (non sans provocation) sur l'efficacité des modèles de classement neuronaux par rapport aux techniques « traditionnelles » d'appariement de mots clés en l'absence des vastes quantités de données d'entraînement nécessaires à ces modèles. Il s'agit d'une question importante, car l'accès à ces données n'est pas toujours possible. Dans quelle mesure les modèles neuronaux de classement fonctionnent-ils avec les quantités limitées de données d'entraînement qui sont accessibles au public ? [YLYL19] ont répondu à cette question en comparant plusieurs modèles de classement neuronal à interaction et à représentation, basés sur une implémentation par « sac de mots » avec une expansion de requête optimisée sur l'ensemble de données du TREC 2004 Robust Track [Voo04]. Dans ces conditions de données limitées, la plupart des méthodes de classement neuronal ont été incapables de battre le modèle référence de recherche par mot-clé. [YJZL20] ont reproduit ce résultat pour un ensemble élargi de méthodes de classement neuronal avec des implémentations complètement différentes, ce qui a renforcé les conclusions des premiers travaux. Bien que de nombreux articles cités ci-dessus fassent état d'améliorations significatives lorsqu'ils sont entraînés sur de grands ensembles de données propriétaires (dont beaucoup comprennent des signaux comportementaux), les résultats sont difficiles à valider et les avantages des méthodes proposées ne sont pas largement accessibles à la communauté. Cependant, avec BERT, tout a changé, presque du jour au lendemain.

2.2.5 L'irruption du modèle BERT

BERT [DCLT19] est apparu en octobre 2018. La première application de BERT au classement de textes a été rapportée par [NC20] en janvier 2019 sur la collection de tests de classement de passages MS MARCO [BCC$^+$18a], dans laquelle la tâche précise utilisée consiste à classer des passages (extraits textuels de longueur de paragraphes) de pages web en fonction de requêtes en langage naturel des utilisateurs. BERT a permis un progrès des performances de classement de textes qui a été rarement observé dans le monde académique et dans le monde industriel, ce qui a provoqué un engouement immédiat de la communauté scientifique.

La simplicité du modèle a conduit à une reproduction rapide et à grande échelle des résultats. En quelques semaines, au moins deux autres équipes ont confirmé l'efficacité de BERT pour le classement de passages, et l'exploration de variantes de modèles fondées sur les idées originales de [NC20] a suivi. Le scepticisme exprimé par [Lin19] s'est dissipé peu de temps après [Lin21], car de nombreux chercheurs ont rapidement démontré qu'avec des modèles de transformeurs pré-entraînés, les grandes quantités de jugements de pertinence n'étaient pas nécessaires pour construire des modèles efficaces pour le classement de textes. La disponibilité de la collection de tests de classement de passages MS MARCO a encore atténué les problèmes de disponibilité des données. La combinaison de ces facteurs a permis l'exploration de modèles neuronaux de pointe pour le classement qui est maintenant à la portée des groupes de recherche universitaires, et n'est plus li-

mitée aux chercheurs de l'industrie ayant accès à de gigantesques quantités de données d'entraînement.

[NC20] ont donné le coup d'envoi de la « révolution BERT » pour le classement de textes, et la communauté scientifique s'est rapidement attelée à exploiter leurs résultats - en corrigeant les limitations et en développant cet axe de recherche de diverses manières. Les publications contemporaines de travaux dans le classement de textes sont dominées par BERT. Étant donné sa centralité évidente dans le domaine, nous apportons dans ce qui suit, un aperçu haut niveau du modèle BERT.

Dans sa version de base, BERT (« Bidirectional Encoder Representations from Transformers ») est un modèle de réseau de neurones permettant de générer des plongements contextuels pour des séquences d'entrée en anglais, avec une variante multilingue (souvent appelée « mBERT ») qui peut traiter des entrées dans plus de 100 langues différentes. Nous nous concentrons ici uniquement sur le modèle monolingue anglais, mais mBERT a également été largement étudié [WD19, PSG19, ARY20]. BERT prend en entrée une séquence de tokens (plus précisément, des représentations vectorielles d'entrée dérivées de ces tokens) et produit une séquence de plongements contextuels, qui fournissent des représentations dépendantes du contexte des tokens d'entrée. Cela contraste avec les représentations indépendantes du contexte (c'est-à-dire statiques), que proposent de nombreuses techniques largement adoptées auparavant, telles que word2vec [MSC+13] ou GloVe [PSM14]. Le comportement entrée-sortie de BERT est représenté par des vecteurs d'entrée qui sont désignés par :

$$[E_{[CLS]}, E_1, E_2, ..., E_{[SEP]}],$$

et les représentations des vecteurs de sortie obtenus sont :

$$[T_{[CLS]}, T_1, T_2, ..., T_{[SEP]}],$$

après avoir traversé un certain nombre de couches de transformeurs et d'encodeurs. Outre le texte à traiter, l'entrée de BERT comprend généralement deux tokens spéciaux, [CLS] et [SEP], que nous détaillons ci-après.

BERT peut être considéré comme un modèle plus sophistiqué mais ayant les mêmes objectifs qu'ELMo [PNI+18], dont BERT tire de nombreuses idées importantes : l'objectif des plongements contextuels est de capturer des caractéristiques complexes du langage (par exemple, la syntaxe et la sémantique) ainsi que la façon dont les significations varient à travers les contextes linguistiques (par exemple, la polysémie). La principale différence réside dans le fait que BERT tire parti des transformeurs, alors qu'ELMo utilise des LSTMs [HS97a]. BERT peut être considéré comme la « moitié encodeur » de l'architecture transformeur complète proposée par [VSP+17], qui a été conçue pour des tâches de séquence à séquence (c'est-à-dire où l'entrée et la sortie sont des séquences de tokens) telles que la traduction automatique.

BERT se distingue également de GPT [RN18]. Si BERT peut être considéré comme un transformeur exclusivement d'encodage, GPT est l'opposé : il représente un transformeur exclusivement de décodage [LSP+18], ou la « moitié décodeur » d'un modèle

complet de transformeur de séquence à séquence. GPT est pré-entraîné pour prédire le prochain mot d'une séquence en fonction de son historique ; en revanche, BERT utilise un objectif différent, ce qui conduit à une distinction importante discutée ci-dessous. BERT et GPT sont souvent regroupés (avec une foule d'autres modèles) et désignés collectivement comme des modèles de langage pré-entraînés, bien que cette caractérisation soit quelque peu trompeuse car, à proprement parler, un modèle de langage en traitement automatique du langage naturel fournit une distribution de probabilité sur des séquences arbitraires de tokens de texte ; voir, par exemple [CG96]. En réalité, l'obtention de telles probabilités à partir de BERT nécessite quelques efforts [SLNK20].

L'avancée significative que GPT et BERT apportent par rapport à la formulation originale du transformeur [VSP+17] est l'utilisation de l'auto-supervision dans le pré-entraînement, alors que [VSP+17] ont commencé par une initialisation aléatoire des poids du modèle et ont procédé à un entraînement direct sur des données étiquetées, c'est-à-dire des paires (séquence d'entrée, séquence de sortie), de manière supervisée. Il s'agit là d'une distinction importante, car l'idée d'un pré-apprentissage basé sur l'auto-supervision est sans doute ce qui améliore la qualité des résultats des modèles dans une multitude de tâches de traitement du langage. L'intérêt de l'auto-supervision est double :

— L'optimisation du modèle n'est plus liée à des données étiquetées. L'auto-supervision signifie que les textes fournissent leurs propres « étiquettes » (dans GPT, l'« étiquette » pour une séquence de tokens est le token suivant apparaissant dans la séquence), et la fonction de perte (« loss ») peut être calculée à partir de la séquence elle-même (sans nécessiter d'autres annotations externes). Étant donné que les données étiquetées découlent en fin de compte de l'effort humain, la suppression du besoin d'étiquettes augmente considérablement la quantité de données pouvant être transmises aux modèles pour le pré-apprentissage. Souvent, la puissance de calcul et les données disponibles deviennent un goulot d'étranglement (« bottleneck ») [KMH+20].

— Les modèles optimisés sur la base d'un ou plusieurs objectifs auto-supervisés, sans référence à une tâche spécifique, fournissent de bons points de départ pour un ajustement ultérieur avec des données étiquetées spécifiques à une tâche. C'est ainsi qu'est née la recette « pré-entraîner d'abord, affiner ensuite » pour travailler avec BERT et les modèles apparentés. Les détails de ce processus de réglage fin sont spécifiques à la tâche, mais les expériences ont montré qu'une quantité modeste de données étiquetées est suffisante pour atteindre un niveau élevé d'efficacité. Ainsi, le même modèle pré-entraîné peut servir de point de départ pour l'exécution de multiples tâches en aval après un réglage fin approprié.

GPT a précédé BERT pour la combinaison des deux composants cruciaux que sont l'utilisation de modèles transformeurs et l'auto-supervision. Néanmoins, ils implémentent cette idée de manières différentes. GPT utilise une fonction objectif avec un modèle traditionnel de langage, étant donné un corpus de tokens $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, le but est de maximiser la fonction de vraisemblance suivante :

$$L(\mathcal{U}) = \sum_i logP(u_i|u_{i-k}, ..., u_{i-1}; \Theta)$$

où k est la taille de la fenêtre contextuelle et la probabilité conditionnelle est modélisée par un transformeur de paramètres Θ.

En revanche, BERT a introduit la fonction objectif de pré-entraînement appelé « modélisation de langage masqué » (MLM pour « Masked Language Modeling »), qui s'inspire de la tâche « Cloze » [Tay53], datant d'il y a plus d'un demi-siècle. MLM est un nom sophistiqué pour une idée assez simple : pendant le pré-apprentissage, nous masquons au hasard un token de la séquence d'entrée et demandons au modèle de le prédire, en s'entraînant avec une fonction de perte d'entropie croisée. L'objectif MLM explique le « B » de BERT, qui signifie bidirectionnel : le modèle est capable d'utiliser à la fois les contextes à la gauche et à la droite d'un token masqué (contextes précédents et suivants) pour faire des prédictions. En revanche, étant donné que GPT utilise une fonction objectif de modélisation linguistique, il ne peut utiliser que les jetons précédents (c'est-à-dire le contexte de gauche dans une langue écrite de gauche à droite ; formellement, cela s'appelle « autorégressif »). Il a été empiriquement observé que la modélisation bidirectionnelle apporte une grande différence, comme le démontre, par exemple, les performances supérieures de tels modèles sur le benchmark GLUE [WSM+19].

Bien que la fonction objectif MLM soit une invention de BERT, l'idée du pré-entrainement a une longue histoire. ULMFiT (« Universal Language Model Fine-tuning ») [HR18] a probablement le mérite d'avoir popularisé l'idée du pré-entraînement à l'aide de fonctions objectif de modélisation du langage, puis d'un réglage fin sur des données spécifiques à une certaine tâche - la méthode qui est devenue universelle aujourd'hui - mais l'application du pré-entraînement au traitement automatique du langage naturel peut être attribuée à [DL15]. Si l'on remonte encore plus loin dans les origines intellectuelles de cette idée, l'inspiration initiale vient de la communauté de la vision par ordinateur [EMB+09]. Les séquences d'entrée de BERT sont généralement tokenisées avec le tokeniseur WordPiece [WSC+16], bien que BPE [SHB16]] soit une alternative courante, utilisée par GPT ainsi que RoBERTa [LOG+19]. Ces tokeniseurs ont pour objectif de réduire l'espace du vocabulaire en découpant les mots en « sous-mots », généralement de manière non supervisée. Dans la suite, nous prenons quelques exemples en anglais pour illustrer le comportement de BERT étant donné que la version la plus utilisée traite cette langue. Avec le vocabulaire WordPiece utilisé par BERT, « scrolling » devient « scroll » + « ##ing ». La convention consistant à ajouter deux croisillons (##) à un sous-mot indique qu'il est « connecté » au sous-mot précédent (c'est-à-dire que dans une langue habituellement écrite avec des espaces, il n'y a pas d'espace entre le sous-mot actuel et le précédent).

Pour l'essentiel, toute correspondance entre des mots et des unités linguistiques significatives doit être considérée comme accidentelle. Par exemple, « walking » et « talking » ne sont pas divisés en sous-mots, et « biking » est divisé en « bi » + « ##king », qui ne correspondent manifestement pas à des morphèmes. Des exemples encore plus extrêmes sont « biostatistics » (« bio » + « ##sta » + « ##tist » + « ##ics ») et « adversarial » (« ad », « ##vers », « ##aria », « ##l »). Néanmoins, le principal avantage de la tokenisation WordPiece (et des méthodes associées) est qu'un vocabulaire relativement restreint (par exemple, 30 000 mots) est suffisant pour modéliser de grands corpus qui peuvent contenir des millions de tokens uniques (sur la base d'une méthode simple telle

que la tokenisation par espacement). Bien que BERT convertisse une séquence de plongements d'entrée en une séquence de plongements contextuels correspondants, dans la pratique, il est principalement appliqué à quatre types de tâches :

— Les tâches de classification à entrée unique, par exemple l'analyse de sentiments sur un seul segment de texte.
— Les tâches de classification à deux entrées, par exemple, détecter si deux phrases sont des paraphrases. En principe, la régression est également possible dans ce cas.
— Les tâches d'étiquetage de tokens à entrée unique, par exemple la reconnaissance d'entités nommées (NER). Pour ces tâches, une étiquette est attribuée à chaque token de l'entrée, contrairement à la classification à entrée unique, où l'étiquette est attribuée à l'ensemble de la séquence.
— Les tâches d'étiquetage de tokens à deux entrées, par exemple la réponse aux questions, formulée comme la tâche d'étiquetage des positions de début et de fin dans un texte réponse candidat (typiquement, la deuxième entrée) étant donné une question (typiquement, la première entrée).

Le premier token de chaque séquence d'entrée de BERT est un token spécial appelé [CLS] ; la représentation finale de ce token spécial est généralement utilisée pour les tâches de classification. Le token [CLS] est suivi de l'entrée ou des entrées : il s'agit généralement, mais pas toujours, de phrases - en effet, les entrées comprennent des textes candidats à classer, qui sont généralement plus longs que des phrases individuelles. Pour les tâches impliquant une seule entrée, un autre token spécial de délimitation [SEP] est ajouté à la fin de la séquence d'entrée. Pour les tâches impliquant deux entrées, celles-ci sont regroupées en une seule séquence contiguë de tokens séparés par le token [SEP], un autre token [SEP] étant ajouté à la fin. Pour les tâches d'étiquetage de tokens sur des entrées uniques (par exemple, la reconnaissance d'entités nommées), l'intégration contextuelle du premier sous-mot est généralement utilisée pour prédire l'étiquette qui devrait être attribuée au token (par exemple, dans un format d'étiquetage IOB, abréviation de « inside, outside, beginning », qui est couramment utilisé pour étiqueter les tokens dans une tâche de regroupement linguistique comme la reconnaissance d'entités nommées). La réponse aux questions (et plus généralement, les tâches d'étiquetage de tokens impliquant deux entrées) sont traitées d'une manière conceptuellement similaire, où le modèle tente d'étiqueter les positions de début et de fin du texte réponse.

Afin que le modèle comprenne la relation entre les différents segments de texte (dans le cas à deux entrées), BERT est également pré-entraîné avec une tâche de « prédiction de la phrase suivante » (NSP pour « Next Sentence Prediction »), où le modèle apprend les plongements des segments, une sorte d'indicateur utilisé pour différencier les deux entrées. Pendant le pré-entraînement, après avoir choisi une phrase du corpus (segment A), la phrase suivante réelle du corpus est sélectionnée une fois sur deux pour être incluse dans l'instance d'entraînement (en tant que segment B), tandis que l'autre moitié du temps une phrase aléatoire du corpus est choisie à la place. La tâche de NSP est de prédire si la deuxième phrase suit effectivement la première. [DCLT19] ont émis l'hypothèse que le pré-entraînement NSP est important pour les tâches en aval, en particulier celles

qui prennent deux entrées. Cependant, les travaux ultérieurs de [LOG+19] ont remis en question la nécessité du NSP ; en fait, sur un large éventail de tâches de traitement automatique du langage naturel, ils n'ont observé aucune dégradation de l'efficacité dans les modèles dépourvus d'un tel pré-entraînement.

Pour résumer, la représentation d'entrée de BERT pour chaque token comprend trois composants :

— le plongement appris du token à partir du tokeniseur WordPiece [WSC+16] ;
— le plongement du segment, qui est un plongement appris indiquant si le token appartient à la première entrée (A) ou à la deuxième entrée (B) dans les tâches impliquant deux entrées ;
— le plongement de la position, qui est un plongement appris capturant la position de l'élément au sein d'une séquence, et qui donne à BERT des informations sur la position des éléments.

La représentation finale de l'entrée de BERT comprend, pour chaque token, la somme de ces plongements. Il convient de souligner que les trois composantes de plongements sont additionnées, et non pas assemblées par concaténation de vecteurs (il s'agit là d'une confusion fréquente). Les représentations constituant la séquence d'entrée de BERT passent par plusieurs couches d'encodage transformeur pour produire les plongements contextuels de sortie. Le nombre de couches, la dimension de la couche cachée (« hidden layer ») et le nombre de têtes d'attention sont des hyperparamètres dans l'architecture du modèle.

Nous concluons notre discussion de haut niveau sur BERT en notant que sa popularité est en grande partie due aux décisions judicieuses des auteurs (et à l'approbation de Google) de ne pas seulement mettre en source ouverte l'implémentation du modèle, mais aussi de rendre publics les modèles pré-entraînés (dont le pré-entraînement est coûteux en calcul). Cela a permis une reproduction rapide des résultats impressionnants obtenus dans l'article original et a fourni à la communauté une implémentation de référence sur laquelle s'appuyer. Aujourd'hui, la bibliothèque Transformers de Hugging Face [WDS+20] s'est imposée comme l'implémentation standard de facto de BERT et fournit de nombreux modèles de transformeurs, pris en charge à la fois par PyTorch [PGM+19] et TensorFlow [ABC+16], qui sont les deux bibliothèques d'apprentissage profond les plus populaires du moment.

Il ne fait aucun doute que nous sommes actuellement dans "l'ère" de BERT et des modèles transformeurs. Il est certain que de nouvelles technologies émergeront et supplanteront complètement ces modèles, marquant ainsi l'avènement d'une nouvelle ère. Néanmoins, il reste encore beaucoup d'investigations à mener avec les transformeurs. Ces investigations peuvent planter les graines ou inspirer la prochaine révolution à venir. Nous espérons que notre étude constituera une feuille de route pour les explorations à venir.

Chapitre 3

Une revue des plongements de graphes bipartis

Sommaire

3.1 Contexte et vue d'ensemble 51
3.2 Notations et formulation du problème 52
3.2.1 Notations 52
3.2.2 Définitions 52
3.3 Taxonomie 55
3.3.1 Approches de préservation de la proximité 56
3.3.2 Approches basées sur le passage de messages 59
3.3.3 Discussion 63
3.4 Outils d'évaluation et d'expérimentation 64
3.4.1 Tâches d'évaluation 64
3.4.2 Jeux de données de référence 68
3.4.3 Bibliothèques en source ouverte 68
3.5 Conclusion et Directions Futures 69

Le contenu de ce chapitre est basé sur l'article « A survey on Bipartite Graphs Embedding » publié dans Social Network Analysis and Mining, 13, 54 (2023). `https://doi.org/10.1007/s13278-023-01058-z`.

Comme nous l'avons mentionné dans le Chapitre 1, les graphes ou réseaux possèdent une structure intéressante pour saisir la complexité du langage naturel. Néanmoins, il s'agit souvent de structures imposantes dont la taille nécessite l'utilisation de techniques adaptées de représentation. La recherche sur l'apprentissage de la représentation des graphes (aussi appelée « embedding » ou plongement en français) a fait l'objet d'une grande attention ces dernières années en apportant des résultats prometteurs pour divers types de réseaux. Cependant, peu d'initiatives se sont penchées sur le cas particulier des

plongements de graphes bipartis, ayant la particularité de posséder deux types distincts de nœuds. Cette disposition offre une modélisation pertinente dans le contexte de nos travaux pour analyser les dépendances entre les documents et les entités qu'ils contiennent. C'est pourquoi nous étudions, dans ce chapitre, les plongements de graphes bipartis, en définissant tout d'abord le problème du plongement de graphes dans le cas biparti. Ensuite, nous proposons une taxonomie des approches utilisées pour répondre à ce problème et décrivons les méthodes les plus récentes. Nous dressons alors les avantages et inconvénients de chacune des catégories d'approches par rapport aux plongements de réseaux conventionnels. Enfin, nous fournissons une description des ressources disponibles pour mener des expériences sur les plongements de graphes bipartis.

3.1 Contexte et vue d'ensemble

Les réseaux constituent un moyen naturel de modéliser un ensemble varié d'informations. Ils sont utilisés dans des domaines tels que la bio-informatique, les systèmes de recommandation, la finance ou les réseaux sociaux [HI18]. Les réseaux permettent de modéliser des relations complexes. Cependant, cet avantage peut se transformer en inconvénient en raison des capacités de traitement qu'ils requièrent. À l'ère du big data, les réseaux complexes peuvent contenir des milliards de nœuds et d'arêtes. Par conséquent, il peut être difficile d'effectuer des procédures d'inférence complexes sur l'ensemble du réseau.

L'apprentissage de la représentation de réseau, également connu sous le nom de plongement de réseau [AM19], est un moyen courant de contourner cette limitation. Le principe de base consiste à représenter dans un espace de faible dimension tout en préservant les principales caractéristiques du réseau (structure, proximité des nœuds, arêtes). Précisément, l'objectif des plongements est de trouver une fonction qui associe à chaque nœud du réseau une représentation latente à faible dimension. De telles représentations nécessitent moins d'efforts pour être manipulées, elles peuvent donc être utilisées comme donnée d'entrée de tâches courantes sur les graphes telles que la classification de nœuds et sont directement utilisables par divers algorithmes communs d'apprentissage automatique.

Avec l'explosion de la quantité de données, ce domaine de recherche a suscité un vif intérêt de la part de la communauté scientifique et a inspiré de nombreux travaux. En particulier, ces dernières années ont vu l'émergence de plusieurs sous-domaines de plongements de réseau avec un accent particulier sur le type du réseau considéré. Par conséquent, diverses méthodes ont été créées pour étudier des types de réseaux spécifiques, par exemple les réseaux homogènes [CZC17], les réseaux attribués [LSQL21], les réseaux hétérogènes [YXZ+20], les hyper-réseaux [HDL16], les réseaux dynamiques [BMVZ21], ou encore les réseaux bipartis [STFR17]. Dans ce chapitre, les termes « graphe » et « réseau » seront utilisés pour décrire la même structure de données, c'est-à-dire un ensemble de nœuds reliés par des arêtes. En effet, ces termes sont utilisés de manière interchangeable dans la littérature.

Les graphes bipartis ont une importance particulière car de nombreuses applications pratiques telles que la modélisation thématique, le diagnostic médical ou la recommandation peuvent être modélisées par ce type de graphe (voir le Tableau 3.2 pour plus d'exemples d'applications). Néanmoins, l'apprentissage de représentation pour les graphes bipartis n'a reçu que peu d'attention, avec une poignée d'articles qui lui sont consacrés. Dans cette étude, nous essayons d'apporter une vue d'ensemble des travaux liés à ce sujet et de rassembler une liste d'outils disponibles pour explorer ce domaine. À notre connaissance, ce travail est la première revue dédiée au plongement de graphes bipartis.

Le reste de cet article est structuré comme suit. La section 3.2 introduit les notations et fournit un contexte mathématique pour définir le problème du plongement de graphes bipartis. La section 3.3 décrit les méthodes de pointe utilisées pour traiter le plongement

de graphes bipartis et les catégorise dans une taxonomie d'approches. En outre, elle établit les avantages et les inconvénients de chaque méthode et indique des pistes pour de futures améliorations. La section 3.4 décrit les ressources disponibles pour mener des expériences sur le plongement de graphes bipartis. Enfin, la section 3.5 conclut ce travail avec une discussion sur plusieurs problèmes ouverts et des directions de recherche possibles.

3.2 Notations et formulation du problème

Nous introduisons ici les notations et les définitions des concepts nécessaires pour formaliser le problème du plongement de graphes bipartis.

3.2.1 Notations

Tout au long de ce chapitre, nous utilisons les notations suivantes : les scalaires sont désignés par des lettres italiques minuscules (par exemple s) ; les matrices sont désignées par des lettres majuscules en gras (par exemple $\boldsymbol{A}$) ; la transposée d'une matrice $\boldsymbol{A}$ est notée $\boldsymbol{A^T}$; le coefficient à la ligne i et la colonne j est noté a_{ij}.

3.2.2 Définitions

Dans cette section, nous introduisons les termes et définitions utilisés pour décrire le plongement de graphes bipartis.

— **Graphe homogène**
Dans ce travail, nous utilisons le terme « graphe homogène » pour désigner un graphe habituel, par opposition avec un graphe biparti (ou tout autre graphe particulier comme les graphes hétérogènes, les hypergraphes etc.). Le terme « homogène » est utilisé pour souligner que le graphe possède un unique type de nœud. Formellement, un graphe homogène $G = (U, E)$ est défini par un ensemble de nœuds U et un ensemble d'arêtes E entre ces nœuds. Une arête allant d'un nœud $u \in U$ à un nœud $v \in U$ est notée $(u, v) \in E$.

— **Graphe biparti**
Un graphe biparti $G = (U, V, E)$ est composé de deux domaines (ou types) différents de nœuds U et V, et d'un ensemble d'arêtes $E \subset U \times V$. Les graphes bipartis ont la particularité de ne considérer que les arêtes inter-domaines : à l'intérieur d'un domaine (U ou V), deux nœuds ne peuvent pas être connectés. La Figure 3.1 donne un exemple de graphes bipartis où l'on peut observer les deux domaines distincts et leurs connexions. Soit $n = |U|$ et $m = |V|$. Soit u_i le i^{eme} nœud de U, $i = 1, 2, ..., n$ et v_j le j^{eme} nœud de V, $j = 1, 2, ..., m$. Chaque arête (u_i, v_j) peut porter un poids non négatif w_{ij} décrivant la force de la relation entre les nœuds u_i et v_j. Si un graphe n'est pas pondéré, nous pouvons considérer que tous les poids sont égaux à 1. De même, si deux nœuds i et j ne sont pas connectés,

nous fixons $w_{ij} = 0$. Nous pouvons donc utiliser une matrice $n \times m$ $\boldsymbol{W} = (w_{ij})$ pour représenter tous les poids dans le réseau biparti. Précisément, $\boldsymbol{W}$ est appelée matrice d'adjacence du domaine U et la matrice d'adjacence du domaine V est la transposée de $\boldsymbol{W}$: $\boldsymbol{W^T} \in \mathbb{R}^{m \times n}$.

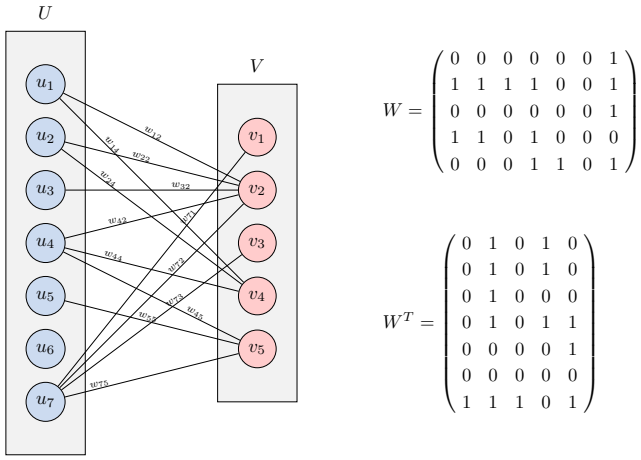

FIGURE 3.1 – Exemple de graphe biparti avec à droite la matrice d'adjacence des deux domaines U et V. Les positions dans la matrice d'adjacence représentent les poids dans le graphe biparti. Le poids $w_{ij} = 1$ si les nœuds u_i et v_j sont connectés et 0 sinon. Dans certains graphes bipartis, les poids peuvent être un scalaire non négatif, décrivant ainsi la force de la relation entre les nœuds.

Les graphes peuvent également être associés à des attributs ou caractéristiques, tels que les informations de profil associées à un utilisateur sur une plateforme en ligne, les mots contenus dans un article ou les genres de films. Le Tableau 3.4 fournit des détails sur les jeux de données de graphes bipartis avec des exemples de types de nœuds de scénarios réels. Les attributs au niveau des nœuds sont des informations représentées par une matrice à valeurs réelles $\boldsymbol{X_U} \in \mathbb{R}^{n \times p}$ et $\boldsymbol{X_V} \in \mathbb{R}^{m \times p'}$, où nous supposons que l'ordre des nœuds est cohérent avec l'ordre de la matrice d'adjacence. Pour les graphes bipartis, chaque type de nœud possède son propre ensemble d'attributs. La Figure 3.2 fournit un exemple de caractéristiques des nœuds associés aux domaines du graphe biparti de la Figure 3.1.

— **Chemin, distance et degré**
Un chemin entre deux nœuds u et v est une suite de nœuds reliés deux à deux par des arêtes (ou relation). La longueur d'un chemin est le nombre d'arêtes qu'il contient. Parmi tous les chemins qui relient deux nœuds, on peut distinguer les plus courts chemins qui sont ceux de longueur minimale. On définit alors la distance entre deux nœuds comme la longueur d'un plus court chemin les reliant. Enfin, le degré d'un nœud est le nombre d'arêtes qui y sont reliées.

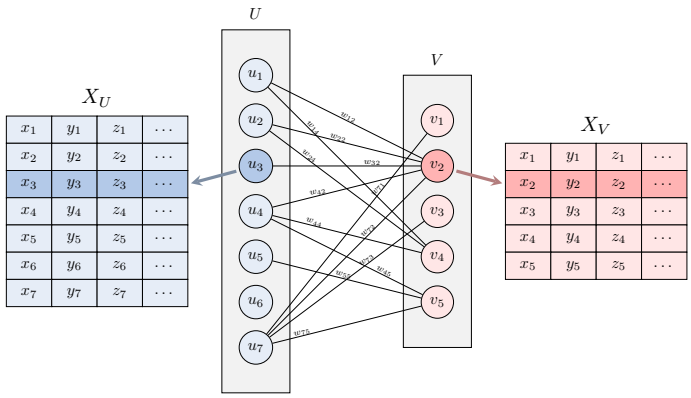

FIGURE 3.2 – Illustration des attributs des nœuds des deux domaines U et V.

— **Relation et distance**
Deux nœuds reliés directement par une arête sont de distance 1. et deux nœuds partageant un voisin commun sont de distance 2. Dans le cas particulier des graphes bipartis, on peut noter que les nœuds dans une relation de distance 2 sont de même type. La Figure 3.4 donne un exemple de ces distances. On parlera de distances supérieures pour désigner des nœuds de distance supérieure à 3. La proximité de distance supérieure représente la similarité entre les ensembles de voisins de deux nœuds. Ces différentes relations font partie des caractéristiques que les plongements de graphes visent à préserver.

— **Plongements de graphes**
L'objectif des plongements de graphes est d'associer à chaque nœud de G un vecteur correspondant dans un espace de faible dimension. Dans cet espace des plongements $\boldsymbol{H} \subset \mathbb{R}^d$, chaque nœud $u \in G$ est représenté par un vecteur de plongement $h \in \mathbb{R}^d$ de dimension $d \in \mathbb{R}$. En général, d est considérablement plus petit que le nombre de nœuds. La Figure 3.3 illustre le processus de plongement, où les nœuds du graphe biparti sont représentés dans un espace de plongement de faible dimension. L'objectif final de ce processus est de faire en sorte que les relations géométriques dans l'espace de plongement reflètent la structure du graphe original. Après un processus d'optimisation permettant une correspondance des plongements avec les nœuds, les plongements peuvent être utilisés comme entrées pour des tâches d'apprentissage automatique en aval, telles que la classification ou le regroupement.

3.3 Taxonomie

Dans cette section, nous présentons et regroupons les algorithmes existants de plongement de graphes bipartis en deux catégories sur la base de caractéristiques communes. Les catégories sont inspirées de Yang et al. [YXZ+20].

Les modèles de plongement de graphes bipartis visent également à préserver les caractéristiques du graphe original, en supposant qu'il contient un grand nombre de nœuds comme illustré par le nombre de nœuds des réseaux dans le Tableau 3.4. Il convient donc d'accorder une attention particulière aux questions de passage à l'échelle.

En raison de cette hypothèse, la plupart des méthodes d'apprentissage de représentation pour les graphes bipartis sont non-supervisées, ou auto-supervisées. En effet, la production d'étiquettes nécessaires à une approche supervisée sur un réseau comportant des millions de nœuds exigerait des efforts massifs d'annotation. Une autre limitation est que les méthodes supervisées actuelles ne peuvent pas intégrer la richesse des informations contenues dans les caractéristiques des nœuds d'un graphe, ainsi que les informations topologiques, dans une unique représentation des nœuds [DLL+22, GCHZ18, BOHG13]. En effet, ces méthodes reposent souvent sur un mécanisme de représentation des nœuds qui agrège l'information du voisinage local au dépend de l'information de la structure globale du graphe.

