Apprentissage de Représentation dans les Réseaux de Documents : Application à la Littérature Scientifique

Robin Brochier

Thèse présentée pour obtenir le grade de
Docteur de l'Université Lumière Lyon 2

École Doctorale Informatique et Mathématiques (ED 512)
Laboratoire ERIC (EA 3083)
Discipline : Informatique

Apprentissage de Représentation dans les Réseaux de Documents : Application à la Littérature Scientifique

Robin BROCHIER

Présentée et soutenue publiquement le 6 juillet 2020, devant un jury composé de :

Christine LARGERON, Professeur des Universités, Université Jean Monnet	Rapportrice
Massih-Reza AMINI, Professeur des Universités, Université Grenoble Alpes	Rapporteur
Maguelonne TEISSEIRE, Directrice de Recherche, INRAE	Examinatrice
Michalis VAZIRGIANNIS, Professeur des Universités, École Polytechnique	Examinateur
Adrien GUILLE, Maître de Conférences, Université Lumière Lyon 2	Co-directeur
Julien VELCIN, Professeur des Universités, Université Lumière Lyon 2	Directeur

Abstract

Representation Learning in Document Networks : Application to the Scientific Literature

The work presented in this thesis, made in collaboration with the company *Digital Scientific Research Technology*, aims to develop representation learning models for networks in order to address the resolution of different tasks of information retrieval, in particular, on data extracted from the scientific literature.

We present GVNR, a network embedding algorithm whose algorithmic time complexity is lower than other representative algorithms in the literature. GVNR-t, its extension, makes it possible to take into account the text associated with the nodes in a network of documents. We then describe MATAN, a model that leverages a mutual attention mechanism between documents. Finally, we present IDNE, a document network embedding model based on a new mechanism, the topic-attention. We experimentally study the performances of these 4 models on transductive and inductive tasks of classification of nodes and of link prediction with 9 datasets. We show that these models achieve state-of-the-art performances in most datasets on all tasks.

In addition, we present our work on expert finding. We introduce a new evaluation methodology and we provide 4 new annotated datasets. We experimentally show the relevance of our evaluation protocol and highlight the remaining steps for the design of an expert finding model based on document network embedding techniques.

Keywords : network embedding ; representation learning ; document network ; information retrieval ; recommender systems ; attention mechanisms

Résumé

Les travaux présentés dans cette thèse, réalisés en collaboration avec l'entreprise *Digital Scientific Research Technology*, ont pour objectif de développer des modèles d'apprentissage de représentation pour les réseaux dans l'optique d'aborder la résolution de différentes tâches de recherche d'information, en particulier sur des données issues de la littérature scientifique.

Nous présentons GVNR, un algorithme de plongement de sommets dans les réseaux dont la complexité algorithmique en temps est plus faible que les algorithmes représentatifs de la littérature. GVNR-t, son extension, permet de prendre en compte le texte associé aux sommets dans un réseau de documents. Nous décrivons ensuite MATAN, un modèle qui entraîne un mécanisme d'attention mutuelle entre documents. Nous présentons enfin IDNE, un modèle d'apprentissage de représentation de documents qui s'appuie sur un nouveau mécanisme, l'attention thématique. Nous étudions expérimentalement les performances de ces 4 modèles sur des tâches transductives et inductives de classification des sommets et de prédiction de liens avec 9 jeux de données. Nous montrons que ces modèles réussissent mieux que l'état de l'art sur la plupart des jeux de données et sur toutes les tâches.

De plus, nous présentons nos travaux sur la recherche automatique d'experts. Nous introduisons une nouvelle méthodologie d'évaluation et nous fournissons 4 nouveaux jeux de données annotés. Nous montrons expérimentalement la pertinence de notre protocole d'évaluation et mettons en lumière les étapes restantes pour la conception d'un modèle de recherche d'experts reposant sur les techniques de plongement de réseau de documents.

Mots-clefs : plongement de réseau ; apprentissage de représentation ; réseau de documents ; recherche d'information ; systèmes de recommandation ; mécanismes d'attention

Remerciements

Je remercie en premier lieu mes encadrants, Julien Velcin et Adrien Guille, pour m'avoir accompagné durant cette thèse. Vos conseils et votre soutien m'ont permis de la réaliser dans les meilleures conditions. Je remercie également les membres du jury Christine Largeron et Massih-Reza Amini en tant que rapporteurs, ainsi que Maguelonne Teisseire et Michalis Vazirgiannis en tant qu'examinateurs, pour avoir accepté d'évaluer ces travaux.

Je tiens à remercier Thibault Honegger et Philippe Chuzel qui m'ont offert l'occasion de poursuivre une thèse CIFRE avec l'entreprise DSRT, dans le cadre du développement de Peerus. Je remercie aussi tous les membres avec qui nous avons partagé cette aventure, particulièrement Benjamin Rothan et François Di Cioccio.

Par ailleurs, je remercie les membres du laboratoire ERIC, tout particulièrement Margot, Antoine, Jean, Clément et Gaël, pour l'excellente ambiance au travail et pour les moments de convivialité passés hors recherche.

Enfin, je remercie ma famille : mes parents, mes sœurs, cousines et cousins, oncles, tantes et grande-tante pour leur soutien constant avec une pensée particulière pour mon frère, Adrien, qui a conforté mon envie de poursuivre un doctorat. Je tiens également à exprimer ma gratitude envers mes amis : ceux de longue date, du lycée, de CPE, de team garage, du hockey et tous ceux qui ont partagé un bout de leur temps avec moi ces trois années durant.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Dans ce chapitre d'introduction, nous présentons tout d'abord le contexte dans lequel s'inscrivent les travaux réalisés dans le cadre de ce doctorat. Ensuite, nous décrivons les notions générales de la théorie des graphes, du traitement automatique des langues et de l'apprentissage de représentation. Ensuite nous décrivons plusieurs jeux de données sur lesquels s'appliquent nos travaux, et décrivons certaines applications possibles. Enfin nous résumons les différentes contributions qui ont été faites durant de cette thèse et présentons le plan de ce manuscrit.

Sommaire

1.1 Contexte de cette thèse

La littérature scientifique forme un large réseau d'information reliant des acteurs variés (laboratoires, entreprises, institutions, etc.). La quantité quotidienne de productions scientifiques se faisant de plus en plus grande, il est devenu particulièrement difficile d'en extraire du contenu d'intérêt pour résoudre une tâche donnée, comme la recherche d'information ou la recommandation automatique. Dans ce contexte, l'entreprise *Digital Scientific Research Technology* (*DSRT*), dans laquelle s'est déroulée cette thèse, a développé à partir de 2015 une plate-forme internet d'agrégation et de filtrage des articles scientifiques. Son application Web, *Peerus*, a pour objectif de présenter chaque jour à ses utilisateurs les publications, auteurs et journaux susceptibles de les intéresser. Dans ce cadre, les travaux présentés dans ce manuscrit ont pour objectif d'améliorer la construction d'un système de recherche d'information (RI) pour la littérature scientifique. Un tel système de RI peut trouver application dans divers domaines tels que la veille scientifique (recommandation d'articles, système d'alerte, identification de brevets), la scientométrie (bibliométrie, identification de tendances et d'impacts) et la recherche d'experts (collaboration, évaluation par les pairs, appel à projet, recrutement).

Un réseau, que l'on modélise en langage formel mathématique comme un graphe, est un ensemble d'éléments, appelés sommets, avec des connexions entre eux, appelées liens. Les systèmes prenant la forme de réseaux abondent dans le monde. On peut citer le *World Wide Web*, les réseaux sociaux, les réseaux organisationnels, les réseaux de distribution tels que les vaisseaux sanguins ou les voies de livraison postale, les réseaux de citations entre articles scientifiques, et bien d'autres encore. L'étude de ces réseaux [Newman, 2003], reposant sur des méthodes statistiques variées, peut grandement bénéficier des fortes capacités de calcul des ordinateurs. Ces derniers permettent en effet de collecter et d'analyser des données à grande échelle. En outre, une propriété qui semble être commune à de nombreux réseaux est la structure en communautés, c'est-à-dire la division des sommets de ces réseaux en groupes au sein desquels les connexions sont denses, mais entre lesquels elles se font plus rares. S'il est facile d'identifier ce type de structure pour des petits réseaux, composés au maximum d'une centaines d'éléments, l'utilisation d'algorithmes spécifiques [Newman and Girvan, 2004] est indispensable lorsque l'on s'intéresse à des réseaux plus grands. En particulier, la littérature scientifique est un immense réseau de collaborations et de citations dont l'analyse algorithmique peut répondre à de nombreuses questions [Kas, 2011]. On estime à plus de 150 millions le nombre d'articles référencés sur l'un des moteurs de recherche les plus utilisés [Orduña-Malea et al., 2015]. Le marché de la publication scientifique grandit chaque année et est estimé à 25,7 milliards de dollars en 2017 [Johnson et al., 2018]. Les algorithmes d'infométrie [Egghe and Rousseau, 1990], dont l'objectif est de traiter et de rendre digeste différents aspects quantitatifs de l'information, ont alors un fort potentiel d'application dans le monde académique.

On peut décrire les données scientifiques comme un réseau hétérogène attribué (RHA). L'hétérogénéité est due au fait que les sommets et les liens constituant son graphe sont de types variés. Les attributs représentent l'ensemble des informations autres que celle de la structure du réseau qui sont associées aux sommets et aux liens (texte, dates, images etc.). Exploiter l'information contenue dans un RHA nécessite de manipuler au moins deux types de données aux formalismes mathématiques différents : celui des graphes et celui des attributs. Les données scientifiques constituent une instance particulière des RHA, celle des réseaux hétérogènes de documents, omniprésents sur le Web. Les forums en ligne, les sites d'information, les e-mails et plus généralement les réseaux sociaux ont en commun avec la littérature scientifique leurs structures de graphe, où des documents (principalement textuels) sont liés entre eux par divers acteurs (utilisateurs, auteurs, etc.) et par divers types de lien (citations, interactions, etc.).

Il existe de nombreuses méthodes éprouvées permettant de réaliser des systèmes de recherche d'information dans des bases de données. Celles-ci reposent souvent sur des algorithmes d'apprentissage automatique résolvant diverses tâches telles que la classification, la régression ou encore le partitionnement des données. Ces algorithmes opèrent généralement dans des espaces vectoriels de dimension finie. Pour tirer profit de ces techniques, il est donc nécessaire de décrire les données que l'on étudie dans ce type espace. L'apprentissage de représentation consiste précisément à construire automatiquement des descriptions des données dans ces espaces tout en conservant certaines de leurs propriétés. Cette approche rend non seulement l'utilisation d'algorithmes usuels possible, par exemple en décrivant les sommets d'un graphe par des vecteurs, mais elle peut aussi en améliorer les performances (mesures d'évaluation, complexité algorithmique, espace mémoire, etc.).

L'approche suivie dans cette thèse est de développer des modèles d'apprentissage de représentation pour les réseaux de documents afin de construire des systèmes de recherche d'information performants. Ces modèles sont testés sur différents réseaux de documents, dont particulièrement des réseaux de publications scientifiques, et sont évalués sur différentes tâches de recherche d'information telles que la classification des sommets, la prédiction de liens et la recherche d'experts.

Dans la suite de ce chapitre, nous présentons en premier lieu les notions générales autour de l'analyse des graphes, du traitement du langage naturel et de l'apprentissage de représentation. Nous décrivons ensuite des cas d'applications et des jeux de données représentés par des réseaux de documents.

1.2 Notions générales

L'analyse des réseaux de documents croisent deux domaines très étudiés dans la littérature : l'analyse des graphes et le traitement automatique des langues, domaines dont nous décrivons les concepts principaux dans cette section. Nous introduisons ensuite les notions fondamentales de l'apprentissage de représentation.

1.2.1 Théorie des graphes

Un graphe $G = (V, E)$ est une structure décrite par deux ensembles V, l'ensemble des n_v sommets, et E, l'ensemble des n_e liens définis par des paires de sommets $(v_i, v_j) \in V \times V$. Un graphe est dit pondéré lorsque ses liens sont associées à des poids $a_{ij} \in \mathbb{R}$. On notera alors un lien par le triplet (v_i, v_j, a_{ij}). Un graphe est dit orienté lorsqu'il existe une notion de sens pour les liens, c'est-à-dire que (v_i, v_j) et (v_j, v_i) désignent deux arcs distincts et plus généralement, pour un graphe pondéré, on peut avoir $a_{ij} \neq a_{ji}$. On note A la matrice d'adjacence du graphe G dont les entrées sont données par les valeurs a_{ij}. Pour un graphe non orienté on a $a_{ij} = a_{ji}, \forall (i, j) \in [\![1, n_v]\!]^2$ et A est alors symétrique. Pour un graphe non pondéré, $a_{ij} \in \{0, 1\}$ et A est dite binaire. La densité d'un graphe est le rapport entre le nombre de liens reliant les sommets et le nombre de liens possibles soit $\frac{n_e}{n_v^2}$. Lorsque ce rapport est faible, on dit que la matrice d'adjacence A est creuse car un nombre important de ses valeurs sont nulles, ce qui signifie que la majorité des sommets ne sont connectés qu'à un nombre très faible de voisins.

Un graphe hétérogène possède plusieurs types de sommets et/ou plusieurs types de liens. On peut alors le décrire avec plusieurs graphes regroupant chacun un unique type de liens et/ou un sous-ensemble des sommets. Un graphe attribué est noté $G = (V, E, X)$ où X est un ensemble d'attributs associés aux sommets V du graphe. Nous ne considérons pas le cas où des attributs sont associés aux liens du graphe. La figure 1.1 montre un graphe hétérogène attribué, hypothétique, représentant un réseau de publications scientifiques.

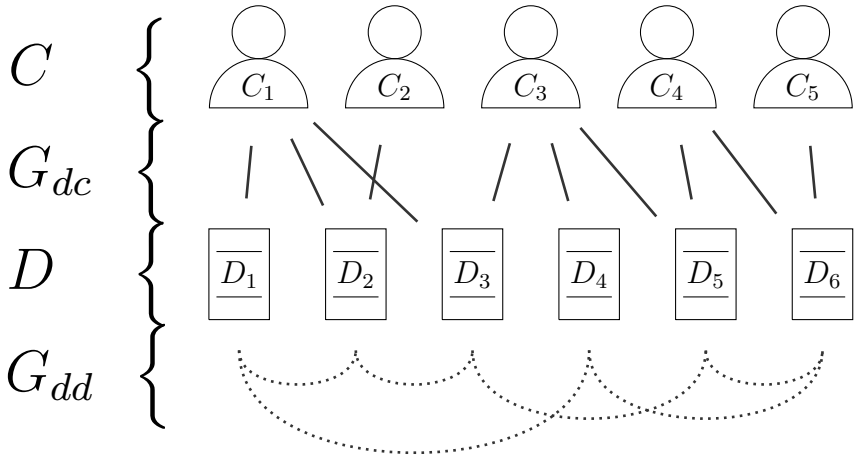


FIGURE 1.1 – Hypothétique réseau de publication. On peut représenter ce graphe hétérogène attribué par deux graphes G_{dc} regroupant les liens de co-auteurs entre les sommets « auteurs » C et les sommets « articles » et G_{dd} regroupant les liens de citations entre les sommets « articles ». De plus, le contenu textuel des articles peut être représenté par une matrice d'attributs X.

On dit que deux sommets v_i et v_j sont connectés si et seulement si $a_{ij} > 0$. Le voisinage direct d'un sommet v_i est l'ensemble $\{v_j \ | a_{ij} > 0\}$. On dit qu'il existe un chemin entre $v_{k_1} = v_i$ et $v_{k_\ell} = v_j$ si et seulement s'il existe une suite $(v_{k_1}, v_{k_2}, ..., v_{k_{\ell-1}}, v_{k_\ell})$ tel qu'il existe un lien entre chaque paire de sommets consécutifs. On dit alors que cette séquence est un chemin de longueur ℓ. On définit le voisinage d'ordre ℓ d'un sommet v_i comme l'ensemble des sommets v_j tels qu'il existe au moins un chemin de longueur inférieure ou égale à ℓ reliant v_i à v_j. La distance géodésique entre deux sommets est définie comme la longueur du chemin le plus court les reliant.

Dans un graphe non pondéré, le nombre de chemins de longueurs inférieures ou égales à ℓ s'obtient par la formule $(A + A^2 + ... + A^\ell)$, où A^k définit la $k^{\text{ième}}$ puissance de la matrice d'adjacence. $(a^k)_{ij}$ désigne alors le nombre exact de chemins distincts de longueurs k depuis le sommet v_i vers le sommet v_j.

Une marche aléatoire sur un graphe, dont un exemple est donné en figure 1.2, est un moyen d'approcher le calcul de $A + A^2 + ... + A^\ell$. Cela consiste à simuler plusieurs marches à partir de chaque sommet d'un graphe. A chaque pas d'une marche, le prochain sommet à visiter est uniformément tiré suivant les probabilités engendrées par les poids du voisinage direct du sommet courant. Pour un sommet donné v_i, cela revient à estimer la probabilité stationnaire sur l'ensemble des sommets de la chaîne de Markov engendrée par la matrice d'adjacence du graphe. Pour garantir la convergence de ce processus itératif, on peut considérer des marches aléatoires avec un facteur d'amortissement γ. Ceci consiste à tirer avec une probabilité $1 - \gamma$ une chance d'effectuer un bond plutôt qu'un nouveau pas. Un bond peut signifier de téléporter le marcheur sur n'importe quel nouveau sommet ou de revenir sur le sommet de départ. Les marches aléatoires sont abordées en détail en section 2.3.2.

1.2.2 Traitement automatique des langues

Le traitement automatique des langues rassemble l'ensemble des techniques employées pour rendre exploitable par la machine des contenus textuels. Ceci inclut entre autres l'extraction d'information, la catégorisation de documents ou la traduction automatique. Nous définissons dans cette section les différentes techniques couramment utilisées pour

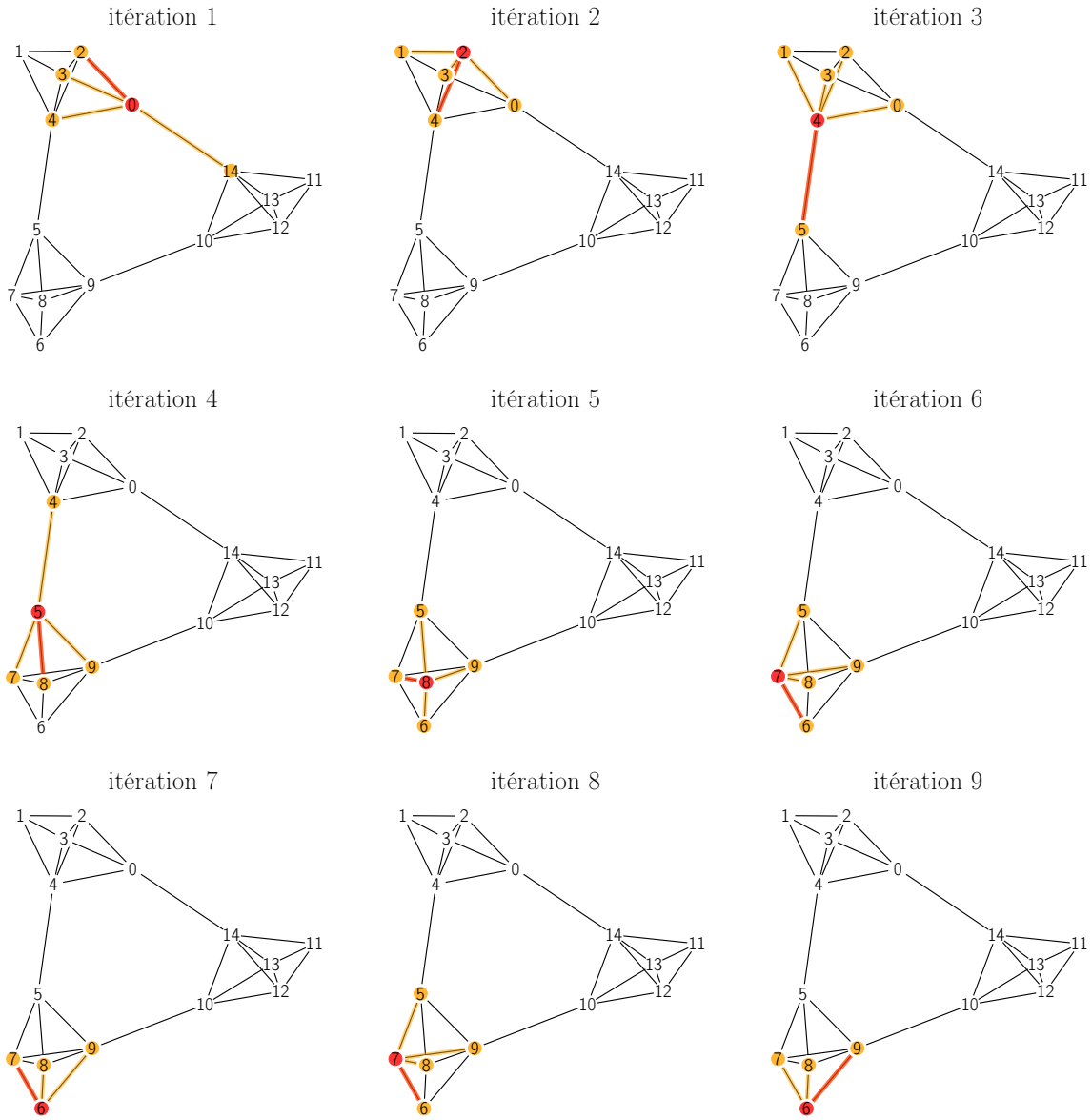


FIGURE 1.2 – Exemple d'une marche aléatoire sur un graphe constitué de 3 communautés. Le sommet visité est en rouge et ses voisins sont en orange. À chaque itération, le marcheur sélectionne aléatoirement le prochain sommet à visiter parmi les voisins. On voit que la marche a tendance à rester au sein d'une communauté. Cependant, comme c'est le cas en itération 3, la marche peut passer d'une communauté à une autre.

transformer un texte en une représentation mathématique exploitable par des algorithmes.

Construction d'un vocabulaire

Un premier traitement que l'on opère sur un corpus de n_d documents consiste à découper les séquences de mots en séquences de jetons (*tokens*). On crée ainsi un vocabulaire de n_ω jetons et l'on peut représenter chaque document comme une suite de longueur ℓ $(\omega_1, ..., \omega_\ell)$ avec $\omega_i \in [\![1, n_\omega]\!]$. La sélection de ces jetons est heuristique et consiste souvent à isoler les mots. On peut aussi utiliser des techniques (comme la *lemmatisation* ou le *stemming*) plus complexes permettant de regrouper sous un même jeton des mots partageant la même racine tels que « documenter » et « documentation » que l'on regroupera sous le jeton « document ». Il existe aussi des méthodes qui apprennent à découper les mots en sous-unités [Sennrich et al., 2015, Kudo, 2018]. De plus, on peut considérer des compositions de jetons, appelés *n-grams*, en enrichissant le vocabulaire avec toutes les compositions possible de $n = 2, 3...$ jetons. Cette technique augmentant considérablement le nombre de jetons dans le vocabulaire, on peut chercher à ne composer que les jetons qui co-occurrent particulièrement ensemble et à les traiter comme des collocations [Bouma, 2009]. Dans la suite de ce rapport, nous utiliserons les expressions « terme » et « mot » indifféremment, en lieu et place de « jeton », considérant que les données ont été préalablement traitées de sorte à les décrire à travers un vocabulaire fixe. La figure 1.3 donne un exemple simple de la construction d'un vocabulaire à partir d'un corpus de deux documents.

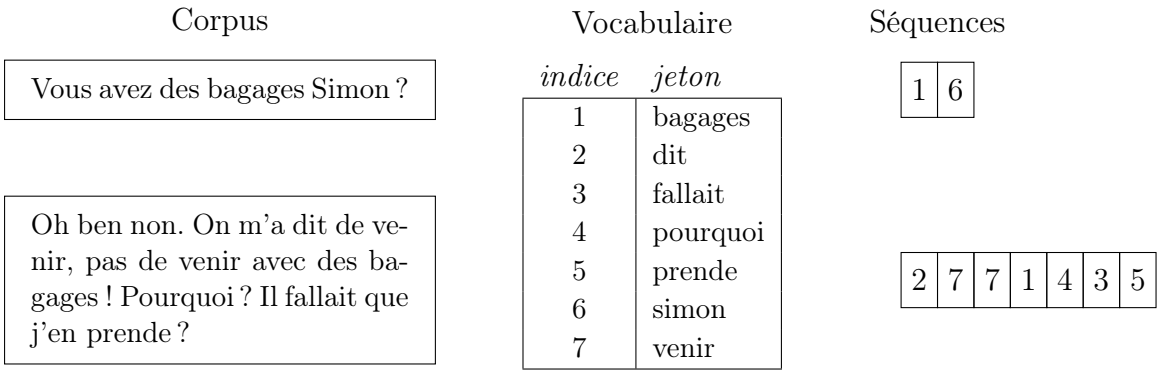


FIGURE 1.3 – Exemple d'un corpus de deux documents à partir desquels nous construisons un vocabulaire. Ces documents peuvent alors être décrits comme des séquences d'indices des mots du vocabulaire.

Par la suite on peut appliquer un pré-traitement sur ces séquences pour réduire la taille du vocabulaire. Une méthode consiste à simplement supprimer des mots vides que l'on aura préalablement déterminés. Ces mots vides sont des mots qui apparaissent souvent mais qui ne portent pas intrinsèquement de sens, tels que « et » ou « pas ». Une seconde méthode consiste à enlever les mots n'apparaissant pas assez fréquemment ou au contraire trop fréquemment. Les mots rares occupent souvent une grande partie du vocabulaire et ne surviennent que trop rarement dans les séquences pour être utiles dans une tâche de recherche d'information. Les mots très fréquents, quant à eux, occupent une grande partie des séquences de jetons et sont trop omniprésents pour être discriminants. Notons que pour certaines tâches, telle que la traduction automatique, ce type de pré-traitement est à proscrire. En effet, un mot très courant et n'ayant pas de sens intrinsèque tel que « pas » peut changer significativement le sens de la phrase.

Représentation par sacs de mots

Indépendemment des techniques de pré-traitement utilisées, on représente un corpus de textes comme un ensemble de documents composés de séquences de jetons issus d'un vocabulaire fini. Cette représentation conserve l'ordre des mots. On peut cependant réaliser une simplification supplémentaire en représentant les documents comme des sacs de mots. On construit alors un vecteur de dimension n_ω où chaque entrée j correspond au nombre d'occurrences dans le document du mot ω_j du vocabulaire, autrement appelée fréquence brute du mot ω_j. On peut par la suite représenter l'ensemble du corpus par une matrice X de dimensions $n_d \times n_\omega$ où l'entrée x_{ij} est le nombre d'apparitions du mot ω_j dans le document d_i. Lorsque chaque ligne de X est normalisée, en la divisant par le nombre de mots dans le document d_i, on obtient des fréquences relatives. Dans ce manuscrit, nous utiliserons la dénomination TF de l'anglais *Term Frequency* pour désigner des fréquences relatives.

Cette représentation matricielle des sacs de mots donne la capacité d'utiliser les outils mathématiques propres aux espaces vectoriels euclidiens. Il est notamment possible d'effectuer entre documents des calculs de similarités, opérations centrales dans les algorithmes d'apprentissage automatique. La similarité cosinus est fréquemment utilisée. Considérant deux vecteurs documents x_1 et x_2, leur similarité cosinus correspond au cosinus de l'angle θ entre eux $\cos(\theta) = \frac{x_1 \cdot x_2}{||x_1||_2 \times ||x_2||_2}$. Dans la mesure où les valeurs x_{ij} sont positives, la similarité cosinus varie de 0 (aucun mots en commun, les documents sont totalement dissimilaires) à 1 (les deux documents partagent les même mots et sont alors parfaitement similaires). De plus, on normalise (par la norme 2) souvent la représentation TF de sorte que $||x_1||_2 = ||x_2||_2 = 1$. Dans ce cas, la similarité cosinus est égale à la distance euclidienne à une constante près. La figure 1.4 donne un exemple de représentation document-terme et d'un calcul de similarité cosinus.

$$\begin{array}{c} \begin{array}{ccccccc} \textit{bagages} & \textit{dit} & \textit{fallait} & \textit{pourquoi} & \textit{prende} & \textit{simon} & \textit{venir} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 2 \end{pmatrix} \begin{array}{c} x_1 \\ x_2 \end{array} \end{array}$$

$$||x_1||_2 = \sqrt{2}, \quad ||x_2||_2 = \sqrt{9}, \quad \cos(\theta) = \frac{x_1 \cdot x_2}{||x_1|| \times ||x_2||} = \frac{1}{\sqrt{18}} \approx 0.24.$$

FIGURE 1.4 – Exemple d'une matrice document-terme d'un corpus constitué de deux documents, décrits précédemment en figure 1.3. Le calcul de la similarité cosinus entre ces deux documents s'appuie sur le produit scalaire entre x_1 et x_2, et les normes euclidiennes $||x_1||_2$ et $||x_2||_2$.

Ces représentations vectorielles permettent d'appliquer un large éventail de techniques d'apprentissage automatique sur des corpus de documents. Néanmoins, elles souffrent de plusieurs désavantages sur lesquels nous revenons en section 3.3.1 lorsque nous abordons en détail l'état de l'art sur l'apprentissage de représentation des documents.

1.2.3 Apprentissage de représentation

Le succès des algorithmes d'apprentissage artificiel dépend principalement des représentations des données sur lesquelles ils sont appliqués. L'apprentissage de représentation

(AR) [Bengio et al., 2013] s'oppose à la construction manuelle de caractéristiques des données en apprenant automatiquement des descriptions qui rendent leur analyse plus efficace. En d'autres termes, plutôt que de construire à la main des représentations des données en utilisant des connaissances expertes, l'AR désigne une approche différente où l'on va construire ces représentations par un algorithme d'apprentissage optimisant un ou plusieurs critères sur les données.

Un critère couramment utilisé en apprentissage de représentation du texte s'appuie sur l'hypothèse distributionnelle. Celle-ci stipule que des mots apparaissant dans les mêmes contextes linguistiques partagent des significations similaires. Ainsi, partant d'une représentation symbolique du texte (séquences de jetons) et transcrivant cette hypothèse dans un espace euclidien muni d'une mesure de similarité, nous sommes en mesure de construire des vecteurs denses associés aux mots du vocabulaire en rapprochant les mots apparaissant dans les mêmes contextes linguistiques. Cette méthodologie est à l'origine des méthodes de plongement de mot (*word embedding*) [Mikolov et al., 2013a] qui a ensuite été étendue au plongement de réseau [Perozzi et al., 2014]. Utilisées sur de grandes quantités de données, les plongements de mot sont devenues les représentations majoritairement utilisées en traitement automatique des langues, remplaçant progressivement l'utilisation des thésaurus, listes de descripteurs construites manuellement.

Les techniques d'apprentissage de représentation pour le texte et pour les réseaux sont intimement liés. En effet, ces deux types de données sont naturellement représentées par des ensembles fini d'éléments (ex : mots et sommets) dont on peut mesurer des similarités deux à deux (ex : nombre de co-occurrences entre mots et nombre de marches où apparaissent des sommets). Il n'est ainsi pas étonnant que les mêmes méthodes de représentations, tels que les plongements ou les mécanismes d'attention, soient appliquées à ces deux types de données. Les avantages des méthodes d'apprentissage de représentation sont multiples :

- les représentations distribuées : les identifiants uniques représentant les mots d'un vocabulaire engendrent des données creuses, c'est-à-dire des points isolés dans un espace vaste et principalement vide. En forçant les représentations des mots à partager des dimensions en commun, par exemple en choisissant un nombre de dimensions très inférieur au nombre de mots dans le vocabulaire, l'espace devient plus restreint et chaque dimension peut capturer certaines régularités distribuées entre les mots ;
- les dépendances linéaires : de nombreux algorithmes d'apprentissage automatique recherchent des frontières de décision linéaires. Un objectif courant en AR est de représenter les données dans un espace où les facteurs de variations qui nous intéressent s'expriment selon des opérations linéaires. De la sorte, il est possible d'appliquer des techniques éprouvées, tel que l'algorithme de régression logistique, pour la résolution de différentes tâches d'apprentissage automatique ;
- les relations locales et globales : très souvent, les hypothèses sur lesquelles l'apprentissage est réalisé sont locales, c'est-à-dire qu'elle ne nous permettent d'observer les interactions d'un élément qu'avec un nombre limité d'autres éléments, souvent définis comme son voisinage. Une force de l'AR est de pouvoir capturer des relations globales entre éléments dont les interactions ne sont pas directement observées, en tirant seulement profit d'une succession d'observations locales.

Un cas simple et largement utilisé d'algorithme de plongement de réseau est l'algorithme de Fruchterman-Reingold [Fruchterman and Reingold, 1991]. Celui-ci permet de représenter les sommets dans un espace vectoriel à deux dimensions, rendant ainsi la visualisation d'un réseau plus digeste. C'est une méthode itérative qui simule, comme un modèle physique, une attraction des sommets connectés et une répulsion latente qui vise à séparer toute paire de sommets. Comme le montre la figure 1.5, l'algorithme place les

sommets dans le plan, après quelques itérations, de sorte que les communautés soient distinctement visibles. Si cet algorithme est pratique pour la visualisation de petits réseaux, sa complexité en $\mathcal{O}({n_v}^3)$ ne permet pas de passer à l'échelle des grands réseaux et ne montre que peut d'intérêt pour d'autres tâches telles la classification supervisée des sommets.

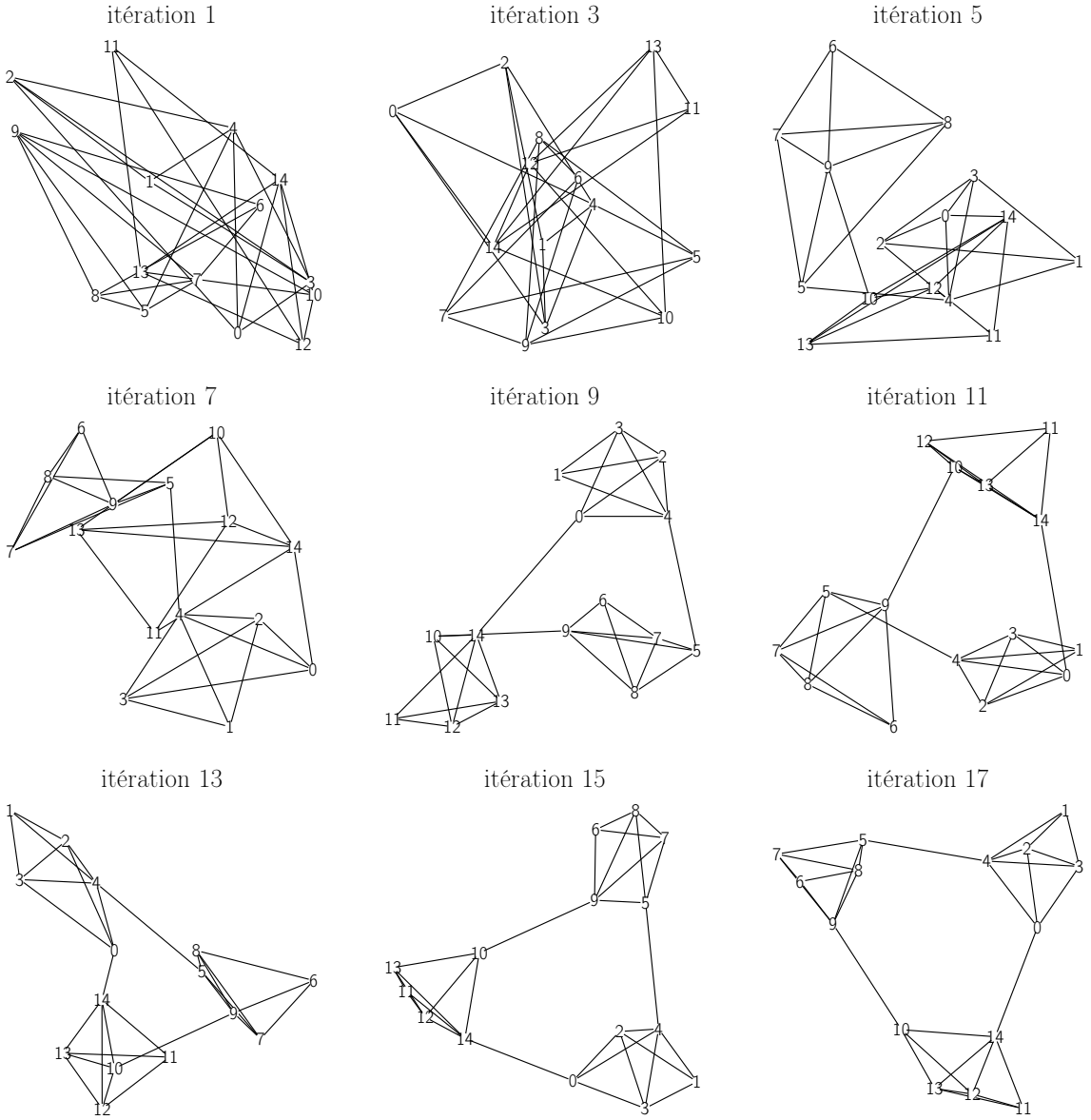


FIGURE 1.5 – Application de l'algorithme de Fruchterman-Reingold [Fruchterman and Reingold, 1991] de dessin de graphe. Afin de représenter en 2 dimensions ce graphe, constitué de 3 communautés, la position de chaque sommet est itérativement mise à jour en assimilant les liens à des ressorts rapprochant les sommets connectés et en éloignant les sommets non connectés. Après 17 itérations, la structure globale du graphe devient lisible.

1.3 Notations

Dans ce manuscrit, nous nous intéressons aux réseaux de documents. Nous décrivons un graphe attribué $G = (V, E, X)$ par sa matrice d'adjacence A et sa matrice d'attributs X correspondant à une certaine représentation des documents que nous détaillerons. Nous notons n_v le nombre de sommets dans le réseaux, n_e le nombre de liens et n_ω le nombre de mots dans le vocabulaire. Nous notons n_d le nombre de documents même si, dans le cadre des réseaux de documents, nous avons toujours $n_d = n_v$. Nous désignons par U et H des représentations en faible dimension que l'on cherche à apprendre et notons leur dimension ρ. Nous rassemblons dans le tableau 1.1 les notations utilisées dans ce manuscrit.

Table 1.1 – Notations utilisées dans ce manuscrit.

Notation	Description
v_i	Sommet i d'un graphe.
d_i	Document i d'un corpus.
ω_i	Mot i d'un vocabulaire.
n_x	Nombre d'éléments dans un ensemble. On aura par exemple n_v sommets, n_d documents et n_ω mots.
ρ_x	Dimension d'une représentation donnée. On aura ρ_v la dimension des représentations des sommets d'un graphe, ρ_d celle des documents et ρ_ω celle des mots.
$A \in \mathbb{N}^{+n_v \times n_v}$	Matrice d'adjacence d'un graphe qui modélise un réseau. En la normalisant en ligne par la norme 1, on obtient la matrice de transition P du graphe.
$X \in \mathbb{R}^{n_d \times n_\omega}$	Matrice d'attributs, souvent une matrice document-terme, TF ou TF-IDF.
$W \in \mathbb{R}^{n_d \times \rho_w}$	Matrice des plongements des mots.
$U \in \mathbb{R}^{n_v \times \rho_v}$	Matrice des plongements des sommets. Lorsque nous avons deux matrices de plongements, nous utiliserons aussi la notation H.
ℓ	Désigne une longueur telle que celle d'une marche aléatoire ou celle d'un document (son nombre de mots).
τ	Désigne la taille d'une fenêtre qui permet de définir le voisinage (ou contexte) de mots ou de sommets.
$X, x_{i.}, x_{.j}, , x_{ij}$	Les matrices sont notées en majuscule, leurs éléments sont notés x_{ij}. La $i^{\text{ème}}$ ligne est notée $x_{i.}$ et la $j^{\text{ème}}$ colonne est notée $x_{.j}$. Nous utiliserons souvent la notation x_i pour désigner un vecteur correspondant à la $i^{\text{ème}}$ ligne de X.
$\|x\|, \|X\|$	Norme 1 d'un vecteur ou d'une matrice, égale à la somme de ses valeurs absolues. Dans le cas d'un graphe G, son volume est noté vol(G) et est égal à la somme des poids de ses arrêtes $\|A\|$.
$\|\|x\|\|, \|\|X\|\|$	Norme 2 (euclidienne) d'un vecteur ou norme de Frobenius d'une matrice, égale à la racine carrée de la somme des carrés de ses valeurs.
$\|X\|_{\neq 0}$	Nombre de valeurs non nulles dans X. Dans le cas d'un graphe, le nombre de valeurs non nulles de la matrice d'adjacence $\|A\|_{\neq 0}$ est égal au nombre de liens du réseau n_e.

Plusieurs algorithmes que nous étudions utilisent des méthodes d'optimisation des paramètres reposant sur l'algorithme du gradient stochastique, où la mise-à-jour des paramètres est effectuée pour chaque individu, ou chaque lot (*mini-batch*) d'individus. Nous utilisons le terme « itération » pour désigner une étape consistant à calculer le gradient pour un lot puis à modifier les paramètres du modèle. Nous utilisons le terme « époque »

(*epoch* en anglais) lorsque l'intégralité des individus ont été parcourus par l'algorithme. Ainsi, si un modèle est entraîné avec des lots constitués de 16 individus, une itération calcule les gradients moyens pour ces 16 individus et réalise une mise à jour des paramètres. Une époque consiste à réaliser $\frac{n}{16}$ itérations, où n est le nombre d'individus dans le jeu de données. L'entraînement complet des paramètres peut être fait en réalisant plusieurs époques.

1.4 Cas d'application

L'apprentissage de représentation de documents en réseau est utile pour différentes applications. Nous décrivons dans cette section les tâches communément réalisées avec de telles représentations et présentons les jeux de données utilisés dans nos travaux.

1.4.1 Tâches à résoudre

Les représentations de documents en réseau sont couramment utilisées pour la résolution de 4 tâches : la classification des sommets/documents, la prédiction de liens, le *clustering* et la visualisation de réseau.

La classification des sommets d'un réseau [Lu and Getoor, 2003, Sen et al., 2008] consiste à associer chaque sommet à une (cas multi-classes) ou plusieurs (cas multi-*labels*) classes étant donné un petit ensemble d'annotation des sommets. Ce processus, dans le cadre du plongement de réseau, s'effectue en trois étapes : (1) des représentations des sommets sont apprises par un algorithme de plongement de réseau puis (2) un modèle de classification linéaire est entraîné sur l'ensemble des représentations des sommets annotés et enfin (3) ce modèle prédit les classes des sommets dont on ne connaît pas les *labels*. La résolution de cette tâche permet d'effectuer à grande échelle des annotations de données en réseau à partir d'ensembles limités d'annotations manuelles. Par exemple, pour un réseau de citation scientifique, on peut chercher à prédire automatiquement les principaux thèmes des publications.

Le *clustering* est analogue à la classification des sommets, sauf qu'aucune annotation n'est disponible pour entraîner un modèle de classification supervisée. De plus, l'évaluation d'un partitionnement des sommets se révèle être plus compliquée que dans le cadre supervisé de la classification. Dans un souci de faisabilité, nous n'explorons dans ce manuscrit que des tâches supervisées, délaissant ainsi le *clustering*.

La prédiction de liens [Liben-Nowell and Kleinberg, 2007, Lü and Zhou, 2011] consiste à identifier les paires de sommets d'un réseau susceptibles d'être connectées. Dans le cadre du plongement de réseau, ce processus consiste à (1) apprendre des représentations des sommets par un algorithme de plongement de réseau puis (2) prédire l'existence des liens entre paires de sommets en se basant sur un calcul vectoriel entre leurs représentations respectives, tels que le produit scalaire ou la similarité cosinus. La résolution de cette tâche permet de réaliser des recommandations. Par exemple, en utilisant un réseau de coauteurs de publications scientifiques, il est possible d'identifier les chercheurs qui sont les plus à même de collaborer ensemble en prédisant des liens encore inexistants.

Enfin, la visualisation de réseau, telle que présentée en figure 1.5, consiste à générer des représentations lisibles d'un réseau en deux ou trois dimensions. Les algorithmes de plongement de réseau sont souvent utilisés en conjonction d'un modèle de réduction de dimension tels que t-SNE [Maaten and Hinton, 2008] et UMAP [McInnes et al., 2018] afin de donner une vue d'ensemble des communautés d'un réseau. Nous présentons de telles visualisations obtenues avec nos modèles GVNR et IDNE, en sections 2.6.2 et 3.6.3.

Nous détaillons plus en détail, dans les chapitres suivants en sections 2.5 et 3.5, les tâches que nous abordons pour l'évaluation de nos modèles. Nous présentons ci-après les

jeux de données que nous avons utilisés pour l'évaluation de la résolution de ces tâches.

1.4.2 Jeux de données

Il existe de nombreuses sources de données organisées en réseaux. Nous présentons ici les différents réseaux que nous avons utilisés comme cas d'application pour tester l'efficacité de nos approches. On peut classer l'origine de ces réseaux selon quatre groupes : les réseaux de citations, les réseaux sociaux, les réseaux du langage et les réseaux biologiques. Le tableau 1.2 présente les propriétés générales des réseaux étudiés dans ce manuscrit.

TABLE 1.2 – Propriétés générales des réseaux étudiés dans ce manuscrit.

	# sommets	# liens	# labels	# mots	# mots par doc	densité	multi-*label*
Cora	2 211	4 771	7	4 333	67 ± 32	0.20%	non
NYT	5 135	3 050 513	4	5 748	24 ± 17	23.14%	non
Gaming SE	22 872	400 664	40	15 760	53 ± 74	0.15%	oui
Travel SE	15 087	465 696	60	14 539	70 ± 73	0.41%	oui
CiteSeer	3 312	4 551	6	3 505	31 ± 5	0.08%	non
Wikipedia	4 777	92 295	40	-	-	0.81%	oui
PPI	3 890	37 845	50	-	-	0.50%	oui
Stats SE	14 834	283 885	59	15 421	103 ± 119	0.26%	oui
Academia SE	20 799	622 720	55	14 034	76 ± 67	0.29%	oui

Réseaux de citations

Nous utilisons 3 réseaux de citations. Cora [McCallum et al., 2000] et CiteSeer [Giles et al., 1998] sont des réseaux de citations scientifiques couramment étudiés. Les sommets de ces réseaux sont des articles et les liens sont des citations. Les documents sont constitués des titres et résumés des articles. Les articles de ces réseaux sont associés à des thématiques de recherches pour Cora et à des conférences pour CiteSeer, qui nous servirons de *labels* pour la classification. Le New York Time [1] (NYT) [Gourru et al., 2020] rassemble les titres des articles de janvier 2007 issus du célèbre quotidien new-yorkais dont les liens ont été construits en connectant les articles associés aux mêmes rubriques (business, arts, technologie, etc.). À chaque article est associé un *label* correspondant à la section dans laquelle il est apparu (opinion, actualités, etc.).