Notre taxonomie s'appuie sur deux observations. D'abord, un objectif essentiel des plongements de graphes est de conserver l'information topologique du graphe en préservant

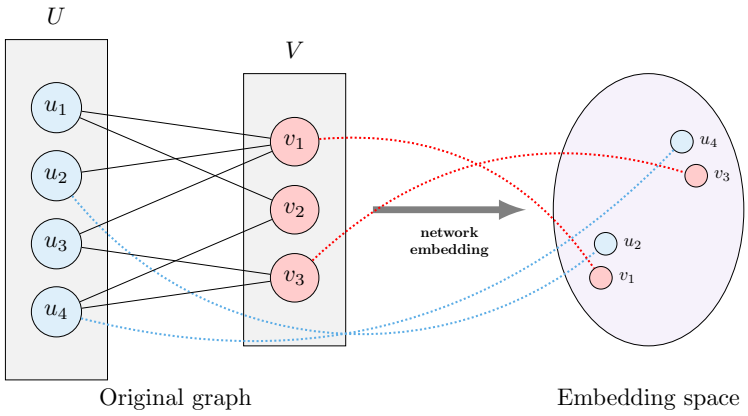

FIGURE 3.3 – Illustration du processus de plongement pour les réseaux bipartis. L'opération de plongement vise à cartographier les nœuds dans un espace de plongement de faible dimension. L'une des difficultés réside dans le fait que les distances dans l'espace de plongement (« Embedding space ») doivent refléter celles entre les nœuds dans le graphe (« Original graph »). Par exemple, les nœuds u_2 et v_1 sont connectés dans le graphe, leurs représentations doivent donc être proches dans l'espace de plongement. Il en va de même pour les nœuds v_3 et u_4.

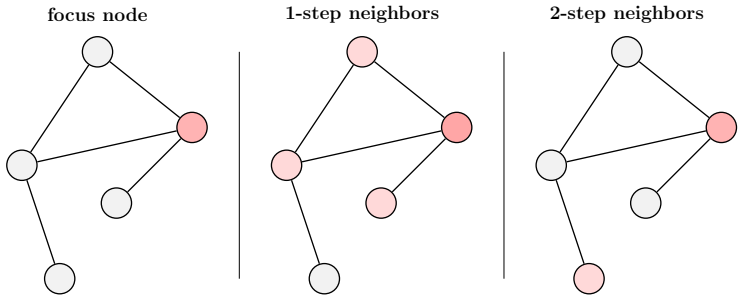

FIGURE 3.4 – Illustration d'une relation à plusieurs ordres dans un graphe homogène. L'image de gauche met en évidence le nœud auquel nous nous intéressons (« focus node »). L'image au centre montre ses voisins directs ou voisins de distance 1 (« 1-step neighbors »), tandis que l'image de droite montre ses voisins de distance 2 (« 2-step neighbors »).

différents types de proximité entre les nœuds. Ensuite, une part importante des méthodes de plongement de graphe a mené à un paradigme qui exploite l'information au niveau des nœuds (les attributs définis précédemment) de différentes manières. Sur la base de ces observations, nous proposons la taxonomie suivante pour les algorithmes existants de plongement de graphes bipartis :

1. **Approches de préservation de la proximité**
2. **Approches fondées sur le passage de messages**.

3.3.1 Approches de préservation de la proximité

Ce groupe de méthodes exploite les structures locales et globales d'un graphe afin de préserver les caractéristiques pertinentes dans le plongement. Pour comprendre l'essence de ces méthodes, il faut prendre en considération les notions de distance 1, de distance 2 et de distance supérieure que nous avons défini dans la section 3.2.2. Les Figures 3.5 et Figure 3.6 illustrent quelques problèmes liés à ces méthodes.

À la suite de solutions proposées pour obtenir des plongements de réseaux homogènes, quelques travaux ont tenté de transposer l'idée de la marche aléatoire, un modèle possédant une dynamique discrète composée d'une succession de pas aléatoires souvent utilisé pour les processus d'échantillonnage. Ces méthodes appliquent généralement une solution en deux étapes. Tout d'abord, des marches aléatoires sont effectuées à partir de plusieurs nœuds afin d'explorer le réseau, l'objectif étant d'établir un « corpus » de nœuds. Les marches aléatoires appliquées au réseau permettent de dresser une liste de séquences de nœuds, semblables aux mots d'une phrase dans le domaine du traitement automatique du langage naturel, qui constitue le corpus des nœuds visités. Dans un second temps, une méthode de plongement de mots (telle que word2vec [MCCD13]) est appliquée aux séquences de nœuds pour obtenir les plongements de nœuds. Ici, les matrices d'attributs des nœuds, définies précédemment, sont utilisées pour obtenir les plongements. Ces mé-

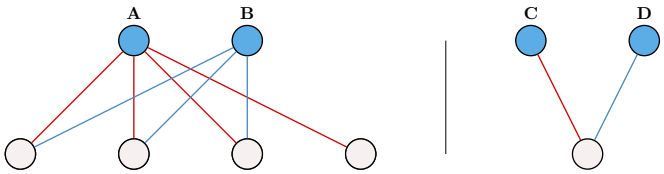

FIGURE 3.5 – Exemple de considération structurelle au sein des réseaux. La figure représente différents motifs de connexion entre les nœuds. Dans la partie gauche de la figure, les nœuds A et B ont trois voisins communs, tandis que dans la partie droite de la figure, les nœuds C et D ont un voisin commun. Il existe différentes méthodes pour évaluer laquelle des paires A-B ou C-D a la relation la plus forte. On pourrait affirmer que les nœuds A et B partagent plus de voisins que les nœuds C et D, et qu'ils devraient donc être considérés comme la connexion la plus forte. Mais d'un autre côté, on pourrait affirmer que les nœuds C et D ont exactement les mêmes voisins, et qu'ils devraient donc être considérés comme la connexion la plus forte.

thodes peuvent être sous-optimales pour le plongement de réseaux bipartis pour deux raisons :

1. La plupart du temps, les deux domaines du graphe biparti sont traités de la même manière et aucune distinction n'est faite entre les attributs des nœuds des deux domaines ;
2. Le corpus généré peut altérer les caractéristiques d'un réseau biparti, telles que la distribution des degrés des nœuds [GL04] qui correspond souvent à une loi de puissance [CF06] pour les graphes bipartis issus de scénario réel.

BiNE [GCHZ18] est une méthode basée sur la marche aléatoire qui fait partie d'un groupe de méthodes inspirées des travaux pionniers de DeepWalk [PARS14] et node2vec [GL16] qui adaptent l'idée de Skip-gram [MSC+13] pour modéliser les réseaux homogènes, et tentent d'étendre cette méthode aux réseaux bipartis.

BiNE effectue des marches aléatoires biaisées dans le but de préserver certaines propriétés du graphe biparti, notamment la distribution des degrés [GL04, NSW01]. Pour ce faire, BiNE prend en compte la modélisation des relations explicites et implicites. Les relations explicites visent à reconstruire le réseau biparti en ciblant les liens observés (liens directs entre les nœuds) tandis que les relations implicites capturent les corrélations de haut niveau (absence de lien ou nœuds à distance 2 ou plus). Les marches aléatoires effectuées par BiNE s'adaptent au nœud de départ, en fonction du degré de ce dernier. Plus précisément, le nombre de marches aléatoires et leur longueur sont définis en fonction de cette importance. De cette manière, la distribution des nœuds dans le corpus généré est plus cohérente avec le réseau biparti original.

BiNE se distingue des méthodes habituelles de plongement de réseaux par ses deux contributions. La première est l'incorporation des relations implicites qui ont permis d'améliorer la représentativité des plongements générés [YZP+18, JCY+16]. La seconde est la définition d'un cadre d'optimisation joint pour prendre en compte les relations

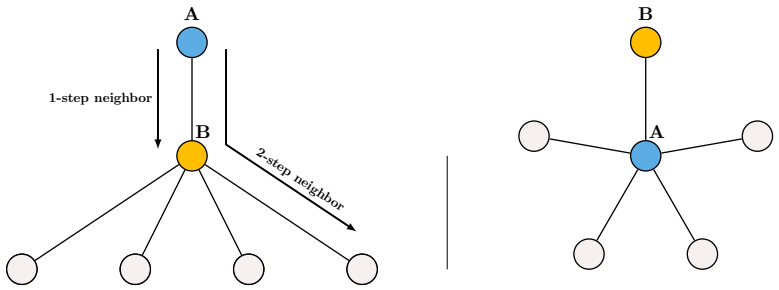

FIGURE 3.6 – La figure de gauche représente un graphe dans lequel le nœud A possède un voisin à distance 1 (« 1-step neighbor ») et quatre voisins à distance 2 (« 2-step neighbor » tandis que la figure de droite représente un graphe où le nœud A possède cinq voisins à distance 1. Si les voisins à distance 1 et 2 sont traités de la même manière, donc que la distance n'est pas prise en compte, alors ces deux graphes structurellement différents sont perçus de la même manière par le nœud A. Ceci est contre-intuitif et montre comment le traitement des voisins à distance k peut avoir un impact sur les considérations structurelles du graphe.

explicites et implicites. Une fonction objectif dédiée est définie pour chaque type de relation et les plongements de nœuds de chaque fonction objectif sont regroupés pour se renforcer mutuellement.

BiANE [HLF+20] vise à améliorer un aspect souvent négligé qui est la prise en compte de la relation entre les informations d'attributs et les informations structurelles des nœuds. En effet, ces deux informations représentent des aspects différents mais complémentaires des nœuds, cependant la plupart des travaux sur les plongements de réseaux ignorent leurs différentes propriétés. Pour ce faire, la principale contribution de BiANE est l'introduction d'une méthode permettant de prendre en compte la structure de proximité de distance supérieure sans allonger le temps de calcul. BiANE définit la proximité inter-domaine comme les liens qui existent entre les deux domaines du graphe biparti et la proximité intra-domaine comme la proximité sous-jacente au sein de chaque domaine (par exemple, la proximité entre les nœuds du même type).

Plus précisément, BiANE extrait et isole les proximités intra-domaine et inter-domaine et utilise ensuite un auto-encodeur [BKG21] pour obtenir un plongement dans un espace latent. Plus précisément, étant donné un réseau attribué biparti, BiANE partitionne d'abord le réseau en fonction du type ou du domaine des nœuds. Il extrait ensuite les informations structurelles et les informations d'attributs de chaque partition et les compresse à l'aide de différents auto-encodeurs. Par la suite, il intègre la proximité de distance 2 dans chaque partition. À ce stade, BiANE effectue un échantillonnage dynamique positif (« dynamic positive sampling ») [MY16] dans l'espace latent et propose un nouvel apprentissage afin d'améliorer la corrélation entre les informations d'attributs et les informations structurelles. Cette approche est intéressante car modéliser la proximité structurelle de distance supérieure peut être complexe. En effet, les travaux

existants [GCHZ18, GHC+19] ajoutent un grand nombre de liens supplémentaires pour le faire, mais rendent ainsi le réseau beaucoup plus dense ce qui introduit des problèmes de passage à l'échelle. Enfin, après avoir effectué un échantillonnage dynamique positif, BiANE agrège les vecteurs d'encodage des auto-encodeurs susmentionnés et modélise la proximité inter-domaine du réseau biparti. BiANE offre un moyen pratique de préserver les proximités intra-domaine et inter-domaine tout en introduisant une stratégie d'échantillonnage dynamique positive efficace qui améliore le processus d'entraînement. Néanmoins, d'autres expériences devraient être menées pour valider cette stratégie, celle-ci étant différente de la méthode d'échantillonnage traditionnelle.

FOBE et HOBE [SS19] visent à préserver les caractéristiques des voisinages proches et éloignés des graphes bipartis. Ce travail introduit deux modèles de plongement qui apprennent des représentations latentes denses de graphes bipartis tout en préservant les informations sémantiques spécifiques au type des nœuds. Les deux méthodes partagent la même approche. Elles observent d'abord les relations structurelles en échantillonnant certains types de relations entre les nœuds. Ensuite, elles apprennent une représentation qui minimise la différence entre les similarités observées entre les nœuds et les similarités estimées correspondantes. Chacune de ces méthodes diffère par la manière dont elle mène ses observations, ses estimations et ses objectifs. La principale contribution de FOBE et HOBE est qu'elles génèrent des estimations de similitudes dans chaque domaine de nœuds séparément et permettent ainsi de mieux préserver les caractéristiques latentes spécifiques au domaine des nœuds.

Techniquement, FOBE échantillonne d'abord les liens directs et les relations de distance 2 entre les nœuds, puis apprend les plongements en minimisant la divergence de Kullback-Leibler [Shl14] entre les observations et les estimations de plongements. D'autre part, HOBE calcule d'abord les similitudes algébriques pour les paires de nœuds de toutes les arêtes. La distance algébrique [CS11] s'avère être un bon moyen de capturer les similarités implicites dans les graphes. Le calcul de la distance algébrique permet à HOBE de prendre en compte les relations de distance 3 en plus des relations de distance 1 et de distance 2. Ensuite, HOBE ajuste les plongements avec l'erreur quadratique moyenne (ou MSE pour Mean Squared Error).

FOBE capture les relations locales, tandis que HOBE capture les relations de distance supérieure. Par conséquent, afin d'unifier ces approches, une méthode de combinaison est proposée pour réunir les différents plongements en un seul, pour capturer ces deux aspects simultanément. Deux versions de la méthode de combinaison sont proposées. La première version favorise la maximisation des performances sur la tâche d'apprentissage (la version directe) tandis que la seconde exploite l'encodage complet des plongements d'entrée (la version auto-régularisée).

3.3.2 Approches basées sur le passage de messages

Les méthodes basées sur le passage de messages (« Message-Passing ») reposent sur une intuition simple : pour chaque nœud, un plongement est construit de manière itérative sur la base des attributs du nœud et des informations agrégées de son voisinage local.

Précisément, deux fonctions différentiables, une fonction de *mise à jour* et une fonction d'*agrégation*, sont définies et sont utilisées à chaque itération du processus d'obtention de plongements. Pour chaque nœud, à chaque itération, la fonction d'agrégation prend en entrée l'ensemble des plongements du voisinage du nœud et produit un message regroupant l'information sur ce voisinage. Ensuite, la fonction de mise à jour fusionne ce message avec les plongements précédents du nœud pour produire le plongement mis à jour. Ce processus est ainsi répété un nombre fini de fois. La Figure 3.7 illustre ce fonctionnement. Le mécanisme de passage de messages est à la base du cadre des réseaux neuronaux de graphe (ou GNN pour « Graph Neural Network ») qui adapte les réseaux neuronaux profonds aux données de type graphe [Ham] et inspire de nombreux travaux sur la représentation de graphe [WPC+21].

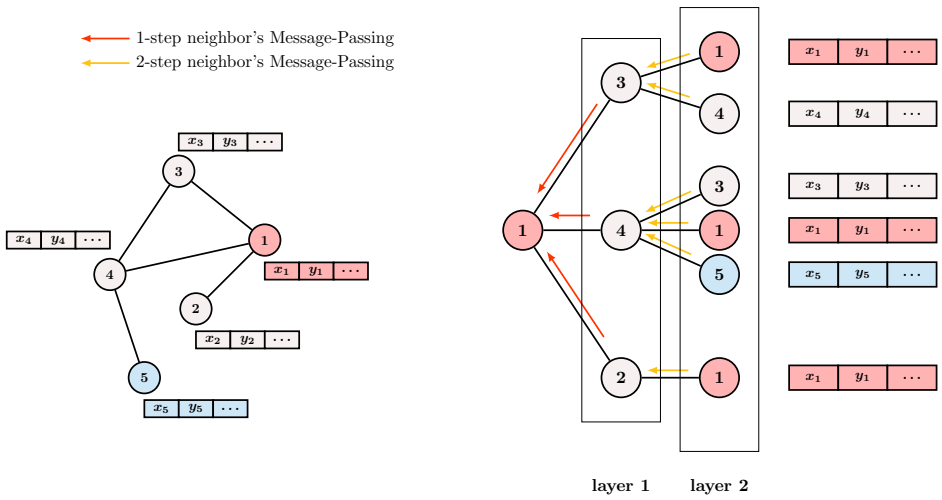

FIGURE 3.7 – Illustration du processus de passage de messages (« Message-Passing »). L'exemple montre un processus de passage de messages en deux étapes pour le nœud 1. En partant du nœud 1, la première couche (« layer 1 ») contient tous les voisins de distance 1 (« 1-step neighbor's ») du nœud 1, à savoir les nœuds 3, 4 et 2. La même opération est ensuite répétée pour chaque nœud de la première couche. Par exemple, tous les voisins de distance 1 du nœud 3 (dans la première couche) sont placés dans la seconde couche, ce qui permet de voir les nœuds 1 et 4 dans la partie supérieure de la seconde couche (« layer 2 »). Pour obtenir les autres nœuds de la seconde couche, le même principe est répété pour les autres nœuds de la première couche. Les flèches convergeant vers un nœud représentent sa mise à jour et la transmission des informations de ses voisins. Cela se fait en deux opérations distinctes. Tout d'abord, tous les attributs des voisins sont agrégés par une fonction d'agrégation. Ensuite, l'état du nœud est mis à jour par une fonction de mise à jour qui tient compte à la fois de l'agrégation des attributs et de l'état précédent du nœud.

BGCN [HXR+19] étend aux réseaux bipartis la notion de réseaux convolutionnels ap-

pliqués aux graphes (GCN pour « Graph Convolutional Networks ») [KW16] un réseau neuronal multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères. Bien que les GCN soient généralement limités aux réseaux homogènes, BGCN les étend aux réseaux bipartis en adaptant les équations d'apprentissage de représentation des réseaux à la spécificité des notations matricielles des réseaux bipartis. Par la suite, il applique une fonction de plongement composée de deux éléments : une agrégation des attributs des voisins de distance 1 et une transformation non linéaire qui prend en entrée l'agrégation susmentionnée. Ensuite, après avoir défini le concept de GCN pour les réseaux bipartis, l'article examine deux types d'architectures de décodeur. Le premier décodeur est un modèle standard de perceptron multicouche (MLP pour « Multi Layer Perceptron ») qui aligne la sortie de la fonction de plongement sur les attributs de chaque nœud par le biais de deux couches de réseau neuronal entièrement connectées. Le second décodeur est basé sur une approche antagoniste : il entraîne un modèle discriminateur à séparer les éléments de la matrice d'attributs originale et les vecteurs de plongement. Le discriminateur vise à maximiser la capacité à identifier la représentation des attributs, et BGCN vise à empêcher le discriminateur de le faire en générant des représentations encodées qui sont aussi similaires que possible.

Enfin, afin de dépasser la limite de l'information du voisinage de distance 1 des modèles précédents, une architecture en cascade basée sur le décodeur BGCN est introduite pour capturer l'information structurelle à plusieurs pas.

Bi-HGNN [LJS+19] propose un système de recommandation pour une plateforme de commerce en ligne qui utilise à la fois les préférences globales au niveau de la communauté des utilisateurs et leurs préférences individuelles. Le modèle fait partie des modèles hiérarchiques de passage de messages, mais se distingue en prenant en compte les informations spécifiques à l'utilisateur, souvent ignorées dans les modèles standards, qui ne sont pas capturées par la généralisation au niveau de la communauté.

Pour ce faire, elle s'inspire des travaux sur les réseaux neuronaux hiérarchiques, couramment utilisés pour la classification des graphes, et applique ces concepts à la prédiction de liens (définie dans la suite).

Bi-HGNN utilise dans un premier temps un modèle GNN standard [HYL18b] pour modéliser les utilisateurs et les articles dans un graphe biparti et encode les caractéristiques brutes des utilisateurs et des articles en utilisant Wide and Deep [CKH+16]. Bi-HGNN s'appuie sur des communautés prédéfinies de nœuds utilisateurs, et génère des plongements de faible dimension pour ces communautés. Ensuite, il utilise ces plongements pour effectuer un regroupement sur chaque nœud utilisateur. Plus précisément, le modèle calcule la distance entre les nœuds utilisateurs et chaque communauté à l'aide des plongements mentionnés ci-dessus. Cette métrique de distance permet d'assigner « de manière plus lisse » chaque nœud utilisateur à quelques communautés. Enfin, les informations sur les nœuds utilisateurs sont décomposées en deux espaces orthogonaux représentant les informations capturées par la généralisation au niveau de la communauté et les préférences individuelles des utilisateurs.

Cascade-BGNN [HXR+20] est une méthode d'apprentissage de représentation auto-

supervisé pour les graphes bipartis de grande taille. L'architecture du modèle de plongement comporte trois éléments clés. Le premier est le passage de messages inter-domaines (IDMP pour « inter-domain message passing »). Son objectif est de permettre à un domaine d'agréger des informations provenant de l'autre domaine par l'intermédiaire des arêtes connectées. L'IDMP n'effectue l'agrégation que sur les nœuds voisins de chaque nœud, sans impliquer le nœud lui-même.

Une fois que les caractéristiques agrégées du domaine opposé sont jointes, l'alignement intra-domaine (IDA pour « intra-domain alignment ») est utilisé pour fusionner ces deux caractéristiques distinctes en une seule représentation. Comme l'IDMP n'inclut pas les informations du nœud lui-même, l'IDA permet de prendre en compte les caractéristiques du nœud. La représentation des deux domaines est obtenue après l'entraînement d'une couche par minimisation d'une fonction de perte antagoniste (« adversarial loss ») pour les deux domaines de manière auto-supervisée.

Les plongements ainsi obtenus ne capturent que la structure topologique de distance 1 entre les domaines ainsi que les informations sur les caractéristiques des deux domaines. Toutefois, l'agrégation à un seul pas ne caractérise pas suffisamment les diverses structures du graphe, d'où la nécessité d'un mécanisme à plusieurs pas. Plutôt que de s'appuyer sur une méthode habituelle d'apprentissage de bout en bout, cet article développe une méthode d'apprentissage en cascade pour conduire le passage de messages de proche en proche. En détail, l'entraînement en cascade est la concaténation de plusieurs blocs BGNN de base, chacun étant composé d'une séquence d'IDMP et d'IDA. Un bloc BGNN de base constitue un entraînement à une couche et complète son entraînement avec un plongement de distance 1. Ensuite, les plongements obtenus sont utilisés comme entrée pour un entraînement ultérieur (par exemple, comme entrée pour le bloc BGNN suivant).

HGCN [He19] propose une extension du modèle convolutionnel pour apprendre les représentations de graphe des réseaux bipartis. Les modèles standards basés sur la convolution sont limités aux graphes homogènes et HGCN est une tentative de surmonter cette limite. En outre, de nombreux algorithmes utilisant des marches aléatoires peuvent altérer la distribution des degrés, qui correspond souvent à une loi de puissance pour les graphes bipartis issus de scénarios réels (voir la section 3.3.1), ce qui entraîne une perte d'information dans les nœuds de degré élevé. HGCN a été conçu pour répondre à plusieurs défis liés aux graphes. Tout d'abord, il est adapté aux graphes à grande échelle et aux caractéristiques hétérogènes des arêtes et des nœuds. De plus, il convient à l'apprentissage non supervisé et prend en compte le type des nœuds dans les graphes bipartis.

Cette approche propose une architecture HGCN en cascade en trois étapes afin de capturer les relations explicites et implicites entre les deux domaines du graphe biparti. Les deux premières étapes sont symétriques et incluent, respectivement, les informations sur les caractéristiques d'un domaine et les informations structurelles du domaine opposé. La troisième étape combine leurs résultats. Plus précisément, la première étape consiste à capturer la représentation explicite en apprenant les représentations des nœuds du premier domaine tout en incluant les caractéristiques des voisins du second domaine. La deuxième étape capture la relation implicite en apprenant les représentations des

nœuds du deuxième domaine tout en incluant les caractéristiques des voisins du premier domaine. Enfin, la troisième étape fusionne les relations implicites et explicites. Deux modèles de sortie sont ensuite proposés : un modèle décodeur et un modèle d'apprentissage antagoniste (« adversarial learning »). Un des inconvénients d'HGCN est le manque de cohérence de la méthode utilisée pour fusionner les informations sur les attributs des nœuds avec les informations structurelles. En effet, lors de leur fusion, la même importance est donnée à ces deux types d'informations ce qui n'est pas adapté à certaines configurations de graphes bipartis.

IGE [ZXKZ17] généralise les techniques de plongement au cas complexe des graphes bipartis dynamiques avec des arêtes attribuées, ou graphes d'interactions attribuées comme dénommé dans l'article. Dans les graphes d'interactions attribuées, une arête représente une interaction entre deux nœuds de domaines différents et les attributs de l'arête représentent le contenu de l'interaction. Pour une meilleure compréhension de ce concept nous pouvons prendre comme exemple le marché boursier et les transactions des investisseurs : chaque transaction implique un investisseur qui achète ou vend une action à un certain prix, dans une certaine quantité et à un certain moment. Ici, les nœuds du graphe biparti sont les investisseurs et les actions, les arêtes entre chaque nœud sont les transactions, et les attributs des arêtes sont les caractéristiques des transactions.

IGE contourne les difficultés liées à la dynamique et aux attributs hétérogènes des arêtes en étudiant la dépendance temporelle des arêtes plutôt que la structure des graphes. Pour ce faire, il introduit une méthode de plongement profond des nœuds. IGE consiste en une architecture à trois réseaux neuronaux contenant deux réseaux neuronaux couplés pour la prédiction et un réseau d'encodage des attributs. Le réseau d'encodage transforme les attributs en vecteurs homogènes de longueur fixe pour gérer l'hétérogénéité des attributs. Ensuite, ces attributs encodés sont fournis aux réseaux de prédiction couplés qui prennent en compte la dépendance temporelle lors de l'apprentissage des représentations de nœuds pour des graphes d'interaction.

3.3.3 Discussion

Les méthodes citées ont des limites que nous résumons et discutons dans cette section.

FOBE et HOBE apportent une approche intéressante, mais peu d'importance est donnée aux attributs des nœuds dans les plongements. De plus, la capacité de passage à l'échelle de ces algorithmes n'est pas abordée.

Malgré ses avantages et ses bons résultats dans les tâches de prédiction de liens et de recommandation, BiNE souffre de certaines limites. En effet, cette approche s'appuie exclusivement sur les arêtes observées et peut donc rencontrer des difficultés pour reconstituer les nœuds qui ont très peu ou pas de connexions. Ce problème peut être résolu en incluant dans le modèle des informations auxiliaires, telles que des attributs numériques ou des descriptions textuelles. Enfin, un axe majeur d'amélioration pour BiNE est la prise en compte de l'aspect dynamique des réseaux.

BGCN et Cascade-BGNN sont des méthodes pertinentes mais les avantages de l'architecture en cascade par rapport aux méthodes traditionnelles ne sont pas explicites, à l'exception de la rapidité d'exécution.

Bi-HGNN propose un moyen intéressant de surmonter une limitation bien connue des systèmes de recommandation classiques, mais le point de vue adopté est spécifique aux plateformes de commerce en ligne. Ainsi, l'innovation apportée n'est pas évidente et les liens avec les problèmes généraux liés aux plongements sont parfois dissimulés. De plus, le potentiel de passage à l'échelle de Bi-HGNN manque de détails.

De manière similaire aux autres méthodes détaillées dans ce travail, IGE se concentre sur les plongements de nœuds sans incorporer les plongements d'arêtes. Cependant, dans le cas d'IGE, qui considère des graphes d'interaction attribués, ce choix est discutable car les arêtes de ces graphes contiennent plus d'informations que les graphes bipartis ordinaires. Ceci constitue un axe d'amélioration du modèle IGE. En outre, IGE n'exploite pas l'information structurelle, ce qui peut constituer un autre axe d'amélioration. Enfin, un inconvénient général du modèle IGE est qu'il n'intègre pas les nœuds de différents types dans le même espace latent. Une amélioration majeure consisterait donc à répondre à cet inconvénient en plaçant les nœuds de types différents dans le même espace.

Les limitations mentionnées ci-dessus montrent que les plongements de graphes bipartis souffrent de problèmes liés au couplage d'informations hétérogènes et un manque de capacité de passage à l'échelle, ce qui devrait motiver des travaux futurs dans ce domaine. En outre, ces limitations ne permettent pas de trancher entre les catégories proposées dans la taxonomie, mais les travaux récents montrent une tendance de la communauté scientifique en faveur des approches de passage de messages.

3.4 Outils d'évaluation et d'expérimentation

Dans cette section, nous énumérons un ensemble d'outils couramment utilisés pour explorer le domaine de l'apprentissage de représentations pour les réseaux bipartis et fournissons une liste de ressources en ligne pour mener des travaux et des expériences dans ce domaine. Cette liste est composée de bibliothèques et de jeux de données traditionnellement utilisés dans les articles portant sur ce sujet. Le Tableau 3.1 présente les tâches de recherche qui ont été précédemment identifiées dans la littérature.

3.4.1 Tâches d'évaluation

Dans la grande majorité des applications courantes des graphes, l'apprentissage de représentation est utilisé pour quatre tâches : la classification de nœuds, la classification de graphes, la prédiction de relations et la reconstruction de graphes. Ces tâches permettent de couvrir un large éventail d'applications réelles. Pour une explication exhaustive des tâches d'apprentissage automatique sur les graphes et de leur interaction avec les catégories d'apprentissage automatique standard, le lecteur intéressé peut se référer au chapitre 1.2 de [Ham].

3.4.1.1 Classification de nœuds

La classification de nœuds est une tâche d'apprentissage automatique qui a gagné en popularité ces dernières années. Elle consiste à construire un modèle capable de classer les nœuds dans des catégories pré-identifiées. Étant donné un petit nombre d'exemples étiquetés manuellement dans l'ensemble du réseau, l'objectif est de prédire l'étiquette associée à tous les nœuds du réseau. La classification des nœuds semble très similaire à la classification supervisée traditionnelle, mais une distinction majeure demeure : les nœuds des réseaux ne sont pas indépendants et identiquement distribués. La raison en est que l'objectif de l'analyse des réseaux est de modéliser un ensemble de nœuds interconnectés et, par conséquent, dépendants. Les exemples d'application de la classification de nœuds sont tous les types de détection de communautés. A titre d'illustration, la classification de nœuds peut être utilisée dans un réseau social pour classer les utilisateurs similaires dans la même communauté ou dans une base de données documentaire pour classer les documents partageant un contenu apparenté.

3.4.1.2 Classification de graphes

La classification de graphes est une tâche courante qui vise à caractériser des graphes dans leur entièreté. L'idée est d'apprendre à partir de données représentées sous forme de graphe. Au lieu d'effectuer une prédiction sur les composantes individuelles (nœuds et arêtes) d'un graphe unique, un modèle de classification de graphes prend en entrée un jeu de données composé de plusieurs graphes et effectue des prédictions indépendantes pour chaque graphe. Ici, chaque graphe est un point de données i.i.d. (indépendant et identiquement distribué) associé à une étiquette et l'objectif est d'utiliser un jeu d'apprentissage de données étiquetées pour apprendre une correspondance et prédire une étiquette pour chaque graphe. La modélisation des propriétés moléculaires est un exemple d'application de la classification de graphes : dans l'identification des enzymes, une collection de graphes, représentant chacun un composé chimique, est utilisée pour entraîner un modèle permettant de prédire si un graphe est une enzyme ou non. Il en va de même pour la découverte de médicaments, où des graphes moléculaires sont considérés pour identifier des propriétés pharmacologiques et ADME/T (absorption, distribution, métabolisme, excrétion et toxicité) souhaitées [BYT19]. Cependant, les algorithmes décrits dans ce travail ne traitent pas la tâche de classification des graphes qui constitue un axe majeur d'exploration pour la recherche future sur le sujet du plongement de graphes bipartis.

3.4.1.3 Prédiction de liens

La prédiction de liens, également connue sous le nom de prédiction de relations ou de complétion de graphes, est l'une des tâches d'apprentissage automatique les plus populaires sur les graphes. Dans la formulation classique de la prédiction de relations, étant donné un ensemble de nœuds et un ensemble incomplet d'arêtes entre ces nœuds, l'objectif de la prédiction de liens est d'utiliser l'information partiellement disponible pour

Méthode	Catégorie	Classification de nœuds	Prediction de liens
BiNE	Marche aléatoire		✓
BGCN	GNN	✓	
Bi-HGNN	GNN		✓
BiANE	Marche aléatoire	✓	✓
FOBE/HOBE	Préservation de proximité		✓
Cascade-BGNN	GNN	✓	
IGE	GNN	✓	
HGCN	GNN	✓	

TABLE 3.1 – Tâches d'évaluation utilisées par les modèles de plongement de graphes bipartis

déduire les arêtes manquantes du graphe. La complexité de la prédiction de liens est hautement corrélée au type de données de graphe étudiées. Par exemple, dans les graphes simples (comme celui représentant les connexions entre les individus dans un réseau social), il existe des heuristiques simples basées sur le nombre de voisins que partagent deux nœuds, alors que dans les graphes multi-relationnels plus complexes (comme les graphes de connaissances biomédicales où des centaines d'interactions biologiques différentes sont considérées), des stratégies de raisonnement et d'inférence complexes sont exploitées. Les systèmes de recommandation sont un exemple d'application de la prédiction de liens [GZL+21]. Cette dernière est utilisée pour la recommandation de contenu dans les réseaux sociaux et la recommandation de produits dans les plateformes de commerce en ligne.

3.4.1.4 Reconstruction de graphes

Comme les plongements de réseaux peuvent être vus comme une forme de compression de l'information contenue dans ces réseaux, une compression sans perte devrait pouvoir permettre de reconstruire le réseau original à partir des plongements. La tâche de reconstruction de graphes utilise les plongements pour prédire les arêtes du réseau original. Cette tâche est étroitement liée à la tâche de prédiction de liens, mais la reconstruction de graphe utilise les arêtes existantes du réseau comme vérité terrain alors que la prédiction de liens vise à prédire la probabilité de formation de liens plausibles. Par exemple, la reconstruction de graphes est utilisée dans l'étude des changements climatiques, pour révéler les modèles de téléconnexions atmosphériques et comprendre leurs mécanismes sous-jacents, ou pour identifier les propriétés fonctionnelles des gènes, lorsque les réseaux de régulation de gènes sont prédits à partir de données d'expression [GdBLC03]. La tâche de reconstruction de graphes n'est pas très répandue dans la littérature et n'est pas représentée dans le Tableau 3.1.