Réseaux sociaux

Nous utilisons les données issues de *Stack Exchange* (SE), un réseau de sites de questions et réponses (Q&A) à édition collaborative. Chaque site traite d'un thème spécifique à l'instar de *Stack Overflow* [2] dédié au développement informatique. Chaque utilisateur de ces plateformes peut poser des questions ou répondre à celles-ci. La qualité des réponses est collaborativement évaluée grâce à un système de votes des utilisateurs. Nous travaillons particulièrement avec quatre communautés. Gaming SE [3] concerne l'univers des jeux vidéo, Travel SE [4] l'univers des voyages, Academia SE [5] s'intéresse au monde de la recherche scientifique et Stats SE [6] porte sur les statistiques et l'analyse de données. Pour chaque communauté, nous ne considérons que les questions qui ont reçu plus de dix votes et dont au moins une réponse a reçu au moins dix votes. Nous construisons un réseau de documents en connectant les questions avec leurs réponses et en connectant les

1. `https://www.nytimes.com/`
2. `https://stackoverflow.com/`
3. `https://gaming.stackexchange.com/`
4. `https://travel.stackexchange.com/`
5. `https://academia.stackexchange.com/`
6. `https://stats.stackexchange.com/`

questions/réponses écrites par un même utilisateur. De plus, chaque question est associée à un ensemble de *tags*, disponibles parmi une liste prédéfinie de thèmes. Lorsqu'un utilisateur pose une question, celui-ci renseigne des *tags* qui sont par la suite validés par d'autres utilisateurs plus expérimentés. Nous utilisons ces annotations comme *labels* lors de nos évaluations.

Réseaux du langage

Wikipedia [Mahoney, 2011] est un réseau de co-occurrences de mots apparaissant dans les premiers millions octets [7] d'une copie du *Wikipedia* anglophone [8] datant du 3 mars 2006. Les sommets sont donc des mots et les liens correspondent à l'existence d'une co-occurrence entre des mots dans une fenêtre de taille $\tau = 2$. Les classes associées aux mots sont des étiquetages morpho-syntaxiques obtenus avec l'algorithme *POS-Tagger Stanford* [Toutanova et al., 2003]. Un mot pouvant être attribué à plusieurs étiquetages, ce jeu de données est multi-*label*.

Réseaux biologiques

PPI [Breitkreutz et al., 2007] est un réseau d'interactions entre protéines chez l'homo sapiens [9]. Les sommets sont des protéines et les liens correspondent à l'existence d'une interaction chimique entre protéines au sein de l'organisme. Les protéines sont aussi associées à des états biologiques issus de la base de donnée d'annotation des gènes [Liberzon et al., 2011]. La réplication de l'ADN est mise en œuvre par d'importants mécanismes moléculaires qui sont constituées d'un grand nombre de protéines organisées grâce aux interactions protéine-protéine.

Accès aux jeux de données et reproductibilité

Les 9 jeux de données que nous avons présentés sont accessibles publiquement. Cora, CiteSeer, Wikipedia et PPI sont régulièrement utilisés dans la littérature scientifique. Nous avons construit les jeux de données NYT et les réseaux issus de *StackExchange* dans le cadre de nos travaux et les avons rendus publiques en marge des nos contributions sur le plongement de réseau de documents [10] et sur la recherche d'experts [11]. Plus généralement, toutes les implémentations des modèles, protocoles d'évaluation et tous les jeux de données présentés dans ce manuscrit sont en accès libre [12].

1.5 Contributions et organisation de ce manuscrit

La thèse présentée dans ce manuscrit a pour objectif d'étudier les algorithmes d'apprentissage de représentation dans les réseaux afin d'aborder des tâches de recommandation dans le domaine de la littérature scientifique. Cette approche a été décrite dans deux articles, le premier en français publié pour les rencontres jeunes chercheurs en marge de la conférence CORIA 2019 et le second en anglais publié pour le *symposium* doctoral en marge de la conférence du Web 2019 :

7. `http://www.mattmahoney.net/dc/textdata`
8. `https://en.wikipedia.org/`
9. `http://konect.uni-koblenz.de/networks/maayan-vidal`
10. `https://github.com/brochier/idne`
11. `https://github.com/brochier/expert_finding`
12. `https://github.com/brochier/`

> Brochier, R. (2019a). Apprentissage de représentation appliqué à la recommandation pour la littérature scientifique. In *In Conférence jointe CORIA-EARIA 2019*

> Brochier, R. (2019b). Representation learning for recommender systems with application to the scientific literature. In *Companion Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Steering Committee

Nous décrivons nos travaux dans la suite de ce manuscrit qui se compose de la façon suivante :

Chapitre 2 - Apprentissage de Représentation des Sommets d'un Réseau

Le chapitre 2 aborde les méthodes de plongement de réseau. Nous introduisons ce domaine en section 2.1 et détaillons l'état de l'art en section 2.3. Nous présentons notre contribution GVNR en section 2.4, nos méthodes d'évaluations en section 2.5 et rapportons nos résultats expérimentaux en section 2.6. Enfin nous concluons ce chapitre en section 2.7. Ce chapitre s'articule principalement autour de notre contribution

> Brochier, R., Guille, A., and Velcin, J. (2019a). Global vectors for node representations. In *Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Steering Committee

dans laquelle non proposons un modèle d'apprentissage de représentation des sommets dans un réseau dont une extension pour les réseaux de documents, GVNR-t, est décrite dans le chapitre suivant en section 3.4.

Chapitre 3 - Apprentissage de Représentation de Documents en Réseau

Le chapitre 3 aborde les méthodes de plongement de documents en réseau. Nous introduisons ce domaine en section 3.1 et abordons l'état de l'art en section 3.3. Nous présentons ensuite trois contributions en section 3.4, de nouvelles méthodes d'évaluation en section 3.5 et rapportons nos résultats expérimentaux en section 3.6. Enfin nous concluons ce chapitre en section 3.7. En plus de GVNR-t, nous détaillons IDNE, présenté dans la publication

> Brochier, R., Guille, A., and Velcin, J. (2020b). Inductive document network embedding with topic-word attention. In *Proceedings of the 42nd European Conference on Information Retrieval Research*. Springer

dans laquelle nous proposons un modèle d'apprentissage de représentation de documents en réseau. Ce modèle présente l'avantage de produire des représentations interprétables selon des thématiques apprises grâce au réseau. De plus, ce modèle peut induire des représentations de documents jamais observés durant l'apprentissage. Enfin, nous présentons MATAN, introduit dans

> Brochier, R., Guille, A., and Velcin, J. (2019b). Link prediction with mutual attention for text-attributed networks. In *Companion Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Stee-

> ring Committee

où nous détaillons un modèle d'apprentissage de représentations mutuelles de documents. L'algorithme permet d'identifier quels sont les caractéristiques textuelles dans des paires de documents qui expliquent l'existence ou l'absence de liens dans un réseau.

Chapitre 4 - Cas d'Application : la Recherche d'Experts

La chapitre 4 aborde un cas d'application de recherche d'information : la recherche d'experts. Nous introduisons ce domaine en section 4.1 et abordons l'état de l'art en section 4.2. Nous présentons ensuite en section 4.3 les jeux de données et les protocoles d'évaluation que nous mettons en œuvre et décrivons des méthodes d'application d'algorithmes de plongement de documents en réseau pour la recherche d'experts. En section 4.4, nous présentons nos résultats expérimentaux avant de conclure ce chapitre en section 4.5. Nous présentons dans ce chapitre la construction de jeux de données et les protocoles d'évaluation pour cette tâche, que nous avons abordés dans notre contribution

> Brochier, R., Guille, A., Rothan, B., and Velcin, J. (2018a). Impact of the query set on the evaluation of expert finding systems. In *3rd Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL) at SIGIR*

où nous comparons une méthodologie traditionnelle d'évaluation avec notre proposition de méthodologie dans laquelle les requêtes sont plus nombreuses et plus représentatives d'applications réelles. Nous approfondissons ces travaux dans

> Brochier, R., Gourru, A., Guille, A., and Velcin, J. (2020a). New datasets and a benchmark of document network embedding methods for scientific expert finding. In *Bibliometric-enhanced Information Retrieval: 10th International BIR Workshop at ECIR*

où nous construisons 4 jeux de données pour notre méthodologie et explorons l'utilisation d'algorithmes de plongement de documents en réseau pour la résolution de la tâche de recherche d'experts. Nos travaux dans ce domaine ont aussi donné lieu au développement d'un outil en ligne d'aide à la recherche de relecteurs scientifiques destiné aux maisons d'édition, en collaboration avec DSRT, qui est présenté dans l'article

> Brochier, R., Guille, A., Velcin, J., Rothan, B., and Cioccio, D. (2018b). Peerus review: a tool for scientific experts finding. In *Extraction et Gestion des Connaissances (EGC)*

Chapitre 5 - Conclusion

Nous concluons ce manuscrit en résumant les contributions de cette thèse et en abordant nos perspectives de recherche.

Chapitre 2

Apprentissage de Représentation des Sommets d'un Réseau

Ce chapitre traite de l'apprentissage de représentation des sommets d'un réseau, autrement appelé plongement de réseau (*network embedding*). Nous abordons en premier lieu l'état de l'art de ce domaine, en revenant sur les méthodes plus anciennes de plongement de graphe (*graph embedding*) et en s'appuyant sur les techniques de parcours de graphe. Ensuite, nous présentons notre contribution, GVNR, un algorithme de plongement de réseau défini comme un problème de factorisation de matrice. Enfin, nous détaillons les protocoles d'évaluation mis en œuvre pour juger la qualité de notre modèle et analysons les résultats expérimentaux obtenus.

Sommaire

2.1 Introduction

Nombre de systèmes complexes prennent la forme de réseaux, tels que les réseaux sociaux, les réseaux de télécommunication et les réseaux biologiques. Le graphe constitue la structure de donnée massivement utilisée par les systèmes informatiques pour stocker les éléments (sommets) et les interactions (liens) de ces données. Cette structure n'est pas seulement utilisée comme formalisme de stockage mais permet aussi d'analyser et de découvrir des propriétés dans ces réseaux. On cherche par exemple à découvrir des communautés au sein d'un réseau social pour mieux identifier les interactions complexes entre utilisateurs. On peut aussi vouloir prédire de nouveaux liens afin de recommander de nouvelles collaborations dans un réseau académique. Enfin, on peut chercher à classer, à partir d'un nombre limité d'exemples, les protéines dans un réseau d'interactions médicamenteuses afin de mieux prédire les effets secondaires d'un traitement. Dans ce sens, l'analyse des réseaux possède un spectre d'applications particulièrement large.

L'apprentissage automatique constitue un angle d'approche naturel pour l'analyse des réseaux. Le problème central est de trouver un moyen d'incorporer l'information sur la structure d'un graphe au sein d'un modèle d'apprentissage. Par exemple, pour prédire de nouveaux liens dans un réseau social, on peut étudier la corrélation entre l'existence d'un lien entre sommets et le nombre d'amis communs entre les utilisateurs. De même, pour identifier les communautés dans un réseau de citations, on peut étudier la densité des liens pour regrouper les sommets ceux qui sont particulièrement connectés les uns des autres.

Les méthodes traditionnelles d'analyse des réseaux reposent souvent sur des mesures de caractéristiques des graphes telles que les degrés moyens des sommets, les distances géodésiques et les coefficients d'agglomération. Ces méthodes sont cependant peu flexibles puisqu'elles reposent sur des variables construites à la main qui ne sont pas apprises à partir des données. De plus, nombre de ces mesures sont coûteuses à calculer sur de grands réseaux à cause de la nature combinatoire de l'exploration d'un graphe.

Récemment, les approches de plongement de réseau proposent d'apprendre directement à partir des données de nouvelles représentations qui préservent les informations structurelles du graphe. L'objectif de ces approches est la construction de représentations qui conservent, dans un espace vectoriel de faible dimension, les caractéristiques individuelles des sommets au sein du graphe. Elles cherchent notamment à contourner les problèmes inhérents aux méthodes traditionnelles d'analyse des graphes :

— leur forte complexité algorithmique : les sommets d'un graphe sont liés entre eux à divers degrés de connexion. Déterminer la force de connexion entre deux sommets (par exemple au sens de la longueur minimale du plus court chemin allant de l'un vers l'autre, dite distance géodésique) est d'une complexité combinatoire fonction de la longueur maximale considérée. Ainsi, les méthodes exactes de calcul de distances dans les graphes ne peuvent être employées en un temps raisonnable sur des graphes de grande taille ;

— leur faible capacité de parallélisation : afin de diminuer les temps d'exécution des méthodes d'analyse de données, une approche classique consiste à paralléliser les calculs, c'est-à-dire à réaliser en simultané plusieurs calculs et à agréger les résultats par la suite. Ceci ne peut se faire qu'au prix d'une indépendance des calculs en question. Cependant, par nature, un réseau représente des données interdépendantes. À titre d'exemple, l'identification de sommets autoritaires (c'est-à-dire localement influents) se fait traditionnellement de manière itérative. Le score d'autorité d'un sommet est calculé en fonction des scores d'autorité de ses voisins, eux mêmes calculés selon ceux de leurs voisins etc. Paralléliser le calcul d'un tel score requière nécessairement d'effectuer des hypothèses fortes d'indépendances entre différentes

sous-parties du graphe, ce qui n'est pas trivial ;

— leur incompatibilité avec les méthodes classiques d'apprentissage : de nombreux modèles d'apprentissage automatique opèrent sur des données représentées dans des espaces vectoriels telles que les images (matrices de pixels). Les méthodes développées pour ces données sont applicables à de nombreux problèmes. Ils reposent souvent sur l'hypothèse que les données sont décrites par des variables indépendantes et identiquement distribuées (hypothèse i.i.d) ne s'appliquant pas aux sommets d'un graphe. En effet, les lignes de la matrice d'adjacence A, représentant les voisinages directs de chaque sommet, sont souvent distribuées selon des phénomènes d'interdépendances tels que l'homophilie. Cette dernière est une caractéristique de nombreux réseaux sociaux qui stipule que deux sommets ayant plusieurs voisins en commun ont nettement plus de chance d'être connectés l'un à l'autre. De plus, la grande dimension et la nature creuse de A constituent deux freins à l'application directe de modèles d'apprentissage automatique classiques. Trouver un espace dense et de faible dimension pour les sommets d'un graphe permettrait l'application d'une grande variété de méthodes éprouvées d'analyse de données ;

— leur faible capacité de réutilisation : il existe de nombreux algorithmes opérants sur les graphes. A titre d'exemple, PageRank [Page et al., 1999] est un algorithme qui permet de déterminer l'autorité des sommets d'un réseau. Si ces sommets sont associés à des données complémentaires (tel que le contenu textuel d'une page Web sur Internet), on peut vouloir recalculer ces scores en fonction des requêtes textuelles d'utilisateurs (recherches par mots clefs). Il est alors nécessaire de faire appel à cet algorithme à chaque requête, ce qui peut être coûteux en calculs. Dans une certaine mesure, il peut être utile de trouver des représentations des sommets qui soient suffisamment génériques pour être réutilisées à moindre coût.

Face à ces problématiques, l'apprentissage de représentation dans les réseaux consiste à trouver des vecteurs denses, continus et de faible dimensions, qui conservent certaines caractéristiques d'un réseau. Dans ce nouvel espace, les connexions entre sommets sont capturées par les distances entre les vecteurs. Ces représentations permettent alors d'aborder différentes problématiques en tirant profit d'algorithmes traditionnels d'apprentissage automatique (voir figure 2.1).

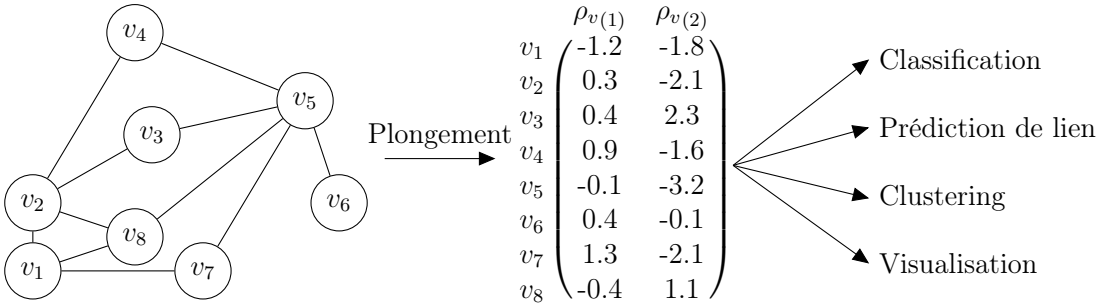


FIGURE 2.1 – Principe général des méthodes de plongement de réseau. Ici, un algorithme génère des vecteurs en dimension $\rho_v = 2$ représentant les sommets. On utilise ensuite ces vecteurs comme représentations d'entrée pour résoudre différentes tâches à l'aide d'algorithmes usuels d'apprentissage automatique.

Afin de construire des représentations des sommets dans un réseau, il est nécessaire de définir un objectif d'apprentissage. Les récents modèles de plongement de réseau reposent particulièrement sur des méthodes de parcours de graphe afin d'estimer une similarité entre sommets sur laquelle est employée une méthode d'optimisation pour la recherche des paramètres des plongements. Après une description formelle du problème du plongement

de réseau dans la section suivante, nous détaillons l'état de l'art autour de l'apprentissage de représentation dans les réseaux.

2.2 Description du problème

On considère un réseau décrit par un graphe pondéré, non-orienté $G = (V, E)$ composé de n_v sommets. Cette configuration est le cas de figure le plus général et nous considérons que l'on peut aisément transformer un graphe orienté en un graphe non orienté en s'intéressant à la matrice $A + A^\top$, où la matrice d'adjacence du graphe est notée $A \in \mathbb{R}^{n_v \times n_v}$ telle que $a_{ij} > 0$ si et seulement si il existe un lien entre les sommets v_i et v_j. On cherche à représenter chaque sommet v_i du graphe par un vecteur u_i de dimension $\rho_v \ll n_v$. On note $U \in \mathbb{R}^{n_v \times \rho_v}$ l'agencement matriciel de ces représentations. On cherche à construire U de sorte à (1) préserver les propriétés de G tout en (2) respectant certains critères de performance. Ces deux points sont au centre des récents développements dans l'apprentissage de représentation dans les réseaux.

Quelles propriétés préserver ? Pour construire des représentations, on définit un objectif d'apprentissage à partir d'un *a priori* (tel que l'hypothèse distributionnelle) qui ne permet pas de directement évaluer la qualité des représentations. L'évaluation de cette qualité se fait alors via des tâches annexes telle que la classification des sommets ou la prédiction de liens. Les protocoles d'évaluation reposent sur des tâches d'apprentissage qui doivent quantifier la qualité des représentations. Notamment, cette qualité se particularise par la capacité à résoudre ces tâches via des modèles linéaires. Cette approche repose sur l'idée que bien représenter des données consiste à pouvoir séparer linéairement dans un certain espace vectoriel les facteurs implicites qui ont générés ces données. On cherchera par exemple à attribuer des catégories aux sommets en utilisant un modèle de classification linéaire et on cherchera à prédire des liens en utilisant une distance définie sur l'espace de représentation. Les méthodes d'évaluation que nous avons utilisées sont décrites plus en détail en section 2.5.

Quels critères respecter ? Au delà de la qualité des représentations, de nombreux aspects d'un algorithme de plongement de réseau sont à considérer dans l'évaluation de ses performances. Tout d'abord, les complexités algorithmiques en temps et en espace sont deux critères essentiels. En effet, il existe de nombreux réseaux dont le nombre de sommets dépassent le million et dont le nombre de liens dépassent le milliard. Pour les traiter, on cherchera un modèle pouvant être parallélisé, ne nécessitant pas de contenir l'intégralité des données d'entraînement en mémoire et/ou dont l'estimation des paramètres ne requière que peu de calculs. De plus, le nombre d'hyperparamètres, la capacité à générer des représentations pour de nouveaux sommets ou encore la robustesse de l'algorithme vis à vis des propriétés du réseau sont quelques exemples de critères parmi d'autres permettant de caractériser un modèle.

De nombreux critères de performance sont à considérer concernant aussi bien la qualité des représentations que la qualité du processus d'apprentissage. L'évaluation de ces modèles est alors une composante non triviale à laquelle il faut accorder une attention particulière.

2.3 État de l'art

Dans cet état de l'art, nous différencions le domaine du plongement de graphe (*graph embedding*), qui constitue un ensemble de techniques de réduction de dimension agnostique au type de données analysées, et le domaine du plongement de réseau (*network embedding*), qui s'inscrit dans le domaine de l'analyse des réseaux naturels. Si le premier a nécessairement nourri le second, nous verrons que les différences entre ses deux domaines motivent l'exploration des techniques de parcours de graphes.

Il existe une riche littérature sur l'apprentissage de représentation reposant sur les graphes [Yan et al., 2005]. De nombreuses méthodes de réduction de dimension s'appuient sur la construction préalable d'un graphe de proximité représentant les données. Néanmoins, ces graphes sont construits à partir d'une mesure de similarité choisie dans l'espace originel des données et la construction du voisinage de leurs sommets est soigneusement contrôlée. À l'inverse, les réseaux naturels, tels que les réseaux sociaux ou les réseaux biologiques, ne possèdent généralement pas les mêmes propriétés que ces graphes de proximité construits à la main. La similarité latente entre les sommets et les mécanismes d'association qui ont engendrés leurs liens sont *a priori* inconnus. Pour cette raison, leurs matrices d'adjacence ne constituent pas nécessairement une bonne mesure des interactions entre les sommets. Ainsi, les méthodes de plongement de réseau s'appuient sur les techniques de parcours de graphe afin de révéler des propriétés du graphe qui ne sont pas directement lisibles dans la matrice A.

Nous présentons tout d'abord en section 2.3.1 certains travaux représentatifs du plongement de graphes, une méthode de réduction de dimension reposant sur la construction d'un graphe de proximité. Ensuite, en section 2.3.2, nous introduisons des méthodes de parcours de graphe couramment utilisées pour capturer différents types d'interactions entre sommets d'un réseau. Enfin, nous détaillons en section 2.3.3 l'état de l'art sur le plongement de réseau.

2.3.1 Plongement de graphe

Originellement, le plongement de graphe est utilisé comme une méthode générale de réduction de dimension. Étant donnés un ensemble de n_x individus $x_1, ..., x_{n_x}$ et une fonction de similarité $s_{ij} \geq 0$ définie sur toute paire d'individus, les méthodes de plongement de graphe construisent une matrice de proximité A pour en réduire sa dimension, construisant ainsi un nouvel espace de représentation U des individus.

Isomap [Tenenbaum et al., 2000] construit un graphe G de voisinage en appliquant la méthode des k plus proches voisins (KNN pour *k-nearest neighbors*). La matrice d'adjacence A de ce graphe est ensuite utilisée pour calculer les distances géodésiques δ_{ij}, correspondant aux longueurs des plus courts chemins entre paires de sommets. On obtient ainsi une matrice des distances Δ sur laquelle une méthode de positionnement multidimensionnel [Cox and Cox, 2000] est appliquée, visant à minimiser la fonction suivante :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\delta_{ij} - ||u_i - u_j||^2\right)^2. \tag{2.1}$$

Les représentations u_i, de dimension très inférieure au nombre d'individus n_x, préservent alors les mesures de distance géodésique dans le graphe construit par KNN. Cependant, le calcul du plus court chemin entre sommets possède une grande complexité algorithmique. La méthode *Locally Linear Embedding* (LLE) [Roweis and Saul, 2000] élimine ce problème en émettant l'hypothèse qu'un individu et ses voisins dans G reposent localement sur un plan. Ainsi, chaque individu peut être reconstruit linéairement selon ses voisins, en

utilisant la matrice d'adjacence A :

$$\min \sum_{i=1}^{n} ||u_i - \sum_{j \in \{k|a_{ik}>0\}} a_{ij} u_j||^2. \tag{2.2}$$

De nombreuses autres méthodes ont été proposées telle que le *Laplacian Eigenmaps* (LE) [Belkin and Niyogi, 2002], mettant en exergue l'utilité du laplacien $L = D - A$ (D étant la matrice de dégrées, voire section 2.3.2) dans l'étude des propriétés des graphes, ou encore le *Locality Preserving Projection* (LPP) [He and Niyogi, 2004], une approximation linéaire de LE constituant une alternative de l'analyse en composante principale [Wold et al., 1987]. Enfin, le partitionnement spectral [Ng et al., 2002, Belkin and Niyogi, 2002] est une technique très utilisée pour l'analyse exploratoire des données. Celui-ci réalise une étape supplémentaire puisque l'objectif final est de trouver un partitionnement des données. Le succès de cette méthode a donné lieu à de multiple contributions, notamment en plongement de graphe. Nous suggérons au lecteur cet article de Von Luxburg [Von Luxburg, 2007] pour une revue approfondie des méthodes de construction de graphe de proximité, des propriétés du laplacien et des algorithmes de réduction de dimension dans les graphes.