Jeu de données	Type de réseau	Tâche d'évaluation	BiNE	BGCN	Bi-HGNN	BiANE	FOBE/HOBE	Cascade-BGNN	IGE	HGCN
Tencent [Ten]	Plateforme web	A, B	✓	✓				✓		✓
Cora [MNRS04]	Réseau de citations	A		✓				✓		
Citeseer [GBL98]	Réseau de citations	A		✓				✓		
MovieLens [HK15a]	Notation de film	A, B	✓		✓	✓				
PubMed [SNB+08]	Réseau de citations	A		✓				✓		
DBLP [SNB+08]	Réseau de citations	A	✓				✓		✓	
LastFM [BMEWL11]	Registre d'écoute	B					✓			
Amazon [LAH07]	Co-achat de produits	B					✓			
YouTube [YL12]	Appartenance à un groupe	B	✓				✓			
Friendster [YL12]	Appartenance à un groupe	B					✓			
Livejournal graphs [BHKL06, LLDM08]	Appartenance à un groupe	B					✓			
AMiner [TZY+08]	Réseau de publication	A, B				✓				
Frappe [BCKO15]	Clics d'application	B								
CiteULike [WCL13]	Références	B								
Netflix [BL07]	Notation de film	B								
MovieLens-Latest [HK15b]	Notation de film	A, B								
Last.fm-360K [Cel10]	Musique d'artiste	B								
Amazon-Book [HM16]	Notation de livre	B								
Epinions-Extend [TGLDS12]	Notation de produit	B								
Echonest [MBMEL12]	Lecture de musique	B								
PPD [WZL+17]	Registre d'investissement	A							✓	
Action [XC18]	Transaction de bourse	A							✓	
Yelp [Asg16]	Commentaire en ligne	A							✓	
Wikipedia [WL12, WPP09]	N/A	B	✓							

TABLE 3.2 – Informations sur les jeux de données, y compris leur référence, le type de réseau et les tâches d'évaluation. La colonne Tâche d'évaluation suit la légende : A-Classification de nœuds, B-Prédiction de liens. Le tableau indique également quel modèle de plongement de graphes bipartis a utilisé quel jeu de données. Les jeux de données non cochés ont été utilisés dans [CWTY19].

Methode	Code source
BiNE	https://github.com/clhchtcjj/BiNE
BGCN	https://tinyurl.com/ABCGraph
Bi-HGNN	Code non disponible
BiANE	Code non disponible
FOBE/HOBE	http://bit.ly/fobe_hobe_code
Cascade-BGNN	https://github.com/chaoyanghe/bipartite-graph-learning
IGE	Code non disponible
HGCN	https://github.com/chaoyanghe/bipartite-graph-learning

TABLE 3.3 – Code source des modèles de plongement de graphes bipartis.

3.4.2 Jeux de données de référence

Cette section présente plusieurs tableaux fournissant des informations sur les jeux de données utilisés dans les articles étudiés.

Le Tableau 3.1 décrit les tâches d'évaluation utilisées dans les méthodes de la taxonomie. Il fournit des informations sur la popularité des tâches d'évaluation dans la littérature récente. Le Tableau 3.2 regroupe des informations de haut niveau sur les jeux de données de graphes bipartis. Plus précisément, il indique le type de graphe, ainsi que les tâches d'évaluation et les méthodes pour lesquelles les graphes ont été utilisés. Le Tableau 3.4 résume les statistiques ainsi que les éléments descriptifs concernant la nature des ensembles de données de graphes bipartis. Il s'agit du tableau le plus complet de notre étude relatif aux jeux de données et il doit être considéré comme le point de référence pour avoir une vue d'ensemble des jeux de données.

3.4.3 Bibliothèques en source ouverte

Nous présentons ici plusieurs bibliothèques en source ouverte utilisées pour les plongements de réseaux en général et qui peuvent être appliquées aux plongements de réseaux bipartis. En outre, le Tableau 3.3 fournit des liens vers le code source des modèles de plongement de graphes bipartis décrits dans la section 3.3.

Pytorch Geometric

Pytorch Geometric [MJE] consiste en diverses méthodes d'apprentissage profond sur des graphes et autres structures irrégulières, également connues sous le nom d'apprentissage profond géométrique, tirées d'une variété d'articles publiés. En outre, il comprend un pipeline de traitement de données facile à utiliser pour gérer de nombreux petits graphes ou des graphes de grande taille, un support multi-gpu, un grand nombre de jeux de données de référence (basés sur des interfaces simples pour créer son propre jeu de données), et des transformations utiles, à la fois pour l'apprentissage sur des graphes arbitraires, sur des maillages 3D ou sur des nuages de points.

DGL - Deep Graph Library

DGL [WZY+19] vise à aider les chercheurs en apprentissage profond sur les graphes en facilitant l'implémentation de modèles de la famille des GNNs. Il vise également à rendre aussi facile que possible la combinaison de modules basés sur des graphes avec des modules basés sur des tenseurs (par exemple PyTorch ou MXNet).

Spektral

Spektral [GA20] est une bibliothèque Python pour l'apprentissage profond sur les graphes, basée sur l'API Keras et TensorFlow 2. L'objectif principal de ce projet est de fournir un cadre de développement simple mais flexible pour créer des réseaux neuronaux de graphe. Spektral peut être utilisé pour classer les utilisateurs d'un réseau social, prédire les propriétés moléculaires, générer de nouveaux graphes avec des réseaux antagonistes génératifs (GAN pour « Generative Adversarial Networks »), regrouper des nœuds, prédire des liens, et toute autre tâche dont les données sont décrites par des graphes.

GraphNets

Graph Nets [Gra] est la bibliothèque de DeepMind [1] pour construire des GNNs dans TensorFlow et Sonnet. La bibliothèque est livrée avec des démonstrations qui montrent comment créer, manipuler et entraîner des réseaux neuronaux de graphes pour raisonner sur des données structurées en graphe, sur une tâche de recherche du plus court chemin, une tâche de tri et une tâche de prédiction physique.

Jraph – Une bibliothèque pour les GNNs dans jax

Jraph [GKB+20] (prononcé « girafe ») est une bibliothèque légère pour travailler avec des réseaux de neurones de graphes dans jax [2]. Elle fournit une structure de données pour les graphes, un kit d'utilitaires pour travailler avec les graphes, et une collection de modèles de réseaux neuronaux de graphes disponibles sur GitHub.

3.5 Conclusion et Directions Futures

Dans ce travail, nous avons réalisé une étude complète de la littérature sur les techniques d'apprentissage de représentation pour les graphes bipartis. Nous avons analysé les approches récentes de plongement de graphes bipartis qui transforment un graphe biparti en une représentation vectorielle de faible dimension tout en préservant autant que possible les propriétés intrinsèques du graphe d'origine. Nous avons proposé une taxonomie des algorithmes de plongement de graphes bipartis qui regroupe ces derniers selon des principes communs. Nous avons décrit les tâches d'évaluation les plus courantes pour

1. https ://www.deepmind.com/
2. https ://github.com/google/jax

Jeu de données	Type (nœud 1)	Type (nœud 2)	Type (relation)	#Classe (nœud 1)	#Classe (nœud 2)	#relation/#nœud	#nœud 1	#nœud 2	#relation
Tencent [Ten]	Utilisateur	Communauté	1 (comportement)	2	N/A	1.399	619,030	90,044	991,734
Cora [MNRS04]	Article	Article	1 (citation)	7	6	1.119	734	877	1,802
Citeseer [GBL98]	Article	Article	1 (citation)	6	6	0.890	613	510	1,000
MovieLens [HK15a]	Utilisateur	Film	5 (notation)	1	1	111.607	162,000	62,000	25,000,000
PubMed [SNB+08]	Article	Article	1 (citation)	3	3	1.114	13,424	3,435	18,782
DBLP [SNB+08]	Auteur	Conference	1 (publication)	5	1	2.908	35,851	20	104,32é
LastFM [BMEWL11]	Artiste	Utilisateur	1 (registre)	1	1	4.755	17,632	1,892	92,834
Amazon [LAH07]	Produit	Produit	1 (co-achat)	1	1	4.711	262,111	N/A	1,234,877
YouTube [YL12]	Utilisateur	Groupe	1 (adhésion)	1	1	2.613	1,134,890	8,385	2,987,624
Friendster [YL12]	Utilisateur	Groupe	1 (adhésion)	1	1	27.132	65,608,366	957,154	1,806,067,135
Livejournal graphs [BHKL06, LLDM08]	Utilisateur	Groupe	1 (adhésion)	1	1	8.093	3,997,962	287,512	34,681,189
AMiner [TZY+08]	Article	Auteur	1 (auteur)	1	1	1.150	80,461	66,107	168,525
Frappe [BCKO15]	Utilisateur	Item	1 (comportement)	1	1	19.092	957	4,082	96,203
CiteULike [WCL13]	Utilisateur	Item	2 (notation)	1	1	9.343	5,551	16,980	210,504
Netflix [BL07]	Utilisateur	Film	5 (notation)	1	1	200.820	480,189	17,770	100,000,000
MovieLens-Latest [HK15b]	Utilisateur	Film	5 (notation)	1	1	79.881	280,000	58,000	27,000,000
Last.fm-360K [Cel10]	Utilisateur	Artiste	many (nombre d'écoute)	1	2	26.876	359,347	294,015	17,559,530
Amazon-Book [HM16]	Utilisateur	Item	1 (commentaire)	2	2	1.970	133,960	431,827	1,114,563
Epinions-Extend [TGLDS12]	Utilisateur	Item	1 (notation)	1	27	2.896	22,166	296,277	922,267
Echonest [MBMEL12]	Utilisateur	Musique	1 (nombre d'écoute)	1	1	34.457	1,019,318	384,546	48,373,586
PPD [WZL+17]	Investisseur	Emprunt	1 (investissement)	1	6	16.181	9,292	4,501	223,190
Action [XC18]	Investisseur	Stock	2 (achat/vente)	3	1	2.916	22,001	939	66,890
Yelp [Asg16]	Utilisateur	Commerce	1 (commentaire)	1	10	3.329	724,884	15,726	2,465,173
Wikipedia [WL12, WPP09]	Article	Navigation	1 (clic d'hyperlien)	1	2	1.488	4,604	76,193	119,882

TABLE 3.4 – Ce tableau résume les statistiques ainsi que les éléments descriptifs sur la nature des jeux de données de graphes bipartis. Il fournit des informations utiles sur la taille des graphes, leur densité et leurs applications.

comparer les différentes méthodes de plongement de graphes bipartis et mesurer la performance des algorithmes. Nous avons également rassemblé des listes de jeux de données et de bibliothèques existantes afin de faciliter la mise en œuvre d'autres recherches sur le sujet.

Afin d'inspirer de futures recherches sur le sujet, nous discutons maintenant brièvement de directions spécifiques qui méritent d'être poursuivies.

Les graphes bipartis sont constitués par nature de composants disparates qui les rendent difficiles à représenter. Les méthodes de plongement des graphes bipartis doivent prendre en compte à la fois les informations au niveau des nœuds et des arêtes (type, attributs et informations sur les arêtes, etc.) et au niveau du graphe dans son ensemble. La fusion de ces différents types d'information n'est pas une tâche facile et des efforts doivent être faits pour préserver autant que possible ces informations dans la représentation finale du graphe biparti.Par ailleurs, les graphes sont souvent de grandes structures qui ont tendance à croître rapidement. Nous nous attendons à voir de plus en plus de graphes de grande échelle. Par conséquent, le passage à l'échelle des méthodes de plongement est essentiel et les futures méthodes de plongement de graphes bipartis doivent être conçues pour traiter cet aspect. S'il n'est pas possible de trouver un compromis entre la qualité des plongements et le passage à l'échelle, alors ce dernier aspect est une caractéristique qui permet de distinguer les approches utilisables de celles qui ne le sont pas. En outre, les travaux existants sur les plongements de graphes bipartis ont peu traité le cas des graphes bipartis dynamiques (c'est-à-dire des graphes qui évoluent dans le temps). Cependant, de nombreux graphes issus de scénarios réels sont dynamiques, tels que les graphes sociaux. Même si le plongement de graphes bipartis est un domaine naissant, la modélisation des mécanismes dynamiques dans les méthodes de plongement est un sujet de recherche important car il répond à de nombreuses problématiques industrielles. Une option pour y parvenir est d'utiliser un graphe biparti avec des arêtes attribuées, comme [ZXKZ17], et de considérer l'information temporelle dans les arêtes de ce graphe.

Chapitre 4

Injection de connaissances temporelles dans la reconnaissance d'entités nommées historiques

Sommaire

4.1 Introduction 75
4.2 Travaux connexes 76
4.2.1 Reconnaissance d'entités nommées dans des données historiques 76
4.2.2 Reconnaissance d'entités nommées avec une base de connaissances 77
4.2.3 Temporalité dans les graphes de connaissances 78
4.3 Jeux de données 79
4.4 Contextes temporels de connaissances pour la reconnaissance d'entités nommées 80
4.4.1 Intégration de l'information temporelle 81
4.4.2 Extraction des contextes 81
4.4.3 Architecture de la reconnaissance d'entités nommées 82
4.5 Configuration expérimentale 84
4.5.1 Résultats 85
4.5.2 Impact des intervalles de temps 86
4.5.3 Impact des erreurs de numérisation 87
4.5.4 Limitations 87
4.6 Conclusions et travaux futurs 87

Les résultats présentés dans ce chapitre sont basés sur l'article « Injecting Temporal-aware Knowledge in Historical Named Entity Recognition » publié dans la conférence « 2023 European Conference on Information Retrieval ».

Les graphes bipartis, qui font l'objet du chapitre précédent, fournissent une approche pertinente pour traiter le langage. Néanmoins, ils sont moins complets que les graphes de connaissances pour tenir compte des disparités des liens contextuels rencontrés dans le langage naturel. En effet, les graphes de connaissances, qui sont des graphes hétérogènes, c'est-à-dire possédant plusieurs types de nœuds et relations, permettent d'outrepasser la limite d'un seul type de relation dans les graphes bipartis. C'est pourquoi, nous explorons leur potentiel pour la tâche de détection d'entité nommée (NER pour « Named Entity Recognition »). Cette tâche, définie dans le Chapitre 1, permet de caractériser de manière succincte le contenu contextuel d'un document en identifiant les mentions d'entités appartenant à certaines catégories. Ainsi, dans ce chapitre, nous nous intéressons à la détection d'entités nommées dans des collections historiques multilingues. Nous soutenons que, outre les multiples défis qui dépendent de la qualité de la numérisation (par exemple, les fautes d'orthographe et les erreurs linguistiques), les documents historiques peuvent poser un autre défi car ils couvrent une période suffisamment longue pour être affectés par les changements et l'évolution du langage naturel. Nous considérons donc que la détection d'entités dans des collections historiques est sensible au temps, et nous explorons l'inclusion de la temporalité dans la tâche de NER en exploitant des graphes de connaissances temporelles. Plus précisément, nous récupérons des contextes additionnels sémantiquement pertinents en explorant les informations temporelles fournies par les collections de données historiques et nous les incluons en tant que représentations moyennes dans un modèle de reconnaissance d'entités nommées basé sur un transformeur. Nous expérimentons avec deux collections récentes composées de journaux historiques du XIX^{e} et du XX^{e} siècle et de commentaires classiques du XIX^{e} siècle en anglais, français et allemand. Les résultats sont prometteurs et montrent l'efficacité de l'injection de connaissances temporelles dans les différents ensembles de données, langues et types d'entités.

4.1 Introduction

Ces dernières années ont vu la production d'un nombre croissant de corpus textuels pour les sciences humaines et sociales. La numérisation de l'imposante collection *Gallica* par la Bibliothèque nationale de France [1] en est un exemple représentatif. L'accès à ces données massives offre de nouvelles perspectives à un nombre croissant de disciplines, allant de l'histoire sociopolitique et culturelle à l'histoire économique, et de la linguistique à la philologie. Des milliards d'images de documents historiques, y compris des documents manuscrits numérisés, des registres médiévaux et de la presse ancienne numérisée, sont capturées et leur contenu est transcrit, manuellement à l'aide d'interfaces dédiées, ou automatiquement à l'aide de la reconnaissance optique de caractères (ROC) ou de la reconnaissance de l'écriture manuscrite (ou HTR pour « Handwritten Text Recognition »). Le processus de numérisation de masse, initié dans les années 1980 avec des projets internes à petite échelle, a conduit à « l'avènement du numérique », qui a atteint une certaine maturité au début des années 2000 avec des campagnes de numérisation à grande échelle dans toute l'industrie [ERBC20, ERFC20b]. Alors que ce processus de numérisation de masse se poursuit, des techniques de plus en plus avancées dans le domaine du traitement automatique du langage naturel sont dédiées aux documents historiques, offrant de nouvelles façons d'accéder à des archives en texte intégral enrichies sémantiquement [OBD+22], telles que la reconnaissance d'entité nommée (NER) [BHP+20, EHLP+22, HLPB+21], la liaison d'entités (ou désambiguïsation d'entités) [LPCDM+22] et la détection d'événements [BNLD22, NBLD20].

Cependant, pour développer de telles techniques, les collections historiques présentent de multiples défis liés à la qualité de la numérisation, à la nécessité de traiter des documents détériorés par l'effet du temps, aux matériaux d'impression de mauvaise qualité ou à des processus de numérisation imprécis, qui sont des problèmes courants dans les documents historiques [HPS+22]. En outre, les collections historiques peuvent faire face à un autre problème du fait que les documents couvrent une période suffisamment longue pour être affectés par le changement et l'évolution de la langue. Cela est particulièrement vrai dans le cas des langues d'Europe occidentale, qui n'ont acquis leurs normes orthographiques modernes que vers le XVIII^{e} ou le XIX^{e} siècle [MF22]. Étant donné que les collections existantes [ERBC20, ERFC20a, ERFC20b] fournissent des métadonnées telles que l'année de publication, nous proposons de tirer parti du contexte temporel des documents historiques afin d'accroître la qualité de leur enrichissement sémantique. Lorsque ces métadonnées ne sont pas disponibles, en raison de l'ancienneté des documents, l'année de publication peut être estimée. En effet, une nouvelle tâche de traitement automatique du langage a récemment vu le jour, son but étant de prédire l'année de publication d'un document donné [RRT+22].

Le NER, définit dans le Chapitre 1 correspond à l'identification d'entités d'intérêt dans les textes, généralement du type personne, organisation ou lieu. Ces entités agissent comme des ancres référentielles qui sous-tendent la sémantique des textes et guident leur interprétation. Par exemple, en Europe, à l'époque médiévale, la plupart des personnes

1. `https://gallica.bnf.fr/`

étaient identifiées simplement par un mononyme ou un nom propre unique. Les noms de famille ou patronymes sont apparus au XIII^{e} siècle, mais beaucoup plus tard dans certaines régions ou classes sociales (XVII^{e} siècle pour les Gallois). De nombreuses personnes portaient le même nom et l'orthographe était variée dans les langues vernaculaires et latines, ainsi qu'à l'intérieur d'une même langue (par exemple, Guillelmus, Guillaume, Willelmus, William, Wilhelm). Les lieux ont pu disparaître ou changer complètement, et ceux qui ont survécus à la préhistoire jusqu'au XXI^{e} siècle (par exemple, l'Écosse, le Pays de Galles, l'Espagne) sont très ambigus et ont des orthographes très différentes, ce qui rend leur identification très difficile [BRM^{+}20]. Dans cet article, nous nous concentrons sur l'exploration de la temporalité dans la détection d'entités à partir de collections historiques. Nous proposons donc une nouvelle technique pour injecter des connaissances temporelles supplémentaires en nous appuyant sur Wikipédia et Wikidata en guise de graphes de connaissances, pour fournir des informations contextuelles connexes. Plus précisément, nous récupérons des contextes supplémentaires sémantiquement pertinents en explorant les informations temporelles fournies par les collections de données historiques et nous les incluons en tant que représentations moyennes dans notre modèle de NER basé sur un réseau neuronal de type transformeur. Nous considérons que l'ajout de contextes grammaticalement corrects peut améliorer les textes sujets aux erreurs induites par le processus de numérisation et qu'ajouter de l'information temporelle peut favoriser la gestion des changements dans le langage ou dans les noms d'entités.

Ce chapitre est structuré comme suit : nous présentons les travaux connexes et les jeux de données respectivement dans les sections 4.2 et 4.3. La section 4.4 décrit notre méthodologie d'extraction de contextes supplémentaires par le biais de graphes de connaissances temporelles et la manière dont les contextes sont inclus dans le modèle proposé. Nous réalisons ensuite plusieurs expériences concernant la relativité des intervalles de temps lors de la sélection des contextes additionnels et présentons nos résultats dans la section 4.5. Enfin, la section 4.6 présente nos conclusions et les travaux futurs à envisager. Le code associé à notre travail est disponible sur le site web suivant : `https://github.com/EmanuelaBoros/clef-hipe-2022-l3i`.

4.2 Travaux connexes

Nous abordons succinctement dans la suite un horizon des domaines de recherche lié à ce travail.

4.2.1 Reconnaissance d'entités nommées dans des données historiques

En raison des multiples défis posés par la qualité de la numérisation ou les variations historiques d'une langue, le NER dans les documents historiques et numérisés est moins performant que dans les documents modernes [WJB^{+}20, YAS^{+}20]. Des campagnes d'évaluation récentes telles que celles organisées par le laboratoire *Identifying Historical People,*

Places, and other Entities (HIPE) à CLEF 2020 [2] [ERFC20b] et 2022 [3] [ERNM+22] ont proposé des tâches de NER et de liaison d'entités sur 200 ans de journaux historiques rédigés dans plusieurs langues (anglais, français, allemand, finnois et suédois) et ont montré avec succès que ces tâches bénéficient des progrès réalisés dans le domaine du traitement automatique du langage naturel utilisant des réseaux de neurones (en particulier grâce aux dernières avancées des approches de modèles de langage pré-entraînés basés sur les transformeurs), car une amélioration considérable des performances a été observée sur les collections historiques, en particulier pour le NER [KR20, SDLT20, TC20].

Les auteurs de [EHLP+22] présentent une étude approfondie du NER sur des ensembles de données historiques et soulignent les défis que doivent relever les méthodes de NER de pointe appliquées à des documents historiques et bruités. Pour surmonter l'impact des erreurs de reconnaissance optique de caractères, des plongements contextualisés au niveau des caractères des textes ont été utilisés pour trouver de meilleures représentations des mots hors vocabulaire (OOV pour « out-of-vocabulary »). Les plongements contextualisés sont appris à l'aide de modèles de langage et permettent de prédire le prochain caractère d'une chaîne de caractères en fonction des caractères précédents. En outre, d'autres recherches ont montré que le réglage fin (« fine-tuning ») de plusieurs encodeurs transformeurs sur des collections historiques pouvait atténuer les erreurs de numérisation [BHP+20]. Pour remédier au manque de ressources historiques, [RP18] a proposé d'utiliser l'apprentissage par transfert afin d'apprendre des modèles sur de grandes ressources contemporaines et de les adapter ensuite à quelques corpus de nature historique. Enfin, pour traiter les variations orthographiques, certains travaux ont développé des règles de transformation pour modéliser l'évolution diachronique des mots et générer une version normalisée adaptée aux systèmes de NER existants [DPRMGB+21, KG20]. Alors que la plupart de ces approches s'appuient généralement sur le contexte textuel local pour détecter les entités dans ces documents, les informations temporelles n'ont généralement pas été prises en compte. À notre connaissance, plusieurs approches ont été proposées pour la liaison d'entités en utilisant des signatures temporelles pour les entités afin de refléter l'importance des différentes années [ASdC+18], telle que l'utilisation de filtres basés sur le temps [LPCDM+22], mais pas pour le NER sur des documents historiques.

4.2.2 Reconnaissance d'entités nommées avec une base de connaissances

Compte tenu des comportements complémentaires des approches basées sur les connaissances et de celles basées sur les réseaux neuronaux pour le NER, plusieurs études ont exploré des approches basées sur les connaissances comprenant différents types de représentations symboliques (par exemple, bases de connaissances, graphes de connaissances statiques, nomenclature toponymique) et ont remarqué des améliorations significatives dans les représentations de tokens et la détection d'entités sur des ensembles de données modernes (par exemple, CoNLL [TKSDM03], OntoNotes 5.0 [PMX+13]). L'information

2. https://impresso.github.io/CLEF-HIPE-2020/
3. https://hipe-eval.github.io/HIPE-2022/

émanant de nomenclature toponymique a été intégrée dans des modèles de NER parallèlement aux représentations au niveau des mots [MFRM21]. En effet, Wikipedia a principalement été utilisé pour augmenter les représentations sémantiques des entités possibles en affinant les modèles de langage récents pré-entraînés sur la tâche de remplissage d'espace vide (« fill-in-the-blank » (« cloze »)) [RYT22, YAS+20].

L'introduction de contextes externes dans des systèmes de NER s'est avérée positive sur les performances d'identification des entités. [WSC+22] a construit un système de base de connaissances à partir d'une instance locale de Wikipédia pour récupérer des documents pertinents à partir de la phrase d'une requête. Les documents récupérés et les phrases de requêtes, après concaténation, sont transmis au système de NER. La méthodologie que nous proposons peut être considérée comme inspirée de leur travail, mais nous incluons les contextes additionnels au niveau du modèle en générant une représentation moyenne pour chaque contexte au lieu de concaténer les contextes avec la phrase initiale. Nous soutenons que le fait d'avoir des représentations regroupées pour chaque contexte additionnel peut réduire le bruit qui pourrait être créé par d'autres entités trouvées dans ces textes.

4.2.3 Temporalité dans les graphes de connaissances

Des avancées récentes ont montré un intérêt croissant pour l'apprentissage de représentations d'entités et de relations incluant des informations temporelles [CXG+22]. D'autres travaux [XNA+20] ont proposé un modèle de plongement de graphes de connaissances temporels (TKG pour « Temporal Knowledge Graph ») pour représenter des faits impliquant des intervalles de temps en concevant l'évolution temporelle des plongements d'entités comme une rotation dans un espace vectoriel complexe. Les entités et les relations ont été représentées sous la forme de plongements complexes où les changements temporels sont représentés par les rotations des plongements d'entités dans l'espace vectoriel complexe. Étant donné que les graphes de connaissances changent au fil du temps dans les données évolutives (par exemple, le fait *Le Président des États-Unis est Barack Obama* n'est valable que de 2009 à 2017), une approche de plongement de graphes de connaissances tenant compte du temps [XCNL21] a également été proposée en allant au-delà des représentations à valeurs complexes. Cette approche introduit des plongements multivectoriels afin de modéliser les entités, les relations et les horodatages pour les TKGs. Nous convenons de nommer horodatage, un point discret dans le temps qui correspond à une année. D'autres recherches [XSL21] ont présenté un modèle de réseau neuronal de graphe (GNN pour « Graph Neural Network ») traitant les informations d'horodatage comme une propriété inhérente à la structure du graphe avec un mécanisme d'auto-attention pour associer des poids appropriés aux nœuds en fonction de la pertinence de leurs relations et des horodatages de leur voisinage. Par conséquent, les horodatages sont considérés comme des propriétés des liens entre les entités.

Les TKGs présentent toutefois de nombreux défauts et une faible qualité des informations qui varie, notamment l'exactitude des faits, l'exhaustivité et l'actualité. Par conséquent, d'autres recherches [DAN22] explorent davantage les TKGs en s'intéressant à la complétude des connaissances avec des informations exactes mais manquantes. En outre,

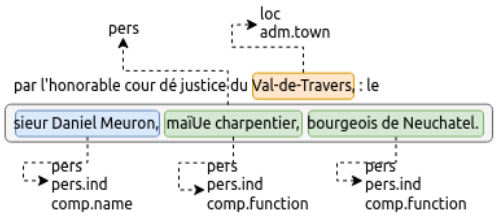

FIGURE 4.1 – Un exemple tiré du jeu de données `hipe-2020`.

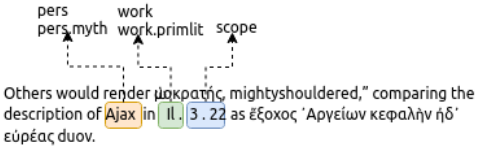

FIGURE 4.2 – Un exemple tiré du jeu de données `ajmc`.

étant donné que ces TKGs souffrent souvent d'incomplétude, les auteurs de [ZCF+21] ont introduit un modèle d'apprentissage de représentation tenant compte du temps qui aide à déduire les faits temporels manquants en s'intéressant aux faits qui se produisent de manière récurrente et en tirant parti d'un mécanisme de copie pour identifier les faits qui se répètent.

Les méthodes mentionnées précédemment montrent que l'utilisation des TKGs est considérée comme un domaine émergent qui est exploré, en particulier dans le domaine du traitement automatique du langage naturel. La disponibilité d'informations sur l'évolution temporelle des entités pourrait non seulement être une solution prometteuse pour améliorer leurs représentations de connaissances sémantiques, mais aussi fournir des informations contextuelles supplémentaires pour améliorer l'efficacité des systèmes de NER. À notre connaissance, notre travail est la première tentative d'exploitation d'informations temporelles à disposition dans des TKGs pour améliorer le NER.

4.3 Jeux de données

Dans cette étude, nous utilisons deux collections composées de journaux historiques et de commentaires classiques couvrant environ 200 ans. Nous utilisons les jeux de données `hipe-2020` et Ajax Multi-Commentary (`ajmc`), récemment proposés par la campagne d'évaluation CLEF-HIPE-2022 [ERDC22].

`hipe-2020` comprend des articles de journaux suisses, luxembourgeois et américains en français, allemand et anglais (XIXe-XXe siècles) et contient 19 848 entités liées dans leurs jeux d'entraînement [ERBC20, ERFC20a, ERFC20b]. Pour chaque langue, le corpus est divisé en sous-ensembles d'entraînement, de développement et de test, à l'exception de l'anglais pour lequel seuls les ensembles de développement et de test ont été produits [ERC+20]. Pour ce dernier, nous avons utilisé les données en français et en allemand pour entraîner les modèles. Un exemple tiré de l'ensemble de données français est présenté dans la Figure 4.1.

`ajmc` est composé de commentaires classiques issus du projet Ajax Multi-Commentary qui comprend des commentaires numérisés du XIXe siècle publiés en français, en allemand et en anglais [RNMR21] annotés avec des entités nommées à la fois universelles et spécifiques à un domaine. Un exemple en anglais est présenté dans la Figure 4.2.

TABLE 4.1 – Aperçu des jeux de données `hipe-2020` et `ajmc`. LOC = Localisation, ORG = Organisation, PERS = Personne, PROD = Produit, DATE = Date, METIER = Métier, OBJET = Objet, et MISSION = Mission. Pour chaque langue, le jeu de données est divisé en trois sous ensembles : un d'entraînement (train), un de développement (dev) et un de test (test).

	`hipe-2020`									`ajmc`								
	Français			Allemand			Anglais			Français			Allemand			Anglais		
Type	train	dev	test	train	dev	test	train	dev	test	train	dev	test	train	dev	test	train	dev	test
LOC	3,089	774	854	1,740	588	595	–	384	181	15	0	9	31	10	2	39	3	3
ORG	836	159	130	358	164	130	–	118	76	–	–	–	–	–	–	–	–	–
PERS	2,525	679	502	1,166	372	311	–	402	156	577	123	139	620	162	128	618	130	96
PROD	200	49	61	112	49	62	–	33	19	–	–	–	–	–	–	–	–	–
DATE	276	68	53	118	69	49	–	29	17	2	0	3	2	0	0	12	5	3
METIER	–	–	–	–	–	–	–	–	–	378	99	80	321	70	74	467	116	95
OBJET	–	–	–	–	–	–	–	–	–	10	0	0	6	4	2	3	0	0
MISSION	–	–	–	–	–	–	–	–	–	639	169	129	758	157	176	684	162	151

Ces deux collections posent plusieurs défis importants : le multilinguisme (elles contiennent toutes deux trois langues : anglais, français et allemand), les documents à code mixte (par exemple, les commentaires, où le grec est mélangé à la langue du commentateur), la granularité des annotations et la richesse des textes caractérisés par une forte densité d'entités nommées. Les deux jeux de données fournissent différentes métadonnées sur les documents dont le niveau de détails varie (par exemple, la langue, le type de document, la source originale, la date). Ils possèdent différents ensembles d'étiquettes d'entités qui ont été construits selon des directives d'annotation différentes. Le Tableau 4.1 présente les statistiques relatives au nombre et au type d'entités dans les jeux de données susmentionnés, divisés en fonction des ensembles d'entraînement, de développement et de test.

4.4 Contextes temporels de connaissances pour la reconnaissance d'entités nommées

La sortie d'un système de ROC (reconnaissance optique de caractères) contient des erreurs qui produisent un texte bruité, similaires à celles étudiées par [MTR19]. Il a été observé qu'il est préférable d'adapter les systèmes de NER pour traiter le bruit de ROC, que d'adapter les corpus de NER eux-mêmes [EHP+21]. Pour traiter les erreurs de ROC, nous introduisons des contextes externes grammaticalement corrects dans les systèmes de NER qui ont un impact positif sur les performances d'identification des entités malgré ces défis [WSC+22]. En outre, inclure ces contextes en tenant compte de la temporalité pourrait encore améliorer la détection des entités sensibles au temps. Nous proposons donc plusieurs paramètres pour l'inclusion de contextes additionnels basés sur Wikidata5m [4] [WGZ+21], un graphe de connaissances comprenant cinq millions d'entités

4. `https://deepgraphlearning.github.io/project/wikidata5m`

Wikidata [5] de domaine général (par exemple, des célébrités, des événements, des concepts, des objets) et qui sont alignées sur une description correspondant au premier paragraphe de la page Wikipédia correspondante.

4.4.1 Intégration de l'information temporelle

Un TKG contient des indications temporelles et des faits associés à une entité qui fournissent des informations sur les changements spontanés ou les transformations temporelles plus progressives de celle-ci tout en informant sur les relations avec d'autres entités. Nous agrégeons la temporalité dans Wikidata5m, y compris le TKG créé par [LC18] et mis au point par [GDDN18] [6]. Ce TKG contient plus de 11 000 entités, 150 000 faits, et une portée temporelle entre les années 508 et 2017. Pour une entité donnée, il fournit un ensemble de faits temporels décrivant les interactions de l'entité dans le temps. Un fait (e, r_i, e_i, t_i) est composé de deux entités e et e_i qui sont liées par la relation r_i et l'horodatage t_i. Il est donc nécessaire de combiner l'information temporelle de ces faits en un élément singulier par le biais d'un opérateur d'agrégation.