Le plongement de graphe étudie donc la réduction de dimension de données à travers une matrice de proximité. Par exemple, pour un corpus d'images, on construit un graphe connectant ces images entre elles en utilisant une similarité basée sur la distance euclidienne dans l'espace vectoriel décrivant leurs niveaux de gris. On peut alors choisir de sélectionner les 5 plus proches images pour construire la matrice d'adjacence du graphe de proximité. La réduction de la dimension de cette matrice, ou du laplacien, permet de capturer cette similarité dans l'espace appris. Les algorithmes de plongement de réseau, à l'inverse, s'intéressent aux graphes pour lesquels nous n'avons pas connaissance de la similarité sous-jacente. Leurs matrices d'adjacence ne reflètent pas nécessairement les réelles proximités entre les données. Pour cette raison, il est nécessaire de préalablement parcourir ces réseaux de sorte à identifier la distribution latente des proximités entre leurs sommets. De nombreux réseaux tels que le réseau de citations des publications scientifiques et le réseau d'Internet ont une distribution des degrés de leurs sommets qui suit une loi puissance [Barabási, 2009], reflétant le dynamisme de leur construction (plus un article scientifique est âgé, plus il a de chances d'être cité). Ces effets ne sont pas considérés dans les techniques de plongement de graphe puisque la construction de la matrice de proximité peut être préalablement contrôlée. Ceci motive l'exploration des techniques de parcours de graphe permettant d'explorer la structure des réseaux naturels.

2.3.2 Parcours de graphe

Un parcours de graphe est une exploration des sommets d'un graphe de proche en proche. Les algorithmes de parcours de graphes s'intéressent, entre autres, au nombre de chemins existant entre les sommets, au nombre moyen ou minimal de sommets les séparants ou encore à l'existence de cycles dans les graphes dirigés. Ces méthodes permettent d'identifier des propriétés locales et globales des graphes, telles que leur connexité ou les distances géodésiques. La difficulté de ces méthodes réside dans leur grande complexité algorithmique en temps due à l'aspect combinatoire du nombre de chemins dans un graphe.

Soit D la matrice des degrés d'un graphe G avec les seuls éléments non nuls situés sur sa diagonale composée des degrés de chaque sommet $d_{ii} = \sum_{j=1}^{n_v} a_{ij}$. Le volume de G correspond à la somme des poids de toutes les arrêtes $\text{vol}(G) = \sum_{i=1}^{n_v} d_{ii}$. La matrice d'adjacence, ainsi que sa version normalisée en ligne $P = D^{-1}A$ (aussi appelée matrice

de transition), peuvent être considérées comme des mesures directes de proximité entre les sommets du graphe. Très souvent, ces matrices sont dites creuses car la plupart des sommets ne sont liés qu'à un petit nombre de voisins. En d'autres termes, ces matrices ne contiennent des informations sur la proximité des sommets que de manière locale. Pour mesurer des relations distantes entre les sommets d'un graphe, il faut parcourir le graphe de sommet en sommet.

Puissances de la matrice d'adjacence

En considérant que deux sommets partageant les mêmes voisins sont similaires, on peut s'intéresser à la matrice A^2. En effet, si deux sommets v_i et v_j ont plusieurs voisins en commun, le produit scalaire entre la ligne $a_{i.}$ et la colonne $a_{.j}$ sera élevé. Dans un graphe non pondéré, sa valeur est égale au nombre de voisins communs aux deux sommets. Au contraire, s'ils ne partagent aucun voisin (bien qu'étant éventuellement eux-mêmes liés entre eux), le produit scalaire sera nul. La matrice A^2 contient donc les relations de longueur 2 du graphe et, dans le cas d'un graphe non pondéré, ses valeurs indiquent le nombre exact de chemins de longueurs 2 qui lient les deux sommets. Plus généralement, la matrice A^ℓ donne une mesure de similarité entre paires de sommets selon les chemins de longueurs ℓ.

Pour capturer la similarité entre les sommets jusqu'à une certaine distance ℓ, on peut alors calculer la somme $\lambda_1 A + \lambda_2 A^2 + ... + \lambda_\ell A^\ell$, où λ_i désigne l'importance que l'on attribue aux chemins de longueurs i. Notons que ce raisonnement peut être appliqué à la matrice de transition P, au quel cas la valeur de $(p^2)_{ij}$ donne la probabilité exacte qu'une marche aléatoire de longueur 2 partant du sommet v_i s'arrête sur le sommet v_j.

Marches aléatoires

Les marches aléatoires constituent un outil mathématique qui décrit une succession de chemins obtenus par des transitions aléatoires sur un graphe. Elles permettent de simuler un marcheur qui se déplace sur les sommets, choisissant successivement le prochain sommet selon les probabilités de transition P. Formellement, on définit une marche de longueur ℓ partant d'un sommet v_i. Ce marcheur va, à chacune des ℓ itérations, sélectionner aléatoirement le prochain sommet v_j selon la distribution donnée par les probabilités de transition $p_{i.}$. On s'intéresse alors à la distribution π_ℓ des probabilités que le marcheur s'arrête sur chacun des sommets du graphe au bout des ℓ itérations. Pour ce faire, on peut calculer cette distribution de manière itérative. Partant d'une distribution de départ connue π_0, on peut calculer ces probabilités après la première itération $\pi_1 = \pi_0 P$ puis à chaque itération jusqu'à la dernière $\pi_\ell = \pi_0 P^\ell$. On peut opérer ce calcul en considérant une distribution de départ concentrée exclusivement sur un sommet v_i pour déterminer la distribution des sommets d'arrivée sur lesquels un marcheur s'arrêterait, c'est-à-dire en prenant $\pi_0 = e_i$, avec e_i un vecteur de la base canonique de $\mathbb{R}^{n_v}$ composé de zéros avec un 1 pour $i^{\text{ème}}$ valeur. Ce calcul peut s'écrire pour l'intégralité des sommets de manière matricielle : $\Pi_\ell = I_{n_v} P^\ell$ où la ligne i de Π_ℓ donne les probabilités d'atteindre chaque sommet en ℓ étapes à partir du sommet v_i. Les probabilités finales sont donc équivalentes à la puissance ℓ de la matrice de transition P.

Le problème du calcul de P^ℓ vient du fait que plus ℓ est grand, plus P^ℓ est dense. En outre, dans la cas d'un graphe non orienté connexe, il existe un ℓ, égal au diamètre du graphe, à partir duquel P^ℓ n'a plus de valeur non nulle (il existe toujours une probabilité d'aller d'un sommet à un autre). Calculer une nouvelle itération demande alors énormément de ressources en calcul et n'est souvent pas possible en pratique. Ainsi, simuler des marches aléatoires permet d'éviter d'effectuer ce calcul exact en réalisant une approximation, moins coûteuse en calculs. Pour ce faire, on réalise μ marches aléatoires

de longueurs ℓ à partir de chaque sommet et l'on enregistre le dernier sommet parcouru. En normalisant par μ les nombres d'occurrences des sommets atteints, on espère ainsi approcher la distribution réelle des probabilités π_ℓ. Notons que pour généraliser ce calcul aux marches de longueurs 1 à ℓ, il suffit d'enregistrer les chemins simulés et d'opérer ces mêmes estimations sur chacun des sommets rencontrés le long des marches. De plus, si un graphe est connexe et non biparti, alors il est garanti qu'il existe une distribution stationnaire des probabilités unique, assurant qu'il existe un μ suffisamment grand à partir duquel les probabilités estimées tendent vers Π_ℓ.

PageRank

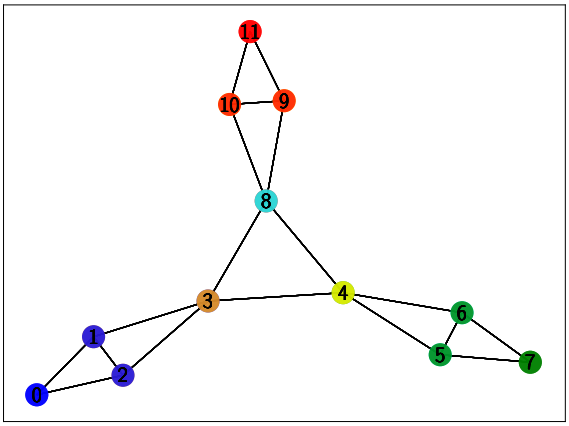
PageRank [Brin and Page, 1998] est un algorithme de mesure de centralité des sommets reposant sur les marches aléatoires dans lequel un marcheur peut à chaque étape, en plus de choisir un prochain sommet, décider de revenir sur son sommet de départ. La matrice de transition du processus s'écrit alors $M = \gamma P + (1-\gamma)\frac{I_{n_v}}{n_v}$ où γ correspond à la probabilité du marcheur de continuer à chaque étape. Ce facteur d'amortissement fut introduit afin d'éviter les problèmes d'impasses posés par les sommets n'ayant pas de liens sortant dans le graphe orienté des pages internet. Pour un graphe non orienté, son utilité réside dans le fait de forcer les marches aléatoires à rester autour de leur sommets de départ permettant ainsi de contrôler l'importance de l'autorité des sommets. De plus, dans le cas d'un graphe non connexe, l'utilisation d'un facteur d'amortissement permet de garantir la stationnarité de la distribution des probabilités Π_ℓ.

Le PageRank personnalisé (PPR pour *Personalized PageRank*) désigne l'application de cet algorithme avec une distribution initiale π_0 savamment choisie. Une application courante consiste, comme pour les marches aléatoires classiques, à calculer le PPR pour chaque sommet individuellement de manière matricielle en calculant $\Pi_\ell = I_{n_v} M^\ell$. D'autres applications, comme nous le verrons dans le chapitre 4, consistent à calculer la distribution initiale π_0 en se basant sur des calculs de similarités entre requêtes et documents. Ceci permet notamment de « personnaliser » le calcul de l'autorité des sommets d'un graphe en fonction du contenu textuel d'une requête utilisateur.

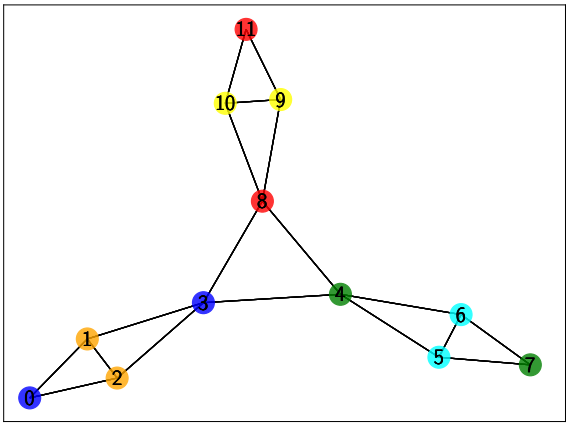
Autres approches

Les marches aléatoires fournissent un cadre flexible de parcours de graphe. En effet, la façon dont un marcheur se déplace peut être facilement biaisée de sorte à favoriser certains types d'exploration. À titre d'exemple, SimRank [Jeh and Widom, 2002] s'intéresse à la similarité structurelle des sommets. L'intuition est que deux sommets sont similaires s'ils sont connectés de la même manière à leurs voisinages. Cela est mesuré en simulant deux marcheurs partant de deux sommets distincts et en calculant le temps moyen nécessaire avant qu'ils ne se croisent sur un même sommet. Ainsi, deux sommets étant structurellement équivalents devraient voir leurs marcheurs se rencontrer régulièrement sur plusieurs sommets tiers.

SimRank et PageRank (illustrés en figure 2.2) sont deux politiques de parcours de graphe opposées. La première repose sur l'hypothèse de l'équivalence structurelle stipulant que deux sommets qui ont un rôle similaire dans le graphe doivent être considérés comme proches. La seconde repose sur l'hypothèse d'homophilie stipulant que deux sommets fortement interconnectés doivent être considérés comme proches. Deux sommets peuvent avoir un rôle similaire et être néanmoins distants dans le graphe comme ils peuvent être fortement connectés entre eux mais ne pas avoir les mêmes rôles structurels. De fait, le choix d'une similarité sur les sommets d'un graphe est souvent fait de manière empirique, en fonction de la nature des données qui composent le graphe.



(a) PageRank rapproche les sommets qui appartiennent aux mêmes communautés.



(b) SimRank rapproche les sommets qui ont les mêmes rôles structurels.

FIGURE 2.2 – Représentation d'un même graphe dont les sommets sont colorés selon 6 *clusters* en appliquant un algorithme de partitionnement hiérarchique agglomératif sur des distances calculées par PageRank (à gauche) et par SimRank (à droite).

Ces techniques de parcours de graphe permettent de construire des mesures de similarité sur les graphes, à l'image des marches aléatoires utilisées pour la recommandation [Fouss et al., 2007] et pour le partitionnement de graphe [Andersen et al., 2006]. S'appuyant sur ces méthodes, les algorithmes de plongement de réseau proposent des techniques d'apprentissage de représentation des sommets.

2.3.3 Plongement de réseau

Dans cette section, nous décrivons les récents modèles de plongement de réseau. Ce domaine a connu une forte popularité depuis l'introduction de DeepWalk [Perozzi et al., 2014] et il existe aujourd'hui un nombre très important d'approches. Nous conseillons au lecteur les articles suivants [Hamilton et al., 2017b, Goyal and Ferrara, 2018] pour une large couverture des contributions de ces dernières années.

L'émergence des algorithmes de plongement de réseau a suivi celle des algorithmes de plongement de mot (*word embedding*). Ces derniers sont récemment devenus la brique de base de la majorité des méthodes de traitement du langage naturel. En effet, ces représentations, qui peuvent être pré-entraînées de manière non supervisée sur des corpus de très grande taille, permettent d'améliorer les performances dans multiples tâches et de s'affranchir de nombre de prétraitements textuels. Les similitudes entre la structure du texte et celle des graphes favorisent l'adaptation de ces méthodes pour l'apprentissage de représentation dans les réseaux.

Nous décrivons dans cette section deux approches majeures de plongement de mots, Word2vec et GloVe avant de présenter les principaux algorithmes de plongement de réseau.

Les plongements de mots

Les méthodes d'apprentissage de représentation pour le texte et pour les réseaux sont fortement liées. En effet, ces deux types de données peuvent être décrits par des éléments symboliques (par exemple les mots et les sommets) dont les relations sont mesurables (par exemple en terme de co-occurrences des mots dans un corpus et de sommets dans des marches aléatoires). De récents travaux permettent de construire des représentations vectorielles de faible dimension des mots sur de très grandes quantités de données et ont

FIGURE 2.3 – Exemple d'une fenêtre de taille $\tau = 2$ glissant sur un corpus de texte. Lors de cette itération, le mot cible est *de* et les mots contextes sont *une*, *démonstration*, *fenêtre* et *glissante*. Les paires de mots (ω_i, ω_j) générées seront alors (*de*, *une*), (*de*, *démonstration*), (*de*, *fenêtre*) et (*de*, *glissante*).

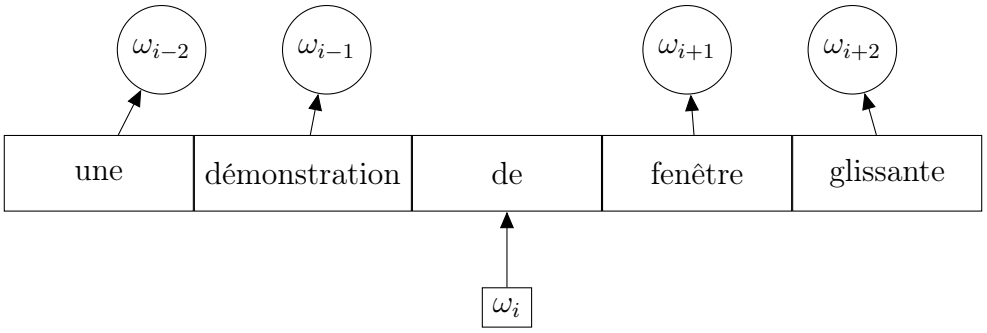


FIGURE 2.4 – Le modèle Skip-Gram prédit les mots contextes $\omega_{i-2}, \omega_{i-1}, \omega_{i+1}$ et ω_{i+2} en fonction d'un mot cible ω_i, extraits à partir d'une fenêtre glissant sur le corpus, ici de taille $\tau = 2$.

permis d'améliorer significativement de nombreuses méthodes de traitement du langage naturel.

L'hypothèse distributionnelle est une hypothèse fondamentale des algorithmes de plongement de mots. Celle-ci stipule que la similarité distributionnelle des mots est fortement corrélée avec la similarité de leurs sens. En d'autres termes, pour construire des représentations vectorielles captant le sens des mots, il suffit d'étudier le voisinage de ceux-ci, c'est-à-dire le contexte dans lequel ils apparaissent. De cette manière, si un modèle est capable de reconstruire les mots contextes d'un certain mot cible, il est capable d'en représenter le sens.

Skip-Gram [Mikolov et al., 2013a], l'une des variantes de la suite logicielle Word2vec [1], est un modèle qui construit deux représentations pour chaque mot ω_i : un vecteur cible u_i et un vecteur contexte h_i. Ces deux vecteurs sont utilisés pour calculer la probabilité conditionnelle d'observer un mot selon son contexte, exprimée comme la fonction softmax du produit scalaire de leurs représentations :

$$p(\omega_j|\omega_i) = \frac{e^{u_i \cdot h_j}}{\sum_{k=1}^{n_\omega} e^{u_j \cdot h_k}}. \tag{2.3}$$

Le modèle maximise la log-vraisemblance d'un ensemble de paires de co-occurrences de mots dans un corpus C :

$$\sum_{(\omega_i,\omega_j)\in C} \log p(\omega_j|\omega_i). \tag{2.4}$$

Cet ensemble est construit en faisant glisser une fenêtre de taille τ sur un corpus de texte. Le mot central de la fenêtre est considéré comme le mot cible ω_i et chacun des τ mots à gauche et τ mots à droite du mot cible est considéré comme mot contexte ω_j (voir figure 2.3). Skip-Gram modélise les probabilités d'occurrence d'un mot cible conditionnellement à chaque mot contexte, de manière indépendante (voir figure 2.4).

Maximiser la log-vraisemblance nécessite néanmoins de calculer le dénominateur de l'équation 2.3, ce qui n'est pas réalisable en pratique puisque la taille du vocabulaire n_ω est

1. `https://patents.google.com/patent/US9037464B1/en`

très élevée et que ce calcul est linéaire en temps $\mathcal{O}(n_\omega)$. Pour cette raison, deux approches ont été proposées :

— la méthode du softmax hiérarchique utilise un arbre binaire pour approcher $p(\omega_j|\omega_i)$ en un temps logarithmique. Chaque mot est associé à chaque feuille de l'arbre. Au lieu d'évaluer l'ensemble des mots dans le dénominateur de l'équation 2.3, il suffit de n'évaluer que le chemin depuis la racine jusqu'à la feuille correspondante. Si le chemin de la racine b_0 jusqu'au mot $b_{\ln(n_\omega)} = \omega_i$ est constitué des nœuds $(b_0, b_1, ..., b_{\ln(n_\omega)})$, alors la probabilité conditionnelle du mot ω_i se calcule comme suit :

$$p(\omega_i|C) = \prod_{j=1}^{b_{\ln(n_\omega)}} p(b_j|\omega_i) = \prod_{j=1}^{b_{\ln(n_\omega)}} \sigma(u_{b_j} \cdot h_i). \tag{2.5}$$

où u_{b_j} est un vecteur coefficient correspondant au nœud b_t. Cette nouvelle paramétrisation permet de réduire la complexité de $\mathcal{O}(n_\omega^2)$ à $\mathcal{O}(n_\omega \ln n_\omega)$. La construction de l'arbre peut être faite de différentes façons. Le choix de l'arbre d'Huffman par les auteurs de Skip-Gram ne prend pas en compte la sémantique des mots. D'autres choix cependant existent et ont montré leur efficacité [Morin and Bengio, 2005, Mnih and Hinton, 2009].

— la méthode de l'échantillonnage négatif [Mikolov et al., 2013b] est encore plus efficace en temps sur les grands corpus, où $\ln(n_\omega)$ devient élevé. Elle consiste tout d'abord à redéfinir la probabilité conditionnelle comme étant la sigmoïde des produits scalaires des représentations du mot cible et du mot contexte, s'affranchissant ainsi du coûteux calcul du dénominateur du softmax :

$$p(\omega_j|\omega_i) = \sigma(u_i \cdot h_j) = \frac{1}{1 + e^{-u_i \cdot h_j}}. \tag{2.6}$$

Ensuite, la maximisation de la log-vraisemblance se fait par l'intermédiaire d'un problème de classification, s'inspirant des méthodes d'estimation contrastive bruitée (NCE pour *noise contrastive estimation*) [Gutmann and Hyvärinen, 2010]. NCE entraîne un modèle de classification binaire qui distingue des paires de mots issus de la distribution empirique des co-occurrences et des paires de mots issus d'une distribution aléatoirement tirée dont voici la vraisemblance :

$$\sum_{(\omega_i,\omega_j)\in C} \Big(\log \sigma(h_j \cdot u_i) + \sum_{k=1}^{n_k} \mathbb{E}_{\omega_k \sim q(\omega_k)} \big[\log \sigma(-h_k \cdot u_i) \big] \Big). \tag{2.7}$$

Pour générer ces paires aléatoires, Skip-Gram avec échantillonnage négatif (SGNS pour *Skip-Gram with Negative Sampling*) tire $n_k \ll n_\omega$ échantillons négatifs pour chaque paire positive extraite du corpus suivant une distribution reflétant les valeurs $q(\omega_i) = (\frac{n_i}{|C|})^{\frac{3}{4}}$, où n_i est le nombre total d'occurrences du mot ω_i dans le corpus C. L'exposant $\frac{3}{4}$ est une valeur sélectionnée empiriquement qui permet de lisser la distribution, diminuant le nombre de tirages de mots fréquents et augmentant le nombre de tirages de mots rares. La complexité du modèle devient alors $\mathcal{O}(n_k n_\omega)$. Sur les grands corpus, une valeur $n_k \simeq 5$ est souvent suffisante.

Ces deux approches sont en pratique optimisées par l'algorithme du gradient stochastique. Le succès de SGNS s'explique par (1) la complexité de l'échantillonnage négatif qui est souvent moindre que celle du softmax hiérarchique car on peut sélectionner n_k de sorte que $n_k < \ln(n_\omega)$, (2) son optimisation qui est plus facilement parallélisable car la rétropropagation du gradient peut se faire de manière distribuée sur des lots de paires (ω_i, ω_j) avec un nombre raisonnable de collisions dans les mises à jour des paramètres u_i et h_j, permettant ainsi de distribuer la descente du gradient sur plusieurs cœurs/serveurs

ceci est une démonstration de fenêtre glissante sur un corpus

	ceci	*est*	*une*	*démonstration*	*de*	*fenêtre*	*glissante*	
	0	1	0	0	0	0	0	ceci
	1	0	2	0	0	0	0	est
	0	2	0	2	0	0	0	une
	0	0	2	0	2	0	0	démonstration
	0	0	0	2	0	2	0	de
	0	0	0	0	2	0	1	fenêtre
	0	0	0	0	0	1	0	glissante

FIGURE 2.5 – Exemple d'une matrice des co-occurrences construite avec une fenêtre de taille $\tau = 1$ glissant sur un corpus de texte. Lors de chaque itération, les entrées dans la matrice correspondant aux paires de mots (ω_i, ω_{i-1}) et (ω_i, ω_{i+1}) sont incrémentées.

et enfin (3) ses meilleures performances empiriques pour la représentation des mots rares d'un corpus.

Enfin, la popularité de Skip-Gram vient aussi des propriétés géométriques des représentations qu'il construit. En effet, celles-ci semblent être en lien direct avec les sens sémantiques et syntaxiques des mots. Il est par exemple possible de vérifier des relations d'analogie telles que $u_{\text{Paris}} = u_{\text{Berlin}} - u_{\text{Allemagne}} + u_{\text{France}}$ démontrant que le modèle a capturé le fait que Paris est à la France ce que Berlin est à l'Allemagne, encodant le concept de capitale dans la différence entre les vecteurs.

GloVe [Pennington et al., 2014] est la principale alternative à SGNS. Pour construire deux représentations u_i et h_i et deux biais b_i^u et b_i^h pour chaque mot, ce modèle factorise une matrice de co-occurrences des mots S d'un corpus en reconstruisant la matrice selon l'objectif suivant :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f(s_{ij})\big(u_i \cdot h_j + b_i^u + b_j^h - \log(s_{ij})\big)^2, \tag{2.8}$$

où $f(s_{ij})$ est une fonction de pondération réduisant l'importance accordée au co-occurrences les moins fréquentes et ignorant les co-occurrences nulles. De plus, un facteur de seuillage $s_{\max}$ est introduit afin de ne pas surreprésenter les hautes fréquences de co-occurrences :

$$f(s_{ij}) = \begin{cases} \left(\frac{s_{ij}}{s_{\max}}\right)^{\frac{3}{4}} & \text{si } s_{ij} < s_{\max}, \\ 1 & \text{sinon.} \end{cases} \tag{2.9}$$

S est une matrice construite, similairement à Skip-Gram, en faisant glisser une fenêtre sur un corpus. Néanmoins, les co-occurrences sont additionnées dans la matrice, à l'image de la figure 2.5. L'entrée s_{ij} donne alors le nombre total de co-occurrences des mots ω_i et ω_j dans le corpus. U et H sont les représentations cibles et contextuelles des mots et b^u et b^h sont leurs biais respectifs.

Ce modèle est motivé par les rapports de probabilités de co-occurrences entre triplets de mots. Soit $p_{ij} = p(\omega_j|\omega_i) = \frac{s_{ij}}{s_i}$ la probabilité que le mot ω_j apparaisse avec le mot ω_i,

s_i étant le nombre total d'occurrences du mot ω_i. L'objectif est que le modèle approche les rapports de probabilités $\frac{p_{ik}}{p_{jk}}$ pour tout mot ω_k. Construire un espace vectoriel dans lequel représenter les mots par des ensembles de vecteurs U et H tels que le rapport de probabilités entre deux mots soit interprété par une soustraction entre leurs vecteurs permet d'obtenir l'équation suivante :

$$F(u_i - u_j, h_k) = \frac{p_{ik}}{p_{jk}}. \tag{2.10}$$

Afin de ramener les paramètres $u_i - u_j$ et h_k (des vecteurs) au ratio des probabilités (un scalaire), on peut choisir une opération simple telle que le produit scalaire $F(u_i - u_j, h_k) = F\big((u_i - u_j)^\top h_k\big)$. En restreignant F à être symétrique telle que $F(u_i - v_j, h_k) = F(u_i, h_k)/F(u_j, h_k)$, on a nécessairement $F = \exp$ et par conséquent $u_i^\top h_k = \log(s_{ik}) - \log(s_i)$. L'équation 2.8 est obtenue en absorbant $\log(s_i)$ dans les biais b^u et b^h.

Pour faire le lien avec Skip-Gram, on peut tout d'abord écrire la probabilité selon le modèle q_{ij} qu'un mot ω_j apparaisse dans le contexte d'un mot ω_i en terme du softmax de $u_i^\top h_j$:

$$q_{ij} = \frac{\exp(u_i^\top h_j)}{\sum_{k=1}^{n_\omega} \exp(u_i^\top h_k)}. \tag{2.11}$$

La fonction à minimiser selon le maximum de vraisemblance est alors :

$$J = -\sum_{(i,j)\in C} \log(q_{ij}). \tag{2.12}$$

En pratique, SGNS utilise une approximation de cet objectif de sorte à réduire le coût du calcul du facteur de normalisation du softmax. Cependant, GloVe effectue déjà une première réduction de cette somme en factorisant tous les termes qui ont les mêmes valeurs (i, j) :

$$J = -\sum_{i}^{n_\omega} \sum_{j}^{n_\omega} s_{ij} \log(q_{ij}). \tag{2.13}$$

Cette réduction, qui en pratique se traduit par la construction explicite de la matrice des co-occurrences de mots dans le corpus, réalise une réduction significative de la complexité en temps du modèle, ce dont nous discutons plus en détails en section 2.4.3. Une autre distinction entre ces deux modèles réside dans le choix de la mesure de l'erreur. SGNS minimise l'entropie croisée entre les distributions p_i et q_i. Les auteurs de GloVe préfèrent la méthode des moindre carrés, motivant leur choix par la capacité de cette mesure à s'affranchir de la normalisation des distribution p_i et q_i ainsi que par son meilleur comportement avec des distributions présentant une longue traîne.

Les plongements de réseaux

Nous présentons dans cette section les principaux travaux en plongement de réseau. Nous verrons que nombre de ces travaux adaptent différents éléments de SGNS par le biais de techniques de parcours de graphe. À l'inverse, notre contribution reprend l'approche adoptée dans GloVe.

DeepWalk [Perozzi et al., 2014] est une méthode de plongement de réseau qui s'inspire de Skip-Gram. L'intuition centrale de cette approche est que les chemins générés par de courtes marches aléatoires dans un graphe sont similaires à des phrases en langage naturel. La fréquence d'apparition des sommets dans ces marches suit une loi puissance, similairement aux fréquences des mots dans un corpus. Ce comportement étant une hypothèse centrale dans de nombreux modèles du langage, à l'image de la loi de Zipf [Apostel et al.,

1957], DeepWalk applique Skip-Gram pour apprendre des représentations des sommets sur les graphes.

L'algorithme original applique Skip-Gram avec softmax hiérarchique sur des marches aléatoires réalisées sur un graphe. Ceci permet alors de capturer les interactions locales entre sommets de sorte que les sommets qui ont un voisinage semblable auront des représentations semblables. L'avantage des marches aléatoires est qu'elles sont parallélisables, ce qui permet de ne pas entraver l'efficacité algorithmique de Skip-Gram.

Formellement, DeepWalk est constitué de deux éléments. En premier lieu, un générateur de marches aléatoires construit des chemins en tirant uniformément des sommets de départ $c_0 = v_i$ et parcourant les sommets avoisinant de manière aléatoire. L'algorithme réalise précisément μ marches par sommet de longueurs fixes ℓ, générant des séquences de sommets $[c_0, c_1, ..., c_\ell]$. Par la suite, le modèle Skip-Gram est appliqué. Une fenêtre est passée sur les chemins extraits par le générateur, constituant ainsi des paires de co-occurrences de sommets (v_i, v_j) et les paramètres sont mis à jour par descente du gradient stochastique maximisant la log-vraisemblance du modèle. La figure 2.6 montre le processus d'échantillonnage des sommets opéré par marches aléatoires dans DeepWalk.

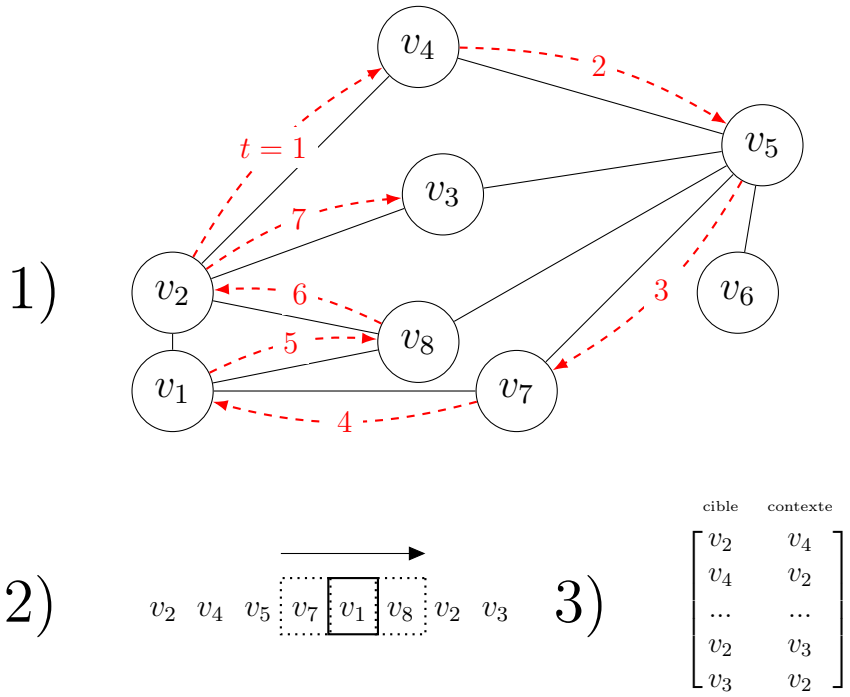


FIGURE 2.6 – Échantillonnage des sommets dans DeepWalk. En (1), une marche aléatoire de longueur $\ell = 8$ est opérée sur un réseau en partant du sommet v_2. On génère ainsi une séquence de sommets, semblable à une phrase, sur laquelle est glissée en (2) une fenêtre de taille $\tau = 1$. On obtient au final, en (3), une série d'échantillons composés chacun d'un sommet cible et d'un sommet contexte.

De nombreux modèles d'apprentissage de représentation dans les graphes reposent sur l'utilisation de Skip-Gram comme modèle d'inférence des paramètres. Cependant, la méthode de l'échantillonnage négatif est quasiment toujours préférée au softmax hiérarchique.

Node2vec [Grover and Leskovec, 2016] est un modèle qui propose d'étendre la notion de marche aléatoire utilisée dans DeepWalk. Celle-ci repose sur des marches qui suivent les probabilités de la matrice de transition du graphe et génèrent des chemins qui, similairement au PageRank, ont tendance à rester au sein d'une seule communauté du graphe et rarement à en sortir. Cependant, il existe d'autres caractéristiques que les commu-

nautés. Par exemple, SimRank a tendance à rapprocher des sommets qui ont une forme d'équivalence structurelle au sein du graphe. Node2vec propose de prendre en compte deux aspects, l'homophilie et l'équivalence structurelle, en biaisant les marches aléatoires. Deux paramètres supplémentaires p et q permettent d'ajuster le compromis entre l'exploration en profondeur (s'éloigner le plus possible d'un sommet) et l'exploration en largeur (parcourir le plus possible le voisinage d'un sommet) des marches. Ainsi, ce modèle est une généralisation de DeepWalk, lui étant équivalent lorsque $p = q = 1$.

En pratique, Node2vec enrichit la notion de voisinage en proposant des marches aléatoires de second ordre. On distingue pour cela deux stratégies d'explorations du voisinage d'un sommet source :

— en largeur : dans laquelle on explore en priorité les sommets immédiatement connectés au sommet source. On explore ensuite les sommets connectés par des marches de longueur 2, puis 3 etc. ;

— en profondeur : dans laquelle on échantillonne des sommets dont la distance géodésique avec le sommet source augmente à chaque fois. On sélectionne un sommet voisin au sommet source, puis un sommet voisin de ce dernier sommet, mais non voisin du sommet source puis séquentiellement des sommets de plus en plus distants.

Ainsi, pour pondérer ces deux notions d'exploration lors de marches aléatoires, Node2vec définit une marche $[c_o, c_1, ..., c_\ell]$ de longueur ℓ partant d'un sommet source c_0 avec une distribution de transition entre les sommets biaisée. Lors d'une marche, si le sommet précédemment parcouru est noté v_i et que l'on s'intéresse à la probabilité de sélectionner v_j comme prochain sommet à parcourir, on a :

$$p(c_k = v_j | c_{k-1} = v_i) = \begin{cases} \frac{\pi_{v_i v_j}}{Z} & \text{si } (v_i, v_j) \in E, \\ 0 & \text{sinon,} \end{cases} \tag{2.14}$$

où $\pi_{v_i v_j}$ est une probabilité de transition non normalisée entre v_i et v_j, et Z une constante de normalisation assurant que $\sum_j p(c_k = v_j | c_{k-1} = v_i) = 1$. Dans ce cadre, DeepWalk est un cas particulier où $\pi_{vx} = a_{ij}$. Node2vec définit alors une marche aléatoire de second ordre. Considérons que le marcheur vient de réaliser une transition du sommet v_l vers v_i, la probabilité de transition vers v_j est rendue dépendante de cette précédente transition $\pi_{v_i v_j} = \alpha_{pq}(l, j) a_{ij}$ où :

$$\alpha_{pq}(l, j) = \begin{cases} \frac{1}{p} & \text{si } \delta_{lj} = 0, \\ 1 & \text{si } \delta_{lj} = 1, \\ \frac{1}{q} & \text{si } \delta_{lj} = 2, \end{cases} \tag{2.15}$$

avec δ_{lj} étant la longueur du chemin le plus court entre les sommets v_l et v_j. Les paramètres p et q permettent ainsi de guider la vitesse à laquelle les marches explorent ou quittent le voisinage d'un sommet. Notamment, p contrôle la probabilité avec laquelle une marche va immédiatement revisiter un sommet. Si sa valeur est petite, alors les chances de revenir sur un sommet déjà visité sera grande, favorisant une exploration en largeur du graphe. q contrôle la prépondérance des marches à s'éloigner du sommet d'origine. Si sa valeur est petite, les marches auront tendance à explorer le graphe en profondeur.

LINE [Tang et al., 2015] est une méthode qui capture les proximités d'ordres 1 et 2 dans un graphe. La proximité de premier ordre entre deux sommets v_i et v_j est définie comme la valeur a_{ij} de la matrice d'adjacence du graphe. Cette proximité est néanmoins incomplète. Dans de nombreux réseaux, les liens observés ne représentent qu'une petite portion des proximités réelles entre les sommets. Par exemple, dans un réseau social, deux personnes peuvent avoir des intérêts parfaitement similaires mais ne pas être amis, car ils ne se sont simplement jamais rencontrés. LINE introduit donc la proximité de second

ordre, en faisant l'hypothèse que des sommets qui sont connectés aux mêmes voisinages sont similaires. Celle-ci est alors définie en terme de similarité entre les contextes (ou voisinages) de deux sommets.

Selon le modèle, la proximité de premier ordre est capturée par la probabilité jointe de v_i et v_j, définie comme la sigmoïde du produit scalaire de leurs représentations :

$$p_1(v_j, v_i) = \sigma(u_i \cdot u_j) = \frac{1}{1 + e^{-u_i \cdot u_j}}. \tag{2.16}$$

La proximité de second ordre est modélisée par la probabilité du sommet contexte v_j d'être généré par le sommet cible v_i. On introduit alors la représentation contextuelle h_i du sommet v_i :

$$p_2(v_j|v_i) = \frac{e^{h_j \cdot u_i}}{\sum_{k=1}^{|V|} e^{h_k \cdot u_i}}. \tag{2.17}$$

LINE optimise les paramètres du modèle en minimisant simultanément la divergence de Kullback-Leibler entre les distributions de proximités de premier ordre p_1 et de second ordre p_2 avec respectivement deux distributions empiriques $\hat{p_1}$ et $\hat{p_2}$, où $\hat{p_1}(v_i, v_j) = \frac{a_{ij}}{|A|}$ et $\hat{p_2}(v_i, v_j) = \frac{a_{ij}}{d_i}$ où d_i est le degré du sommet v_i. L'optimisation se fait par descente du gradient en utilisant la méthode de l'échantillonnage négatif pour approcher l'équation de la proximité du second ordre 2.17.

NetMF [Qiu et al., 2018] montre que DeepWalk, Node2vec et LINE peuvent être unifiés dans le cadre des méthodes de factorisation de matrice. Leur démonstration reprend les travaux de Levy et Goldberg [Levy and Goldberg, 2014] qui montrent que Skip-Gram avec échantillonnage négatif factorise implicitement la matrice, à une constante près, des informations mutuelles des co-occurrences $\#(\omega_i, \omega_j)$ des mots dans un corpus C :

$$\log\left(\frac{\#(\omega_i, \omega_j)|C|}{\#(\omega_i) \cdot \#(\omega_j)}\right) - \log n_k, \tag{2.18}$$

où n_k est le nombre d'échantillons négatifs et $\#(\omega_i)$ est le nombre total d'occurrences du mot ω_i dans le corpus. En considérant que les marches aléatoires convergent vers une distribution stationnaire $\pi(\omega_i) = \frac{d_i}{|A|}$, les auteurs montrent que DeepWalk calcule une approximation de rang faible de cette matrice :

$$\log\left(|A|\left(\frac{1}{\tau}\sum_{r=1}^{\tau}\left(D^{-1}A\right)^r\right)D^{-1}\right) - \log n_k, \tag{2.19}$$

où τ est la taille de la fenêtre et n_k le nombre d'échantillons négatifs. De cette équation, il est possible d'expliciter la matrice factorisée par LINE en considérant ce modèle comme un cas particulier de DeepWalk où la taille de la fenêtre est $\tau = 1$:

$$\log\left(|A|D^{-1}AD^{-1}\right) - \log n_k. \tag{2.20}$$

Ces équivalences permettent aux auteurs de tirer profit d'algorithmes efficaces de factorisation de matrices pour introduire un nouvel algorithme de plongement de réseau, NetMF, pour lequel deux variantes sont présentées. La première est une version exacte, lorsque la taille de la fenêtre τ considérée est petite et la seconde est une version approchée, qui exploite le faible rang de la matrice laplacienne du graphe. En définissant M la matrice de DeepWalk, telle que, selon l'équation 2.19 :

$$M = \frac{|A|}{\tau n_k}\left(\sum_{r=1}^{\tau} P^r\right)D^{-1}, \tag{2.21}$$

où $P = D^{-1}A$, la version exacte consiste donc à calculer successivement les puissances de P. Seulement, DeepWalk factorise le logarithme de M qui devient alors (1) indéfini lorsque $m_{ij} = 0$ et (2) très dense. On peut cependant factoriser la matrice $M' = \max(M, 1)$. De cette façon, $\log(M')$ est creuse et peut-être factorisée par décomposition en valeurs singulières.

La version approchée est préférable lorsque τ est grand car le calcul des puissances de P devient trop coûteux en pratique. Celle-ci consiste à approcher la matrice normalisée du laplacien avec ses l plus grandes valeurs propres :

$$D^{-1/2}AD^{-1/2} \approx U_l \Lambda_l U_l^\top. \tag{2.22}$$

Ensuite, on construit une approximation de M de la façon suivante :

$$\hat{M} = \frac{|A|}{n_k} D^{-1/2} U_l \left(\frac{1}{\tau} \sum_{r=1}^{\tau} \Lambda_l^r \right) U_l^\top D^{-1/2}. \tag{2.23}$$

Finalement, la matrice $\hat{M}' = \max(\hat{M}, 1)$ est décomposée de la même manière que pour la version exacte.

Autres approches

De nombreuses approches sont proposées dans la littérature pour représenter les sommets d'un graphe. GraRep [Cao et al., 2015] reprend LINE en le généralisant aux proximités d'ordre $K > 2$. Verse [Tsitsulin et al., 2018] étend le constat établi par Node2vec en proposant un modèle versatile en terme de similarité. Partant d'une mesure de similarité quelconque sur un graphe, ce modèle utilise une méthode d'estimation contrastive bruitée pour minimiser la différence entre la proximité mesurée dans le graphe original et la proximité mesurée selon les représentations apprises. Les expérimentations sont menées avec différentes mesures de similarités telles que le PageRank personnalisé [Brin and Page, 1998] et SimRank [Jeh and Widom, 2002]. SDNE [Wang et al., 2016] propose un réseau de neurones profond pour apprendre la haute non linéarité de la structure d'un graphe. À travers un autoencodeur multi-couches, l'estimation des similarités de premier et de second ordre sont apprises.

Il existe de plus des approches qui apprennent des représentations dans des espaces non euclidiens. Les plongements de Poincaré [Nickel and Kiela, 2017] cherchent à trouver des représentations dans un espace hyperbolique, permettant de capturer simultanément les similarités et les hiérarchies entre les données. Graph2gauss, que nous verrons plus en détail dans le prochain chapitre, est une méthode de plongement de réseau qui représente les sommets comme des distributions gaussiennes capturant une notion d'incertitude sur leurs représentations.

Enfin, l'apprentissage de représentation dans les graphes dépasse largement le cadre des algorithmes de plongement de réseau. Récemment, l'essor des *graph convolutional networks* [Schlichtkrull et al., 2018] a fortement nourri ce domaine de recherche. Ces approches cherchent à généraliser les réseaux neuronaux convolutifs, très performants sur certains types de données aux structures régulières (images, texte), aux données dont les structures sont arbitraires et donc décrites par des graphes. Ces méthodes, telles que GraphSage [Hamilton et al., 2017a] et GAT [Veličković et al., 2017] ont montré d'excellentes performances pour représenter des graphes entiers, où les sommets sont associés à des attributs. Ces modèles apprennent à agréger le voisinage d'un sommet à partir de leurs attributs. Cependant, à notre connaissance, leurs performances pour l'apprentissage non supervisé de sommets non attribués n'atteignent pas celles des méthodes décrites précédemment.

2.4 Méthode proposée : GVNR

Dans cette section, nous présentons notre méthode de plongement de réseau, GVNR. L'objectif est de construire des représentations $U \in \mathbb{R}^{n_v \times \rho_v}$ des sommets, telles que $\rho_v \ll n_v$, de sorte à capturer les caractéristiques locales et globales d'un réseau.

Suivant le succès de Skip-Gram pour l'apprentissage de représentation des mots, de nombreux modèles ont été proposés pour représenter les sommets d'un réseau. Certaines méthodes reprennent la technique de l'échantillonnage négatif appliquée sur un corpus de séquences de sommets. Si cette technique réduit grandement la complexité de l'algorithme par rapport à sa formulation originale, elle reste linéaire par rapport à la taille du corpus.

GloVe [Pennington et al., 2014] constitue l'alternative principale à Skip-Gram pour apprendre des représentations des mots. Ce modèle définit un problème de factorisation sur la matrice des comptages de co-occurrences des mots dans le corpus. Sa complexité est sublinéaire par rapport à la taille du corpus (voir section 2.4.3). De plus, le fait de factoriser explicitement une matrice de co-occurrences permet de la modifier afin d'en réduire le bruit.

La méthode que nous proposons, GVNR [Brochier et al., 2019a], reformule GloVe pour l'apprentissage de représentation des sommets d'un réseau. Dans cette section, nous présentons en premier lieu GVNR, la matrice qui est factorisée et la méthode d'optimisation mise en œuvre. Ensuite, nous montrons que la complexité du modèle est doublement réduite, lors de la construction de la matrice de similarité et lors de la factorisation.