Nous effectuons une transformation sur l'information temporelle de chaque fait d'une entité afin de les combiner en une seule information temporelle. Soit e une entité décrite par les faits :

$$\{F_e\}_{i=1}^{n} = \{(e, r_1, e_1, t_1), (e, r_2, e_2, t_2), \dots (e, r_i, e_i, t_i), \dots (e, r_n, e_n, t_n)\},$$

L'opérateur d'agrégation est la fonction $AGG \rightarrow t_e$ qui prend en entrée les informations temporelles de F_e et produit les informations temporelles associées à e. Plusieurs opérateurs d'agrégation sont possibles. Parmi eux, les options naturelles sont les opérateurs moyenne, médiane, minimum et maximum. Le minimum d'un ensemble de faits est défini comme le fait le plus ancien, et le maximum comme le fait le plus récent. Par exemple, si une entité est associée à quatre faits s'étalant sur les années 1891, 1997, 2006 et 2011, les quatre opérateurs précédents donnent respectivement : 1976, 2001, 1891 et 2011. Étant donné que nos ensembles de données correspondent à des documents datant du XIX^{e} au XX^{e} siècle, l'opérateur minimum est plus susceptible de créer un contexte temporel approprié pour les entités. En effet, puisque nous considérons des textes anciens, nous souhaitons mettre en évidence les entités en adéquation avec la période de temps correspondante afin d'accentuer les faits plus anciens. À la fin de l'opération d'agrégation, 8 176 entités de Wikidata5m sont associées à une année comprise entre 508 et 2001, ce qui permet de filtrer la plupart des faits survenus au cours du XXI^{e} siècle.

4.4.2 Extraction des contextes

Notre système de base de connaissances repose sur une instance locale d'Elastic-Search [7] et suit un alignement de la similarité sémantique multilingue, ce qui présente un

5. https://www.wikidata.org/
6. https://github.com/mniepert/mmkb/tree/master/TemporalKGs/wikidata
7. https://www.elastic.co/guide/en/elasticsearch/reference/8.1/release-highlights.html

avantage pour les requêtes multilingues, et est réalisé avec des index de champs vectoriels denses (« dense vector field »). Ainsi, étant donné le vecteur d'une requête, ElasticSearch récupère les k plus proches voisins (k-NN pour « k-Nearest-Neighbors »), c'est-à-dire les k vecteurs les plus proches, et renvoie les documents correspondants comme résultats de la recherche. Pour chaque entité Wikidata5m, nous créons une entrée ElasticSearch comprenant un champ d'identification, un champ de description et un champ du plongement de la description que nous obtenons à l'aide d'un modèle Sentence-BERT multilingue pré-entraîné [RG19, RG20]. Nous construisons un index sur l'identifiant de l'entité et un index vectoriel dense sur le plongement de la description. Nous proposons deux configurations différentes pour la recherche de contexte :

— `non-temporel` : Cette configuration n'utilise aucune information temporelle. Étant donné une phrase d'entrée pendant la recherche de contexte, nous obtenons d'abord la représentation vectorielle dense correspondante avec le même modèle Sentence-BERT que celui utilisé pendant la phase d'indexation. Ensuite, nous interrogeons la base de connaissances pour extraire les entités les plus sémantiquement similaires sur la base d'une recherche de similarité cosinus avec l'algorithme des k plus proches voisins sur l'index du vecteur dense du plongement de la description. Le contexte C est finalement composé de k descriptions d'entités.

— `temporel-`δ : Cette configuration intègre les informations temporelles. Pour chaque entité sémantiquement similaire extraite par `non-temporel`, nous appliquons une opération de filtrage afin de conserver ou d'écarter l'entité du contexte. Étant donné l'année t_{entre} liée aux métadonnées de la phrase d'entrée lors de la recherche de contexte, l'entité est conservée si l'année qui lui est associée t_e se trouve dans l'intervalle $[t_{entre} - \delta; t_{entre} + \delta]$, où δ est le seuil d'intervalle d'année, sinon elle est rejetée. Plusieurs valeurs sont testées dans la suite afin de voir l'impact de ce paramètre. Après application de AGG, t_e est l'année la plus ancienne de l'ensemble des faits de l'entité e dans le TKG. Si t_e n'existe pas, e est également conservé.

4.4.3 Architecture de la reconnaissance d'entités nommées

Modèle de base Notre modèle consiste en une approche d'apprentissage hiérarchique et multitâche, avec un encodeur finement réglé basé sur le modèle pré-entraîné BERT [DCLT19]. Ce modèle comprend un encodeur avec deux couches transformeur [VSP+17] avec des modules adaptateurs [HGJ+19, PVGR20] sur le modèle BERT. Les adaptateurs sont ajoutés à chaque couche transformeur après la projection suivant le mécanisme d'attention multi-tête (multi-headed attention) et ils s'adaptent non seulement à la tâche mais aussi à l'entrée bruitée, ce qui s'est avéré bénéfique pour les performances du NER dans ces conditions particulières [BHP+20]. Enfin, la couche de prédiction consiste en une couche de champ aléatoire conditionnel (CRF pour Conditional Random Field).

Plus précisément, supposons que $TokRep = x = (x_i)_{i=1}^{l}$ est une séquence d'entrée composée de l tokens, désignée par $(x_i)_{i=1}^{l} = (x_1, x_2, \ldots, x_i, \ldots, x_l)$, où x_i fait référence au i-ième token de la séquence. Nous appliquons d'abord un modèle de langage pré-entraîné de type transformeur comme encodeur que nous allons ensuite finement régler. On obtient alors $(h_i)_{i=1}^{l}$, avec $H_{[CLS]} = encodeur((x_i)_{i=1}^{l})$ où $(h_i)_{i=1}^{l} =$

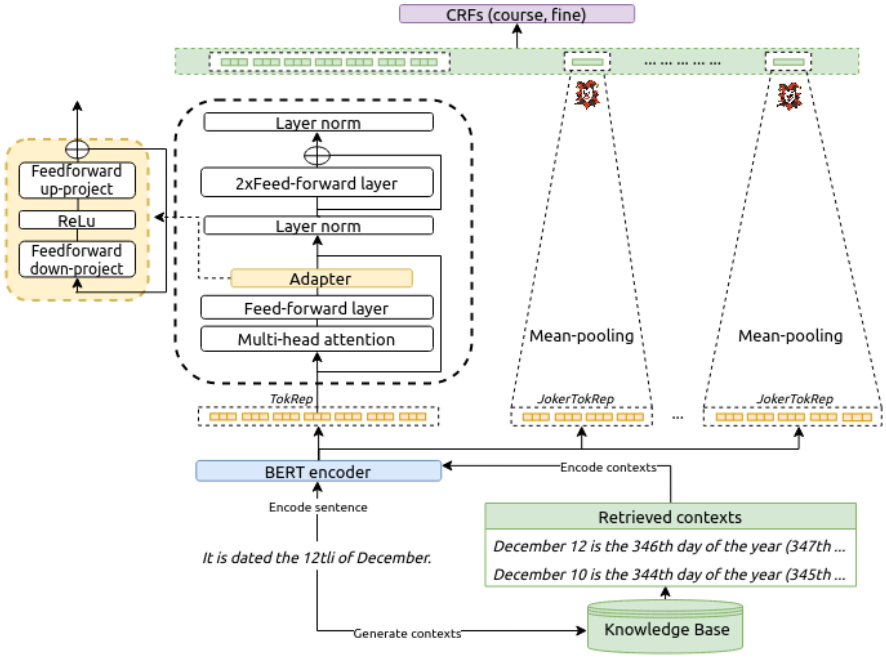

FIGURE 4.3 – Architecture du modèle de NER avec les contextes temporels (*jokers de contexte*).

$(h_1, h_2, \ldots, h_i, \ldots, h_l)$ est la représentation de chaque token dans la séquence x et $H_{[CLS]}$ représente le vecteur d'état caché final du token spécial $[CLS]$ désignant l'ensemble de la séquence x. L'encodeur contient un certain nombre de couches qui prennent en entrée la matrice $H = \{h_i\}_{i=1}^{l} \in \mathbb{R}^{l \times d}$ où d est la dimension d'entrée (dimension de sortie de l'encodeur). Une couche transformeur comprend un mécanisme d'auto-attention à plusieurs têtes $Head(h) : Q^{(h)}, K^{(h)}, V^{(h)} = HW_q^{(h)}, HW_k^{(h)}, HW_v^{(h)}$ et $MultiHead(H) = (Head^{(1)}, \ldots, Head^{(n)})W_O$ [8] où n est le nombre de têtes et l'exposant h représente l'indice de tête. Q_t est le vecteur de requête du t-ième token, j est le token auquel le t-ième token participe. K_j est la représentation du vecteur clé du j-ième token. Le softmax [Bri89] $Attn$ se situe le long de la dernière dimension. $MultiHead(H)$ est la concaténation sur la dernière dimension de taille $\mathbb{R}^{l \times d}$ où d_k est le facteur d'échelle $d_k \times n = d$. W_O est un paramètre d'apprentissage de taille $\mathbb{R}^{d \times d}$.

En combinant la sous-couche de propagation avant en fonction de la position et l'attention à têtes multiples, nous obtenons une couche à propagation avant $FFN(f(H)) = max(0, f(H)W_1)W_2$ où W_1 et W_2 sont des paramètres pouvant être appris et max est la fonction d'activation $ReLU$. $W_1 \in \mathbb{R}^{d \times d_{FF}}$, $W_2 \in \mathbb{R}^{d_{FF} \times d}$ sont des matrices de projection entraînées, et d_{FF} est un hyperparamètre. L'adaptateur de tâche est appliqué à ce niveau sur $TokRep$ sur chaque couche et consiste en une projection vers le bas $D \in \mathbb{R}^{h \times d}$ où h est la taille cachée du modèle transformeur et d est la dimension de l'adaptateur,

8. Nous laissons de côté les détails qui peuvent être consultés dans [VSP+17].

également suivie d'une activation $ReLU$ et d'une projection vers le haut $U \in \mathbb{R}^{d \times h}$.

Jokers de contexte Pour inclure les contextes supplémentaires générés comme expliqué dans la section 4.4, nous introduisons les *jokers de contexte*. Chaque contexte supplémentaire passe par l'encodeur [9] générant un $JokerTokRep$ qui est ensuite traité via une couche « Mean-Pooling » le long de l'axe de la séquence. Nous appelons ces représentations *jokers de contexte*. Nous les considérons comme des jokers insérés dans la représentation de la phrase courante afin d'améliorer la reconnaissance des entités de granularité fine. Cependant, nous considérons également que ces jokers peuvent affecter les résultats d'une manière qui n'est pas immédiatement apparente et qu'ils peuvent nuire aux performances d'un système de NER. La Figure 4.3 illustre l'architecture NER décrite.

4.5 Configuration expérimentale

Notre dispositif expérimental se compose d'un modèle de référence et de quatre configurations avec différents niveaux de contextes de connaissances :

- `non-contextuel` : notre modèle tel que décrit dans la section 4.4.3. Dans cette configuration de référence, aucun contexte n'est ajouté aux représentations des phrases d'entrée.
- `non-temporel` : les contextes sont générés avec la première configuration de recherche de contexte sans information temporelle et intégrés dans le modèle par l'intermédiaire des *jokers de contexte*.
- `temporel-(50|25|10)` : les contextes sont générés avec la deuxième configuration de recherche de contexte avec $\delta \in \{50, 25, 10\}$ (où δ est l'intervalle de temps ou le seuil d'intervalle d'année) et intégrés dans le modèle via les *jokers de contexte*.

Hyperparamètres

Afin de disposer d'un cadre expérimental uniforme, nous avons choisi un modèle pré-entraîné multilingue basé sur BERT [10]. Nous désignons le nombre de couches (c'est-à-dire les blocs transformeurs basés sur l'adaptateur) par L, la taille cachée par H et le nombre de têtes d'auto-attention par A. Pour BERT, les valeurs sont L=12, H=768 et A=12. Nous avons ajouté deux couches avec H=128, A=12, et les adaptateurs (qui sont entraînés sur la tâche) ont une taille de 128×12. Pour toutes les configurations de recherche contextuelle, la taille du contexte $|C|$ d'une phrase d'entrée a été fixée à $k = 10$. Pour l'indexation des documents dans ElasticSearch, nous avons utilisé le modèle multilingue pré-entraîné Sentence-BERT [11].

9. Nous n'utilisons pas dans ce cas les couches de transformeur supplémentaires avec adaptateurs, car elles ont été spécifiquement conçues pour les textes bruités/non standard et n'apportent aucune amélioration de performances sur les textes standard [BHP+20].

10. `https://huggingface.co/bert-base-multilingual-cased`

11. `https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2`

TABLE 4.2 – Résultats en français, allemand et anglais pour les jeux de données `hipe-2020` et `ajmc`.

	Français						**Allemand**						**Anglais**					
	`hipe-2020`			`ajmc`			`hipe-2020`			`ajmc`			`hipe-2020`			`ajmc`		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
`non-contextuel`																		
CS	0.755	0.757	0.756	0.829	0.806	0.817	0.754	0.730	0.742	0.910	0.877	0.893	0.604	0.563	0.583	0.789	0.859	0.823
CF	0.857	0.859	0.858	0.883	0.858	0.870	**0.853**	0.826	0.839	0.935	0.901	0.917	0.778	0.726	0.751	0.855	0.931	0.891
`non-temporel`																		
CS	0.762	**0.767**	**0.765**	0.829	0.783	0.806	0.759	**0.767**	**0.763**	**0.930**	0.898	0.913	0.565	0.601	0.583	0.828	0.871	0.849
CF	0.862	**0.869**	0.866	**0.906**	0.856	0.880	0.847	0.856	0.852	**0.949**	0.916	**0.932**	0.741	0.788	0.764	0.885	0.931	0.908
`temporel-50`																		
CS	**0.765**	0.765	**0.765**	0.839	0.822	0.830	0.748	0.756	0.752	0.921	**0.911**	**0.916**	**0.643**	0.617	**0.630**	0.855	0.882	0.868
CF	**0.867**	0.867	**0.867**	0.901	0.883	0.892	0.833	0.842	0.838	0.937	**0.927**	**0.932**	**0.794**	0.762	0.777	0.916	**0.945**	0.931
`temporel-25`																		
CS	0.759	0.756	0.757	**0.848**	**0.839**	**0.844**	0.757	0.743	0.750	0.925	0.903	0.914	0.621	0.630	0.625	0.833	0.876	0.854
CF	0.863	0.859	0.861	0.902	**0.892**	**0.897**	0.852	0.835	0.843	0.938	0.916	0.927	0.787	0.800	**0.793**	0.893	0.940	0.916
`temporel-10`																		
CS	0.762	0.764	0.763	**0.848**	**0.839**	**0.844**	**0.760**	0.765	0.762	0.917	0.898	0.907	0.605	**0.646**	0.625	**0.866**	**0.888**	**0.877**
CF	0.863	0.866	0.865	0.902	**0.892**	**0.897**	0.852	**0.857**	**0.854**	0.936	0.916	0.926	0.760	**0.811**	0.784	**0.922**	**0.945**	**0.933**
`L3i@HIPE-2022`																		
CS	0.782	0.827	0.804	0.810	0.842	0.826	0.780	0.787	0.784	0.946	0.921	0.934	0.624	0.617	0.620	0.824	0.876	0.850
CF	0.883	0.933	0.907	0.856	0.889	0.872	0.870	0.878	0.874	0.965	0.940	0.952	0.793	0.784	0.788	0.868	0.922	0.894

Évaluation

L'évaluation est effectuée sur la tâche de NER à granularité grossière. Le NER à granularité grossière se réfère à l'identification et à la catégorisation des mentions d'entités selon les types d'entités de haut niveau énumérés dans le Tableau 4.1. Nous utilisons les métriques de précision (P), de rappel (R) et la F-mesure micro (F1) [ERBC20, MKS+99] qui prend en compte tous les vrais positifs, faux positifs, vrais négatifs et faux négatifs des données. Nous exploitons deux configuration de délimitation des mentions d'entités dans le texte : une configuration stricte (correspondance exacte des délimitations) et une configuration flou de correspondance [12] (« fuzzy boundary matching »). Nous appelons ces mesures « grossières-strictes » (***CS***) et « grossières-floues » (***CF***).

4.5.1 Résultats

Le Tableau 4.2 présente nos résultats dans les trois langues et ensembles de données (les meilleurs résultats sont en gras). On constate que les modèles comportant des *jokers de contexte* supplémentaires basés sur des connaissances apportent une amélioration par rapport au modèle de référence sans contexte ajouté. En outre, l'inclusion d'informations temporelles donne de meilleurs résultats que les contextes non temporels. Les scores de `ajmc` sont plus élevés que ceux de `hipe-2020`, indépendamment de la langue et des contextes. Nous expliquons ce comportement par la faible diversité de certains types d'entités du jeu de données `ajmc`. Par exemple, les dix entités les plus fréquentes du type « personne » représentent respectivement 55%, 51.5% et 62.5% dans les ensembles d'entraînement, de développement et de test. De plus, huit des dix entités les plus fréquentes sont à la fois dans les ensembles d'entraînement et de test. L'anglais `hipe-2020` présente

12. Nous avons utilisé l'évaluateur HIPE `https://github.com/hipe-eval/HIPE-scorer`.

des scores plus faibles comparativement au français et à l'allemand, indépendamment des contextes. Nous attribuons cette baisse de performance à l'utilisation des ensembles français et allemand pendant l'entraînement, étant donné l'absence d'un ensemble d'entraînement spécifique pour l'anglais.

Les deux dernières lignes du Tableau 4.2 montrent les résultats de notre meilleur système [BGGG$^+$] pendant la campagne d'évaluation HIPE-2022 [ERFC20a]. Ce système est similaire à celui décrit dans la section 4.4.3, mais il utilise, pour chaque langue, un modèle linguistique spécifique et n'inclut aucune connaissance temporelle. Le modèle linguistique supplémentaire est à l'origine des résultats légèrement supérieurs. Nous nous attendrions à des résultats plus élevés si nous avions des informations temporelles, mais pour cette configuration expérimentale, nous étions limités en termes de ressources. Pour la moitié des ensembles de données, ce système est plus performant que les configurations tenant compte de la dimension temporelle (valeurs soulignées), mais au prix d'une dépendance à l'égard de la langue, un inconvénient qui affecte principalement l'ensemble de données anglais `hipe-2020`, pour lequel aucune donnée d'entraînement n'est disponible.

4.5.2 Impact des intervalles de temps

`ajmc` contient des commentaires du XIX^{e} siècle sur des textes grecs [RNMR21] et a été créé dans le cadre du projet Ajax MultiCommentary [13], et donc les jeux de données français, allemand et anglais concernent une tragédie grecque de Sophocle, l'Ajax, datant de la période antique [14]. Le `ajmc` allemand contient des commentaires de deux années (1853 et 1894), de même que le `ajmc` anglais (1881 et 1896), et le `ajmc` français ne contient des commentaires que d'une seule année (1886). En raison de la taille de la collection, `hipe-2020` couvre un plus grand nombre d'années. En termes d'étendue, les articles français ont été collectés de 1798 à 2018, les articles allemands de 1798 à 1948 et les articles anglais de 1790 à 1960.

Nous avons donc examiné la différence entre les contextes récupérés par les configurations non temporelles et temporelles.

Le Tableau 4.3 résume ces différences pour les ensembles d'entraînement et de test et affiche le nombre de contextes qui ont été filtrés et remplacés à partir de `non-temporel` pour chaque intervalle de temps, c'est-à-dire $\delta \in \{50, 25, 10\}$. Globalement, plus l'intervalle d'années est faible, plus le nombre de contextes remplacés est important. On peut remarquer que le nombre de contextes remplacés est plus faible pour `ajmc` que pour `hipe-2020`. Cela s'explique par le nombre restreint d'années et le manque de diversité des entités au cours de ces périodes. En comparant les résultats du Tableau 4.2, nous pouvons déduire qu'en général il est avantageux de mettre en œuvre des intervalles de temps plus courts tels que $\delta = 10$. En fait, `temporel-10` présente des scores F1 plus

13. `https://mromanello.github.io/ajax-multi-commentary/`

14. Bien que la date exacte de sa première représentation soit inconnue, la plupart des spécialistes la situent relativement tôt dans la carrière de Sophocle (peut-être s'agit-il de la plus ancienne pièce de Sophocle encore existante), quelque part entre 450 et 430 avant J.-C., peut-être autour de 444 avant J.-C.

TABLE 4.3 – Nombre de contextes introduits selon l'intervalle de temps.

	Français		**Allemand**		**Anglais**	
	train	test	train	test	train	test
`temporel-50/25/10`						
`hipe-2020`	120/154/217	42/47/61	325/393/482	12/14/14	192/222/246	77/85/96
`ajmc`	10/12/12	0/0/0	71/71/73	20/20/20	2/2/2	0/0/0

élevés que `ajmc` dans presque tous les cas. Toutefois, cela varie en fonction de la langue et de la répartition par année de l'ensemble de données.

4.5.3 Impact des erreurs de numérisation

Les commentaires de la littérature grecque classique présentent les difficultés typiques de la ROC historique. Avec des mises en page complexes, souvent avec plusieurs colonnes et lignes de texte, la qualité de la numérisation des commentaires peut avoir un impact important sur la détection d'entités nommées et d'autres tâches en aval telles que la liaison d'entités. Statistiquement, environ 10% des entités sont affectées par la ROC dans les jeux de données anglais et allemand `ajmc` et 27,5% des entités sont contaminées dans le corpus français. Les modèles avec contexte additionnel, en particulier les approches temporelles, contribuent à la reconnaissance des entités, qu'elles soient contaminées ou correctes. Cette contribution est encore plus significative sur les entités avec des erreurs de numérisation. Elle s'est traduite par une meilleure amélioration de la reconnaissance des entités contaminées par rapport aux entités correctes malgré leur prédominance dans les données. Dans le corpus allemand, par exemple, le gain est d'environ 14% en utilisant `temporel-50` par rapport au modèle référence, alors qu'il n'est que de 2% pour les entités correctes. En outre, les trois quarts des entités avec un taux d'erreur de 67% des caractères sont correctement reconnues alors que le modèle référence n'en reconnaissait qu'un quart. Enfin, toutes les performances des modèles sont complètement amoindries lorsque les taux d'erreur sont supérieurs à 70% sur les entités.

4.5.4 Limitations

Idéalement, le système nécessite des métadonnées sur l'année où les ensembles de données ont été rédigés ou au moins un intervalle de temps. Sinon, il est nécessaire d'utiliser d'autres systèmes pour prédire l'année de publication [RRT+22]. Cependant, les erreurs de ces systèmes sont propagées et peuvent avoir un impact sur les résultats du NER.

4.6 Conclusions et travaux futurs

Dans ce chapitre, nous avons exploré une stratégie afin d'injecter des informations temporelles dans la tâche de reconnaissance d'entités nommées sur des collections historiques. En particulier, nous nous sommes appuyés sur l'utilisation de contextes sémanti-

quement pertinents en explorant les informations temporelles fournies dans les métadonnées des collections et dans des graphes de connaissances temporelles. Les modèles que nous avons proposés incluent des contextes en tant que représentations moyennes d'un modèle de réseau neuronal transformeur. Nous avons observé plusieurs tendances concernant l'importance de la temporalité pour les journaux historiques et les commentaires classiques, en fonction des intervalles de temps et du taux d'erreur de numérisation. Tout d'abord, nos résultats ont montré qu'une courte période de temps fonctionne mieux pour les collections avec une diversité d'entités limitée et de petits intervalles d'années, tandis qu'une période de temps plus longue profite aux grands intervalles d'années. Ensuite, nous avons également montré que notre approche est efficace pour détecter les entités contaminées par des erreurs de numérisation, même avec un taux d'erreur de 67% des caractères. Enfin, nous avons remarqué que la qualité des contextes retrouvés dépend de l'affinité entre la collection historique et la base de connaissances.

Pour favoriser de futures études dans ce domaine, nous abordons brièvement quelques orientations qui méritent d'attirer l'attention pour de potentielles investigations.

Dans des travaux futurs. il pourrait être intéressant d'inclure des informations sur la temporalité en prédisant les années d'un large ensemble de pages Wikipédia à utiliser comme contextes complémentaires. Par ailleurs, cette étude confirme l'intérêt des méthodes utilisant l'information de graphe de connaissances sur des tâches linguistiques. Il serait pertinent d'étendre ces recherches sur d'autres tâches connexes, comme le classement de textes.

Chapitre 5

Impact du bruit sur la recherche d'information

Sommaire

5.1 Introduction 91
5.2 État de l'art 92
5.3 Méthodologie 93
5.3.1 Jeu de données 94
5.3.2 Injection de bruit 94
5.3.3 Outil d'injection de bruit 97
5.3.4 Modèles de classement 98
5.3.5 Évaluation en environnement bruité 99
5.4 Résultats et discussion 99
5.5 Conclusion 102

Les résultats présentés dans ce chapitre sont basés sur l'article « A Quantitative Analysis of Noise Impact on Document Ranking » publié dans the 2023 IEEE Conference on Systems, Man, and Cybernetics (IEEE SMC 2023).

Après des décennies de numérisation massive, une quantité substantielle de documents existe sous forme numérique. L'accessibilité de ces documents est fortement influencée par la qualité de l'indexation des documents. Cependant, la plupart de ces documents sont indexés dans des versions bruitées qui comportent de nombreuses erreurs. Le bruit peut être dû à des erreurs de saisie manuelle ou à un processus de reconnaissance optique de caractères. Il se traduit par des fautes d'orthographe, des caractères manquants, etc. Ce chapitre présente une étude de l'impact du bruit sur le classement de textes, une tâche essentielle à la recherche d'information que nous avons défini dans le Chapitre 1. Nous fournissons une analyse approfondie et quantitative de l'impact des erreurs sur le classement de textes en évaluant deux modèles de classement populaires sur plusieurs

versions bruitées du jeu de données de classement de passages MS MARCO. Ces jeux de données ont été produits pour l'occasion, avec différents niveaux, types et distributions de bruit. Ils représentent une ressource rare et complète que nous partageons librement pour explorer la problématique du bruit dans le traitement automatique du langage naturel. Cette problématique est paradoxalement peu traitée par la communauté scientifique bien qu'elle possède de nombreuses applications pratiques. Notre étude donne donc un aperçu des défis que pose le classement de textes dans des conditions bruitées et plaide en faveur du développement de modèles de classement plus robustes au bruit.

5.1 Introduction

Grâce aux récentes avancées technologiques et à la disponibilité croissante des plateformes numériques, une quantité importante de documents existe sous forme numérique. Les efforts de numérisation d'institutions du patrimoine culturel contribuent de plus en plus à la production de quantités massives de documents numérisés. Ces documents sont transcrits soit manuellement à l'aide d'efforts humains, soit automatiquement à l'aide de systèmes de reconnaissance optique de caractères (ROC) ou encore de reconnaissance automatique de la parole (RAP), en fonction de leur format. Par conséquent, leur contenu textuel inclut un bruit en raison d'erreurs de frappe humaines, d'erreurs de ROC/RAP ou encore d'autres facteurs. Or, l'accessibilité de ces documents est corrélée à la manière dont ils sont indexés. D'autant que ce sont précisément ces versions du contenu textuel contenant certains mots erronés qui sont utilisées lors de l'indexation par les moteurs de recherche. Ceci représente un problème sérieux pour l'indexation des documents et, par conséquent, pour la recherche de documents [GVB03, CDC+17].

Comme abordé dans le Chapitre 2, le classement est une tâche fondamentale pour la recherche d'information, avec un large éventail d'applications pratiques et industrielles. Le classement permet aux utilisateurs de trouver rapidement et efficacement les documents les plus pertinents à partir d'un large corpus de textes. Cependant, dans des scénarios réels, la performance des modèles de classement peut être significativement affectée par les différents types de bruit [TBCE94]. Il est donc essentiel de comprendre l'impact du bruit pour développer des modèles plus robustes aux erreurs.

Alors que des travaux antérieurs ont été menés pour évaluer l'impact des erreurs de ROC [TBC96], peu de travaux s'intéressent à des types de bruit moins spécifiques qui peuvent être introduits par d'autres sources d'erreur que les systèmes de ROC. C'est pourquoi, dans ce travail, nous étudions l'impact de différents types de bruit sur la tâche de classement de textes, en particulier les erreurs d'insertion, de suppression et de substitution, afin de mieux comprendre la robustesse des modèles de classement de textes dans des conditions bruitées. Nous analysons également l'impact de la distribution des erreurs dans les mots en utilisant deux distributions différentes du bruit dans le texte. Les résultats de ce travail peuvent contribuer au développement d'applications pratiques plus performantes, telles que les moteurs de recherche qui utilisent le contenu de sites web, eux-mêmes sujets à de nombreuses erreurs. À notre connaissance, il s'agit de la première étude qui explore l'impact des erreurs dans le texte sur la tâche de classement de textes avec une granularité de l'intensité de bruit étudié de manière aussi fine. En mettant à disposition nos jeux de données bruitées, nous espérons également contribuer au développement de modèles de traitement automatique du langage naturel capables de traiter efficacement des contenus textuels bruités.

Dans cette étude, nous nous intéressons aux performances de deux modèles populaires de classement de textes, BM25 [RJ76], une fonction de classement probabiliste introduite dans le Chapitre 2, et DistilBERT [SDCW20], un modèle d'apprentissage profond basé sur un transformeur. Nous utilisons diverses mesures pour évaluer les performances de ces modèles en fonction de différents niveaux, types et distributions de bruit.

Plus précisément, notre travail apporte trois contributions principales :
— une évaluation exhaustive et quantitative de l'impact des erreurs de caractère sur la performance de deux méthodes populaires de classement de textes ;
— une analyse de l'effet de différents types et distributions de bruit ;
— la création et le partage de plusieurs jeux de données avec différents niveaux et types de bruit, afin de soutenir les efforts de recherche visant à développer des modèles de classement plus robustes aux entrées textuelles bruitées.

Le reste du présent chapitre est structuré comme suit. La section 5.2 présente des travaux connexes sur le classement de textes et le traitement du bruit dans les textes. La section 5.3 décrit le protocole expérimental mené pour étudier l'impact du bruit et apporte des détails sur les aspects techniques. La section 5.4 présente la performance des modèles de classement de textes sous contraintes de bruit et interprète les résultats obtenus. Enfin, la section 5.5 conclut le document par une discussion autour de plusieurs problèmes en suspens et sur les orientations de recherche possibles à propos de ce sujet.

5.2 État de l'art

Impact des données bruitées sur la performance de tâches de traitement automatique du langage naturel De nombreux travaux de traitement automatique du langage naturel se sont concentrés sur le traitement de données bruitées pour la délimitation des phrases, la tokenisation et l'étiquetage de parties du discours [Lop05, Lop08]. D'autres travaux ont évalué les effets des textes bruités sur d'autres tâches telles que la catégorisation de textes [ILA95, ZMO+04], le résumé de documents [JLS03], la traduction automatique [Yas05], la reconnaissance d'entités nommées [HJCS+20], la liaison d'entités [LPHSD19] et la modélisation de sujets [MDOJ18]. Van Strien et al. [VSBA+20] ont, par exemple, démontré qu'une qualité amoindrie du texte dans les documents nuit aux performances de six tâches de traitement automatique du langage naturel, y compris la segmentation des phrases et l'analyse syntaxique des dépendances. Hamdi et al. [HPS+23] ont montré que les performances des systèmes de reconnaissance des entités nommées peuvent connaître une baisse significative du F1-score de 90% à 50% pour des taux d'erreur de caractères compris entre 2% et 30%.

Classement de textes La recherche d'information consiste à extraire des documents pertinents d'une vaste collection de documents en réponse à une requête de l'utilisateur. Ce domaine étant abordé plus en détails dans le Chapitre 2, nous rappelons ici des éléments strictement nécessaires à la bonne compréhension de ce chapitre. Les approches existantes pour le classement comprennent des modèles probabilistes traditionnels tels que BM25 [RJ76], qui reposent sur une représentation parcimonieuse des requêtes et des documents. Avec l'avènement de l'apprentissage profond, des modèles plus avancés tels que les modèles basés sur les transformeurs comme BERT et DistilBERT [DCLT19, SDCW20] ont été proposés. Ces modèles utilisent une représentation dense des documents et des requêtes et ont montré des performances de pointe dans diverses tâches de traitement automatique du langage naturel, y compris le classement

de textes. Dans ce travail, nous nous concentrons sur les modèles de représentation parcimonieuses et denses dans un contexte bruité.

Impact des données bruitées sur la recherche d'information De nombreux travaux se sont intéressés à la recherche d'information à partir de données bruitées [CHTB94]. L'impact des erreurs dans les documents est bien étudié dans certaines tâches de recherche [TVOH15]. Chiron et al. [CDC+17], par exemple, ont proposé une méthode pour estimer le risque que la requête d'un utilisateur ne corresponde pas aux documents bruités stockés dans les bibliothèques numériques. Taghva et al. [TBC96] ont mis en avant que des taux d'erreur ROC modérés n'ont pas un impact élevé sur l'efficacité des mesures classiques de recherche d'information. Un travail plus récent a montré que la recherche d'information peut être dégradée avec seulement 5% de taux d'erreur au niveau des caractères du texte [BLSVM20]. Il est intéressant de noter que Oliveira et al. [dOVA+23] ont démontré que, pour des taux d'erreur comparables, les documents longs sont plus affectés par les erreurs de ROC que les documents courts.

Notre travail s'inscrit dans le même esprit que celui de Bazzo et al. [BLSVM20] : nous évaluons quantitativement l'impact du bruit sur la tâche de classement de textes. Cependant, contrairement à eux, nous examinons plus en détail l'impact des différents types d'opérations génératrices de bruit (insertion, substitution, suppression) avec différents niveaux d'intensité et distributions d'erreur.