2.4.1 Description du modèle

Motivés par l'approche de GloVe [Pennington et al., 2014] pour le plongement de mots, nous proposons un algorithme de représentation des sommets d'un réseau, GVNR (*Global Vectors for Node Representation*), plus efficace en terme de complexité en temps que DeepWalk [Perozzi et al., 2014]. Néanmoins, si GloVe formule une factorisation ne prenant en compte que les valeurs non nulles de la matrice des co-occurrences, nous faisons l'hypothèse que prendre les non occurrences de paires de sommets permet d'améliorer les représentations apprises. De plus, nous proposons de filtrer la matrice des co-occurrences de sorte à éliminer ses faibles valeurs, que nous considérons comme du bruit associé aux marches aléatoires.

GVNR formule ainsi un problème de factorisation sur une matrice seuillée S des comptages des co-occurrences des sommets d'un réseau dans des marches aléatoires. Spécifiquement, GVNR construit des représentations U et H des sommets de manière à reconstruire S en prenant en compte aussi bien ses valeurs strictement positives que ses valeurs nulles. Le filtrage de S et la sélection des valeurs considérées pour la factorisation permettent non seulement d'améliorer les performances de l'algorithme mais aussi d'améliorer sa complexité en temps par rapport à la formulation originale de GloVe. Nous décrivons tout d'abord la construction de la matrice S et détaillons par la suite la formulation de la factorisation.

Matrice seuillée des co-occurrences

GVNR réalise des marches aléatoires similaires à celle présentées dans DeepWalk pour capturer les similarités entre les sommets. À partir de chaque sommet, μ marches de longueurs ℓ sont opérées formant ainsi un corpus de séquences de sommets. Sur ce corpus, une fenêtre de taille τ est glissée de façon à incrémenter les co-occurrences de sommets dans la matrice $S \in \mathbb{R}^{n_v \times n_v}$. À chaque paire de sommets formée par le sommet cible v_i et un sommet contexte v_j dans la fenêtre, on augmente la valeur s_{ij} par $\frac{1}{q}$, q étant la longueur du chemin entre v_i et v_j dans la fenêtre. Cette méthode de pondération, introduite dans

GloVe [Pennington et al., 2014], permet de décroître l'influence des co-occurrences en fonction de la distance parcourue entre les sommets. Elle constitue un détail important des améliorations de performances rapportées dans GloVe [Levy et al., 2015]. D'autres méthodes de pondérations existent telle que $\frac{\tau-q+1}{\tau}$ utilisée dans SGNS, donnant ainsi plus de poids aux longs chemins que notre fenêtre. D'autre méthodes encore, telles que FastText [Bojanowski et al., 2017], apprennent une représentation pour chaque position dans la fenêtre. Nous étudions l'impact du choix de la pondération de la fenêtre en section 2.6.4.

La matrice S construite est une interpolation à mi-chemin entre la matrice pondérée des marches aléatoires de longueurs 1 à τ : $\sum_{q=1}^{\tau} \frac{1}{q} A^q$ et la matrice des probabilités Π_ℓ issues de marches de longueurs ℓ telle que $\Pi_\ell = I_{n_v} A^\ell$. En effet, réaliser μ marches partant de chaque sommet en appliquant une fenêtre de taille τ réalise un échantillonnage du voisinage de ces derniers. Seulement, en pratique, on utilise souvent une longueur de marche ℓ très supérieure à τ ce qui amène ces marches aléatoires, après quelques étapes, à parcourir les sommets suivant les distributions données par Π_ℓ.

L'approximation par marches aléatoires présente l'avantage d'attribuer des zéros à la plupart des co-occurrences de paires de sommets qui ont une probabilité faible d'apparaître dans un même chemin. Néanmoins, elles produisent tout de même de nombreuses valeurs s_{ij} très faibles, correspondant à des co-occurrences que l'on peut qualifier d'accidentelles. Pour cette raison, GVNR procède à un seuillage de la matrice S en éliminant les valeurs inférieures à un seuil $s_{\min}$. Outre la réduction du bruit, dont l'impact est mis en lumière par les résultats expérimentaux en section 2.6, un avantage de ce seuillage est la forte diminution de la densité de la matrice permettant de réduire la complexité de GVNR, comme nous le présentons en section 2.4.3.

Factorisation

Nous formulons une factorisation de la matrice S en mesurant l'erreur de reconstruction aussi bien sur les valeurs non nulles que sur certaines valeurs nulles. Soient U et H les représentations cibles et contextuelles que nous cherchons à construire pour les sommets d'un graphe et b^u, b^h leurs biais respectifs. Nous introduisons une fonction $\delta(s_{ij})$ qui permet de sélectionner les valeurs à considérer dans le calcul de l'erreur, suivant ce problème d'optimisation :

$$\underset{U,H,b^u,b^h}{\operatorname{argmin}} \sum_{i=1}^{n_v} \sum_{j=1}^{n_v} \delta(s_{ij}) \big(u_i \cdot h_j + b_i^u + b_j^h - \log(1+s_{ij})\big)^2, \tag{2.24}$$

avec

$$\delta(s_{ij}) = \begin{cases} 1 & \text{si } s_{ij} > 0, \\ m_i & \text{sinon, où } m_i \sim \text{Bernoulli}(\alpha_i). \end{cases} \tag{2.25}$$

$\delta(s_{ij})$ prend la valeur 1 pour toutes les entrées non nulles de S alors que pour les valeurs nulles, elle est déterminée par une variable aléatoire suivant une loi de Bernoulli, de paramètre α_i spécifique à chaque sommet du graphe. En notant $p_i = \frac{|s_{i.}|_{\neq 0}}{n_v}$ la densité de la $i^{\text{ème}}$ ligne de S, α_i est calculé selon la cote :

$$\alpha_i = \begin{cases} n_k \times \frac{p_i}{1-p_i} & \text{si } p_i \leq (n_k+1)^{-1}, \\ 1 & \text{sinon.} \end{cases} \tag{2.26}$$

$n_k \geq 0$ est un hyperparamètre qui contrôle la proportion maximale globale de coefficients nuls incorporés dans le calcul de l'erreur. Le nombre de coefficients nuls tirés par sommet est alors en moyenne égal au nombre de sommets contextuels avec lesquels il apparaît

multiplié par n_k. Cet hyperparamètre est similaire au nombre d'échantillons négatifs utilisé dans SGNS et permet de pondérer l'importance donnée aux co-occurrences non observées de sommets. Une visualisation des valeurs s_{ij} considérées lors de la factorisation est donnée en figure 2.7.

$$S = \begin{pmatrix} 0 & 4 & 8 & 0 & 0 & 0 \\ 4 & 0 & 0 & 3 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 6 & 1 & 0 \end{pmatrix}$$

s_{ij} : entrées strictement positives après seuillage

s_{ij} : entrées nulles échantillonnées aléatoirement

FIGURE 2.7 – Une matrice des comptages des co-occurrences à partir de laquelle les entrées strictement positives et certaines entrées nulles sont échantillonnées. Ici, $n_k = 1$, ce qui signifie que le même nombre d'entrées positives et d'entrées nulles sont échantillonnées.

Similairement à GloVe, la factorisation est effectuée par un modèle log-linéaire guidé par les probabilités de co-occurrence. D'un point de vue théorique, l'utilisation du logarithme permet de transformer les rapports de probabilités en différences entre vecteurs dans l'espace de représentation. Le fait d'ajouter 1 aux valeurs de S assure que $\log(1 + s_{ij}) > 0$ pour toutes les valeurs strictement positives de s_{ij} et $\log(1 + s_{ij}) = 0$ si et seulement si $s_{ij} = 0$. Intuitivement, le logarithme limite l'étendu des coefficients approchés par $u_i \cdot h_j + b_i^u + b_j^h$ facilitant ainsi l'optimisation des paramètres. GVNR réalise ainsi une régression des moindres carrés parcimonieuse de la matrice S.

2.4.2 Interprétation du seuillage

Comme nous le présentons dans nos expérimentations quantitatives en section 2.6.1, le seuillage de la matrice S par une valeur minimum $s_{\min}$ permet d'améliorer significativement dans de nombreux cas les performances obtenues. Selon [Levy et al., 2015], le nombre d'échantillons négatifs n_k utilisé dans SGNS a deux fonctions distinctes. Tout d'abord, il est utilisé pour mieux estimer la distribution des exemples négatifs puisque plus n_k est élevé, meilleure est l'estimation de la distribution des données (au prix d'une plus grande complexité). Ensuite, il agit comme un *a priori* sur la probabilité d'observer un exemple positif plutôt qu'un exemple négatif. Un n_k plus élevé signifie que les exemples négatifs sont plus probables. Avec GVNR, ces deux aspects sont considérés à travers deux hyperparamètres. Le nombre moyen n_k d'éléments nuls échantillonnés joue un rôle similaire au nombre d'échantillons négatifs dans SGNS, mais le seuillage $s_{\min}$ permet de n'agir que sur le deuxième aspect, le ratio entre exemples positifs et négatifs.

Si l'on considère la tâche de prédiction de liens dans le réseau, il semble raisonnable de vouloir entraîner le modèle avec des quantités équilibrées d'échantillons représentant des paires de sommets liées et des paires de sommets non liées. Néanmoins, SGNS est souvent appliqué avec un nombre d'échantillons négatifs d'un ordre de grandeur supérieur à celui des échantillons positifs ($n_k \simeq 10$). À l'inverse, GVNR réalise les meilleures performances (voir section 2.6.3) pour $n_k = 1$, c'est-à-dire lorsque les deux ensembles sont équilibrés. Nous pensons que le seuillage par $s_{\min}$ permet de calibrer l'*a priori* sur la probabilité d'observer une paire liée plutôt qu'une paire non liée, réduisant ainsi fortement la nécessité

d'augmenter le nombre d'échantillons négatifs n_k. Ceci permet en outre de réduire de deux manières la complexité en temps de l'algorithme puisque (1) un n_k plus bas signifie moins d'exemples négatifs à considérer et (2) $s_{\min}$ réduit le nombre d'exemples positifs. Nous étudions cela dans la section suivante.

De manière intéressante, GloVe gère la pondération de cas positifs et négatifs très différemment de SGNS. En effet, dans la formulation de sa factorisation, les valeurs nulles de la matrice sont ignorées. Néanmoins, une fenêtre pondérée de type $\lambda = \frac{1}{q}$ étant utilisée pour calculer les comptages de co-occurrence des mots, de nombreuses valeurs de s_{ij} sont inférieures à 1. La factorisation se faisant sur le logarithme de S, selon l'équation 2.8, $\log(s_{ij})$ prend très souvent des valeurs négatives, jouant ainsi le rôle des échantillons négatifs puisque forçant le produit scalaire $u_i \cdot h_j$ à être négatif et donc u_i et h_j à avoir des directions opposées. Si GloVe ne prend pas en compte les non co-occurrences dans la matrice S, il réalise cependant, par effet de bord, un pseudo-échantillonnage négatif. Celui-ci n'est cependant pas uniforme puisque les sommets v_j induisant des valeurs $\log(s_{ij})$ négatives sont toujours sélectionnés dans le voisinage du sommet v_i. Notre approche est plus concise puisque nous échantillonnons les sommets négatifs v_j de manière uniforme sur l'ensemble du réseau. En somme, GloVe réalise un échantillonnage négatif local alors que GVNR, similairement à SGNS, réalise un échantillonnage négatif global.

2.4.3 Complexité algorithmique en temps

Au-delà de la qualité des représentations apprises, que l'on étudie en section 2.6, GVNR présente une meilleure complexité en temps que les méthodes basées sur SGNS. Nous présentons d'abord une analyse théorique de cette complexité puis montrons les gains obtenus sur des jeux de données réels. Enfin, nous discutons brièvement de la complexité de LINE.

La complexité de SGNS est linéaire avec la taille du corpus $\mathcal{O}(|C|)$. Dans le cas d'un algorithme de plongement de graphe, le corpus est constitué de l'ensemble des co-occurrences entre sommets obtenues par marches aléatoires. La complexité de GVNR est au pire de $\mathcal{O}(|V|^2)$ car l'algorithme opère directement sur la matrice de similarité entre sommets S. En s'appuyant sur l'observation des auteurs de DeepWalk [Perozzi et al., 2014] montrant que la distribution des occurrences de sommets lors de marches aléatoires suit une loi puissance, similairement aux occurrences des mots, nous constatons dans nos données que la fréquence de co-occurrence des sommets suit également une loi puissance (voir figure 2.8), on peut estimer que s_{ij} peut être modélisée comme une loi puissance de coefficient α fonction du rang r_{ij} du nombre de co-occurrences de la paire de sommets (v_i, v_j) :

$$s_{ij} \propto \frac{1}{{r_{ij}}^{\alpha}}. \tag{2.27}$$

En reprenant le raisonnement suivi par les auteurs de GloVe [Pennington et al., 2014], on peut faire le lien entre la taille du corpus $|C|$ et le nombre de paires de sommets aux co-occurrences non nulles $|S|$:

$$|S| = \begin{cases} \mathcal{O}(|C|) & \text{si } \alpha < 1, \\ \mathcal{O}(|C|^{\frac{1}{\alpha}}) & \text{si } \alpha > 1. \end{cases} \tag{2.28}$$

Les valeurs de α pour les graphes sur lesquels nous travaillons sont souvent supérieures à 1 ce qui permet d'obtenir un gain par rapport aux méthodes s'appuyant sur SGNS. Cependant, contrairement à GloVe, GVNR introduit des coefficients nuls dans sa formulation. Le nombre de coefficients ajoutés est égal à $n_k \times |S|_{\neq 0}$. Seulement, le seuillage de la matrice de co-occurrences permet de très fortement réduire le nombre de valeurs

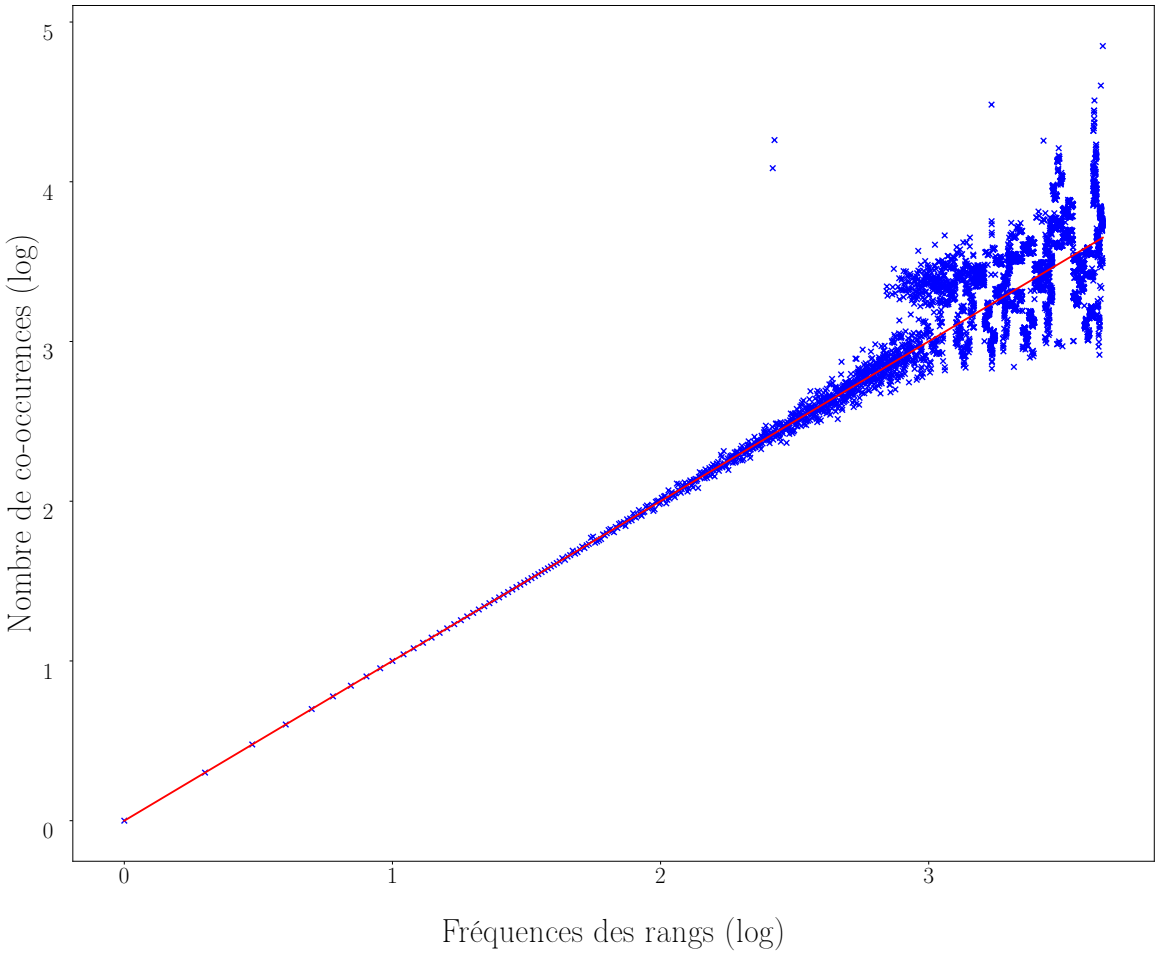


FIGURE 2.8 – Distributions des comptages de co-occurrences s_{ij} en fonction du rang de leurs valeurs dans le réseau de citations Cora.

non nulles. En effet, puisque s_{ij} suit une loi puissance, plus sa valeur est proche de zéro, plus elle est fréquente. En pratique, nous utilisons toujours $n_k = 1$ ce qui implique que la complexité de l'algorithme est comparable à celle de GloVe, à un facteur 2 près. Nous montrons empiriquement ci-dessous que le seuillage $s_{\min} = 1$, que nous utilisons dans nos expérimentations, permet néanmoins de fortement contrer cet effet en réduisant par plus de 2 le nombre de valeurs non nulles dans S.

Le tableau 2.1 montre, pour plusieurs jeux de données utilisés durant nos expérimentations, le nombre d'itérations réalisées selon les approches de type SGNS, selon GloVe et selon GVNR. Les coefficients directeurs α correspondant à l'équation 2.27 sont indiqués en dernière colonne. On voit que les nombres d'itérations de GloVe et GVNR sont nettement plus faibles car ces algorithmes n'opèrent qu'une itération pour chaque paire de sommets tandis que SGNS réalise autant d'itérations qu'il y a d'occurrences de cette paire dans le corpus C. De plus, on voit qu'à l'exception du jeu de données NYT le nombre d'itérations de GVNR est inférieur à celui de GloVe, grâce au seuillage des faibles valeurs de S. Notons que ces algorithmes réalisent en pratiques plusieurs époques sur le jeux de données qui sont du même ordre de grandeur. En l'occurrence, dans nos expérimentations, SGNS réalise 5 époques, GloVe 10 et GVNR 2 seulement.

Pour compléter cette analyse sur la complexité en temps de ces algorithmes, notons que LINE [Tang et al., 2015], présenté en section 2.3, est une méthode de plongement de réseau opérant directement sur la matrice d'adjacence du réseau. Les densités des matrices d'adjacence (voir tableau 1.2) des réseaux étudiés sont mécaniquement plus faibles que les

TABLE 2.1 – Comparaison des nombres d'itérations par époque dans SGNS, GloVe et GVNR selon différents jeux de données en considérant $\mu = 80$ marches aléatoires par sommet de longueurs $\ell = 40$ et une fenêtre de taille $\tau = 5$. GVNR utilise $n_k = 1$ échantillon négatif et un seuillage $s_{\min}$=1. On note que les coefficients des lois puissance des réseaux sont majoritairement supérieurs à 1.

Jeu de données	|V|	|E|	SGNS	GloVe	GVNR	α
Cora	2 211	9 542	67 214 400	858 315	**793 326**	0.98
NYT	5 135	6 101 026	156 104 000	**6 106 114**	12 205 092	1.00
Gaming SE	22 872	801 328	695 308 800	12 259 288	**7 754 520**	1.04
Travel SE	15 087	931 392	458 644 800	9 921 904	**5 752 762**	1.00
CiteSeer	3 312	9 102	100 684 800	**431 660**	493 024	1.00
Wikipedia	4 777	184 590	145 220 800	10 158 398	**5 741 006**	1.14
PPI	3 890	75 690	118 256 000	8 055 838	**7 230 492**	1.04
Stats SE	14 834	567 770	450 953 600	11 436 290	**7 376 892**	1.03
Academia SE	20 799	1 245 440	632 289 600	25 544 057	**13 129 272**	1.02

matrices de co-occurrences des sommets dans des marches aléatoires. Par exemple, cette densité et de 0.2% pour la matrice d'adjacence du réseau Cora alors que sa matrice de co-occurrences correspondant à nos expérimentations présente 4% de densité. Nous avons donc cherché à comparer notre méthode à LINE, or le nombre d'itérations employées dans les expérimentations présentées dans l'article est de 10 milliards. Lorsque nous avons testé l'algorithme avec un nombre d'itérations équivalents à SGNS (500 millions) nous n'avons pu aller au bout de l'apprentissage en un temps raisonnable sur nos machines. Lorsque nous avons testé l'algorithme avec un nombre d'itérations équivalents à GVNR (10 millions), les résultats obtenus sont très largement inférieurs aux autres méthodes. Pour ces raisons, nous ne nous comparons pas à LINE lors de nos expérimentations.

2.5 Méthodologies d'évaluation

Les algorithmes de plongement de réseau sont généralement utilisés à deux fins distinctes : la reconstruction des liens et la prédiction des classes attribuées aux sommets. La première consiste à prédire des liens entre sommets qui n'ont pas été directement observés par le modèle tandis que la seconde consiste à classer les sommets parmi des catégories fixées à l'avance. Dans cette section, je détaille ces deux tâches permettant d'évaluer la qualité des représentations apprises par un algorithme de plongement de réseau.

2.5.1 Prédiction de liens

Les représentations des sommets doivent être capables de préserver la structure d'un réseau. De fait, reconstruire uniquement les liens présents dans la matrice d'adjacence constitue une tâche sans intérêt car tout modèle avec suffisamment de paramètres sera capable de restituer l'intégralité des liens par surapprentissage.

Pour évaluer la capacité des représentations à capturer la structure du graphe, il est nécessaire de cacher durant l'apprentissage une portion des liens pour ensuite vérifier que la similarité entre les représentations des paires de sommets dont les liens ont été cachés est plus grande que la similarité entre les représentations de paires de sommets non connectés. De la sorte, on s'assure d'évaluer la capacité de l'algorithme à capturer les facteurs latents à l'origine des liens entre les sommets.

Dans l'espace des représentations, la similarité entre deux sommets peut être calculée de multiples façons. Néanmoins, la plupart des algorithmes de plongement de réseau sont

formulés de sorte à minimiser le produit scalaire entre les représentations de paires de sommets lorsque ceux-ci sont proches dans le graphe. Pour cette raison, la probabilité de l'existence d'un lien selon le modèle est souvent considérée comme étant proportionnelle à la similarité cosinus entre les vecteurs des deux sommets considérés. Lorsque ces vecteurs sont normalisés, on peut directement utiliser le produit scalaire, étant alors inversement proportionnel à la distance euclidienne.

Formellement, étant donnés un algorithme de plongement de réseau et un graphe de matrice d'adjacence A, l'évaluation consiste à cacher un certain pourcentage r des liens décrits dans A. On échantillonne de même autant de paires de sommets qui ne sont pas liés dans A afin de comparer les prédictions de liens existants avec des liens non-existants. Ensuite, l'algorithme construit des représentations U des sommets à partir du graphe partiellement caché et ces représentations sont utilisées pour calculer la similarité des paires de sommets issus des liens précédemment échantillonnés. Enfin, on compare ces prédictions à l'existence ou non d'un lien dans le graphe original. Pour ce faire, l'aire sous la courbe ROC (AUC pour *area under the receiver operating characteristic curve*) permet de mesurer empiriquement la probabilité que la similarité entre des sommets connectés soit plus grande que celle de deux sommets non connectés [Fawcett, 2006].