5.3 Méthodologie

Dans cette section, nous présentons la méthodologie de notre travail qui explore l'impact du bruit sur les modèles de classement de textes et plus précisément sur le classement de passages. Le classement de passages, traité en détail dans le Chapitre 2, est une tâche de recherche essentielle de la recherche d'information qui consiste à extraire d'un grand corpus de textes les passages pertinents pour une requête utilisateur donnée et à classer ces passages en fonction de leur pertinence.

Nous nous concentrons ici sur les défis liés au bruit, mais les jeux de données bruités sont une ressource rare et aucun de ceux disponibles dans la littérature ne répond à nos besoins. C'est pourquoi nous créons des jeux de données synthétiques simulant différents types de bruit. La création de nos propres jeux de données bruités nous permet d'avoir un contrôle précis de la distribution et de l'intensité du bruit considéré, ce qui est crucial pour une évaluation complète des modèles de classement en conditions bruitées.

En résumé, notre méthodologie est la suivante :

— nous travaillons sur le jeu de données MS MARCO Passage Ranking fourni par les organisateur de TREC 2020 (deep learning track) ;
— nous injectons du bruit de différents types et intensités dans les passages de ce corpus en utilisant la bibliothèque `nlpaug`, et obtenons ainsi 72 versions bruitées du jeu de données ;
— nous appliquons deux modèles de classement, BM25 et DistilBERT, pour classer les passages de ces jeux de données bruitées ;

— nous utilisons quatre mesures (NDCG, Recall@1000, MAP et MRR) pour évaluer la performance des deux modèles en environnement bruité.

5.3.1 Jeu de données

Nous menons notre étude sur le jeu de données MS MARCO Passage Ranking [BCC+18b] car il a gagné en popularité ces dernières années dans la communauté de la recherche d'information et fait figure de référence. Le jeu de données MS MARCO Passage Ranking est une collection comprenant un corpus de passages, des requêtes de test et des jugements de pertinence de test [Sto06]. Le jeu de données comprend un corpus de 8,8 millions de passages extraits des résultats du moteur de recherche Bing. La longueur moyenne de chaque passage est de 56 tokens (médiane : 50, max : 362). De plus amples détails sur la distribution des longueurs de passage sont disponibles dans [CMYC21].

Ce jeu de données est divisé en trois sous-ensembles : un ensemble d'entraînement, un ensemble de développement et un ensemble de test, chacun étant constitué de paires de requêtes utilisateur et de passages pertinents. Chaque requête comporte en moyenne un passage pertinent.

Pour nos expériences, nous utilisons l'ensemble MS-Marco-passagetest2020-top1000 [CMYC21]. Il s'agit d'un classement initial de 1000 passages par requête de test. Nous considérons l'union de ces passages candidats et nous nous concentrons sur la tâche de classement sur le corpus résultant. La raison de ce choix est la quantité importante d'injections de bruit et d'exécutions de classements à effectuer, ce qui nécessite un corpus plus petit que le corpus officiel complet de classement, mais suffisamment important pour être pertinent. Ce jeu de données contient environ 170k passages.

Il faut noter que la tâche que nous effectuons est différente de la tâche de classement complet où le corpus est beaucoup plus grand. Elle est également différente de la tâche de reclassement où seuls 1000 candidats par requête sont pris en compte pour une requête donnée, et non l'ensemble de tous les candidats.

5.3.2 Injection de bruit

L'injection de bruit dépend principalement de trois paramètres : le type de bruit, la quantité de bruit au niveau des caractères et la façon dont il est distribué dans le passage.

Pour mesurer la quantité de bruit injectée dans un texte, nous utilisons le taux d'erreur de caractère (ou CER pour Character Error Rate) qui est une mesure couramment utilisée pour évaluer la performance des systèmes de reconnaissance vocale ou de traduction automatique. Dans le contexte de notre étude, le CER compare, pour un texte dégradé donné, le nombre total de caractères, y compris les espaces, au nombre minimum d'insertions, de substitutions et de suppressions de caractères nécessaires pour obtenir le texte original. Pour un CER donné, la distribution des erreurs peut être uniforme dans les mots ou, au contraire, concentrée sur certains mots. Ainsi, nous utilisons également le taux d'erreur de mot (ou WER pour Word Error Rate) qui est défini de la même manière que le CER mais au niveau des mots. Pour un CER donné, il peut y avoir peu de mots fortement modifiés (d'où un WER faible) ou beaucoup de mots faiblement modifiés (d'où

un WER élevé). Notons que dans notre étude, lorsqu'il est utilisé pour mesurer le bruit dans un jeu de données particulier, le terme CER (respectivement WER) se réfère au CER (respectivement WER) moyen sur l'ensemble des passages du corpus.

Nous nous concentrons sur trois types de bruit au niveau des caractères - l'insertion, la suppression et la substitution - qui sont couramment rencontrés dans des scénarios applicatifs réels et dont il a été démontré qu'ils ont un impact significatif sur la performance des modèles de traitement automatique du langage naturel [ASLS21]. En nous concentrant sur ces types de bruit, nous pouvons explorer n'importe quelle *nature* de bruit, par exemple le bruit aléatoire, le bruit de ROC ou les erreurs de frappe sur clavier, indépendamment de la source. En effet, toute nature de bruit peut être reconstruite comme une combinaison d'insertions, de suppressions et de substitutions au niveau des caractères. Par conséquent, le fait de se concentrer sur le type de bruit plutôt que sur sa nature nous permet d'étudier l'impact du bruit d'une manière plus complète et générique, sans être lié à un scénario spécifique.

En suivant cette idée, nous injectons des erreurs aléatoires au niveau des caractères dans le texte : l'insertion correspond à l'injection aléatoire d'un nouveau caractère, la suppression à la suppression aléatoire d'un caractère et la substitution au remplacement d'un caractère (dans un mot) par un autre caractère aléatoire. Comme indiqué ci-dessus, bien que des injections plus réalistes puissent être envisagées, par exemple en injectant des erreurs de frappe sur clavier (un caractère sera plus probablement remplacé par un caractère proche sur un clavier), ou des erreurs liées à la ROC (un caractère sera remplacé par un caractère de forme similaire), nous soutenons que, dans le contexte de cette étude, la modification de la nature des erreurs injectées ne ferait pas de différence significative, étant donné que nous nous concentrons sur l'impact des différentes intensités et distributions de bruit dans un texte donné, plutôt que de nous attaquer à des scénarios applicatifs réels spécifiques. Le choix de travailler avec des erreurs injectées de manière aléatoire est donc naturel puisqu'il s'agit de la manière la plus générique de créer du bruit.

Nous injectons chacun des trois types de bruit avec des intensités différentes en faisant varier le CER de 0 à 36% avec des intervalles de 3%, comme décrit dans la Figure 5.2. Pour chaque valeur de CER, nous étudions également deux régimes différents de distribution des erreurs : l'un où les erreurs sont réparties sur quelques mots dans le texte, mais où les mots affectés sont sévèrement modifiés (faible WER), que nous appelons *Batch 1*, et l'autre où les erreurs sont plus uniformément réparties entre les mots (WER plus élevé) que nous appelons *Batch 2*. Dans le Tableau 5.1, nous montrons un exemple de passage et ses versions dégradées pour les batch 1 et 2 et pour les trois types de bruit. La Figure 5.1 présente les caractéristiques de chaque jeu de données bruitées que nous avons produit.

Nous pouvons constater que le WER et le CER sont assez similaires pour les trois types de bruit. Nous notons également que le WER mesuré dans le *Batch 2* est environ deux fois plus élevé que dans le *Batch 1*, à CER mesuré égal. Ceci est cohérent avec le fait que, dans le *Batch 2*, nous injectons du bruit qui est plus uniforme et qui a donc un fort impact sur le WER, à CER égal. Pour l'insertion et la substitution, le WER du *Batch 2* peut même dépasser 100% (il apparaît lorsque le nombre de modifications dépasse le

nombre total de mots), même si le CER est maintenu en dessous de 40%.

TABLE 5.1 – Exemple d'injection de bruit avec un CER ($\simeq 10-15\%$) pour les Batch 1 et 2

Passage original		The definition of expressed is to convey by words, gestures or conduct. You can learn more about the definition of the word expressed at the Dictionary website.
Batch 1	**insertion**	The definition of expressed is to convey by nw9oKrrdNs, gestures or kcGogn%d8uBcFt. #YRoTu can learn more about the definition of the word expressed at the tDmiGc%tJidofnGaQrfy website.
	suppression	The definition of expressed to convey by words, gestures or conduct. You can learn more about definition of word expressed at the Dictionary.
	substitution	The O3_C_WS(M3 of expressed is to convey by words, gestures or conduct. You can learn more about the definition &D 5c& %E5y expressed at 7kA Dictionary website.
Batch 2	**insertion**	The d0ef(initsion of expressed is Hto convey b_y owoxrds, gestures or cFonducst. YJou cZan lxearjn more about tMhe definition #of the word expressed aKt thZe Dictionary website.
	suppression	Th definition of expreed is to onve by wors, gestures or coduc. You cn lern ore abou th definition o the wrd exprssd a he Dictionary wsite
	substitution	Jhe definition Gf expressed ik to c#Qvey by 6ords, gestures or connucg R You Can leaEn more abeut the dQfinmtion of whe word expressed at thO Dictio$ar6 wessiXe.

Pour résumer, nous injectons du bruit avec trois types de bruit, deux distributions différentes et une intensité de CER croissante en prenant 12 valeurs cibles différentes. Au total, nous générons donc 72 versions bruitées (2 distributions x 3 types de bruit x 12 CER cibles) de la collection de tests MS MARCO Passage Ranking. Les données produites et utilisées dans ce travail sont disponibles en ligne [1].

1. https://www.dropbox.com/scl/fo/7mvun4nh3et3ak6bbcrz1/h?dl=0&rlkey=tgw7lrt5kglrdvehpu461ghq3

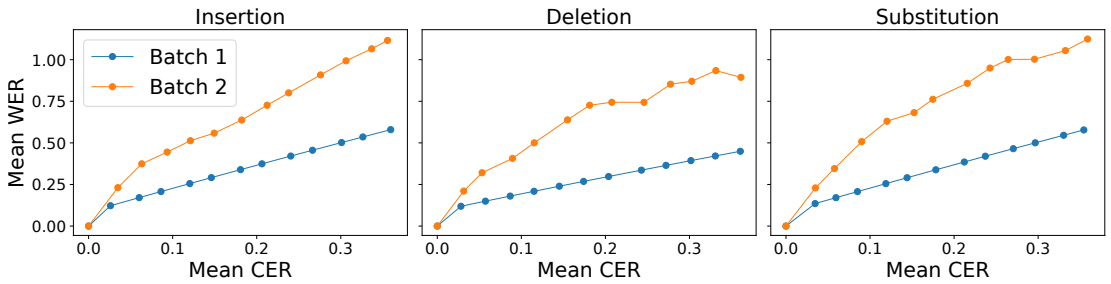

FIGURE 5.1 – Distribution du bruit dans les Batch 1 et 2 décrivant la proportion de WER en fonction du CER des jeux de données bruités. Chaque point de la figure correspond à un jeu de données bruitées, à l'exception du point 0 (CER et WER) qui correspond au corpus original.

5.3.3 Outil d'injection de bruit

Pour simuler les types de bruit choisis dans notre étude, nous utilisons la bibliothèque `nlpaug` [Ma19], une bibliothèque d'augmentation de traitement automatique du langage naturel populaire et en libre accès qui fournit diverses fonctions pour l'augmentation des données. Cette bibliothèque offre un large éventail de techniques d'augmentation qui peuvent être facilement intégrées dans notre chaîne de traitement expérimentale.

Cependant, `nlpaug` n'a pas été spécifiquement conçu pour l'injection de bruit dans le texte et peut donc présenter certaines limites en termes de précision. Par exemple, nous avons constaté que l'obtention d'un niveau de bruit souhaité à l'aide de `nlpaug` était souvent difficile et entraînait des imprécisions dans le CER et le WER obtenus, par rapport à la cible. Pour résoudre ce problème, nous tolérons une légère marge d'erreur en prenant le CER moyen du corpus comme point de référence. Plus précisément, pour chaque jeu de données bruitées, nous générons un bruit dont le CER moyen est relativement proche du CER cible, de sorte que la différence absolue entre les deux valeurs ne dépasse pas 0,75%. Par exemple, si nous souhaitons générer un jeu de données bruitées avec un CER cible de 12%, `nlpaug` pourrait générer un CER réel de 11,8%. Dans ce cas, nous conservons le jeu de données bruitées puisque nous tolérons toute valeur comprise entre 11,25% et 12,75%. Les jeux de données bruitées ont été obtenus à l'aide d'un recherche par quadrillage (« *grid-search* ») sur les paramètres de `nlpaug` et seuls les jeux de données les plus pertinents, en termes de CER, ont été conservés. Nous illustrons cette marge dans la Figure 5.2. Il convient de noter que les courbes des Figures 5.1 et 5.3 ainsi que toutes les discussions ultérieures dans ce chapitre se réfèrent toujours au CER réel des jeux de données bruités et non au CER cible.

Malgré ces limites, nous pensons que `nlpaug` est un choix approprié pour notre étude car il fournit un moyen pratique et efficace de simuler différents types de bruit et nous permet d'obtenir des niveaux significatifs d'intensité du bruit dans nos jeux de données. En outre, il a été largement utilisé dans des études antérieures [AK20, Nov21] et a démontré son efficacité dans l'amélioration de la performance des modèles de traitement

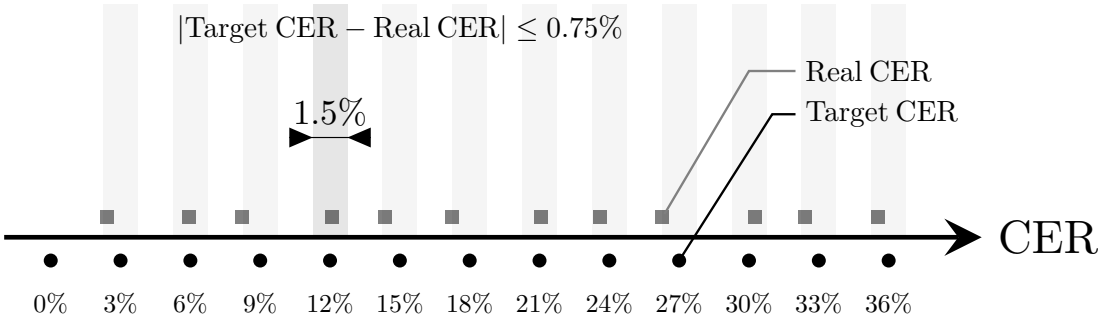

FIGURE 5.2 – Schéma montrant le CER moyen obtenu pour le corpus bruité par rapport aux valeurs de CER initialement visées. Les intervalles de tolérance sont représentés par des bandes grises et blanches.

automatique du langage naturel dans certains contextes [DS].

5.3.4 Modèles de classement

L'objectif de cette étude est d'évaluer les performances des modèles de classement de textes sur les jeux de données bruitées que nous avons créés. Nous utilisons deux modèles classiques, à savoir BM25 et DistilBERT pour la tâche de classement de passages. Nous utilisons la bibliothèque `Pyserini` pour réaliser nos expériences [LML+21].

Pour la bonne compréhension du lecteur, nous rappelons quelques caractéristiques de ces modèles qui ont été traités dans le Chapitre 2. BM25 est un modèle de représentation parcimonieuse qui calcule un score basé sur la fréquence des termes de la requête utilisateur dans le document, ainsi que sur leur fréquence inverse dans le document. À l'opposé, DistilBERT est un modèle de langage récent qui utilise une approche de représentation dense basée sur des réseaux neuronaux profonds. DistilBERT est un modèle pré-entraîné qui peut être affiné pour diverses tâches de traitement automatique du langage naturel, y compris le classement des passages. Précisément, il s'agit d'une version distillée [HVD15] de BERT [DCLT19], conservant 97% de performance mais étant 60% plus rapide en utilisant seulement la moitié des paramètres de BERT. Contrairement à BM25, DistilBERT génère des représentations denses qui encodent des informations sémantiques et syntaxiques pouvant capturer des relations complexes entre les mots et les phrases.

Les représentations parcimonieuses et denses divergent dans la manière dont elles encodent les requêtes et les documents. Comme leur nom l'indique, les modèles de représentation parcimonieuse (respectivement dense) encodent les documents sous forme de vecteurs creux (respectivement denses). En comparant ces deux types de modèles dans un contexte bruité, nous pouvons nous faire une idée de leurs forces et faiblesses, et de la manière dont ils se comportent en fonction du niveau, du type et de la distribution du bruit. Cette comparaison est particulièrement intéressante car les deux modèles utilisent des approches fondamentalement différentes pour représenter le texte. De plus, nous pensons que BM25 et DistilBERT sont représentatifs de leur famille de modèles respective

et nous nous attendons à des résultats similaires si nous menons des expériences avec d'autres modèles des mêmes familles.

5.3.5 Évaluation en environnement bruité

Pour évaluer les performances des modèles de classement de passages en environnement bruité, nous nous concentrons sur plusieurs mesures couramment utilisées dans la recherche d'information. Ces mesures évaluent la pertinence des passages retrouvés sur la base de jugements fournis par des évaluateurs humains. Nous utilisons le gain cumulatif actualisé normalisé (ou NDCG pour « Normalized Discounted Cumulative Gain ») [JK02], le rappel (plus précisément le Recall@1000), la précision moyenne (ou MAP pour « Mean Average Precision ») [MRS08] et le rang réciproque moyen (ou MRR pour « Mean Reciprocal Rank ») [Cra09]. Le NDCG mesure l'efficacité d'un modèle de classement en tenant compte de la pertinence des documents retrouvés à différentes positions du classement renvoyé. Le Recall@1000, quant à lui, mesure la proportion de documents pertinents retrouvés par un modèle parmi les 1000 premiers documents. Le MAP mesure la moyenne de la précision moyenne d'un modèle sur l'ensemble des requêtes. Enfin, le MRR est la moyenne sur toute les requêtes, de l'inverse du rang du premier document pertinent retrouvé par un modèle de classement.

Ces mesures fournissent des informations instructives et exhaustives sur les performances des modèles de classement de passages en environnement bruité. Nous avons mis en œuvre ces mesures avec la procédure d'évaluation officielle mise à disposition par TREC 2020 deep learning track avec les étiquettes de NIST [CMYC21]. Notons également que nous nous assurons que notre mise en œuvre de BM25 et DistilBERT indexée sur le corpus complet de classement de passages MS MARCO obtient les mêmes performances que dans la matrice de reproduction de Pyserini `Pyserini two-click reproduction matrix` [2].

5.4 Résultats et discussion

Nous appliquons BM25 et DistilBERT sur les jeux de données bruitées que nous avons générés. Les résultats pour toutes les combinaisons de jeux de données et de batch (Batch 1 et Batch 2) sont présentés dans la Figure 5.3. Comme prévu, les performances de classement diminuent considérablement à mesure que le niveau de CER augmente. Néanmoins, nous observons parfois une légère amélioration. Ce phénomène peut être dû à des perturbations dans les mesures de performance et souligne que des études de variance devraient idéalement être réalisées.

Nos résultats indiquent que la quantité de bruit injectée dans le texte a un impact sur toutes les mesures utilisées pour évaluer la performance des modèles de classement. Nous pouvons remarquer que Recall@1000 et MRR ne sont que légèrement affectés par le bruit dans Batch 1. Cependant, ces mesures sont nettement plus dégradées dans Batch 2.

2. `https://castorini.github.io/pyserini/2cr/msmarco-v1-passage.html`

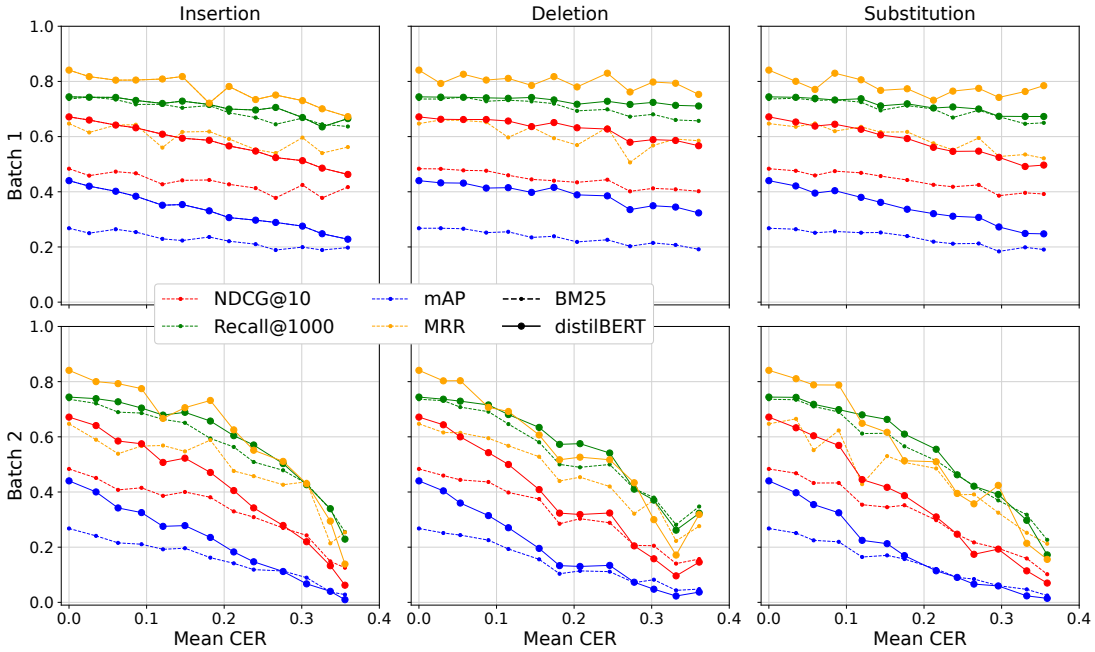

FIGURE 5.3 – Performance de BM25 et DistilBERT avec une quantité croissante de CER injecté dans le jeu de données de test MS MARCO Passage Ranking. Les différentes méthodes de classement sont représentées par deux types de marqueurs et deux styles de lignes différents, les différentes mesures étant représentées par des couleurs différentes.

Nous avons constaté que les trois types de bruit ont un effet similaire sur les performances des modèles de classement, bien qu'il y ait quelques différences observées en fonction de la distribution des erreurs dans le texte. En particulier, l'insertion est légèrement plus préjudiciable que les autres types de bruit dans Batch 1. Paradoxalement, l'insertion est moins préjudiciable aux performances de classement dans Batch 2. Nous estimons que ces observations ne sont pas significatives en raison de la très faible différence entre les trois types de bruit en termes de performance.

Par conséquent, nous considérons que BM25 et DistilBERT se comportent de manière comparable en présence d'erreurs d'insertion, de suppression ou de substitution au niveau des caractères, ce qui suggère que le type de bruit n'a pas d'impact significatif sur les modèles de classement évalués. C'est plutôt la quantité globale et la distribution du bruit qui affectent les performances.

La comparaison des performances de classement entre les Batch 1 et 2 nous permet de tirer des conclusions générales sur la sensibilité des modèles de classement de textes à la distribution du bruit dans le texte. Les différences observées entre les Batch 1 et 2 indiquent que les performances de classement de textes en environnement bruité sont fortement influencées par la distribution des erreurs au niveau des caractères dans les mots du texte. Plus précisément, nous observons que lorsque les erreurs sont fortement concentrées sur quelques mots, l'impact sur les performances de recherche est moins prononcé que lorsque les erreurs sont uniformément réparties sur plusieurs mots. Cette constatation est particulièrement pertinente car, à notre connaissance, c'est la première fois qu'une étude explore l'impact de la distribution du bruit sur les performances de classement. Nous pensons que cela est dû au processus de tokenisation dont dépendent les deux méthodes : un seul mot fortement dégradé dans un passage aurait un faible effet sur le vocabulaire, tandis que de nombreux mots légèrement dégradés auraient un impact important, en créant une grande quantité de tokens bruités dans le vocabulaire. Par conséquent, le principal facteur de dégradation des performances semble être le WER.

La dépendance de BM25 et de DistilBERT vis-à-vis de la tokenisation expliquerait pourquoi les deux modèles montrent un comportement similaire en réponse au bruit injecté malgré le fait qu'ils utilisent des approches différentes pour le classement de textes. En fait, dans Batch 2, lorsque le CER moyen dépasse 20%, les deux méthodes de classement ont des performances comparables, quels que soient la quantité, le type et la distribution des erreurs. Il est intéressant de noter que BM25 et DistilBERT ont un Recall@1000 très proche. De plus, dans le Batch 2, les performances en termes de MAP et de NDCG des modèles se rapprochent au fur et à mesure que le CER augmente et dans certains cas, BM25 est même plus performant que DistilBERT. Cela tend à montrer que DistilBERT est plus fortement affecté par des niveaux élevés de bruit que BM25. Ces observations sont instructives car elles soulignent que les modèles de représentation dense ne présentent pas de robustesse particulière au bruit par rapport aux modèles de représentation parcimonieuse. Par conséquent, le développement de modèles adaptés à la présence de bruit dans le texte est nécessaire pour atténuer l'impact négatif sur les performances de classement.

5.5 Conclusion

Dans ce travail, nous avons exploré l'impact du bruit sur le classement de textes en utilisant deux modèles bien connus, BM25 et DistilBERT, sur le jeu de données de test MS MARCO Passage Ranking. Nous avons injecté du bruit dans ce jeu de données avec trois types d'erreurs au niveau des caractères. Nous avons défini deux régimes d'injection de bruit, l'un avec un faible WER et un nombre élevé de caractères bruités par mot (Batch 1), et l'autre avec un WER élevé et un faible nombre de caractères bruités par mot (Batch 2).

Nous avons mené une évaluation complète des modèles de classement de passages à l'aide de plusieurs mesures : NDCG, Recall@1000, MRR et MAP.

Nos résultats indiquent que, comme prévu, la performance de classement diminue considérablement avec le niveau de CER. Nous observons que les performances de BM25 et de DistilBERT sont affectées de manière similaire par différents types de bruit, ce qui suggère que ni les modèles à représentation parcimonieuse ni les modèles à représentation dense ne sont particulièrement adaptés au classement de textes en environnement bruité. Nous pensons que ces observations sont dues au processus de tokenisation de ces types de modèles. En outre, nos résultats montrent que les performances de classement en environnement bruité dépendent de la distribution des erreurs au niveau des caractères dans les mots du texte, ce qui est bien représenté par le WER associé à notre niveau de CER : pour une valeur de CER donnée, plus le WER est élevé, plus les performances de classement seront faibles.

De plus, en mettant à disposition les jeux de données bruitées produits pour cette étude, nous pensons que notre travail sera utile aux chercheurs et aux praticiens travaillant sur des tâches de traitement automatique du langage naturel qui impliquent des données d'entrée bruitées.

Dans le but d'encourager de futures investigations sur ce sujet, nous abordons à présent de manière succincte des directions qui méritent d'être explorées.

Des travaux futurs pourraient explorer l'impact du mélange de différents types de bruit sur le classement de textes, ou étudier d'autres natures de bruit telles que celles liées à la reconnaissance optique de caractères, à la reconnaissance automatique de la parole, ou encore aux erreurs de frappe sur clavier. En outre, il serait intéressant d'étudier l'impact de la longueur des documents sur les performances du classement en environnement bruité, étant donné que le jeu de données que nous avons utilisé contient des documents textuels relativement courts. Enfin, nos résultats suggèrent que ni les modèles à représentation dense ni les modèles à représentation parcimonieuse ne sont supérieurs à l'autre dans le contexte du classement dans des conditions bruitées. Les modèles à représentation dense, bien que plus performants pour une quantité raisonnable de CER, subissent une dégradation plus importante quand le CER augmente. Des recherches supplémentaires sont donc nécessaires pour déterminer quel type de représentation est le plus efficace pour traiter les textes contenant du bruit. Une piste intéressante pour étudier cette question serait d'explorer le comportement de différents tokeniseurs en contexte bruité.

Chapitre 6

Représentation spécifique à la requête pour le classement de document

Sommaire

6.1 Introduction 105
6.2 Travaux connexes 106
6.3 Méthode de travail 108
6.4 Construction de sous-graphes spécifiques aux requêtes 111
6.5 Apprentissage de plongements spécifiques aux requêtes . . . 113
6.5.1 Conditionnement à la requête 113
6.5.2 Mise à jour hétérogène des entités 114
6.5.3 Mise à jour hétérogène des documents 115
6.5.4 Apprentissage de classements sur les plongements de documents 115
6.5.5 Inférence 117
6.6 Conclusion 117

En vue de clôturer nos travaux, nous proposons ici un chapitre de perspectives basé sur un projet de recherche en partenariat avec l'Indian Institute of Technology Bhilai. Ce projet est en cours et ses résultats ne sont pas encore disponibles. Une fois les résultats obtenus, le contenu de ce chapitre sera retranscrit dans un article pour être soumis à une conférence internationale.

Comme discuté dans le Chapitre 2, les méthodes usuelles de classement de textes s'appuient fortement sur l'appariement lexical et les modèles statistiques, qui ne parviennent souvent pas à saisir la véritable pertinence des textes par rapport aux requêtes en raison

du manque de compréhension du contexte. Par ailleurs, l'étude menée dans le Chapitre 5 nous a montré le comportement similaire de ces méthodes en présence de bruit dans les textes, démontrant que les méthodes traditionnelles comme les plus récentes ne sont pas adaptées au traitement du texte en environnement bruité.

En perspective des travaux menés jusqu'ici et des résultats encourageants obtenus au Chapitre 4 pour la tâche de liaison d'entités nommées (« Entity Linking »), nous proposons dans ce travail une nouvelle méthode de classement de textes basée sur l'utilisation de graphes. Cette approche est générique et exploite l'information à disposition dans une base de connaissance quelconque. L'intuition derrière notre travail est de tirer parti d'une représentation des entités, dans le graphe de connaissances, qui soit spécifique à la requête afin d'améliorer la qualité du classement de textes. Pour faciliter l'utilisation de cette méthode dans le contexte des travaux menés au chapitre précédent, nous nous restreignons à la tâche de classement de passages mais notre méthode reste valable pour le classement de textes en général. Notre méthode utilise des techniques de traitement automatique du langage naturel de pointe pour extraire et représenter les entités à partir du texte dans la requête et les passages potentiellement pertinents. En nous concentrant sur les entités importantes mentionnées dans la requête et le passage, nous visons à capturer les relations sémantiques et la pertinence thématique entre les requêtes et les passages de manière plus précise. La méthode développée dans ce chapitre est inspirée des travaux de [SDZ+18b] qui étudient la tâche de réponse aux questions définie dans le chapitre 1.

6.1 Introduction

Dans le domaine de la recherche d'information, il est primordial de parvenir à un classement précis et contextuellement pertinent des passages en réponse aux requêtes des utilisateurs. Le domaine du classement de documents a connu un changement de paradigme pour aller vers des modèles de réseaux neuronaux profonds. Dans ces circonstances, l'exploitation des informations contextuelles présentes dans les graphes de connaissances (KG pour « Knowledge Graph ») s'avère être une stratégie prometteuse pour comprendre l'intention de recherche de l'utilisateur. En effet, les graphes de connaissances, qui sont des graphes hétérogènes, se caractérisent par leur capacité à représenter diverses relations par le biais de plusieurs types d'arêtes et à saisir les subtilités structurelles par le biais de représentations spécifiques aux graphes.

Ces dernières années, diverses méthodes [CVD21, YZL+22] ont été conçues pour exploiter ces informations contextuelles, allant d'approches heuristiques à des modèles récents basés sur des réseaux neuronaux. Indépendamment de ces grandes familles de méthodes, la plupart des modèles existants utilisent une représentation sémantique générale basée sur des descriptions textuelles statiques d'entités [XPC17] disponibles dans les KG. Néanmoins, des études antérieures ont montré que les descriptions textuelles statiques d'entités présentaient une utilité limitée dans l'amélioration de la précision du classement par le biais d'expériences sur la collection ClueWeb12 [DSP14]. De même, les plongements d'entités obtenus par des techniques régulières de plongements de graphes [BUGD+13, LLS+15, RH20, WZFC14] contiennent des connaissances sémantiques générales mais manquent de flexibilité pour s'adapter à des contextes d'interrogation spécifiques. Malgré les récentes tentatives d'intégration des informations issues des graphes de connaissances dans des modèles de langage avancés tels que ERNIE [ZHL+19] et E-BERT [PWS20], ces derniers continuent de s'appuyer sur des descriptions textuelles statiques des entités. La Figure 6.1 illustre ce problème avec un exemple montrant deux requêtes différentes pour lesquelles la même description de l'entité « Espagne » est utilisée. Si la description de l'entité peut être utile pour la requête A, ce n'est pas le cas pour la requête B.

À la lumière de ces considérations, ce chapitre propose une nouvelle approche pour améliorer le classement de documents. Nous proposons d'utiliser des représentations d'entités spécifiques aux requêtes afin de combler le fossé entre les plongements statiques des entités et l'intention des requêtes des utilisateurs. En intégrant des représentations d'entités contextualisées dans le processus d'apprentissage des modèles de classement des documents, nous visons à améliorer la compréhension du contexte et l'évaluation de la pertinence des documents par rapport aux requêtes de l'utilisateur.

Dans les sections suivantes, nous développons cette approche et la méthodologie employée. La section 6.2 décrit un état de l'art succinct des domaines scientifiques abordés dans nos travaux. La section 6.3 précise notre cadre de travail et les hypothèses utilisées pour notre méthode. La section 6.4 détaille les étapes de construction d'un graphe spécifique qui est à la base de notre méthode. La section 6.5 quant à elle développe le processus d'apprentissage de représentations de notre modèle QuSpec. Enfin, la section 6.6 discute qualitativement l'intérêt de notre méthode, ses avantages et inconvénients.

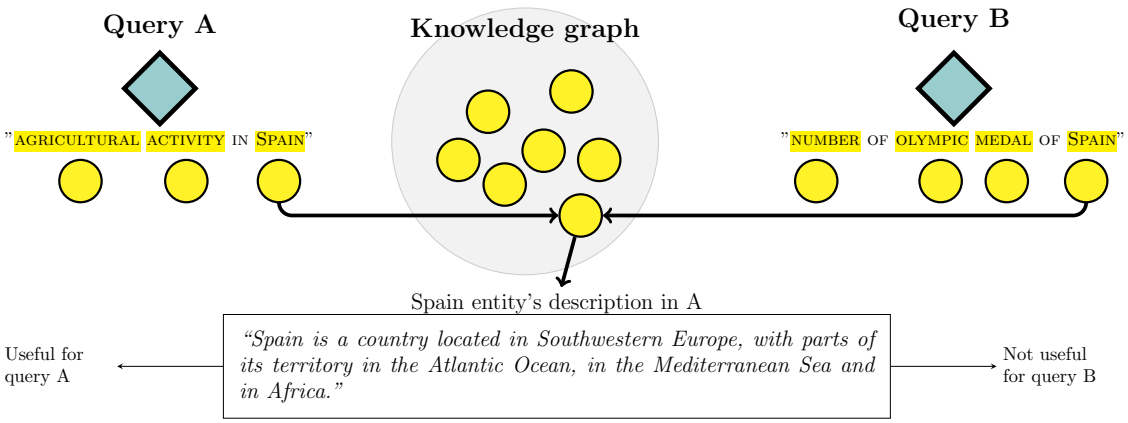

FIGURE 6.1 – Représentation schématique de la limite des plongements statiques d'entités lors de l'utilisation d'informations provenant d'un graphe de connaissances. Ici, les deux requêtes A et B mentionnent l'entité « Espagne » (« Spain ») pour laquelle le graphe de connaissances contient une description unique. La description d'« Espagne » est utile pour répondre à la requête A, car elle donne des informations sur la géographie de l'« Espagne » et donc sur une éventuelle activité agricole dans ce pays. En revanche, la même description d'« Espagne » n'est pas informative pour la requête B, qui concerne les médailles olympiques obtenues par l'Espagne.

Elle apporte également des informations sur l'évaluation expérimentale nécessaire pour valider l'efficacité de la méthode proposée.

6.2 Travaux connexes

Notre travail s'inscrit dans plusieurs domaines de recherche que nous présentons dans cette section. Il s'agit du classement de documents et de l'apprentissage de classements, qui sont abordés plus en détails dans le Chapitre 2 pour lesquels nous rappelons des éléments strictement nécessaires à la bonne compréhension de ce chapitre, et l'apprentissage de plongements de graphes dont le cas biparti est traité par le Chapitre 3.

Classement de documents Ce domaine, traité plus en détails dans le Chapitre 2, consiste à extraire des documents pertinents d'une vaste collection de documents en réponse à une requête de l'utilisateur. Les documents récupérés sont généralement classés en fonction d'une notion de pertinence. Le classement est utilisé dans de nombreuses applications réelles, telles que les moteurs de recherche, les systèmes de recommandation et les systèmes de recherche documentaire [GAB12, Pen22]. Les approches existantes pour le classement comprennent des modèles probabilistes traditionnels tels que BM25 [RJ76], qui reposent sur une représentation parcimonieuse des requêtes et des documents. Avec

l'avènement de l'apprentissage profond, des modèles plus avancés tels que les modèles basés sur des transformeurs comme BERT et DPR [DCLT19, KOM+20] ont été proposés. Ces modèles utilisent des représentations denses des requêtes et des documents et ont montré des performances de pointe dans diverses tâches de traitement automatique du langage naturel, y compris le classement de documents. Des travaux plus récents ont également exploré l'utilisation de grands modèles de langage pré-entraînés (nommés LLMs pour « Large Language Models » dans la littérature) tels que T5 et UniLM pour le classement de documents, ce qui a entraîné des améliorations significatives des performances [RSR+20, DYW+19].

Apprentissage de classements L'apprentissage de classements se réfère à une approche d'apprentissage automatique utilisée pour améliorer le classement des documents en réponse à la requête d'un utilisateur. Puisque nous le traitons plus en détails dans le Chapitre 2, nous rappelons ici des éléments nécessaires pour la bonne compréhension de ce chapitre. L'objectif est de développer un modèle de classement capable d'apprendre automatiquement à trier une liste de documents de manière à mieux répondre aux besoins d'information de l'utilisateur, c'est-à-dire que l'ordre de classement représente la pertinence relative des documents par rapport à la requête formulée par l'utilisateur. Les méthodes d'apprentissage de classements sont généralement regroupées en trois catégories : par points, par paires et par listes.

- *Approches par points :* La tâche d'apprentissage est formulée comme un problème de classification ou de régression où chaque paire requête-document est traitée comme une instance indépendante, de sorte que la position d'un document dans le classement final n'est pas prise en compte dans la fonction de perte. Étant donné la généralité de cette méthode, la plupart des algorithmes classiques de classification et de régression peuvent être mis en œuvre, y compris la régression logistique [Gey94b], les arbres de régression [KWPDG01], les réseaux neuronaux convolutionnels [SM15] et les modèles de transformeur séquence à séquence tels que T5 [NJPL20].
- *Approches par paires :* Ces approches considèrent explicitement l'ordre relatif des documents dans une liste classée. Elles formulent le problème du classement comme une tâche de classification binaire dans laquelle les documents sont présentés par paire et une caractéristique de la paire indique lequel des deux documents est préféré par l'utilisateur. Parmi les exemples, citons RankNet [BSR+05] et RankT5 [ZQJ+23]
- *Approches par listes :* Pour une requête donnée, ces approches considèrent la liste complète des documents classés comme une entité unique. Par rapport aux autres approches, celles-ci visent à optimiser directement les mesures de performance du classement. Parmi les exemples, citons RankGP [YLKY07] et SoftRank [TGRM08].

Apprentissage de plongements de graphes Ces dernières années, l'application des réseaux neuronaux de graphes (GNN pour « Graph Neural Network ») aux données rela-

tionnelles a généré des résultats remarquables. Par essence, un GNN peut être conceptualisé comme l'utilisation de la structure du graphe d'entrée en tant que graphe de calcul pour effectuer le passage de messages. Ce processus implique l'agrégation des informations du voisinage local des nœuds du graphe pour obtenir une représentation contextuelle plus nuancée.

Le passage de messages suit deux étapes clés :

— les représentations des nœuds sont d'abord initialisées ;
— ensuite, une fonction de mise à jour est appliquée de manière itérative L fois pour modifier la représentation du nœud tout en propageant les informations provenant des voisins du nœud.

Ici, L représente le nombre de couches intégrées dans le modèle et correspond à la longueur maximale des chemins le long desquels l'information est propagée dans le graphe. Une fois le processus de propagation d'information terminé, les représentations des couches finales, désignées par h_v, sont utilisées pour accomplir la tâche souhaitée, telle que la prédiction des liens dans les graphes de connaissances.

Dans ce cadre, plusieurs architectures de GNN ont été proposées. [KW17] introduit un réseau convolutionnel pour graphe (GCN pour « Graph Convolutional Network ») qui moyenne les voisins de distance 1 de chaque nœud dans le graphe, puis applique une projection linéaire suivie d'opérations non linéaires d'activation. [HYL18a] propose GraphSAGE, qui étend l'opération d'agrégation du GCN au-delà de la moyenne pour inclure des fonctions comme la somme ou le maximum. [VCC+18] présente le réseau d'attention de graphe (GAT pour « Graph Attention Network »), qui incorpore un mécanisme d'attention dans les GNNs. Cela permet à GAT d'attribuer des niveaux variables d'importance aux nœuds d'un même voisinage, améliorant ainsi son expression.

6.3 Méthode de travail

Nous nous intéressons au classement de passages qui est une tâche de recherche essentielle de la recherche d'information. Cette tâche est identique au classement de documents mais s'applique à des passages, qui sont des segments de texte plus courts, de l'ordre de quelques paragraphes, extraits de documents (pages web, documents textuels etc.). Les défis associés au classement de passages ne lui sont pas propres, et s'appliquent également au classement de documents.

Dans le contexte de ce chapitre, un graphe de connaissances est représenté par $G = (V, E, R)$, où V représente la collection d'entités contenues dans le graphe de connaissances. Les arêtes, désignées par E, sont composées de faits, qui sont des triplets de la forme (s, r, o), indiquant qu'une relation désignée par $r \in R$ existe entre l'entité sujet, $s \in V$, et l'entité objet, $o \in V$.

En parallèle, un corpus de textes, appelé D, est défini comme un ensemble de documents $d_1, ..., d_{|D|}$. Chaque document de ce corpus est représenté par une séquence de mots $d_i = (w_1^i, ..., w_{|d_i|}^i)$. On suppose, en outre, qu'un système de liaison d'entités [LMI+20, CIRP21] a été appliqué à cette collection de documents. Le résultat de

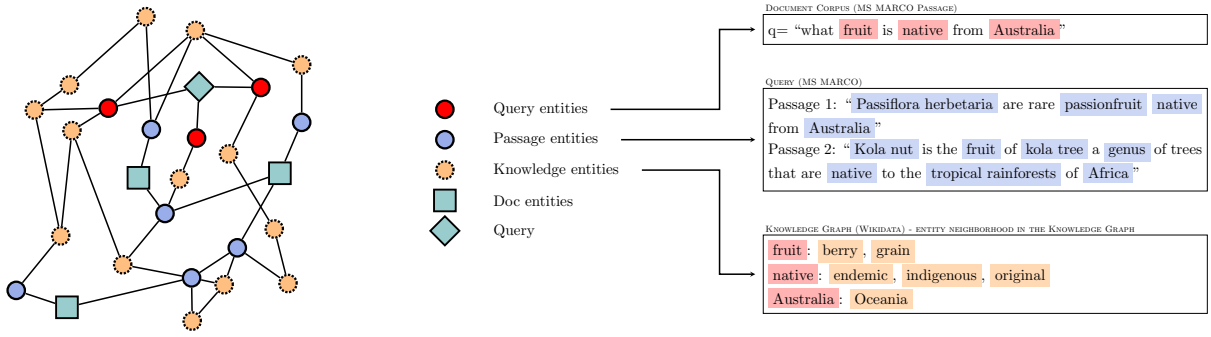

FIGURE 6.2 – Résultat du processus de construction du sous-graphe. Le sous-graphe spécifique à une requête G_q pour une requête q contient 2 types de nœuds : des entités (par exemple des entités du graphe de connaissances global G) et des documents textuels (par exemple des requêtes et des passages du corpus D). Ces deux types de nœuds sont sélectionnés à l'aide d'une méthodologie améliorant la pertinence par rapport à la requête des entités et des documents textuels composant le sous-graphe. On peut constater que les entités du sous-graphe spécifique à la requête sont de trois types : elles peuvent provenir de la requête initiale, des documents jugés pertinents pour la requête (extraits à partir d'un classement initial établi via une méthode parcimonieuse telle que BM25) ou elles peuvent être des entités provenant de chemins du graphe de connaissances G, reliées à des entités dans les documents pertinents pour la requête.

ce processus de liaison d'entités est représenté par un ensemble L, comprenant des liens (v, d_p^i) qui établissent des connexions entre une entité $v \in V$ et un mot spécifique situé à la position p dans le document d_i. Pour rendre compte succinctement de tous les liens entre entités dans un document d_i particulier, nous utilisons la notation L_{d_i}. Dans le cas où les mentions d'entités couvrent plusieurs mots dans le document d_i, notre notation inclut les liens reliant tous les mots couverts par la mention dans L. Considérons par exemple le passage $d =$ « It is known that Cristiano Ronaldo Santos eats apple. » contenant les entités « Cristiano Ronaldo Santos » et « apple » dont les mots couvrent respectivement les positions $(5, 6, 7)$ et (9). Alors chacun des mots $(5, 6, 7)$ est lié à l'entité correspondant identifiée par le code 457 dans le graphe de connaissances. Le mot (9) est, quant à lui, lié à l'entité identifiée par le code 12 correspondant à « apple ». Le système de correspondance mots - entités représentera, par exemple, les mots du passage d comme la séquence $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$ et les entités du passage comme $(0, 0, 0, 0, 457, 457, 457, 0, 12)$, avec le code 0 pour les mots qui ne sont pas des entités. On définit alors $G^a = (V \cup D \cup \{q\}, E \cup E', R \cup \{doc\})$ le graphe augmenté hétérogène contenant des entités, les documents D et la requête q, ainsi que des liens $E' = \{(e, doc, d), e \in V, d \in D \cup \{q\}\}$ de type doc reliant les documents et la requête aux entités qu'ils contiennent.

L'objectif principal de cette recherche est d'opérer la tâche de classement de do-

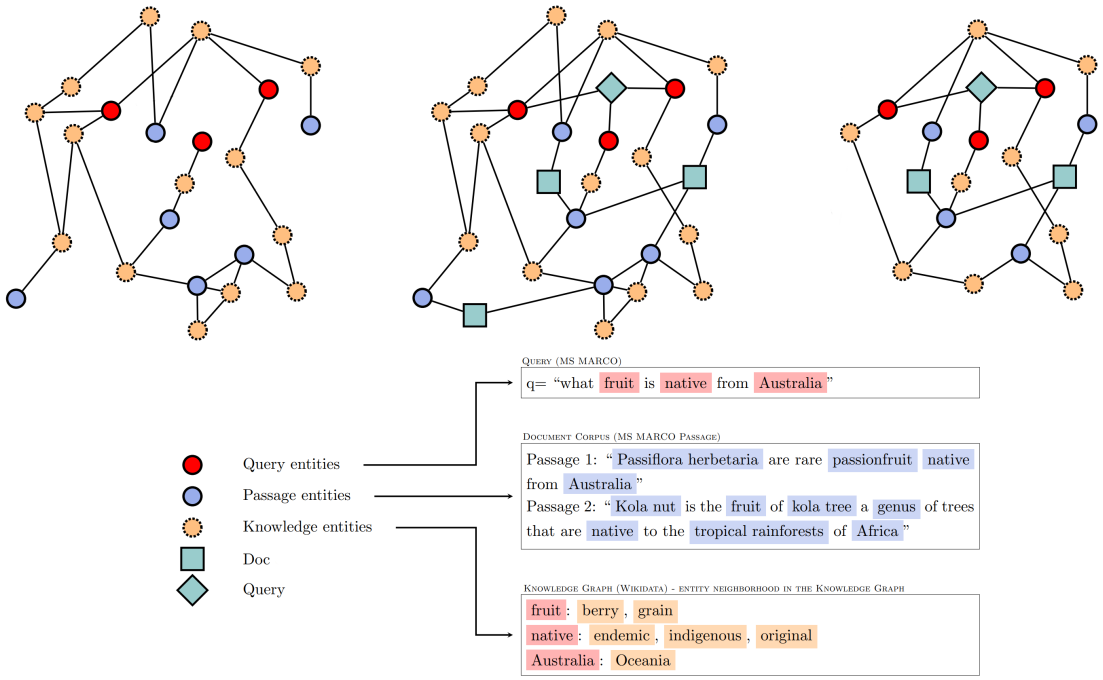

FIGURE 6.3 – Processus de construction du sous-graphe. (en haut à gauche) le graphe de connaissances initial ne contenant que des entités ; (en haut au milieu) le graphe de connaissances hétérogène obtenu en ajoutant des sommets de type « document » qui peuvent correspondre soit à la requête soit à des documents du corpus, ainsi que les connexions vers les entités qu'ils contiennent ; (en haut à droite) le sous-graphe hétérogène spécifique à la requête dans lequel seule une partie des documents et des entités est conservée. Les documents conservés sont ceux jugés pertinents par un modèle préliminaire de recherche comme BM25. Les entités de ce sous-graphe spécifique sont de trois types : celles provenant de la requête initiale, celles provenant des documents pertinents pour la requête, et celles situées sur des chemins dans le graphe entre les entités de la requête et celles des documents pertinents. La figure est illustrative et le graphe ne correspond pas aux passages et à la requête.

cuments du corpus en fonction de leur degré de pertinence par rapport à une requête en langage naturel $q = (w_1^q, ..., w_{|q|}^q)$. Notons que le processus de liaison d'entité décrit précédemment est également appliqué à la requête.

Pour s'attaquer efficacement à cette tâche, nous adoptons une approche en deux étapes. Dans un premier temps, pour chaque requête q, nous procédons à l'extraction d'un sous-graphe hétérogène G_q, contenant des entités de G le graphe de connaissances générales. G_q est un graphe hétérogène qui se réfère au sous-graphe spécifique à la requête et qui contient des entités et des documents jugés hautement pertinents pour la requête q, nommés documents candidats. Il convient de noter que ces documents candidats font

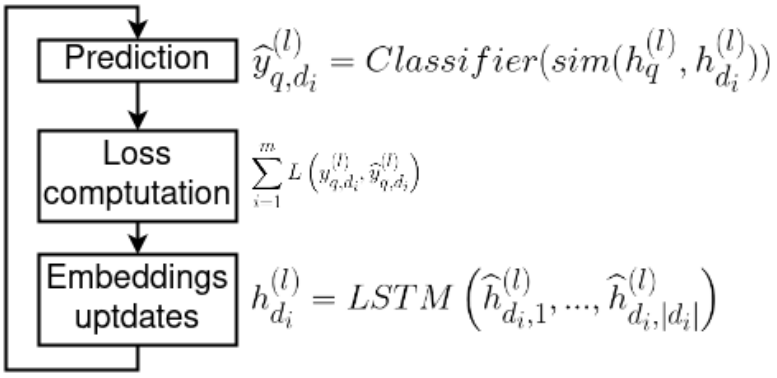

FIGURE 6.4 – Processus d'entraînement du modèle QuSpec. Le sous-graphe spécifique à la requête construit dans la Figure 6.2 est à la base du processus d'entraînement. À chaque étape de l'entraînement, les entités et les documents du sous-graphe sont mis à jour. Ensuite, les plongements des documents sont utilisés pour prédire leur pertinence vis-à-vis de la requête. Les prédictions ainsi obtenues sont utilisées pour calculer la perte d'apprentissage du classement décrite dans 6.5.4. Ensuite, la fonction de perte est minimisée pour l'étape courante et les paramètres résultants sont rétro-propagés dans le sous-graphe pour l'étape d'entraînement suivante.

partie du sous-graphe et sont liés aux entités qu'ils contiennent. Le sous-graphe hétérogène spécifique à la requête obtenu lors de cette étape est détaillé dans la Figure 6.2. La raison de cette étape initiale est de garantir un rappel élevé des entités pertinentes tout en limitant la taille du graphe à des dimensions compatibles avec la mémoire du GPU pour l'apprentissage ultérieur basé sur le gradient. Ensuite, nous utilisons notre modèle proposé, QuSpec, pour obtenir des représentations de nœuds dans le sous-graphe $G_q = (V_q, E_q, R_q)$. Ces représentations de nœuds sont conditionnées par la requête q et sont ensuite utilisées pour classer les documents du corpus en fonction de leur pertinence par rapport à la requête. La Figure 6.4 décrit les étapes globales de notre méthode, allant de la construction du sous-graphe hétérogène de la requête à l'entraînement du modèle QuSpec. Ces aspects sont discutés plus en détail aux sections 6.4 et 6.5.

6.4 Construction de sous-graphes spécifiques aux requêtes

La requête q et l'ensemble des documents pertinents $relevant(D, q) \subseteq D$ pour cette requête conduisent à une catégorisation des nœuds des documents dans V_q. En effet, nous désignons $y_d = 1$ si $d \in relevant(D, q)$ et $y_d = 0$ sinon, où $d \in V_q$ représente un nœud de type document. Par conséquent, la tâche de classement peut être considérée comme une classification binaire appliquée aux nœuds de type document dans notre graphe G_q. Néanmoins, il est important de souligner que deux distinctions notables existent dans notre scénario par rapport aux tâches de classification conventionnelles sur les graphes. Premièrement, nous cherchons à conditionner les représentations des nœuds dans le graphe sur

la base de la question en langage naturelle q. Deuxièmement, notre graphe G_q est caractérisé par des nœuds hétérogènes. Certains nœuds de ce graphe correspondent à des entités du graphe de connaissances, représentant des objets symboliques, tandis que d'autres représentent des documents textuels, qui se manifestent sous la forme de séquences de mots de longueur variable.

Pour traiter ces aspects, nous construisons un sous-graphe de requête G_q qui favorise une représentation basée sur le contexte sémantique de la requête et nous introduisons un mécanisme de conditionnement sur cette dernière pour traiter la première distinction. Ensuite, nous utilisons des mises à jour hétérogènes pour traiter la deuxième distinction.

Pour construire le sous-graphe spécifique à une requête q, nous commençons par trouver les documents les plus pertinents pour la requête en nous basant sur un extracteur préliminaire. En effet, le sous-graphe G_q contient à la fois des entités du graphe de connaissances G, la requête q et des documents du corpus D. Pour extraire les documents pertinents pour la requête q, nous appliquons BM25 [RJ76] et récupérons les N premiers documents candidats renvoyés. Ici, N est un paramètre que nous pouvons choisir en fonction de nos besoins (par exemple 50 ou 100). Une fois que nous avons les N meilleurs documents, nous récupérons toutes les entités de ces documents à l'aide d'un outil de détection d'entités nommées (NED pour « Named Entity Detection ») comme ELQ [1]. Nous désignons les entités de la requête et de ses documents candidats comme les entités de base. Les entités de départ provenant des documents sont appelées S_d. De même, nous récupérons toutes les entités S_q du texte de la requête.

Nous identifions des connexions entre ces entités afin de former des chemins et de constituer notre sous-graphe spécifique à la requête. Pour cela, nous utilisons CLOCQ [2] pour trouver des chemins entre les entités de la requête et des documents, ce qui peut nous donner des chemins (lorsqu'ils existent) de distance 1 ou 2 entre les entités. Notez qu'ici, distance 1 signifie que les entités appartiennent au même fait (par exemple un triplet (o, r, s) tel que défini dans 6.3), et distance 2 signifie que les deux entités ont une entité commune dans leurs voisinages de distance 1.

Nous disposons de deux types de paires entre lesquelles nous voulons vérifier les connexions (c'est-à-dire vérifier qu'il existe un chemin dans le graphe entre les éléments de la paire) :

— les paires entité de requête / entité de document (S_q, S_d)
— les paires entité de document / entité de document (S_d, S_d)

Toutefois, en pratique, des considérations relatives au temps de calcul doivent être prises en compte pour faciliter l'implémentation de cette méthode. Par exemple, si nous prenons $N = 100$, et en supposant 3 à 4 entités par document, cela implique 300 à 400 entités de base pour chaque sous-graphe, ce qui fait environ 10^4 paires à vérifier par requête. Le nombre de requêtes requis pour l'entraînement pouvant être grand, nous souhaitons nous limiter à un nombre raisonnable de paires à vérifier, 10^4 étant déjà trop élevé. Pour éviter cet inconvénient, nous choisissons un sous-ensemble de ces paires au

1. https ://github.com/facebookresearch/BLINK/tree/main/elq
2. https ://github.com/PhilippChr/CLOCQ

hasard et les vérifions. Bien que cela puisse nuire quelque peu à la précision, cela permet de gagner en temps de calcul, ce dont nous tirons parti.

6.5 Apprentissage de plongements spécifiques aux requêtes

Dans la suite, nous exposons en détails les différentes étapes qui constituent notre modèle QuSpec.

6.5.1 Conditionnement à la requête

Dans la poursuite de notre objectif de construire une représentation basée sur le contexte de la requête, nous introduisons également un mécanisme qui incorpore la dépendance à la requête. Ce dernier favorise la propagation de l'information le long des chemins du graphe de connaissances qui impliquent des entités mentionnées dans la requête. En outre, nous utilisons un mécanisme d'attention pour évaluer la pertinence des relations associées aux entités de la requête, garantissant ainsi que les plongements sont principalement propagés le long des arêtes pertinentes pour la requête. Ces aspects sont décrits plus en détail ci-après.

La requête q est représentée comme une séquence de mots, notée $(w_1^q, ..., w_{|q|}^q)$. À l'initialisation, h_q^0, est calculée en donnant cette séquence à un réseau récurrent à mémoire court et long terme (LSTM) [HS97b], dont nous prenons l'état final. Les couches suivantes mettent à jour la représentation de la requête, h_q, à l'aide d'un réseau de neurones à propagation avant (FFN pour « feed-forward network ») appliqué à la somme de toutes les entités, S_q, mentionnées dans la requête. Les équations de mise à jour de la requête sont décrites ci-dessous :

— l'initialisation est effectuée au moyen d'un réseau LSTM : $h_q^{(0)} = LSTM(w_1^q, ..., w_{|q|}^q)$
— par la suite, à chaque couche, l'équation de mise à jour est obtenue par :

$$h_q^{(l)} = FFN(\sum_{v \in S_q(v)} h_v^{(l)}) \tag{6.1}$$

De nombreuses requêtes nécessitent une forme de raisonnement multi-sauts, le long de chemins partant d'un nœud explicitement mentionné dans la requête et s'étendant jusqu'à un nœud dans le document. Pour encourager l'adoption d'un tel comportement au cours du processus de propagation des plongements, nous nous inspirons du concept de PageRank personnalisé, une notion introduite à l'origine dans le domaine de la recherche d'information par [Hav02]. Le PageRank a été initialement proposé pour mesurer quantitativement la popularité d'une page web. Plusieurs variantes ont ensuite été adaptées à diverses sous domaines et en particulier au classement de textes.

La propagation des plongements commence à partir des entités de la requête, désignées par S_q. En plus des plongements vectoriels, représentés par h_v, associés à chaque nœud, nous conservons une mesure scalaire appelée score « PageRank », désignée par $p_r v$. Ce score sert à quantifier le poids cumulé attribué aux chemins provenant d'une

entité de la requête et allant jusqu'au nœud courant. Mathématiquement, ce score est formulé comme suit :

— l'initialisation commence par $pr_v^0 = 1/S_q$ si $v \in S_q$ et 0 sinon.

— Ensuite, à chaque couche l, le score PageRank est :

$$pr_v^{(l)} = (1-\mu)pr_v^{(l-1)} + \mu \sum_{v' \in N_r(v)} \sum_{r \in R} \omega_{v'}^r pr_v^{(l-1)} \tag{6.2}$$

Le poids d'attention figurant dans le troisième terme de l'équation 6.2 est déterminé par un processus de calcul qui implique les plongements des requêtes et des relations. Mathématiquement, il s'exprime sous la forme $\omega_{v'}^r = Softmax(t_r h_q^{(l-1)})$, où une normalisation softmax est effectuée sur toutes les arêtes sortantes issues du nœud v'. Ici, t_r représente la représentation vectorielle associée à la relation r. Ce mécanisme est essentiel pour piloter la propagation des plongements d'une manière qui favorise l'importance des arêtes contextuellement pertinentes pour la requête.

6.5.2 Mise à jour hétérogène des entités

Les nœuds correspondant aux entités sont initialisés à l'aide de vecteurs de dimension fixe, désignée par $h_v = x_v \in \mathbf{R}^n$. Ici, x_v peut être soit un plongement pré-entraîné de graphe de connaissances, soit un vecteur généré de manière aléatoire. La variable n représente la taille des plongements. Soit $M(v) = (d, p)$ l'ensemble des positions p dans les documents d qui correspondent à des mentions de l'entité v. La procédure de mise à jour des nœuds d'entité consiste à appliquer un réseau feed-forward (FFN) à une couche sur la concaténation de quatre états distincts, comme l'exprime l'équation 6.3. La Figure 6.5 illustre le processus de mise à jour et la manière dont les différents composants impliqués sont agrégés.

$$h_v^{(l)} = \mathrm{FFN}\begin{bmatrix} h_v^{(l-1)} \\ h_q^{(l-1)} \\ \sum_{v' \in N_r(v)} \sum_{r \in R} \omega_{v'}^r \sigma_r(h_{v'}^{(l-1)}) \\ \sum_{(d,p) \in M(v)} h_{d,p}^{(l-1)} \end{bmatrix} \tag{6.3}$$

Dans l'équation 6.3, les deux premiers termes sont respectivement la représentation de l'entité et la représentation de la requête, toutes deux héritées de la couche précédente.

Le troisième terme implique l'agrégation des états dérivés des entités voisines du nœud courant, $N_r(v)$. Cette agrégation est réalisée après le redimensionnement des états avec un poids d'attention ω_v^r (un concept décrit dans la suite) et l'application de transformations spécifiques aux relations dénotées par T_r. Les recherches antérieures, en particulier

dans le contexte des réseaux convolutionnels de graphe relationnel [SKB+17], utilisent une projection linéaire pour T_r. Cependant, cette approche peut se révéler computationnellement prohibitive lorsqu'il s'agit de sous-graphes étendus [SDZ+18a].

Par conséquent, dans ce travail, nous optons pour des vecteurs de relation, notés t_r pour $r \in R_q$, au lieu de matrices. La mise à jour le long d'une arête est calculée selon l'équation 6.4 :

$$\sigma_r(h_{v'}^{(l-1)}) = pr_{v'}^{(l-1)}\text{FFN}(t_r, h_{v'}^{(l-1)}) \tag{6.4}$$

Dans l'équation 6.4, $pr_{v'}^{(l-1)}$ désigne un score PageRank qui régit la propagation des plongements le long des chemins partant des nœuds de la requête, dont une explication détaillée est fournie dans la section suivante.

Enfin, le quatrième terme agrège les états issus des tokens qui correspondent à des mentions de l'entité v dans les documents constituant le sous-graphe. Il est essentiel de noter que le processus de mise à jour dépend de la position des entités dans les documents qui les contiennent.

6.5.3 Mise à jour hétérogène des documents

Les nœuds des documents dans le graphe correspondent à des séquences de texte de longueur variable. Étant donné que plusieurs entités peuvent être liées à différentes positions dans un document donné, nous gardons une représentation flexible du document à chaque couche. Cette représentation dynamique du document est définie par $h_d \in \mathbf{R}^{|d| \ timesn}$, où $|d|$ représente la longueur du document. Étant donné les mots constitutifs du document, désignés par $(w_1, ..., w_{|d|})$, nous établissons la représentation cachée initiale comme suit : $h_d = LSTM(w_1, ..., w_{|d|})$. Plus précisément, nous représentons la p-ième ligne de h_d, qui correspond au plongement du p-ième mot dans le document d à la couche l, par $h_{d,p}$. Considérons $L(d, p)$, la collection de toutes les entités associées au mot situé à la position p dans le document d. Le processus de mise à jour de la représentation du document se déroule en deux étapes. Dans un premier temps, nous procédons à l'agrégation des états des entités liées à chaque position individuelle dans le document. Cette opération est décrite mathématiquement par l'équation 6.5.

$$\hat{h}_{d,p}^{(l)} = FFN(h_{d,p}^{(l-1)}, \sum_{v \in L(d,p)} h_v^{(l-1)}) \tag{6.5}$$

Dans cette équation, les variables h_v sont normalisées par le nombre d'arêtes sortantes au niveau du nœud v. Ensuite, nous agrégeons les différents états du document en utilisant un mécanisme de LSTM.

6.5.4 Apprentissage de classements sur les plongements de documents

Le processus d'apprentissage de classements est une tâche supervisée dont l'objectif est d'obtenir de manière autonome une fonction prédictive $F(x)$, étant donné un ensemble

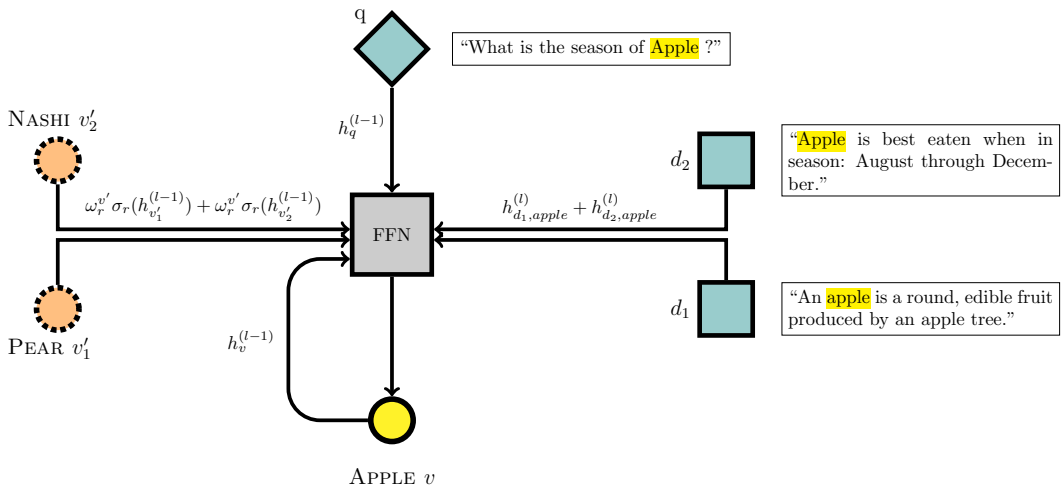

FIGURE 6.5 – Mécanisme de mise à jour des plongements d'entité. L'entité « Apple » est mise à jour avec les quatre composantes impliquées dans l'équation 6.3 qui sont données en entrée d'un réseau neuronal feed-forward (FFN). À côté de l'entité « Apple », on peut observer une boucle prenant en compte le plongement de l'entité « Apple » de l'étape précédente. Sur le côté gauche de la figure, les informations provenant des entités « Pear » et « Nashi », les voisines de l'entité « Apple » dans le graphe de connaissances, sont véhiculées. En haut, le plongement de la requête de l'étape précédente est inclus dans le processus. Enfin, dans la partie droite de la figure, les plongements des documents du sous-graphe spécifique à la requête dans lesquels l'entité « Apple » est mentionnée sont transmis au mécanisme de mise à jour.

de données d'apprentissage comprenant des paires de la forme $x = ((x_1, y_1), (x_2, y_2), ..., (x_m, y_m))$. Ici, (x_i) désigne les plongements de documents, et (y_i) les étiquettes associées à ces documents. Ici, F est un classifieur binaire qui prédit si le document est pertinent ou non par rapport à la requête.