La figure 2.9 illustre ce protocole d'évaluation. Pour réduire l'impact du tirage aléatoire des liens dans A sur l'évaluation, on opère cette évaluation plusieurs fois (par exemple 10) et on calcule la moyenne des scores obtenus.

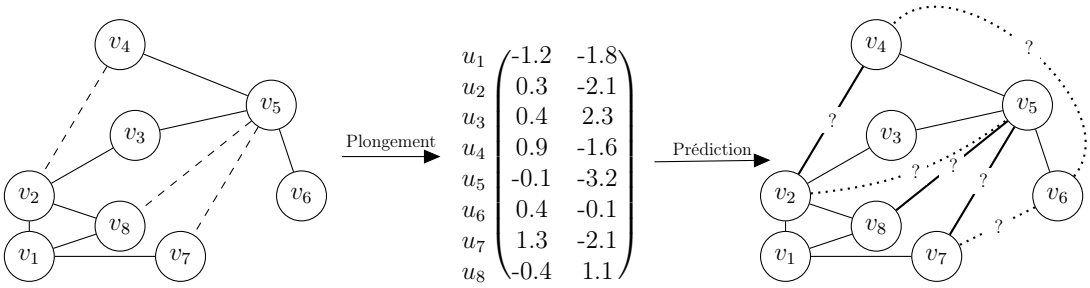


FIGURE 2.9 – Protocole d'évaluation utilisé pour évaluer GVNR sur la prédiction de liens. Certains liens d'un réseau sont retirés et des représentations des sommets sont construites à partir du réseau partiellement caché. Les liens cachés et certains liens inexistants sont prédits à partir des représentations.

2.5.2 Classification des sommets

La classification des sommets d'un graphe est le type d'évaluation le plus utilisé dans la littérature. Ce cas correspond à une situation où une petite proportion des sommets est associée à un ensemble de classes (ou *labels*) et où l'on voudrait étendre cette catégorisation à l'ensemble du graphe. Pour ce faire, on construit des représentations des sommets avec un algorithme de plongement de réseau. Afin d'évaluer la qualité de ces représentations, on utilise un classifieur linéaire pour prédire les catégories. En effet, on considère qu'un algorithme de plongement de réseau construit de bonnes représentations si les sommets n'appartenant pas aux mêmes catégories sont linéairement séparables. Dans le cas général, on peut associer plusieurs catégories au même sommet. On note alors L la matrice des catégories dans laquelle la ligne $l_{i.}$ désigne les indicateurs binaires d'appartenance à chaque catégorie du sommet v_i.

Formellement, étant donné un algorithme de plongement de réseau, une matrice d'adjacence A et un ensemble d'associations sommets-catégories L, l'évaluation consiste en

premier lieu à construire des représentations U des sommets par l'algorithme de plongement. On sélectionne ensuite un certain pourcentage r des sommets avec leurs catégories associées que l'on utilise pour entraîner un classifieur linéaire tel qu'un modèle de régression logistique. Enfin, on prédit les catégories pour chaque sommet n'ayant pas servi de donnée d'entraînement au classifieur. Pour mesurer la qualité des prédictions, on peut utiliser les probabilités de prédiction sur les catégories du modèle de régression logistique et les comparer aux véritables catégories L en utilisant là aussi la courbe ROC et l'AUC. Cette méthode présente l'avantage de prendre en compte l'ordre du classement produit par ces probabilités dans le calcul de l'erreur. On peut aussi calculer le score F_1 qui est la moyenne harmonique entre la précision et le rappel. Cette mesure permet de mieux estimer la capacité absolue du modèle linéaire à classer les sommets. Pour obtenir un score F_1 élevé, il faut que le classifieur ait une précision élevée, c'est-à-dire qu'il assigne les sommets à leurs classes réelles, et qu'il ait un rappel élevé, c'est-à-dire qu'il n'assigne pas de sommets aux mauvaises classes. Lorsque l'on calcule ces deux mesures pour plusieurs sommets, on peut agglomérer les résultats soit en moyennant les mesures sans pondération (*micro*) soit en pondérant selon les nombres de sommets appartenant à chaque classe (*macro*). Dans nos expérimentations, nous utilisons toujours la première méthode (*micro*) car l'impact de la distribution des *labels* sur la classification nous intéresse moins que la capacité des algorithmes de plongement à représenter chaque sommet. De plus, dans nos expérimentations, nous observons que les deux méthodes d'agglomération produisent des scores positivement corrélés.

La figure 2.10 illustre ce protocole d'évaluation. Similairement à l'évaluation pour la prédiction de liens, on réalise en pratique plusieurs fois cette évaluation pour limiter l'impact du tirage aléatoire des sommets d'entraînement. De plus, on fait varier la proportion r de ces sommets pour avoir une idée plus fine de la capacité de généralisation de l'algorithme dans l'apprentissage des représentations vis-à-vis de la quantité d'information disponible. Aussi, le modèle de régression logistique peut intégrer un terme de régularisation, telle qu'une pénalité L_2, dont le facteur de pondération est sélectionné par validation croisée. Enfin, certains jeux de données sont dits multi-*labels* car un sommet peut être associé à plusieurs classes à la fois. Afin de conserver la distribution des classes dans les données d'entraînements du classifieur, nous utilisons l'algorithme itératif de stratification [Sechidis et al., 2011].

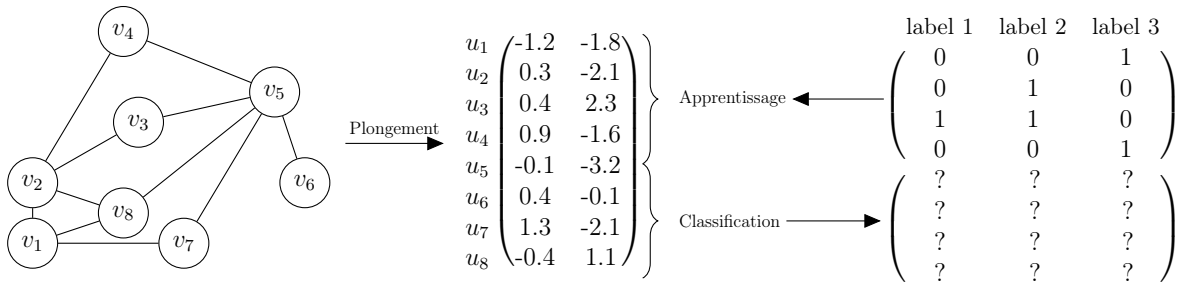


FIGURE 2.10 – Protocole d'évaluation utilisé pour évaluer GVNR sur la classification des sommets. Des représentations vectorielles des sommets sont apprises sur le réseau et une certaine proportion d'entre elles servent à l'apprentissage d'un modèle de classification linéaire. Le reste des représentations permet de prédire les *labels* cachés.

2.6 Expérimentations

Nous présentons ici les résultats des expérimentations mettant en valeur les performances de GVNR vis-à-vis des algorithmes de l'état de l'art. Nous montrons en premier lieu les résultats quantitatifs sur les tâches de classification et de prédiction de liens. Nous montrons ensuite des visualisations en deux dimensions des représentations apprises par GVNR et DeepWalk puis étudions l'effet des hyperparamètres de notre modèle. Enfin nous analysons l'impact du choix de la pondération de la fenêtre sur les marches aléatoires en terme de complexité en temps et de performances de classification.

2.6.1 Résultats quantitatifs

Nous présentons dans cette section les résultats expérimentaux obtenus sur la classification et la prédiction de lien. Nous détaillons tout d'abord les paramètres des évaluations puis présentons les algorithmes et implémentations utilisées pour conduire les expériences. Nous discutons ensuite les résultats en comparant GloVe avec GVNR et en analysant les performances de GVNR vis-à-vis des autres algorithmes de plongement de réseau.

Évaluations

Nous évaluons GVNR sur deux tâches, la classification des sommets et la prédiction de liens, toutes deux détaillées précédemment en section 2.5. Ces évaluations sont réalisées sur chacun des 9 jeux de données présentés en introduction de ce manuscrit en section 1.4.2.

Pour la classification, nous réalisons un apprentissage de représentations des sommets pour chaque algorithme. Nous utilisons de 10% à 50% de ces représentations et leurs *labels* respectifs pour entraîner un modèle de régression logistique [Kleinbaum et al., 2002] utilisant l'implémentation *LIBLINEAR* [Fan et al., 2008] et nous opérons une validation croisée du paramètre de régularisation L_2 à chaque entraînement. La prédiction des *labels* des sommets n'ayant pas été utilisés pour l'entraînement est ensuite réalisée afin de calculer le score F_1 et l'AUC des prédictions. Pour chaque pourcentage de données d'entraînement, nous opérons l'évaluation 10 fois et moyennons les scores obtenus, ceci afin de limiter l'impact du tirage aléatoire sur les performances obtenues.

Pour la prédiction de liens, nous enlevons aléatoirement 50% des liens constituant le réseau et appliquons les algorithmes de plongement sur le sous-réseau résultant. Les représentations apprises sont directement utilisées pour prédire l'existence de liens entre paires de sommets. Ceux-ci sont tirés de sorte que 50% des paires de sommets sont des paires dont un lien a été retiré du réseau original et que 50% de paires de sommets ne sont pas liés dans le réseau original. Notez que nous pourrions, comme pour la classification, faire varier le pourcentage de liens cachés. Cependant, ceci est en pratique plus délicat. D'une part il est nécessaire de ne pas trop réduire ce pourcentage afin d'éviter de créer des îlots de sommets déconnectés du reste du réseau et, d'autre part, si l'on augmente trop ce pourcentage la tâche devient trop facile à résoudre et il devient impossible de départager les algorithmes évalués. Nous opérons cette évaluation 10 fois afin de réduire l'impact du tirage aléatoire sur les performances. La probabilité d'existence d'un lien entre deux sommets est calculée à partir de la similarité cosinus entre leurs représentations respectives.

Algorithmes

Pour nos expérimentations, nous considérons 3 algorithmes de références : DeepWalk, NetMF et GloVe, et deux variations de GVNR, l'une sans seuillage de la matrice ($s_{\min} =$

0) et l'autre avec seuillage ($s_{\min} = 1$). De plus, nous montrons pour comparaison les performances obtenues par un algorithme aléatoire. La dimension de toute représentation est fixée à $\rho = 256$. Lorsqu'une méthode repose sur des marches aléatoires, nous réalisons systématiquement $\mu = 80$ marches de longueurs $\ell = 40$ et appliquons un fenêtre de taille $\tau = 5$. Ces valeurs permettent d'obtenir en moyenne les meilleurs résultats pour DeepWalk et GVNR sur l'ensemble des jeux de données et permettent alors de comparer les performances de ces deux algorithmes à prétraitements égaux. Voici les détails des implémentations que nous avons utilisées :

- Aléatoire : génère des représentations aléatoires ;
- DeepWalk : nous utilisons une version fournie par les auteurs de Node2Vec basée sur SGNS, utilisons $n_k = 10$ échantillons négatifs et réalisons 5 époques ;
- NetMF : nous utilisons la version approchée fournie pas les auteurs en appliquant les mêmes hyperparamètres que DeepWalk ;
- GloVe : nous implémentons GloVe avec l'algorithme d'optimisation ADAM [Kingma and Ba, 2014] en l'appliquant sur la matrice des comptages des co-occurrences dans les marches aléatoires. Nous prenons $s_{\max} = 10$ et réalisons 10 époques avec un taux d'apprentissage initial de 0.001 ;
- GVNR ($s_{\min} = 0$) : nous utilisons $n_k = 1$ échantillon négatif par échantillon positif sans seuillage et réalisons 2 époques avec un taux d'apprentissage initial de 0.001 ;
- GVNR ($s_{\min} = 1$) : nous utilisons $n_k = 1$ avec un seuillage $s_{\min} = 1$ et réalisons 2 époques avec un taux d'apprentissage initial de 0.001.

Les tableaux 2.2 à 2.10 montrent les résultats des évaluations. Nous les analysons dans les sections suivantes.

GloVe, GVNR et seuillage

GVNR et GloVe constituent deux modèles relativement proches. Les différences majeurs entre ceux-ci résident dans la prises en compte des valeurs nulles dans la matrice S factorisée ainsi que dans le seuillage de cette dernière. Les résultats montrent que GVNR est nettement meilleur en classification sur la majorité des jeux de données, à l'exception du réseau du langage Wikipedia et du réseau d'interactions protéine-protéine PPI. Ces deux jeux de données ont la particularité de présenter les densités les plus grandes (mis à part le New York Time qui constitue un cas très particulier sur lequel nous revenons ultérieurement), c'est-à-dire que le nombre moyen de voisins des sommet est particulièrement élevé. Le grand nombre de faibles comptages de co-occurrences générés dans ce genre de réseau peut expliquer ces bonnes performances. En effet, comme nous en discutons en section 2.4.2, GloVe réalise implicitement une forme d'échantillonage négatif grâce aux valeurs de s_{ij} inférieurs à 1. Ce nombre de valeurs est plus probable de se produire lorsque le nombre de voisins moyens est élevé, puisque le nombre de chemins accessibles au marcheur est décuplé. En prédiction de lien, GVNR réalise systématiquement de meilleures prédictions que GloVe, à l'exception des jeux de données de Q&A issus de *Stack Exchange*, où les performances sont difficilement départageables. Enfin, GVNR avec seuillage ($s_{\min} = 1$) réalise toujours de meilleurs scores en classification que sans seuillage ($s_{\min} = 0$) mais réalise néanmoins de moins bonnes prédictions de liens sur les deux réseaux de citation Cora et CiteSeer, les performances des deux variantes étant tout de même nettement meilleures que celles des autres méthodes. Nous étudions en section 2.6.4 plus particulièrement l'impact du seuillage en combinaison du choix de la pondération de la fenêtre.

TABLE 2.2 – Résultats expérimentaux obtenus sur le réseau de citations Cora

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	14.44	13.62	13.85	14.26	13.18	52.61	51.71	51.24	51.60	51.59	51.24
NetMF	68.22	74.97	77.60	79.80	81.05	89.38	92.81	93.77	94.70	95.69	66.53
DeepWalk	74.09	76.98	78.53	78.36	79.82	94.21	95.20	95.70	95.80	96.13	70.50
GloVe	65.01	65.82	68.95	68.68	69.64	90.42	90.78	92.37	92.52	93.34	71.97
GVNR ($s_{\min} = 0$)	74.04	77.12	78.64	78.60	79.86	94.07	95.22	95.88	96.04	96.39	**78.10**
GVNR ($s_{\min} = 1$)	**76.01**	**78.50**	**80.13**	**80.54**	**81.45**	**95.12**	**95.99**	**96.41**	**96.55**	**96.90**	76.26

TABLE 2.3 – Résultats expérimentaux obtenus sur le réseau de citations CiteSeer

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	15.79	16.26	15.49	14.96	15.48	49.95	50.43	49.95	49.13	49.29	48.32
NetMF	51.72	**57.74**	**60.17**	**63.27**	**65.68**	77.43	82.16	82.58	83.61	84.77	60.55
DeepWalk	51.19	56.32	57.77	59.14	60.19	79.83	82.96	83.97	84.94	85.88	59.75
GloVe	38.24	40.43	41.49	42.24	42.13	69.15	71.17	72.75	73.69	73.97	65.04
GVNR ($s_{\min} = 0$)	50.61	52.54	53.49	52.81	54.25	80.37	82.49	83.15	83.39	84.13	**73.18**
GVNR ($s_{\min} = 1$)	**53.58**	55.46	57.02	57.84	59.28	**82.07**	**83.45**	**84.82**	**85.13**	**85.98**	71.19

TABLE 2.4 – Résultats expérimentaux obtenus sur le réseau New York Time

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	25.98	22.52	20.20	21.15	20.24	58.03	54.73	53.22	53.36	53.30	49.91
NetMF	63.06	60.55	61.54	61.23	59.03	**87.65**	**87.35**	86.51	**87.53**	**87.58**	55.61
DeepWalk	**64.34**	**63.57**	**62.08**	**62.86**	**61.32**	86.69	86.81	**86.77**	85.96	86.31	**99.78**
GloVe	59.41	57.40	57.03	57.05	56.22	84.36	83.78	82.77	82.99	83.07	50.60
GVNR ($s_{\min} = 0$)	59.81	58.40	56.88	57.17	56.39	86.02	84.69	84.60	82.88	82.30	99.76
GVNR ($s_{\min} = 1$)	61.43	60.53	59.29	58.09	56.40	86.79	86.73	85.72	83.73	83.73	99.73

TABLE 2.5 – Résultats expérimentaux obtenus sur le réseau Gaming SE

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	5.89	5.32	5.29	5.33	5.34	60.10	56.41	54.45	53.67	52.84	49.75
NetMF	26.09	26.33	29.11	30.15	32.45	**73.80**	**78.65**	79.20	80.61	83.98	97.69
DeepWalk	**29.47**	**31.59**	**32.62**	**34.74**	**34.49**	71.38	77.65	**80.83**	**83.29**	**84.85**	97.21
GloVe	11.99	13.12	12.58	13.19	13.66	49.75	59.95	68.53	72.90	75.59	**98.09**
GVNR ($s_{\min} = 0$)	19.93	23.53	26.49	28.80	30.53	64.48	73.84	75.20	77.08	80.25	97.54
GVNR ($s_{\min} = 1$)	25.86	29.04	31.23	32.65	33.36	69.47	75.25	77.21	79.91	82.15	98.06

TABLE 2.6 – Résultats expérimentaux obtenus sur le réseau Travel SE

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	7.25	6.89	6.67	6.91	6.61	58.47	54.14	53.66	53.20	52.47	52.03
NetMF	10.12	11.57	13.02	13.88	13.88	58.16	63.12	64.00	66.23	66.68	97.99
DeepWalk	**12.39**	**13.28**	13.42	**14.90**	**15.16**	59.54	**63.45**	**66.48**	**67.86**	**68.22**	97.44
GloVe	8.85	9.85	10.44	10.32	10.61	53.62	55.60	59.64	62.41	63.70	98.82
GVNR ($s_{\min} = 0$)	9.96	11.50	12.13	13.17	13.54	58.31	61.62	63.95	65.13	65.65	**98.94**
GVNR ($s_{\min} = 1$)	9.77	12.56	**13.48**	14.57	14.45	**63.64**	62.99	63.55	65.85	65.81	98.82

TABLE 2.7 – Résultats expérimentaux obtenus sur le réseau Wikipedia

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	20.41	21.74	18.53	21.33	19.74	55.11	54.85	53.35	53.55	52.28	50.21
NetMF	20.67	22.94	23.87	21.89	23.15	65.55	53.73	56.25	58.21	59.45	51.23
DeepWalk	**39.20**	**41.09**	**40.50**	**40.80**	**39.51**	**74.36**	75.47	**77.20**	**76.90**	**77.04**	58.78
GloVe	31.75	39.14	37.04	37.75	39.37	70.39	**76.34**	74.16	71.46	73.62	57.55
GVNR ($s_{\min} = 0$)	26.80	28.01	27.96	28.62	28.76	70.48	66.59	65.58	64.38	65.49	61.76
GVNR ($s_{\min} = 1$)	31.24	34.18	34.01	34.91	33.82	70.28	75.95	69.62	70.54	71.48	**68.27**

TABLE 2.8 – Résultats expérimentaux obtenus sur le réseau PPI

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	6.87	6.55	6.26	6.32	6.32	56.71	53.75	52.51	51.64	51.61	48.44
NetMF	10.77	13.13	13.90	**15.03**	**14.76**	**59.79**	**62.52**	**62.71**	**64.37**	62.28	68.70
DeepWalk	11.52	13.53	12.48	13.76	13.22	56.21	60.28	59.45	61.70	64.22	71.20
GloVe	**13.85**	**14.97**	**14.21**	14.79	14.17	57.55	61.08	61.31	62.11	64.11	69.31
GVNR ($s_{\min} = 0$)	11.61	11.62	11.73	12.16	11.84	53.45	58.42	57.95	58.39	61.96	65.96
GVNR ($s_{\min} = 1$)	12.33	13.29	13.85	13.40	13.56	57.71	59.70	61.62	62.02	**64.51**	**76.69**

TABLE 2.9 – Résultats expérimentaux obtenus sur le réseau Stats SE

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	7.31	6.44	6.29	6.26	6.23	60.62	55.42	54.04	52.09	52.58	48.41
NetMF	13.40	13.86	15.25	15.95	17.01	**64.89**	66.38	68.20	68.49	69.20	92.48
DeepWalk	14.86	**15.31**	16.22	17.30	18.06	63.00	**68.04**	**69.75**	**71.13**	**71.13**	97.08
GloVe	11.46	11.93	13.21	15.00	13.64	60.42	63.27	64.88	66.16	68.38	**98.01**
GVNR ($s_{\min} = 0$)	**14.94**	15.19	16.22	17.72	17.38	63.57	67.12	68.20	67.94	69.51	97.98
GVNR ($s_{\min} = 1$)	13.83	15.20	**17.31**	**18.45**	**18.45**	62.85	65.62	67.18	68.77	70.56	97.94

TABLE 2.10 – Résultats expérimentaux obtenus sur le réseau Academia SE

	Classification										Prédiction
	F1					AUC					AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	50%
Aléatoire	7.46	7.66	7.36	7.19	7.44	59.63	56.71	53.05	52.86	53.48	48.25
NetMF	9.56	11.18	11.97	13.00	14.18	**61.52**	61.68	64.50	59.75	64.62	98.66
DeepWalk	**11.58**	10.71	**12.52**	13.48	**14.84**	60.59	**65.09**	**66.66**	**67.42**	**69.15**	98.17
GloVe	11.11	**12.72**	12.00	13.21	13.20	54.92	59.84	61.45	64.81	67.49	98.57
GVNR ($s_{\min} = 0$)	10.04	12.04	12.30	12.92	13.72	59.81	62.77	64.95	65.84	67.91	**99.08**
GVNR ($s_{\min} = 1$)	10.37	11.64	12.35	**14.17**	14.68	60.62	62.10	64.38	66.22	68.05	98.84

Classification

Sur la tâche de classification des sommets, le duo DeepWalk et NetMF réalisent généralement des résultats similaires sauf sur Wikipedia où DeepWalk est nettement le meilleur algorithme. Ces deux méthodes sont plus performantes que GVNR sur NYT, Gaming SE et Wikipedia et sont moins bonnes sur PPI, Cora et CiteSeer. Sur les autres jeux de données, GVNR réalise des performances similaires à DeepWalk, étant meilleur lorsque pourcentage de données d'entraînement est grand et étant légèrement moins bon sur les petits pourcentages. Deux jeux de données méritent une attention particulière. Premièrement, sur Wikipedia, DeepWalk est largement le meilleur modèle en classification alors que GVNR l'est en prédiction de liens, confirmant que ces deux tâches ne reposent pas sur les mêmes mécanismes. Deuxièmement, sur CiteSeer en classification, NetMF est nettement meilleur en terme de score F_1 alors que GVNR l'est en terme de AUC. Ceci montre que, sur ce jeu de données, si NetMF est le modèle le plus à même de produire des représentations efficaces pour prédire de manière exacte (en terme de précision et rappel) les bonnes classes associées aux sommets, GVNR produit de meilleures représentations pour l'estimation des probabilités d'appartenances des sommets aux classes. En d'autres termes, lorsque la régression logistique entraînée sur les représentations de GVNR se trompe de prédiction, elle attribue tout de même de plus grandes probabilités d'appartenir aux bonnes classes qu'avec les représentations de NetMF.