Pour évaluer la performance prédictive de la fonction F, nous utilisons une fonction de perte $L(.,.)$. Le processus d'apprentissage revient à la minimisation de la fonction de risque empirique, comme pour d'autres tâches d'apprentissage. Cependant, la minimisation peut être plus ou moins compliquée selon les caractéristiques de la fonction de perte. En particulier, le mécanisme de rétropropagation requis dans le processus d'apprentissage contraint la fonction de perte d'être différentiable. Par conséquent, afin de garantir le respect de cette exigence, nous optons pour l'erreur quadratique moyenne (MSE pour « Mean Squared Error »), une fonction de perte par point qui est continue.

Étant donné les données d'apprentissage $x = ((x_1, y_1), (x_2, y_2), ..., (x_m, y_m))$, la fonc-

tion de perte est définie comme suit :

$$L(F) = \frac{1}{m} \sum_{i=1}^{m} L(F(x_i, y_i)) \tag{6.6}$$

La tâche d'apprentissage consiste alors à minimiser la fonction de perte, comme pour les autres tâches d'apprentissage.

6.5.5 Inférence

Les représentations finales obtenues lors de l'apprentissage sont ensuite utilisées pour classer les documents. Plus précisément, durant l'apprentissage un classifieur binaire est entraîné sur les représentations des documents. Une fois l'apprentissage terminé, ce classifieur est directement appliqué sur les représentations des documents lors de l'étape d'inférence.

6.6 Conclusion

Dans ce travail, nous avons proposé une nouvelle méthode de classement de textes basée sur l'utilisation de graphe de connaissances. Malgré l'absence de résultats, que nous n'avons pas encore obtenus pour cause de temps, nous tenions à détailler cette approche car elle constitue une approche générique pour exploiter l'information d'une base de connaissances quelconque pour le classement de textes. En particulier, elle propose une représentation des entités qui répond aux limites identifiées dans les méthodes similaires de l'état de l'art. En détails, la majorité des méthodes existantes de classement avec des graphes de connaissances adoptent une représentation sémantique généralisée qui est établie à partir de descriptions textuelles statiques des entités dans le graphe. Ces descriptions statiques montrent des avantages limités dans l'amélioration des performances du classement. En effet, si l'intérêt d'utiliser les entités est de répondre à l'insuffisance d'une compréhension contextuelle des méthodes usuelles, une description statique constitue une information généralisée et rigide qui ne peut pas s'adapter aux contextes particuliers développés dans les textes. Afin de répondre à cette lacune, notre méthode utilise des représentations d'entités spécifiques aux requêtes, plus aptes à discerner l'intention exprimée par les requêtes utilisateurs.

Pour obtenir ces représentations spécifiques, notre méthode s'articule en deux étapes. Dans un premier temps, nous procédons à l'extraction d'un sous-graphe spécifique à la requête qui contient des documents et des entités estimés fortement pertinents pour la requête. Pour cette première étape, nous sélectionnons les documents les plus pertinents pour la requête au moyen d'un extracteur préliminaire de documents, permettant un premier tri de documents potentiellement pertinents. Nous récupérons ensuite toutes les entités de ces documents et celles de la requête puis identifions des connexions entre ces entités. Ces connexions sont utilisées pour créer des chemins dans le graphe de connaissances et construire notre sous-graphe spécifique à la requête. Dans un second temps,

nous utilisons notre méthode, QuSpec, pour obtenir des représentations des nœuds dans le sous-graphe spécifique. Ces représentations sont conditionnées par la requête et sont utilisées ultérieurement pour classer les documents du corpus en fonction de leur pertinence par rapport à la requête. Nous employons un processus de mise à jour hétérogène de représentation qui, pour une entité donnée, prend en compte l'information de la requête, des documents qui mentionnent l'entité, et de l'entité et de son voisinage dans le graphe de connaissances. Ce processus de mise à jour permet un apprentissage des représentations des entités qui incorpore les dépendances de chacune de ces composantes. Il favorise la propagation de l'information au long de chemins du graphe de connaissances impliquant des entités mentionnées dans la requête. Ce processus, basé sur un mécanisme d'attention, garantit une représentation des entités basée sur le contexte de la requête.

Nous donnons dans la suite de cette conclusion des informations sur les prochaines étapes de ce projet, toujours en développement. Nous en profitons pour décrire les limites liées à la méthode que nous avons rencontrées, et proposer des pistes d'amélioration.

Au moment de l'écriture de cette thèse, ce projet de recherche est toujours en cours et nécessite quelques développements, notamment pour la phase d'apprentissage. En effet, la méthode est inspirée de travaux menés pour la tâche de réponse aux questions (définie dans le Chapitre 1) et certaines composantes (paramètres, classifieur) doivent être adaptées pour le classement de texte. En outre, un protocole d'évaluation doit être conduit pour évaluer la performance de la méthode. Pour ce dernier point, nous avons élaboré un processus de comparaison avec deux modèles de référence de pointe, ColBERT [KZ20] et DPR [KOM$^+$20] que nous souhaitons évaluer dans des conditions similaires. Effectivement, ces modèles ayant été entraînés sur de vastes jeux de données, il est ainsi important, pour obtenir une comparaison équitable, de les reproduire à une échelle plus modeste, étant données nos capacités de calcul limitées. C'est pourquoi nous souhaitons ré-entraîner ces modèles sur un jeu de données comprenant 10 000 requêtes, le nombre de requêtes que nous utilisons actuellement pour l'entraînement de QuSpec. Enfin, pour finaliser ce projet, nous envisageons une étude d'ablation pour déterminer le rôle des différentes composantes de notre méthode. En particulier, nous souhaitons évaluer l'apport des mécanismes de conditionnement sur la requête et l'apport de la mise à jour hétérogène, deux éléments essentiels de notre approche.

Pour conclure ce chapitre, nous décrivons les limites que nous avons rencontrées dans la mise en œuvre de notre méthode et suggérons des améliorations possibles pour des travaux futurs.

La principale limite de notre méthode est liée à l'accessibilité des entités qui sont obtenues au moyen d'une API (CLOCQ [3] dans notre cas). Chaque entité est acquise par l'intermédiaire d'une requête API. Le nombre abondant d'entités dans les documents induit ainsi un requêtage massif de l'API dont le fonctionnement est régulièrement grippé par le trop grand nombre de requêtes envoyées. Par ailleurs, l'utilisation de l'API peut être très consommatrice en ressource de calcul, et représente dans notre cas 365 gigaoctets de mémoire vive. Additionné au processus d'extraction d'entités qui représente dans notre

3. https ://github.com/PhilippChr/CLOCQ

cas 150 gigaoctets de mémoire vive, on obtient un besoin de calcul supérieur à 500 gigaoctets qui constitue une spécification très contraignante même sur un serveur informatique de haute performance. Il est donc important d'alléger cette contrainte en ressource. Pour cela, il est envisageable d'utiliser des techniques d'échantillonage pour sélectionner les entités qui seront effectivement extraites. Une autre alternative est de sélectionner plus finement les documents potentiellement pertinents pour construire le sous-graphe de la requête évoqué dans la méthode. Pour ce faire, on peut modifier l'extracteur préliminaire de documents (BM25 dans notre cas) par un extracteur plus performant. En effet, DPR, que nous citons comme modèle de référence, a été initialement proposé par ses auteurs comme un extracteur, utilisant une représentation dense, qui peut se substituer à BM25. Son mécanisme de représentation étant plus sophistiqué, on peut ainsi espérer un gain en performance permettant de réduire le nombre de documents pertinents nécessaires dans la phase de sélection préliminaire. Enfin, d'autres améliorations de notre méthode sont envisageables, comme la modification du processus de mise à jour hétérogène (des travaux plus récents [HDWS20] dans ce domaine ayant été proposés), ou encore l'initialisation des plongements [CD22] qui est susceptible de réduire les efforts d'entraînement du modèle QuSpec.

Chapitre 7

Conclusion

Sommaire

7.1 Problématique traitée dans cette thèse 121
7.2 Contributions 122
7.3 Perspectives 124

Dans cette thèse, nous nous sommes intéressés au problème de la recherche d'information qui est un domaine essentiel du traitement automatique du langage naturel. Dans ce chapitre, nous revenons sur la problématique de la thèse, nous détaillons les différentes contributions que nous avons apportées et nous proposons des perspectives afin d'inspirer de prochaines investigations dans la continuité de nos travaux.

7.1 Problématique traitée dans cette thèse

Par le biais de nos contributions, nous explorons comment le langage naturel peut être limité par les techniques majoritairement utilisées de découpage du texte en unités linguistiques. En particulier, nous nous sommes intéressés aux tâches de classement de textes (souvent nommé « document ranking » ou classement de documents dans la littérature) et de reconnaissance d'entités nommées (« named entity recognition ») qui sont emblématiques de la recherche d'information. Nous avons questionné l'apport d'une représentation du langage basée sur des graphes sur ces tâches, afin de montrer l'intérêt d'exploiter la structure qu'ils offrent. Par ailleurs, en étudiant l'impact du bruit, qui se manifeste par des erreurs dans les mots, sur le classement de textes, nous avons traité une problématique technique omniprésente dans le traitement automatique du langage naturel. Nos travaux soulignent l'utilité que peut avoir une représentation du langage en terme d'entités, un concept possédant une signification contextuelle se démarquant du mot, à la base de la représentation du langage. Les entités mettent plus en avant le

contexte du texte en proposant un découpage du langage orienté sur les liens sémantiques des différents contextes en présence.

7.2 Contributions

Le Chapitre 2 présente un état de l'art de la recherche d'information. Il décrit les concepts sur lesquels repose ce domaine et revient sur les outils communément utilisés pour l'explorer (comme les méthodes d'évaluation et les jeux de données). Après avoir défini ces notions fondamentales nous donnons un aperçu chronologique des méthodes développées pour traiter le classement de textes. Cette tâche de recherche est essentielle à la recherche d'information, et de nombreux travaux lui ont été consacrés depuis 1940. Nous retraçons les premières initiatives de recherche liées à cette tâche et détaillons les différents courants d'approches qui ont constitué ce domaine de recherche. Enfin, nous décrivons les catégories de méthodes de classement de textes plus récentes, avec une attention particulière pour les méthodes neuronales de classement qui sont à l'origine d'un changement de paradigme du domaine. Ces dernières influencent considérablement les publications scientifiques récentes et sont aujourd'hui incontournables pour aborder la recherche d'information.

Dans le Chapitre 3, nous abordons les techniques de représentation des graphes, et étudions les mécanismes sur lesquels reposent les plongements, une approche de représentation récente qui est à la base de l'exploitation du concept d'entité. Nous nous intéressons à un cas particulier de graphes, les graphes bipartis, qui offrent une configuration permettant de mieux comprendre les interdépendances entre les textes et les entités. Les plongements de graphes bipartis ont reçu peu d'attention malgré le potentiel que représente ce type de graphe pour modéliser un nombre d'applications variées (plateformes d'achat en ligne, réseaux sociaux, etc.). Notre travail apporte une étude complète de la littérature sur les techniques d'apprentissage de représentation pour les graphes bipartis. Nous proposons une catégorisation des approches existantes qui regroupe ces dernières selon des principes communs et nous analysons les bénéfices et désavantages de chacune des catégories proposées par rapport aux plongements de graphes simples. Nous répertorions également l'ensemble des ressources disponibles pour explorer ce domaine.

Dans le Chapitre 4, nous montrons l'intérêt d'un modèle de langage basé sur les graphes pour la tâche de reconnaissance d'entités nommées, une tâche liée au découpage du texte en unités linguistiques et qui est essentielle à notre problématique. Le modèle que nous proposons améliore les performances de l'état de l'art sur un jeu de données de textes multilingues historiques. Cette contribution montre l'apport des graphes, et en particulier des graphes de connaissances, sur une tâche permettant d'identifier les entités dans le texte. Ce travail est également lié au problème du bruit dans le texte, étant donné que la collection de textes considérée est issue de textes historiques qui sont souvent détériorés et comportent des erreurs induites par le processus de numérisation (par exemple, des fautes d'orthographe et des erreurs linguistiques). Les résultats que nous avons obtenus suggèrent que les méthodes basées sur des graphes de connaissances sont prometteuses pour traiter le bruit dans les collections historiques. En effet, nous avons

montré que notre approche est efficace pour détecter les entités contaminées par des erreurs de numérisation, avec des taux d'erreur des caractères allant jusqu'à 67%. Cette contribution s'intéresse également à l'intérêt d'injecter des informations temporelles dans la tâche de reconnaissance d'entités nommées sur des collections historiques et montre qu'il s'agit d'une piste pertinente à explorer. Effectivement, nos résultats montrent l'intérêt d'introduire une dimension temporelle dans la tâche de NER. Les résultats obtenus suggèrent que l'introduction d'une courte période de temps améliore la détection d'entités dans les collections de documents avec une diversité limitée d'entités et couvrant de petits intervalles d'années. Ils suggèrent également qu'une période de temps plus longue profite aux documents couvrant de grands intervalles d'années. En outre, ce travail a permis au laboratoire L3i de l'Université de La Rochelle de gagner la compétition HIPE 2022 (Identifying Historical People, Places and other Entities), dédiée à la maîtrise des processus d'extraction et de liaison d'entités nommées dans les journaux historiques multilingues.

Le Chapitre 5 est dédié à la problématique du bruit dans le texte et propose une étude exhaustive de l'impact du bruit sur les performances du classement de textes. En effet, une part importante des documents numériques textuels contient des erreurs (par exemple humaines). Bien que la problématique du bruit dans le texte soit régulièrement rencontrée dans le traitement automatique du langage naturel, il y a finalement peu de types et de distributions de bruit qui ont été étudiés dans la littérature. Notre travail évalue quantitativement l'impact de différents types et distributions d'erreurs sur le classement de textes. Cette contribution a permis la production et le partage de plusieurs jeux de données de textes bruités. La mise en source ouverte de ces jeux de données vise à soutenir les efforts de recherche afin de développer des modèles de classement plus robustes aux entrées textuelles bruitées. Elle a par ailleurs inspiré la création par Preligens d'un logiciel en source ouverte, *textnoisr*[1], dédié à l'injection de bruit dans du texte qui est plus adapté que l'outil que nous avons utilisé, ce dernier n'étant pas conçu pour synthétiser du texte bruité. Ce logiciel, ainsi que le protocole expérimental décrit dans ce travail, offrent à la communauté scientifique une base et des instructions claires pour la création de jeux de données de textes bruités, une ressource en nombre insuffisant dont la rareté entrave l'étude de ce domaine. En outre, en évaluant un modèle traditionnel (BM25 basé sur l'occurrence de mots) et un modèle de pointe (basé sur BERT) de classement de textes, notre étude montre qu'il n'existe pas aujourd'hui de méthode qui soit robuste au bruit et que les modèles de pointe perdent leur avantage dès que le niveau de bruit devient élevé. Ces modèles étant tous deux basés sur un découpage en unités linguistiques classiques, nous pointons une éventuelle limite de cette approche qu'il serait utile d'investiguer davantage. Notre étude donne un aperçu des défis que pose le classement de textes dans des conditions bruitées et plaide en faveur du développement de modèles de classement de textes plus robustes au bruit.

Enfin, le Chapitre 6, qui est un travail d'ouverture, propose un modèle de classement de textes basé sur une base de connaissances. Notre modèle constitue une approche générique pour tirer parti de l'information à disposition dans une base de connaissances

1. https ://github.com/earthcube-lab/textnoisr

quelconque. Il propose une représentation des entités qui s'attaque aux limites identifiées dans les méthodes similaires de l'état de l'art. En effet, la majorité de ces méthodes adoptent une représentation sémantique généralisée qui est obtenue à partir de descriptions textuelles statiques des entités dans le graphe de connaissances. Ces descriptions statiques montrent des bénéfices limités pour améliorer les performances du classement de textes. Pour répondre à ce défaut, notre méthode adopte des représentations d'entités qui sont spécifiques aux requêtes, de manière à mieux discerner l'intention exprimée par les requêtes utilisateurs. Nous procédons en deux étapes pour obtenir ces représentations spécifiques. Dans un premier temps pour chaque requête, un sous-graphe hétérogène spécifique à la requête est extrait et contient des documents et des entités jugés en adéquation avec le contexte de la requête. Cette étape permet un premier tri de supports d'informations potentiellement pertinents. Dans un second temps, nous appliquons notre modèle, QuSpec, sur le sous-graphe spécifique pour obtenir des représentations de ses nœuds. Ces représentations sont conditionnées par la requête et sont utilisées ultérieurement pour classer les documents du corpus en fonction de leur pertinence par rapport à la requête. Les représentations sont obtenues au moyen de mises à jour hétérogènes qui, pour une entité donnée, combinent l'information de la requête, des documents qui mentionnent l'entité, de l'entité et de son voisinage dans le graphe de connaissances. Ce processus de mise à jour permet un apprentissage des représentations des entités qui réunit les dépendances de chacune des ces composantes. Les représentations finales sont alors utilisées pour opérer la tâche de classement.

7.3 Perspectives

En conclusion de notre travail, les graphes prouvent leurs capacités à bien modéliser le texte en améliorant les performances de la tâche linguistique de reconnaissance d'entités nommées. En proposant un découpage du langage en unités linguistiques favorisant une compréhension contextuelle des textes, les graphes questionnent les limites des méthodes de découpages communes aux domaines traitant le langage.

D'autres travaux doivent être menés pour explorer l'étendue du potentiel des techniques utilisant des graphes, en particulier pour la tâche de classement de textes, pour laquelle nous avons montré qu'il n'existe pas de méthode robuste au bruit aujourd'hui. Nous proposons dans la suite des directions d'exploration afin d'inspirer de futures recherches sur le sujet.

Les erreurs dans le texte induites par le bruit pénalisent de manière équivalente les méthodes de classement de textes traditionnelles comme les plus modernes. Dans ce contexte, les modèles de langage basés sur les entités peuvent être une alternative aux méthodes couramment utilisées, en particulier dans une configuration où les erreurs dans le texte sont réparties dans peu de mots. Cette configuration correspond à un nombre peu élevé d'entités contaminées par le bruit. L'intuition laisse alors penser que les dommages induits par les erreurs peuvent être compensés par le surplus contextuel apporté par les entités. Dans la continuité de nos travaux, d'autres études doivent être conduites pour déterminer si le découpage du langage en unités linguistiques est limitant en particulier

dans un environnement bruité : il serait ainsi intéressant de voir comment les modèles de classement de textes basés sur les entités se comportent sous de telles contraintes. Peuvent-ils améliorer les performances ? Nos contributions des chapitres 5 et 6 esquissent une piste de recherche pour répondre à cette question. En effet, il serait intéressant d'étudier les performances des modèles de classement de textes basés sur les entités sous différentes configurations de bruit. Ainsi, nos jeux de données bruitées et notre modèle QuSpec offrent un support pour mener une analyse comparative des modèles à entités dans un environnement bruité. Les résultats d'une telle analyse permettraient d'évaluer l'aptitude à gérer le bruit des modèles à entités et de confronter leurs performances à celles des méthodes plus communes en faisant varier l'intensité et la distribution des erreurs dans le texte. Ces pistes permettraient de poursuivre le travail de cette thèse en contexte bruité et viseraient à étudier l'impact du découpage en unités linguistiques sur la robustesse au bruit dans le texte.

Une autre piste de recherche peut être d'explorer l'impact de la longueur des documents sur le découpage en unités linguistiques. Il s'agit d'un axe de travail que nous n'avons pas traité car nous nous sommes souvent intéressés à des textes courts. Cet axe de recherche indiquerait des orientations dans les travaux à poursuivre pour mieux saisir l'apport des différentes méthodes du découpage en unités linguistiques. Des recherches supplémentaires seraient nécessaires pour déterminer si une méthode de découpage est plus efficace selon la longueur des textes ou si elles se comportent globalement de la même manière quelle que soit la longueur traitée.

Il serait également possible, dans le prolongement de l'exploration de la problématique du bruit, de synthétiser des jeux de données plus représentatifs de bruit couramment rencontrés dans des scénarios réels. En effet, dans le Chapitre 5, nous avons décidé de nous concentrer sur des erreurs aléatoires dans le texte car elles permettent de recomposer n'importe quelle nature de bruit. Ainsi, une piste pertinente d'étude serait d'étendre nos travaux en investiguant d'autres types de bruit, par exemple les erreurs de frappe sur clavier.

Enfin, nos travaux se sont principalement intéressés aux graphes bipartis et aux graphes de connaissances, mais d'autres types de graphe pourraient être examinés. En effet, notre contribution au Chapitre 4 donne une option possible en étudiant un graphe temporel pour la tâche de NER. Par ailleurs, des graphes de nature dynamique, c'est-à-dire qui évoluent dans le temps, pourraient également être considérés pour explorer l'apport des graphes sur le langage naturel.

Publications issues de ces travaux de recherche

- Giamphy Edward, Guillaume Jean-Loup, Doucet Antoine, Sanchis Kevin, « A survey on bipartite graphs embedding », Social Network Analysis and Mining, 2023.

- Gonzàlez-Gallardo Carlos-Emiliano, Boros Emanuela, Giamphy Edward, Hamdi Ahmed, Moreno, José G., Doucet Antoine, « Injecting Temporal-Aware Knowledge in Historical Named Entity Recognition », Advances in Information Retrieval, ECIR, 2023.

- Giamphy Edward, Sanchis Kevin, Dashyan Gohar, Guillaume Jean-Loup, Hamdi Ahmed, Sanselme Lilian, Doucet Antoine, « A Quantitative Analysis of Noise Impact on Document Ranking », IEEE Conference on Systems, Man, and Cybernetics, 2023.

Bibliographie

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow : A system for large-scale machine learning. USENIX Association, 2016.

Voir page 47.

[AK20] Vasudev Awatramani and Anupam Kumar. Linguist geeks on WNUT-2020 task 2 : COVID-19 informative tweet identification using progressive trained language models and data augmentation. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pages 414–418, Online, November 2020. Association for Computational Linguistics.

Voir page 97.

[AM19] Nino Arsov and Georgina Mirceva. Network embedding : An overview, 2019.

Voir page 51.

[ARY20] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. Association for Computational Linguistics, 2020.

Voir page 43.

[ASdC+18] Prabal Agarwal, Jannik Strötgen, Luciano del Corro, Johannes Hoffart, and Gerhard Weikum. diaNED : Time-aware named entity disambiguation for diachronic corpora. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers), pages 686–693, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Voir page 77.

[Asg16] Nabiha Asghar. Yelp dataset challenge : Review rating prediction, 2016.

Voir pages 67 et 70.

[ASLS21] Khetam Al Sharou, Zhenhao Li, and Lucia Specia. Towards a better understanding of noise in natural language processing. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), pages 53–62, Held Online, September 2021. INCOMA Ltd.

Voir page 95.

[BCC+18a] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco : A human generated machine reading comprehension dataset, 2018.

Voir pages 31 et 42.

[BCC+18b] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco : A human generated machine reading comprehension dataset, 2018.

Voir page 94.

[BCKO15] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. Frappe : Understanding the usage and perception of mobile app recommendations in-the-wild, 2015.

Voir pages 67 et 70.

[BGGG+] Emanuela Boros, Carlos-Emiliano González-Gallardo, Edward Giamphy, Ahmed Hamdi, Jose G Moreno, and Antoine Doucet. Knowledge-based contexts for historical named entity recognition & linking. pages 1064–1078.

Voir page 86.

[BHKL06] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks : Membership, growth, and evolution. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, page 44–54, New York, NY, USA, 2006. Association for Computing Machinery.

Voir pages 67 et 70.

[BHP+20] Emanuela Boroş, Ahmed Hamdi, Elvys Linhares Pontes, Luis-Adrián Cabrera-Diego, Jose G Moreno, Nicolas Sidere, and Antoine Doucet. Alleviating digitization errors in named entity recognition for historical documents. In Proceedings of the 24th conference on computational natural language learning, pages 431–441, 2020.

Voir pages 75, 77, 82 et 84.

[BKG21] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

Voir page 58.

[BL07] James Bennett and Stan Lanning. The netflix prize. 2007.

Voir pages 67 et 70.

[BLSVM20] Guilherme Torresan Bazzo, Gustavo Acauan Lorentz, Danny Suarez Vargas, and Viviane P Moreira. Assessing the impact of ocr errors in information retrieval. In Advances in Information Retrieval : 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42, pages 102–109. Springer, 2020.

Voir page 93.

[BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011), 2011.

Voir pages 67 et 70.

[BMVZ21] Claudio D. T. Barros, Matheus R. F. Mendonça, Alex B. Vieira, and Artur Ziviani. A survey on embedding dynamic graphs, 2021.

Voir page 51.

[BNLD22] Emanuela Boros, Nhu Khoa Nguyen, Gaël Lejeune, and Antoine Doucet. Assessing the impact of ocr noise on multilingual event detection over digitised documents. International Journal on Digital Libraries, pages 1–26, 2022.

Voir page 75.

[BOHG13] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. Knowledge-Based Systems, 46 :109–132, 2013.

Voir page 55.

[Bri89] John S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Proceedings of the 2nd International Conference on Neural Information Processing Systems, 1989.

Voir page 83.

[BRM+20] Emanuela Boroş, Verónica Romero, Martin Maarand, Kateřina Zenklová, Jitka Křečková, Enrique Vidal, Dominique Stutzmann, and Christopher Kermorvant. A comparison of sequential and combined approaches for named entity recognition in a corpus of handwritten medieval charters. In 2020 17th International conference on frontiers in handwriting recognition (ICFHR), pages 79–84. IEEE, 2020.

Voir page 76.

[BSR+05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent.

In Proceedings of the 22nd international conference on Machine learning, pages 89–96, 2005.
Voir page 107.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.
Voir page 105.

[Bur10] Christopher Burges. From ranknet to lambdarank to lambdamart : An overview. Learning, 2010.
Voir pages 38 et 39.

[Bus45] V. Bush. As we may think. Atlantic Monthly, 1945.
Voir page 33.

[BYT19] Anna O. Basile, Alexandre Yahi, and Nicholas P. Tatonetti. Artificial intelligence for drug toxicity and safety. Trends in Pharmacological Sciences, 40(9) :624–635, 2019.
Voir page 65.

[CD22] Shubham Chatterjee and Laura Dietz. Bert-er : Query-specific bert entity representations for entity ranking. Association for Computing Machinery, 2022.
Voir page 119.

[CDC+17] Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, Muriel Visani, and Jean-Philippe Moreux. Impact of ocr errors on the use of digital libraries : towards a better access to information. In Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries, pages 249–252. IEEE Press, 2017.
Voir pages 91 et 93.

[Cel10] O. Celma. Music Recommendation and Discovery in the Long Tail. Springer, 2010.
Voir pages 67 et 70.

[CF06] Deepayan Chakrabarti and Christos Faloutsos. Graph mining : Laws, generators, and algorithms. ACM Comput. Surv., 38, 03 2006.
Voir page 57.

[CG96] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Association for Computational Linguistics, 1996.
Voir page 44.

[CHTB94] WB Croft, SM Harding, K Taghva, and J Borsack. An evaluation of information retrieval accuracy with simulated ocr output. In Symposium on Document Analysis and Information Retrieval, pages 115–126, 1994.
Voir page 93.

[CIRP21] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval, 2021.
Voir page 108.

[CKH+16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems, 2016.
Voir page 61.

[CMY+20] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. Overview of the trec 2019 deep learning track. arXiv preprint arXiv :2003.07820, 2020.
Voir page 39.

[CMY+21] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. Ms marco : Benchmarking ranking models in the large-data regime. 2021.
Voir page 31.

[CMYC21] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track, 2021.
Voir pages 94 et 99.

[Cra09] Nick Craswell. Mean Reciprocal Rank, pages 1703–1703. Springer US, Boston, MA, 2009.
Voir page 99.

[CS11] Jie Chen and Ilya Safro. Algebraic distance on graphs. SIAM Journal on Scientific Computing, 33(6) :3468–3490, 2011.
Voir page 59.

[CVD21] Janneth Chicaiza and Priscila Valdiviezo-Diaz. A comprehensive survey of knowledge graph-based recommender systems : Technologies, development, and contributions. Information, 2021.
Voir page 105.

[CWTY19] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. Collaborative similarity embedding for recommender systems, 2019.
Voir pages 9 et 67.

[CXG+22] Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. Temporal knowledge graph completion : A survey. arXiv preprint arXiv :2201.08236, 2022.
Voir page 78.

[CZC17] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding : Problems, techniques and applications, 2017.
Voir page 51.

[DAN22] Ioannis Dikeoulias, Saadullah Amin, and Günter Neumann. Temporal knowledge graph reasoning with low-rank and model-agnostic representations. arXiv preprint arXiv :2204.04783, 2022.
Voir page 78.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding, 2019.
Voir pages 19, 23, 42, 46, 82, 92, 98 et 107.

[DL15] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning, 2015.
Voir page 45.

[DLL+22] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In International Conference on Learning Representations, 2022.
Voir page 55.

[dOVA+23] Lucas Lima de Oliveira, Danny Suarez Vargas, Antônio Marcelo Azevedo Alexandre, Fábio Corrêa Cordeiro, Diogo da Silva Magalhães Gomes, Max de Castro Rodrigues, Regis Kruel Romeu, and Viviane Pereira Moreira. Evaluating and mitigating the impact of ocr errors on information retrieval. International Journal on Digital Libraries, pages 1–18, 2023.
Voir page 93.

[DPRMGB+21] Mª Luisa Díez Platas, Salvador Ros Munoz, Elena González-Blanco, Pablo Ruiz Fabo, and Elena Alvarez Mellado. Medieval spanish (12th–15th centuries) named entity recognition and attribute annotation system based on contextual information. Journal of the Association for Information Science and Technology, 72(2) :224–238, 2021.
Voir page 77.

[DS] Alvin Deng and Evan Shrestha. Bert-based transfer learning with synonym augmentation for question answering.
Voir page 98.

[DSP14] Laura Dietz, Michael Schuhmacher, and Simone Ponzetto. Queripidia : Query-specific wikipedia construction. In 4th Workshop on Automated Knowledge Base Construction (AKBC 2014), 2014.
Voir page 105.

[DXCL18] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018.
Voir page 41.

[DYW+19] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation, 2019.
Voir page 107.

[EHLP+22] Maud Ehrmann, Ahmed Hamdi, Elvys Linhares Pontes, Matteo Romanello, and Antoine Douvet. A Survey of Named Entity Recognition and Classification in Historical Documents. ACM Computing Surveys, 2022.
Voir pages 75 et 77.

[EHP+21] Maud Ehrmann, Ahmed Hamdi, Elvys Linhares Pontes, Matteo Romanello, and Antoine Doucet. Named entity recognition and classification on historical documents : A survey. arXiv preprint arXiv :2109.11406, 2021.
Voir page 80.

[EMB+09] Dumitru Erhan, Pierre-Antoine Manzagol, Y. Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. Journal of Machine Learning Research - Proceedings Track, 2009.
Voir page 45.

[ERBC20] Maud Ehrmann, Matteo Romanello, Stefan Bircher, and Simon Clematide. Introducing the CLEF 2020 HIPE shared task : Named entity recognition and linking on historical newspapers. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, Advances in information retrieval, pages 524–532, Cham, 2020. Springer International Publishing.
Voir pages 75, 79 et 85.

[ERC+20] Maud Ehrmann, Matteo Romanello, Simon Clematide, Phillip Benjamin Ströbel, and Raphaël Barman. Language resources for historical newspapers : the impresso collection. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 958–968, 2020.
Voir page 79.

[ERDC22] Maud Ehrmann, Matteo Romanello, Antoine Doucet, and Simon Clematide. Introducing the hipe 2022 shared task : Named entity recognition and linking in multilingual historical documents. In European Conference on Information Retrieval, pages 347–354. Springer, 2022.
Voir page 79.

[ERFC20a] Maud Ehrmann, Matteo Romanello, Alex Flückiger, and Simon Clematide. Extended overview of clef hipe 2020 : named entity processing on historical newspapers. In CEUR Workshop Proceedings, number 2696. CEUR-WS, 2020.

Voir pages 75, 79 et 86.

[ERFC20b] Maud Ehrmann, Matteo Romanello, Alex Flückiger, and Simon Clematide. Overview of clef hipe 2020 : Named entity recognition and linking on historical newspapers. In International Conference of the Cross-Language Evaluation Forum for European Languages, pages 288–310. Springer, 2020.

Voir pages 75, 77 et 79.

[ERNM+22] Maud Ehrmann, Matteo Romanello, Sven Najem-Meyer, Antoine Doucet, and Simon Clematide. Overview of HIPE-2022 : Named Entity Recognition and Linking in Multilingual Historical Documents. In Alberto Barrón-Cedeño, Giovanni Da San Martino, Mirko Degli Esposti, Fabrizio Sebastiani, Craig Macdonald, Gabriella Pasi, Allan Hanbury, Martin Potthast, Guglielmo Faggioli, and Nicola Ferro, editors, Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings o Lecture Notes in Computer Science (LNCS). Springer, 2022.

Voir page 77.

[FB01] Huyghe François-Bernard. Stratégies étatiques face aux enjeux de l'information, 20101.

Voir page 13.

[FLGD87] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. Commun. ACM, 1987.

Voir page 36.

[Fuh89] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. ACM Trans. Inf. Syst., 1989.

Voir pages 18 et 37.

[GA20] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral, 2020.

Voir page 69.

[GAB12] G.Sudeepthi, G. Anuradha, and Maddali Surendra Prasad Babu. A survey on semantic web search engine. 2012.

Voir page 106.

[GBL98] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer : an automatic citation indexing system. In DL '98, 1998.

Voir pages 67 et 70.

[GCHZ18] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine : Bipartite network embedding. In The 41st international ACM SIGIR conference on research & development in information retrieval, pages 715–724, 2018.

Voir pages 55, 57 et 59.

[GCL11] Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. 2011.

Voir page 38.

[GdBLC03] Timothy S Gardner, Diego di Bernardo, David Lorenz, and James J Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. Science (New York, N.Y.), 301(5629) :102—105, July 2003.

Voir page 66.

[GDDN18] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. arXiv preprint arXiv :1809.03202, 2018.

Voir page 81.

[Gey94a] Fredric C. Gey. Inferring probability of relevance using the method of logistic regression. Springer-Verlag, 1994.

Voir pages 18 et 37.

[Gey94b] Fredric C Gey. Inferring probability of relevance using the method of logistic regression. pages 222–231. Springer, 1994.

Voir page 107.

[GFAC16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. Association for Computing Machinery, 2016.

Voir page 41.

[GHC+19] Ming Gao, Xiangnan He, Leihui Chen, Tingting Liu, Jinglin Zhang, and Aoying Zhou. Learning vertex representations for bipartite networks, 2019.

Voir page 59.

[GKB+20] Jonathan Godwin, Thomas Keck, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Veličković, and Alvaro Sanchez-Gonzalez. Jraph : A library for graph neural networks in jax., 2020.

Voir page 69.

[GL04] Jean-Loup Guillaume and Matthieu Latapy. Bipartite structure of all complex networks. Information processing letters, 90(5) :215–221, 2004.

Voir page 57.

[GL16] Aditya Grover and Jure Leskovec. node2vec : Scalable feature learning for networks, 2016.
Voir page 57.

[Gra] Graph nets, `https://www.deepmind.com/open-source/graph-nets`.
Voir page 69.

[GRMJ15] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth Jones. Word embedding based generalized language model for information retrieval. 2015.
Voir page 40.

[GVB03] David Grangier, Alessandro Vinciarelli, and Hervé Bourlard. Information retrieval on noisy text. Technical report, IDIAP, 2003.
Voir page 91.

[GZL+21] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. A survey of graph neural networks for recommender systems : Challenges, methods, and directions, 2021.
Voir page 66.

[Ham] William L. Hamilton. Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 14(3) :1–159.
Voir pages 60 et 64.

[Har19] Donna Harman. Information retrieval : The early years. Foundations and Trends® in Information Retrieval, 2019.
Voir pages 34 et 35.

[Hav02] Taher H. Haveliwala. Topic-sensitive pagerank. WWW '02, page 517–526, New York, NY, USA, 2002. Association for Computing Machinery.
Voir page 113.

[HDL16] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.
Voir page 51.

[HDWS20] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. Association for Computing Machinery, 2020.
Voir page 119.

[He19] Chaoyang He. Heterogeneous graph convolutional networks for bipartite graph embedding. 2019.
Voir page 62.

[HGJ+19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pages 2790–2799. PMLR, 2019.
Voir page 82.

[HHG+13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. Association for Computing Machinery, 2013.
Voir page 40.

[HI18] Tim Hegeman and Alexandru Iosup. Survey of graph analysis applications, 2018.
Voir page 51.

[HJCS+20] Ahmed Hamdi, Axel Jean-Caurant, Nicolas Sidère, Mickaël Coustaty, and Antoine Doucet. Assessing and minimizing the impact of ocr quality on named entity recognition. In Digital Libraries for Open Knowledge : 24th International Conference on Theory and Practice of Digital Libraries, TPDL 2020, Lyon, France, August 25–27, 2020, Proceedings 24, pages 87–101. Springer, 2020.
Voir page 92.

[HK15a] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets : History and context. ACM Trans. Interact. Intell. Syst., 5(4), dec 2015.
Voir pages 67 et 70.

[HK15b] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets : History and context. ACM Trans. Interact. Intell. Syst., 5(4), dec 2015.
Voir pages 67 et 70.

[HLF+20] Wentao Huang, Yuchen Li, Yuan Fang, Ju Fan, and Hongxia Yang. Biane : Bipartite attributed network embedding. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 149–158, 2020.
Voir page 58.

[HLPB+21] Ahmed Hamdi, Elvys Linhares Pontes, Emanuela Boros, Thi Tuyet Hai Nguyen, Günter Hackl, Jose G Moreno, and Antoine Doucet. A multilingual dataset for named entity recognition, entity linking and stance detection in historical newspapers. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2328–2334, 2021.
Voir page 75.

[HM16] Ruining He and Julian McAuley. Ups and downs. In Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, apr 2016.
Voir pages 67 et 70.

[HPS+22] Ahmed Hamdi, Elvys Linhares Pontes, Nicolas Sidere, Mickaël Coustaty, and Antoine Doucet. In-depth analysis of the impact of ocr errors on named entity recognition and linking. Natural Language Engineering, pages 1–24, 2022.
Voir page 75.

[HPS+23] Ahmed Hamdi, Elvys Linhares Pontes, Nicolas Sidere, Mickaël Coustaty, and Antoine Doucet. In-depth analysis of the impact of ocr errors on named entity recognition and linking. Natural Language Engineering, 29(2) :425–448, 2023.
Voir page 92.

[HR18] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. Association for Computational Linguistics, 2018.
Voir page 45.

[HS97a] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 1997.
Voir page 43.

[HS97b] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8) :1735–1780, nov 1997.
Voir page 113.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.
Voir page 98.

[HXR+19] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. Adversarial representation learning on large-scale bipartite graphs, 06 2019.
Voir page 60.

[HXR+20] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. Cascade-bgnn : Toward efficient self-supervised representation learning on large-scale bipartite graphs, 2020.
Voir page 61.

[HYBdM17] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. PACRR : A position-aware neural IR model for relevance matching. Association for Computational Linguistics, 2017.
Voir page 41.

[HYBdM18] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-pacrr : A context-aware neural ir model for ad-hoc retrieval. Association for Computing Machinery, 2018.
Voir page 41.

[HYL18a] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.
Voir page 108.

[HYL18b] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs : Methods and applications, 2018.
Voir page 61.

[ILA95] David J Ittner, David D Lewis, and David D Ahn. Text categorization of low quality images. In Symposium on Document Analysis and Information Retrieval, pages 301–315. Citeseer, 1995.
Voir page 92.

[JCY+16] Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. Little is much : Bridging cross-platform behaviors through overlapped crowds. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, page 13–19. AAAI Press, 2016.
Voir page 57.

[JGP+07] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM Trans. Inf. Syst., 2007.
Voir page 30.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst., 20(4) :422–446, oct 2002.
Voir page 99.

[JLS03] Hongyan Jing, Daniel Lopresti, and Chilin Shih. Summarizing noisy documents. In Proceedings of the Symposium on Document Image Understanding Technology, pages 111–119, 2003.
Voir page 92.

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. Association for Computing Machinery, 2002.
Voir page 38.

[KdVBL20] Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. Which bm25 do you mean ? a large-scale reproducibility study of scoring variants. Springer-Verlag, 2020.
Voir page 36.

[KG20] Eleni Kogkitsidou and Philippe Gambette. Normalisation of 16th and 17th century texts in french and geographical named entity recognition. In Proceedings of the 4th ACM SIGSPATIAL Workshop on Geospatial Humanities, pages 28–34, 2020.
Voir page 77.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.
Voir page 44.

[KOM+20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.
Voir pages 107 et 118.

[KR20] Tanti Kristanti and Laurent Romary. Delft and entity-fishing : Tools for clef hipe 2020 shared task. In CLEF 2020-Conference and Labs of the Evaluation Forum, volume 2696. CEUR, 2020.
Voir page 77.

[KT] Mei Kobayashi and Koichi Takeda. Information retrieval on the web.
Voir page 35.

[KW16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.
Voir page 61.

[KW17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.
Voir page 108.

[KWPDG01] Stefan Kramer, Gerhard Widmer, Bernhard Pfahringer, and Michael De Groeve. Prediction of ordinal classes using regression trees. Fundamenta Informaticae, 47(1-2) :1–13, 2001.
Voir page 107.

[KZ20] Omar Khattab and Matei Zaharia. Colbert : Efficient and effective passage search via contextualized late interaction over bert, 2020.
Voir page 118.

[LAH07] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. ACM Trans. Web, 1(1) :5–es, may 2007.
Voir pages 67 et 70.

[LC18] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In Companion Proceedings of the The Web Conference 2018, pages 1771–1776, 2018.
Voir page 81.

[Li11] Hang Li. Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition. 2011.
Voir page 38.

[Lin19] Jimmy Lin. The neural hype and comparisons against weak baselines. 2019.
Voir page 42.

[Lin21] Jimmy Lin. The neural hype, justified ! a recantation. 2021.
Voir page 42.

[Liu09] Tie-Yan Liu. Learning to rank for information retrieval. Found. Trends Inf. Retr., 2009.
Voir page 38.

[LJS+19] Chong Li, Kunyang Jia, Dan Shen, C.J. Richard Shi, and Hongxia Yang. Hierarchical representation learning for bipartite graphs. In

Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pages 2873–2879. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

Voir page 61.

[LLDM08] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks : Natural cluster sizes and the absence of large well-defined clusters, 2008.

Voir pages 67 et 70.

[LLS+15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, page 2181–2187. AAAI Press, 2015.

Voir page 105.

[LM14] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

Voir page 40.

[LMI+20] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen tau Yih. Efficient one-pass end-to-end entity linking for questions, 2020.

Voir page 108.

[LML+21] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini : A Python toolkit for reproducible information retrieval research with sparse and dense representations. In Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), 2021.

Voir page 98.

[LNY21] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking : Bert and beyond, 2021.

Voir page 24.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta : A robustly optimized bert pretraining approach, 2019.

Voir pages 45 et 47.

[Lop05] Daniel Lopresti. Performance evaluation for text processing of noisy inputs. In Proceedings of the 2005 ACM symposium on Applied computing, pages 759–763. ACM, 2005.

Voir page 92.

[Lop08] Daniel Lopresti. Measuring the impact of character recognition errors on downstream text analysis. In Document Recognition and Retrieval

XV, volume 6815, page 68150G. International Society for Optics and Photonics, 2008.
Voir page 92.

[LPCDM+22] Elvys Linhares Pontes, Luis Adrián Cabrera-Diego, Jose G Moreno, Emanuela Boros, Ahmed Hamdi, Antoine Doucet, Nicolas Sidere, and Mickaël Coustaty. Melhissa : a multilingual entity linking architecture for historical press articles. International Journal on Digital Libraries, 23(2) :133–160, 2022.
Voir pages 75 et 77.

[LPHSD19] Elvys Linhares Pontes, Ahmed Hamdi, Nicolas Sidere, and Antoine Doucet. Impact of ocr quality on named entity linking. In Digital Libraries at the Crossroads of Digital Information for the Future : 21st International Conference on Asia-Pacific Digital Libraries, ICADL 2019, Kuala Lumpur, Malaysia, November 4–7, 2019, Proceedings 21, pages 102–115. Springer, 2019.
Voir page 92.

[LSP+18] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences, 2018.
Voir page 43.

[LSQL21] Cong Li, Min Shi, Bo Qu, and Xiang Li. Deep attributed network representation learning via attribute enhanced neighborhood, 2021.
Voir page 51.

[Luh58] H. P. Luhn. The automatic creation of literature abstracts. IBM Journal of Research and Development, 1958.
Voir page 33.

[LW07] Jimmy Lin and W. Wilbur. Pubmed related articles : A probabilistic topic-based model for content similarity. BMC bioinformatics, 2007.
Voir page 17.

[LX14] Hang Li and Jun Xu. Semantic matching in search. Found. Trends Inf. Retr., 2014.
Voir page 36.

[Ma19] Edward Ma. Nlp augmentation. https ://github.com/makcedward/nlpaug, 2019.
Voir page 97.

[Mar94] Mitchell Marcus. New trends in natural language processing : Statistical natural language processing. National Academy Press, Washington DC, 1994.
Voir page 34.

[MBMEL12] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, page 909–916, New York, NY, USA, 2012. Association for Computing Machinery.

Voir pages 67 et 70.

[MC18] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval t. Foundations and Trends® in Information Retrieval, 2018.

Voir pages 27 et 39.

[MC19] Bhaskar Mitra and Nick Craswell. An updated duet model for passage re-ranking, 2019.

Voir page 41.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

Voir page 56.

[MDC16] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search, 2016.

Voir page 41.

[MDOJ18] Stephen Mutuvi, Antoine Doucet, Moses Odeo, and Adam Jatowt. Evaluating the impact of ocr errors on topic modeling. In Maturity and Innovation in Digital Libraries : 20th International Conference on Asia-Pacific Digital Libraries, ICADL 2018, Hamilton, New Zealand, November 19-22, 2018, Proceedings 20, pages 3–14. Springer, 2018.

Voir page 92.

[MF22] Enrique Manjavacas and Lauren Fonteyn. Adapting vs pre-training language models for historical languages. 2022.

Voir page 75.

[MFRM21] Tao Meng, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. Gemnet : Effective gated gazetteer representations for recognizing complex entities in low-context input. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, pages 1499–1512, 2021.

Voir page 78.

[MJE] Fey Matthias and Lenssen Jan Eric. Fast graph representation learning with pytorch geometric. `https://github.com/pyg-team/pytorch_geometric`.

Voir page 68.

[MK60] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. 1960.

Voir page 33.

[MKS+99] John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, et al. Performance measures for information extraction. In Proceedings of DARPA broadcast news workshop, pages 249–252. Herndon, VA, 1999.

Voir page 85.

[MNCC16] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking, 2016.

Voir page 40.

[MNRS04] Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. Information Retrieval, 3 :127–163, 2004.

Voir pages 67 et 70.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008.

Voir page 99.

[MRS+21] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models : A survey, 2021.

Voir page 23.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

Voir pages 43 et 57.

[MTR19] Stephen Mayhew, Tatiana Tsygankova, and Dan Roth. ner and pos when nothing is capitalized. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6256–6261, Hong Kong, China, November 2019. Association for Computational Linguistics.

Voir page 80.

[MY16] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2016.

Voir page 58.

[NBLD20] Nhu Koha Nguyen, Emanuela Boros, Gaël Lejeune, and Antoine Doucet. Impact analysis of document digitization on event extraction. In

4th workshop on natural language for artificial intelligence (NL4AI 2020) co-located with the 19th international conference of the Italian Association for artificial intelligence (AI* IA 2020), volume 2735, pages 17–28, 2020.

Voir page 75.

[NC20] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.

Voir pages 42 et 43.

[NJPL20] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Findings of the Association for Computational Linguistics : EMNLP 2020, pages 708–718, 2020.

Voir page 107.

[NMCC16] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. International World Wide Web Conferences Steering Committee, 2016.

Voir page 40.

[Nov21] Jekaterina Novikova. Robustness and sensitivity of BERT models predicting Alzheimer's disease from text. In Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021), pages 334–339, Online, November 2021. Association for Computational Linguistics.

Voir page 97.

[NSW01] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. Physical Review E, 64(2), jul 2001.

Voir page 57.

[OBD+22] Sarah Oberbichler, Emanuela Boroş, Antoine Doucet, Jani Marjanen, Eva Pfanzelter, Juha Rautiainen, Hannu Toivonen, and Mikko Tolonen. Integrated interdisciplinary workflows for research on historical newspapers : Perspectives from humanities scholars, computer scientists, and librarians. Journal of the Association for Information Science and Technology, 73(2) :225–239, 2022.

Voir page 75.

[ON15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

Voir page 40.

[OZ18] Kezban Dilek Onal and Ye Zhang. Neural information retrieval : at the end of the early years. Information Retrieval Journal, 2018.

Voir page 39.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, aug 2014.
Voir page 57.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking : Bringing order to the web. 1999.
Voir page 37.

[PBW+19] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking : Scalable tensorflow library for learning-to-rank. ACM, 2019.
Voir page 39.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. Association for Computing Machinery, 1998.
Voir page 34.

[Pen22] Yuanzhe Peng. A survey on modern recommendation system based on big data, 2022.
Voir page 106.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch : An imperative style, high-performance deep learning library, 2019.
Voir page 47.

[PLG+16] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. A study of matchpyramid models on ad-hoc retrieval, 2016.
Voir page 41.

[PMX+13] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pages 143–152, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
Voir page 77.

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. Association for Computational Linguistics, 2018.
Voir page 43.

[PSG19] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT ? Association for Computational Linguistics, 2019.
Voir page 43.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe : Global vectors for word representation. Association for Computational Linguistics, 2014.
Voir page 43.

[PVGR20] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. MAD-X : An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7654–7673, Online, November 2020. Association for Computational Linguistics.
Voir page 82.

[PWS20] Nina Poerner, Ulli Waltinger, and Hinrich Schütze. E-BERT : Efficient-yet-effective entity embeddings for BERT. In Findings of the Association for Computational Linguistics : EMNLP 2020, pages 803–818. Association for Computational Linguistics, November 2020.
Voir page 105.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-BERT : Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.
Voir page 82.

[RG20] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4512–4525, 2020.
Voir page 82.

[RH20] Andrew Runge and Eduard Hovy. Exploring neural entity representations for semantic information. In Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 204–216. Association for Computational Linguistics, November 2020.
Voir page 105.

[RJ76] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. Journal of the American Society for Information Science, 1976.
Voir pages 18, 27, 91, 92, 106 et 112.

[RJ05] Filip Radlinski and Thorsten Joachims. Query chains : Learning to rank from implicit feedback. Association for Computing Machinery, 2005.

Voir page 38.

[RLY+14] Xiang Ren, Jialu Liu, Xiao Yu, Urvashi Khandelwal, Quanquan Gu, Lidan Wang, and Jiawei Han. Cluscite : Effective citation recommendation by information network-based clustering. Association for Computing Machinery, 2014.
Voir page 17.

[RN18] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.
Voir page 43.

[RNMR21] Matteo Romanello, Sven Najem-Meyer, and Bruce Robertson. Optical character recognition of 19th century classical commentaries : the current state of affairs. In The 6th International Workshop on Historical Document Imaging and Processing, pages 1–6, 2021.
Voir pages 79 et 86.

[Rob04] Stephen Robertson. Understanding inverse document frequency : On theoretical arguments for idf. Journal of Documentation - J DOC, 2004.
Voir page 18.

[RP18] Martin Riedl and Sebastian Padó. A named entity recognition shootout for german. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers), pages 120–125, 2018.
Voir page 77.

[RRT+22] Iiro Rastas, Yann Ciarán Ryan, Iiro Tiihonen, Mohammadreza Qaraei, Liina Repo, Rohit Babbar, Eetu Mäkelä, Mikko Tolonen, and Filip Ginter. Explainable publication year prediction of eighteenth century texts with the bert model. In Proceedings of the 3rd Workshop on Computational Approaches to Historical Language Change, pages 68–77, 2022.
Voir pages 75 et 87.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.
Voir page 107.

[RWJ+94] Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. 1994.
Voir pages 35 et 36.

[RYT22] Ryokan Ri, Ikuya Yamada, and Yoshimasa Tsuruoka. mLUKE : The power of entity representations in multilingual pretrained language models. In ACL 2022 (to appear), 2022.
Voir page 78.

[RZ09] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework : Bm25 and beyond. Foundations and Trends in Information Retrieval, 2009.
Voir page 35.

[RZLL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD : 100,000+ questions for machine comprehension of text. Association for Computational Linguistics, 2016.
Voir page 32.

[SA06] Mark Smucker and James Allan. Find-similar : Similarity browsing as a search tool. 2006.
Voir page 17.

[Sal68] G. Salton. Automatic content analysis in information retrieval. 1968.
Voir page 34.

[Sal72] G. Salton. A new comparison between conventional indexing (medlars) and automatic text processing (smart). Journal of the American Society for Information Science, 1972.
Voir page 34.

[Sar75] Tefko Saracevic. Relevance : A review of and a framework for the thinking on the notion in information science. Journal of the American Society for Information Science, 1975.
Voir page 26.

[Sar16] Tefko Saracevic. The notion of relevance in information science : Everybody knows what relevance is. but, what is it really ? Synthesis Lectures on Information Concepts, Retrieval, and Services, 2016.
Voir page 26.

[SB88] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. Inf. Process. Manag., 1988.
Voir page 35.

[Sch19] Robin M. Schmidt. Recurrent neural networks (rnns) : A gentle introduction and overview, 2019.
Voir page 40.

[SDCW20] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter, 2020.
Voir pages 91 et 92.

[SDLT20] Pedro Javier Ortiz Suárez, Yoann Dupont, Gaël Lejeune, and Tian Tian. Sinner@ clef-hipe2020 : Sinful adaptation of sota models for named entity recognition in french and german. In CLEF (Working Notes), 2020.
Voir page 77.

[SDZ+18a] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.
Voir page 115.

[SDZ+18b] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. Open domain question answering using early fusion of knowledge bases and text, 2018.
Voir page 104.

[SG01] Amanda Spink and Howard Greisdorf. Regions and levels : Measuring and mapping users' relevance judgments. J. Am. Soc. Inf. Sci. Technol., 2001.
Voir page 26.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016.
Voir page 45.

[SHG+14] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. Association for Computing Machinery, 2014.
Voir page 40.

[SHH18] Ian Soboroff, Shudong Huang, and Donna K. Harman. Trec 2018 news track overview. 2018.
Voir page 17.

[Shl14] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood, 2014.
Voir page 59.

[Sim65] R. F. Simmons. Answering english questions by computer : A survey. Commun. ACM, 1965.
Voir page 16.

[SKB+17] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017.
Voir page 115.

[SLNK20] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. Association for Computational Linguistics, 2020.
Voir page 44.

[SM15] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, pages 373–382, 2015.
Voir page 107.

[SNB+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. AI Magazine, 29(3) :93, Sep. 2008.
Voir pages 67 et 70.

[SS19] Justin Sybrandt and Ilya Safro. Fobe and hobe : First- and high-order bipartite embeddings, 05 2019.
Voir page 59.

[STFR17] Michael Stauffer, Thomas Tschachtli, Andreas Fischer, and Kaspar Riesen. A survey on applications of bipartite graph edit distance. 05 2017.
Voir page 51.

[Sto06] Nicola Stokes. TREC : Experiment and Evaluation in Information Retrieval Ellen M. Voorhees and Donna K. Harman (editors) (National Institute of Standards and Technology), Cambridge, MA : The MIT Press (Digital libraries and electronic publishing series, edited by William Y. Arms), 2005, x+462 pp ; hardbound, ISBN 0-262-22073-3,. Computational Linguistics, 32(4) :563–567, 12 2006.
Voir page 94.

[SWY75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. Commun. ACM, 1975.
Voir pages 18 et 34.

[Tay53] Wilson L. Taylor. “cloze procedure” : A new tool for measuring readability. Journalism & Mass Communication Quarterly, 1953.
Voir page 45.

[TBC96] Kazem Taghva, Julie Borsack, and Allen Condit. Effects of ocr errors on ranking and feedback using the vector space model. Inf. Process. Manage., 32(3) :317–327, 1996.
Voir pages 91 et 93.

[TBCE94] Kazem Taghva, Julie Borsack, Allen Condit, and Srinivas Erva. The effects of noisy data on text retrieval. Journal of the American Society for Information Science, 45(1) :50–58, 1994.
Voir page 91.

[TC20] Konstantin Todorov and Giovanni Colavizza. Transfer learning for named entity recognition in historical corpora. In CLEF (Working Notes), 2020.
Voir page 77.

[Ten] Tencent records the watching behaviors of users on movies in qqlive. https://v.qq.com/.
Voir pages 67 et 70.

[TGLDS12] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. Etrust : Understanding trust evolution in an online world. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, page 253–261, New York, NY, USA, 2012. Association for Computing Machinery.
Voir pages 67 et 70.

[TGRM08] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank : Optimizing non-smooth rank metrics. In Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08, page 77–86, New York, NY, USA, 2008. Association for Computing Machinery.
Voir page 107.

[TKSDM03] Erik F. Tjong Kim Sang and Fien De Meulder. 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, pages 142–147, 2003.
Voir page 77.

[TVOH15] Myriam C Traub, Jacco Van Ossenbruggen, and Lynda Hardman. Impact analysis of ocr quality on research tasks in digital archives. In International Conference on Theory and Practice of Digital Libraries, pages 252–263. Springer, 2015.
Voir page 93.

[TZY+08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer : Extraction and mining of academic social networks. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, page 990–998, New York, NY, USA, 2008. Association for Computing Machinery.
Voir pages 67 et 70.

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.
Voir page 108.

[Voo02] Ellen Voorhees. Overview of the trec 2001 question answering track. 2002.
Voir page 16.

[Voo04] Ellen Voorhees. Overview of the trec 2004 robust track. 01 2004.
Voir page 42.

[VSBA+20] Daniel Van Strien, Kaspar Beelen, Mariona Coll Ardanuy, Kasra Hosseini, Barbara McGillivray, and Giovanni Colavizza. Assessing the impact of ocr quality on downstream nlp tasks. 2020.
Voir page 92.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
Voir pages 43, 44, 82 et 83.

[WCL13] Hao Wang, Binyi Chen, and Wu-Jun Li. Collaborative topic regression with social regularization for tag recommendation. In IJCAI, 2013.
Voir pages 67 et 70.

[WCY93] S. K. M. Wong, Y. J. Cai, and Y. Y. Yao. Computation of term associations by a neural network. Association for Computing Machinery, 1993.
Voir pages 18, 37 et 39.

[WD19] Shijie Wu and Mark Dredze. Beto, bentz, becas : The surprising cross-lingual effectiveness of BERT. Association for Computational Linguistics, 2019.
Voir page 43.

[WDS+20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers : State-of-the-art natural language processing. Association for Computational Linguistics, 2020.
Voir page 47.

[WGZ+21] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler : A unified model for knowledge embedding and pre-trained language representation. Transactions of the Association for Computational Linguistics, 9 :176–194, 2021.
Voir page 80.

[WJB+20] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. Automated concatenation of embeddings for structured prediction. arXiv preprint arXiv :2010.05006, 2020.
Voir page 76.

[WL12] Robert West and Jure Leskovec. Human wayfinding in information networks. In Proceedings of the 21st International Conference on World Wide Web, WWW '12, page 619–628, New York, NY, USA, 2012. Association for Computing Machinery.

Voir pages 67 et 70.

[WPC+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1) :4–24, jan 2021.

Voir page 60.

[WPP09] Robert West, Joelle Pineau, and Doina Precup. Wikispeedia : An online game for inferring semantic distances between concepts. In IJCAI, 2009.

Voir pages 67 et 70.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system : Bridging the gap between human and machine translation, 2016.

Voir pages 45 et 47.

[WSC+22] Xinyu Wang, Yongliang Shen, Jiong Cai, Tao Wang, Xiaobin Wang, Pengjun Xie, Fei Huang, Weiming Lu, Yueting Zhuang, Kewei Tu, et al. Damo-nlp at semeval-2022 task 11 : A knowledge-based system for multilingual named entity recognition. arXiv preprint arXiv :2203.00545, 2022.

Voir pages 78 et 80.

[WSM+19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue : A multi-task benchmark and analysis platform for natural language understanding, 2019.

Voir page 45.

[WZFC14] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence, 28(1), Jun. 2014.

Voir page 105.

[WZL+17] Meng Wang, Jiaheng Zhang, Jun Liu, Wei Hu, Sen Wang, Xue Li, and Wenqiang Liu. Pdd graph : Bridging electronic medical records and biomedical knowledge graphs via entity linking, 2017.

Voir pages 67 et 70.

[WZY+19] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library :

A graph-centric, highly-performant package for graph neural networks, 2019.

Voir page 69.

[XC18] Yumo Xu and Shay B. Cohen. Stock movement prediction from tweets and historical prices. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers), pages 1970–1979, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Voir pages 67 et 70.

[XCNL21] Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri, and Jens Lehmann. Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, pages 2569–2578, Online, June 2021. Association for Computational Linguistics.

Voir page 78.

[XDC+17] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. ACM, 2017.

Voir page 41.

[XHL20] Jun Xu, Xiangnan He, and Hang Li. Deep learning for matching in search and recommendation. Foundations and Trends® in Information Retrieval, 2020.

Voir page 39.

[XNA+20] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. TeRo : A time-aware knowledge graph embedding via temporal rotation. In Proceedings of the 28th International Conference on Computational Linguistics, pages 1583–1593, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

Voir page 78.

[XPC17] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. International World Wide Web Conferences Steering Committee, 2017.

Voir page 105.

[XSL21] Chengjin Xu, Fenglong Su, and Jens Lehmann. Time-aware relational graph attention network for temporal knowledge graph embeddings. 2021.

Voir page 78.

[Yas05] AI-Onaizan Yaser. Effect of degraded input on statistical machine translation. In 2005 Symposium on Document Image Understanding Technology, page 103, 2005.
Voir page 92.

[YAS+20] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE : Deep contextualized entity representations with entity-aware self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6442–6454, Online, November 2020. Association for Computational Linguistics.
Voir pages 76 et 78.

[YJZL20] Andrew Yates, Kevin Martin Jose, Xinyu Zhang, and Jimmy Lin. Flexible ir pipelines with capreolus. Association for Computing Machinery, 2020.
Voir page 42.

[YL12] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth, 2012.
Voir pages 67 et 70.

[YLKY07] Jen-Yuan Yeh, Jung-Yi Lin, Hao-Ren Ke, and Wei-Pang Yang. Learning to rank for information retrieval using genetic programming. In Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval (LR4IR 2007). Citeseer, 2007.
Voir page 107.

[YLYL19] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the "neural hype" : Weak baselines and the additivity of effectiveness gains from neural ranking models. pages 1129–1132, 07 2019.
Voir page 42.

[YXZ+20] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning : A unified framework with survey and benchmark, 2020.
Voir pages 51 et 55.

[YZL+22] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. A survey of knowledge-enhanced text generation. 2022.
Voir page 105.

[YZP+18] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. Walkranker : A unified pairwise ranking model with multiple relations for item recommendation. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications

of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 2596–2603. AAAI Press, 2018.

Voir page 57.

[ZCF+21] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. Learning from history : Modeling temporal knowledge graphs with sequential copy-generation networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 4732–4740, 2021.

Voir page 79.

[ZHL+19] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE : Enhanced language representation with informative entities. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

Voir page 105.

[ZMO+04] Guowei Zu, Mayo Murata, Wataru Ohyama, Tetsushi Wakabayashi, and Fumitaka Kimura. The impact of ocr accuracy on automatic text classification. In Chi-Hung Chi and Kwok-Yan Lam, editors, Content Computing, pages 403–409, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Voir page 92.

[ZQJ+23] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5 : Fine-tuning t5 for text ranking with ranking losses. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, page 2308–2313, New York, NY, USA, 2023. Association for Computing Machinery.

Voir page 107.

[ZXKZ17] Yao Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. Learning node embeddings in interaction graphs. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, page 397–406, New York, NY, USA, 2017. Association for Computing Machinery.

Voir pages 63 et 71.

Graphes de connaissances pour la représentation du contexte en traitement automatique du langage naturel

Résumé : La numérisation grandissante de nos sociétés induit une prolifération des textes numérisés. Le traitement automatique du langage naturel (TALN) devient alors indispensable pour gérer le nombre croissant de documents numérisés. Le domaine de la recherche d'information a ainsi émergé pour faciliter l'accessibilité des contenus textuels numérisés recherchés par un utilisateur. Ce dernier usage constitue un comportement qui s'est démocratisé dans le monde socio-économique moderne par l'utilisation massive des moteurs de recherche. La majorité des approches traitant les textes écrits, reposent sur un découpage au préalable des textes en unités linguistiques, le mot étant l'exemple d'unité linguistique le plus évident. Cette thèse s'interroge sur l'usage du découpage du texte en unités linguistiques et explore des méthodes plus aptes à capturer la proximité sémantique et contextuelle des textes. C'est pourquoi nous étudions l'apport des graphes pour le TALN et notamment leur capacité à représenter de manière structurée les relations complexes entre les concepts contextuels des textes. Nous nous intéressons d'abord aux plongements de graphes, une technique de représentation compressée permettant d'exploiter la structure de graphe. Nous étudions ensuite le potentiel des graphes de connaissances pour la reconnaissance d'entités nommées, une technique qui caractérise contextuellement un texte. Enfin, nous analysons l'impact du bruit dans le texte sur la recherche d'information, un problème peu étudié dans la littérature. Pour conclure, nous proposons une méthode de recherche d'information reposant sur l'utilisation d'un graphe de connaissances et d'une représentation contextuelle des unités linguistiques.

Mots clés : Traitement automatique du langage naturel ; recherche d'information ; graphes ; apprentissage de représentation ; apprentissage machine.

Knowledge Graphs for context representation in Natural Language Processing

Abstract: The increasing digitization of our societies is leading to a proliferation of digitized texts. Natural Language Processing (NLP) has become indispensable for managing the growing number of digitized documents. The field of Information Retrieval has thus emerged to facilitate the accessibility of digitized textual content sought by an user. The latter is a behavior that has been democratized in the modern socio-economic world by the massive use of search engines. Most approaches to the treatment of written texts are based on a prior breakdown of texts into linguistic units, the word being the most obvious example of a linguistic unit. This thesis examines the use of text breakdown into linguistic units, and explores methods that are better suited to capture the semantic and contextual proximity of texts. This is why we study the benefits of graphs for NLP, and in particular their ability to represent in a structured way the complex relationships between contextual concepts in texts. We first address graph embeddings, a compressed representation technique to exploit graph structure. We then study the potential of knowledge graphs for named entity recognition, a technique that contextually characterizes a text. Finally, we analyze the impact of noise in text on Information Retrieval, a problem little studied in the literature. In conclusion, we propose an Information Retrieval method based on the use of a knowledge graph and a contextual representation of linguistic units.

Keywords: Natural Language Processing ; Information Retrieval ; Graphs ; Representation Learning ; Machine Learning.

Laboratoire Informatique, Image, Interaction
Institut LUDI - La Rochelle Université
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1