Prédiction de liens

Sur la tâche de prédiction de liens, GVNR est constamment l'un des meilleures algorithmes. En particulier, sur Cora, CiteSeer, PPI et Wikipedia, GVNR avec ou sans seuillage réalise des scores fortement supérieurs aux autres méthodes. Sur les réseaux Stack Exchange, les performances réalisées par tous les algorithmes sont toutes très proches du score parfait. Une particularité sur Wikipedia et particulièrement sur NYT est que NetMF et GloVe réalisent des scores proches de l'aléatoire, montrant qu'ils sont presque incapables de reconstruire le réseau original. Le point commun entre ces deux réseaux est leur degré moyen élevé. Pour étudier ce phénomène, nous pourrions appliquer ces deux algorithmes sur des réseaux générés artificiellement en faisant varier le degré moyen. En identifiant le point de rupture où les scores diminuent, nous pourrions alors mieux identifier les raisons de ces performances.

2.6.2 Visualisation des représentations

Les figures 2.11 et 2.12 montrent des visualisations en deux dimensions des représentations apprises sur le réseau Cora respectivement avec GVNR et DeepWalk, dont les couleurs représentent les *labels* qui leurs sont associés. Pour faire ces visualisations, nous avons utilisé l'algorithme UMAP (*Uniform Manifold Approximation and Projection for Dimension Reduction*) [McInnes et al., 2018] sur les représentations apprises en dimension $\rho = 256$ considérant un nombre de voisins par individus de 100 et une distance minimale de 0.5.

Pour GVNR, les représentations sont généralement bien séparées selon leurs *labels*, à l'exception de la catégorie « Reinforcement Learning » qui présente trois îlots distincts et des catégories « Theory » et « Rule Learning » qui sont difficilement séparables entre elles. Nous verrons dans le chapitre suivant, en section 3.6.3, que la prise en compte de l'information textuelle dans l'apprentissage des représentations permet d'améliorer la visualisation de ce réseau dans le plan. À titre de comparaison, cette visualisation pour DeepWalk semble mieux séparer le *label* « Reinforcement Learning » mais les groupements sont généralement plus éparses. Il n'est cependant ici pas question de tirer de plus amples

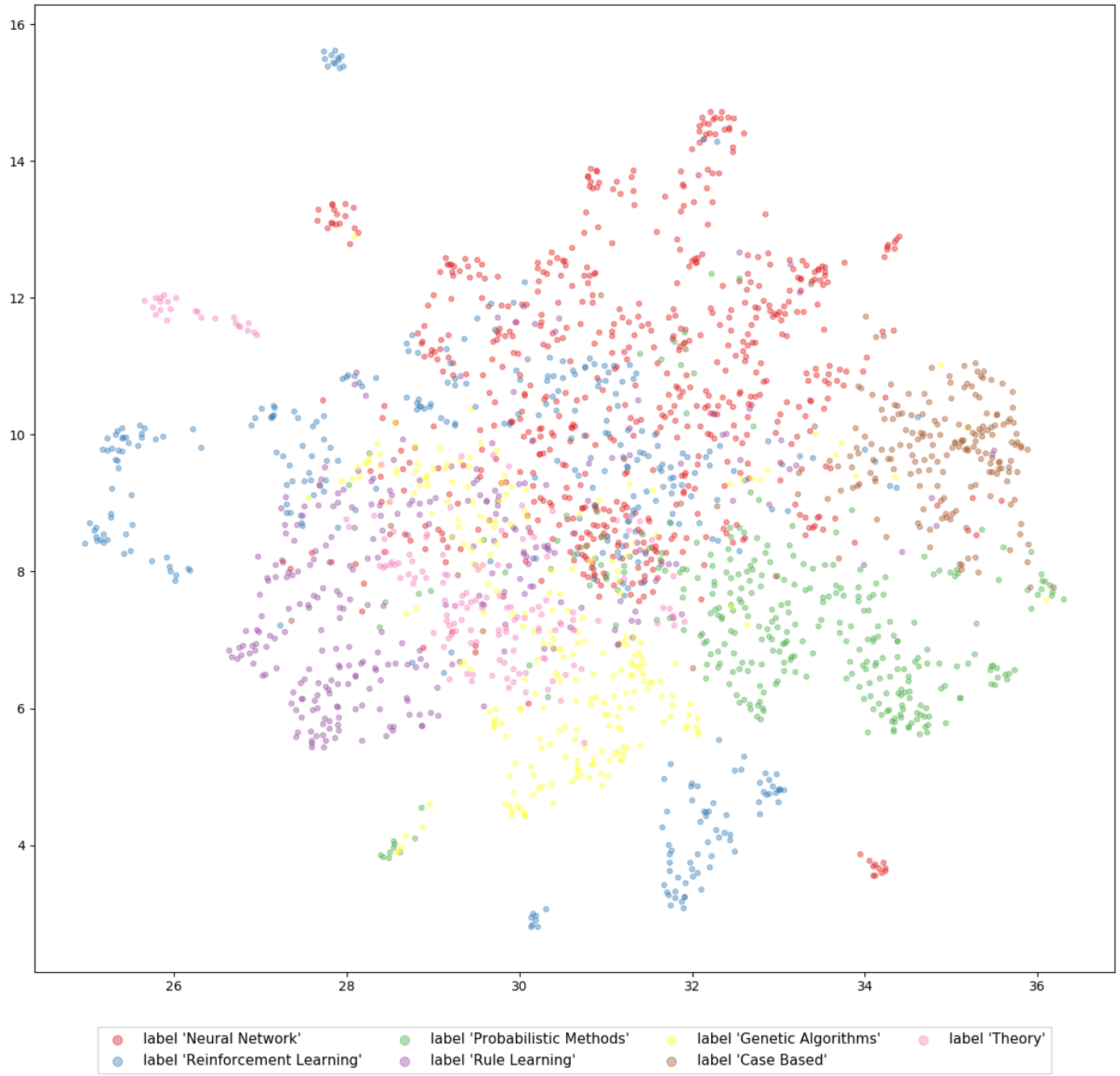


FIGURE 2.11 – Visualisation des représentations obtenues par GVNR sur Cora, colorées selon leurs *labels*.

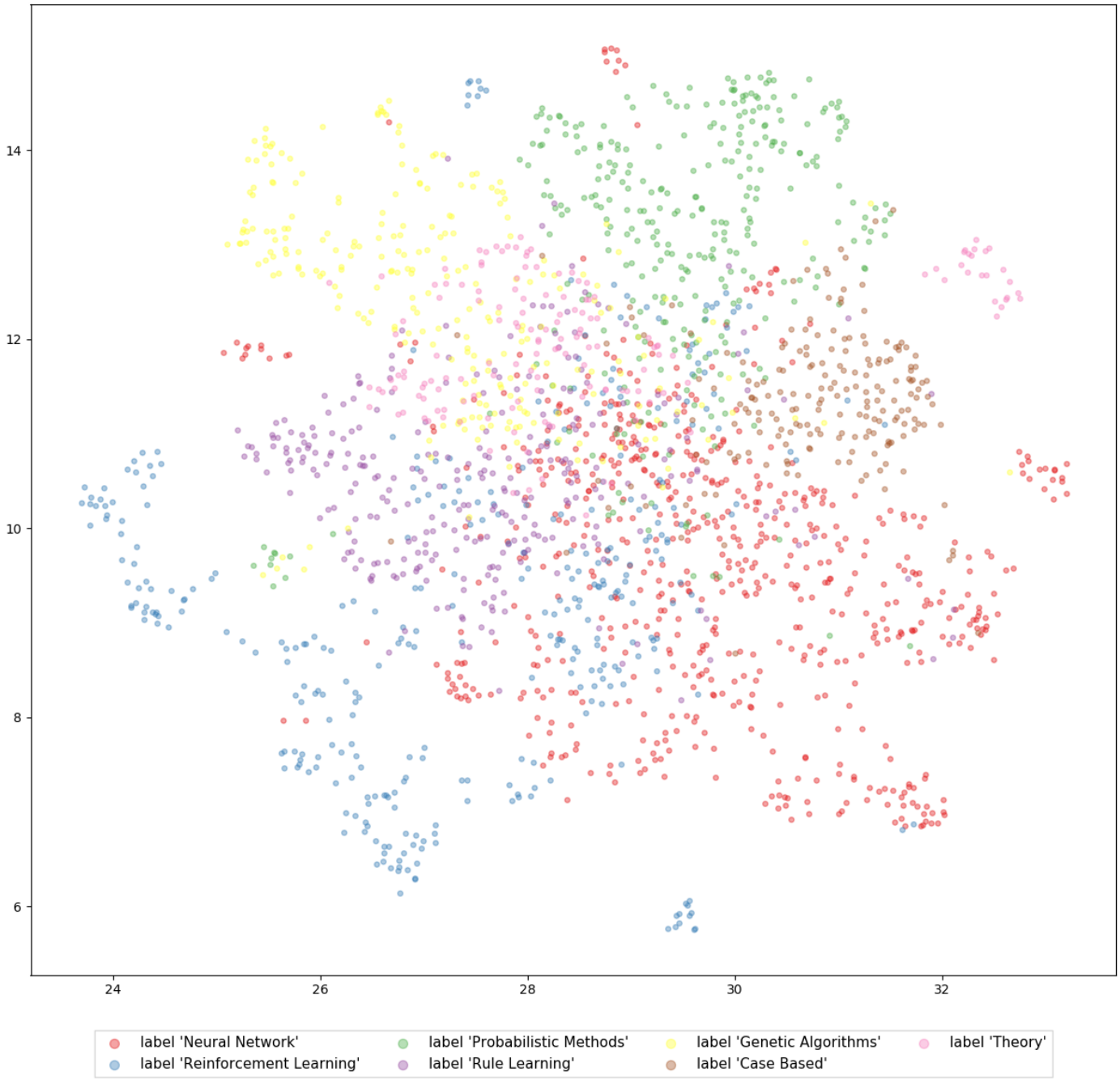


FIGURE 2.12 – Visualisation des représentations obtenues par DeepWalk sur Cora, colorées selon leurs *labels*.

conclusions sur ces visualisations car elles sont très sensibles aux hyperparamètres de l'algorithme UMAP et à la géométrie des représentations apprises par les deux algorithmes de plongement. Pour aller plus loin, il faudrait réaliser une analyse similaire à [Mimno and Thompson, 2017] comparant les propriétés des représentations apprises par SGNS et GloVe.

Dans la section suivante, nous étudions l'impact des hyperparamètres de GVNR sur la qualité des représentations apprises.

2.6.3 Étude des hyperparamètres

Nous étudions ici le comportement de GVNR en fonction des hyperparamètres. Nous rapportons en figure 2.13 les performances obtenues par l'algorithme sur Cora en classification en faisant varier la valeur du seuillage $s_{\min}$, le nombre d'échantillons négatifs n_k, la taille de la fenêtre τ et la dimension des représentations ρ. Pour toutes les expérimentations, nous utilisons les hyperparamètres suivants (lorsqu'ils ne sont pas étudiés) : $s_{\min} = 1$, $n_k = 1$, $\tau = 5$, $\rho = 256$ et réalisons au préalable $\mu = 80$ marches de longueurs $\ell = 40$.

Le modèle fonctionne mieux avec une valeur de seuillage $s_{\min}$ comprise entre 1 et 8. Notamment, le choix d'une valeur strictement supérieure à zéro permet d'améliorer nettement les performances. Sur plusieurs jeux de données, notamment sur les plus volumineux, le choix de $s_{\min} = 1$ est le meilleur compromis. Nous observons qu'un nombre d'échantillons négatifs $n_k = 1$ apporte constamment une amélioration par rapport à $n_k = 0$, des valeurs encore plus élevées ne provoquant sur certains jeux de données que des améliorations ponctuelles, marginales et uniquement pour des ratios de données d'entraînement élevés. Les performances varient grandement selon les pourcentages d'entraînement avec une taille de fenêtre τ comprise entre 2 et 5, avec une forme de stabilisation pour des valeurs plus grandes. La taille de fenêtre optimale est très spécifique au jeu de données étudié, mais la valeur $\tau = 5$ constitue un compromis acceptable sur tous les jeux. Enfin, la dimension ρ des représentations affecte nécessairement les performances. Plus grande est celle-ci, meilleurs sont les résultats avec une forme de stabilisation au-delà de $\rho = 128$.

2.6.4 Pondération de la fenêtre

Dans cette section nous étudions l'impact du choix de la pondération utilisée dans la fenêtre calculant les coefficients de co-occurrences entre sommets sur les marches aléatoires. Étant donnée q la distance séparant deux sommets v_i et v_j dans la fenêtre, on note λ le coefficient par lequel on incrémente l'entrée correspondante dans la matrice des comptages s_{ij}. Trois schémas de pondérations sont communément employés dans la littérature. Le premier consiste simplement à ne pas pondérer selon la distance entre le sommet cible et le sommet contexte dans la fenêtre. Ainsi, une paire de sommets adjacents ($q = 1$) et une paire de sommets distants de $q = 5$ dans la fenêtre seront toutes deux incrémentées de 1 dans la matrice des comptages, soit $\lambda = 1$. Deuxièmement, GVNR utilise le même schéma que GloVe avec un coefficient $\lambda = \frac{1}{q}$. Troisièmement, Word2vec utilise une pondération qui diminue linéairement avec la distance, soit $\lambda = \frac{\tau - q + 1}{\tau}$, où τ est la taille de la fenêtre. La figure 2.14 montre le profil de ces trois schémas de pondération.

Dans un premier temps, nous montrons dans le tableau 2.11 l'effet de ces pondérations sur plusieurs jeux de données, pour des valeurs de seuillage $s_{\min} = 0$ et $s_{\min} = 1$, sur la densité de la matrice S. Nous mettons ainsi en lumière l'efficacité de la réduction lorsque la fonction de pondération $\lambda(q)$ décroît fortement.

Nous montrons ensuite dans les tableaux 2.12 et 2.13 l'effet de ces pondérations et seuillages sur les performances obtenues sur Cora et Wikipedia. On voit que les deux

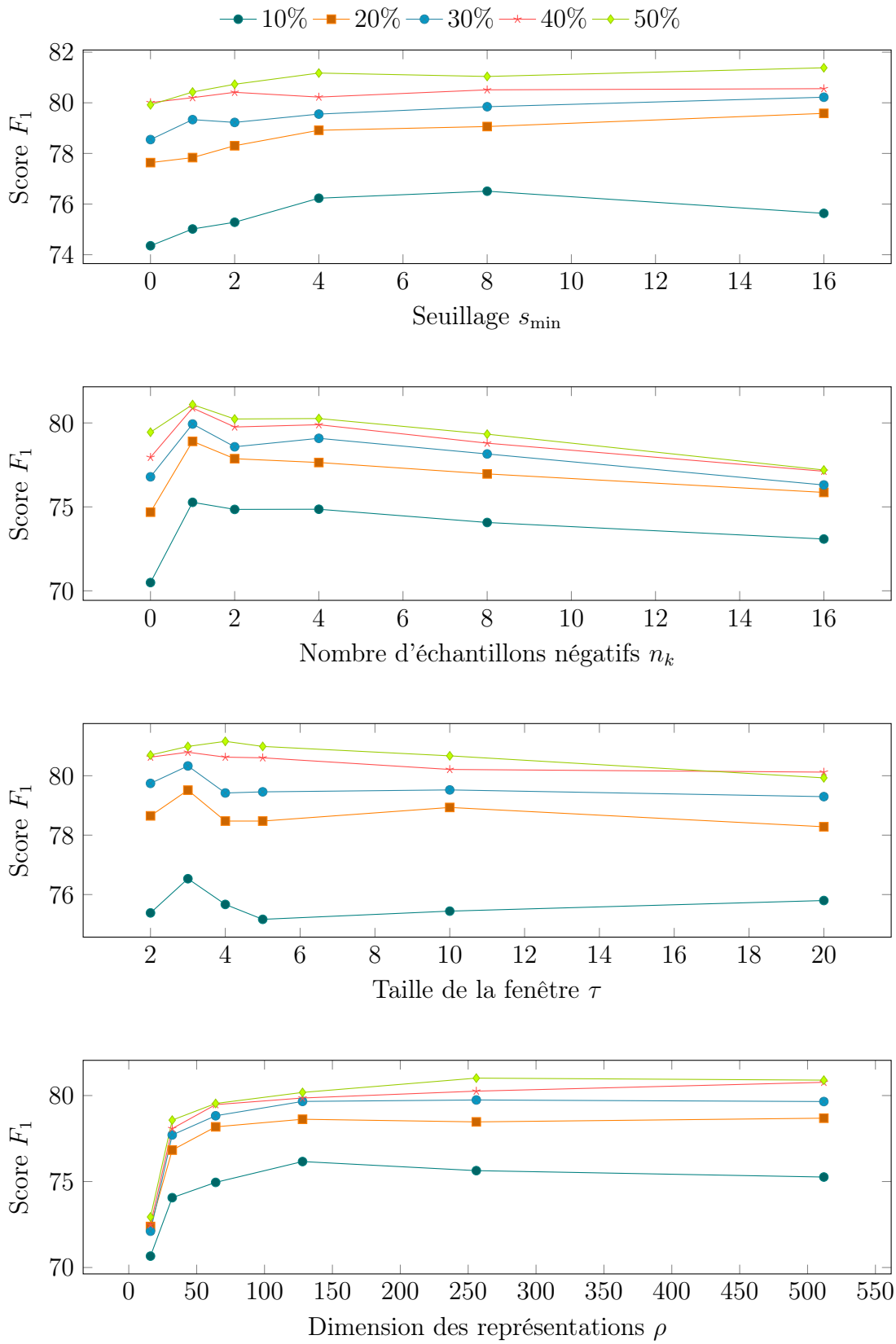


FIGURE 2.13 – Effets des hyperparamètres $s_{\min}$, n_k, τ et ρ sur les performances de GVNR pour la classification des sommets du jeu de données Cora selon plusieurs pourcentages de données d'entraînement.

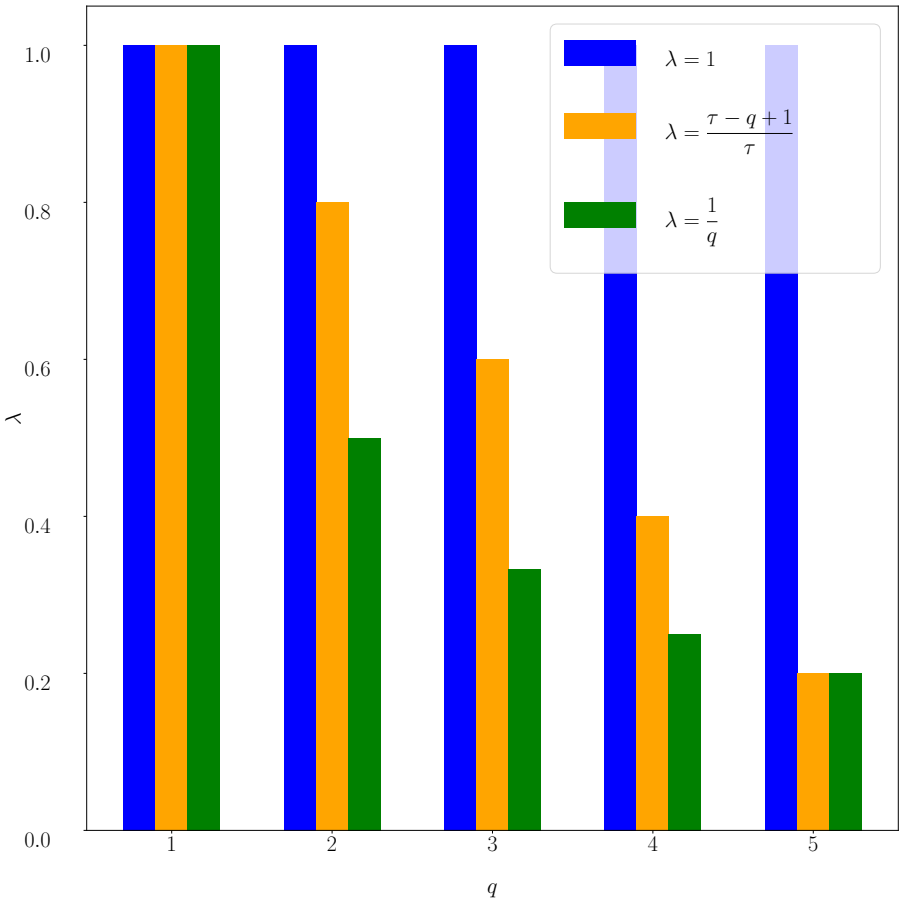


FIGURE 2.14 – Visualisation des trois schémas de pondération λ utilisés dans nos expérimentations avec une fenêtre de taille $\tau = 5$, en fonction de la distance absolue q entre un sommet cible et un sommet contexte dans une séquence.

schémas $\lambda = \frac{1}{q}$ et $\lambda = \frac{\tau - q + 1}{\tau}$ combinés avec le seuillage $s_{\min} = 1$ réalisent significativement de meilleures performances que les autres combinaisons possibles, particulièrement sur Wikipedia. Seulement, à performances égales, la pondération $\lambda = \frac{1}{q}$ utilisée avec GVNR nécessite 3% moins de temps sur Cora et 30% moins de temps sur Wikipedia pour réaliser l'apprentissage.

2.6.5 Résumé des résultats expérimentaux

Nous avons vu que GVNR obtient généralement de meilleurs scores que GloVe, sauf sur deux jeux de données particulièrement denses : un réseau d'interactions protéine-protéine et un réseau du langage. Le seuillage de la matrice S apporte constamment une nette amélioration des représentations apprises. Comparé aux algorithmes de la littérature en classification, GVNR semble meilleur sur les réseaux de citations, obtient des scores similaires sur les réseaux issus de forum en ligne et présente des difficultés sur des réseaux denses tels que celui du New York Time et Wikipedia, là où DeepWalk présente de nettement meilleures performances. En prédiction de liens, GVNR produit de meilleurs scores sur l'ensemble des jeux de données.

L'étude de l'impact des hyperparamètres montre que $1 \leq s_{\min} \leq 10$ est un bon choix de seuillage et que $n_k = 1$ semble la valeur optimale du nombre d'échantillons négatifs. La taille de la fenêtre devrait être choisie spécifiquement à chaque jeu de données, mais par souci de simplicité, nous avons identifié que $\tau = 5$ réalise un bon compromis sur la plupart

TABLE 2.11 – Densités de matrices des co-occurrences en fonction de la pondération λ de la fenêtre et du filtrage $s_{\min}$ appliqués. En gras sont indiqués les plus petits pourcentages de densité pour chaque jeu de données.

λ	$s_{\min}$	Cora	Travel SE	Wikipedia	PPI
1, $\frac{\tau-q+1}{\tau}$ et $\frac{1}{q}$	0	17.56%	4.35%	44.55%	53.27%
1	1	12.72%	2.24%	23.85%	37.82%
$\frac{\tau-q+1}{\tau}$	1	8.84%	1.52%	17.72%	28.77%
$\frac{1}{q}$	1	**8.09%**	**1.26%**	**12.58%**	**23.89%**

TABLE 2.12 – Effets de la pondération λ de la fenêtre et du filtrage $s_{\min}$ sur les performances de classification de GVNR sur Cora

λ	$s_{\min}$	F1					AUC					Temps
		10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	
1	0	75.04	78.65	79.46	79.92	80.69	94.89	95.81	96.20	96.36	96.75	772s
$\frac{\tau-q+1}{\tau}$	0	74.84	78.11	79.55	79.77	81.14	94.39	95.28	96.04	96.09	96.57	702s
$\frac{1}{q}$	0	74.34	78.18	79.70	79.85	80.94	94.04	95.24	95.84	96.01	96.48	718s
1	1	75.48	**78.98**	**80.01**	80.14	**81.46**	**94.99**	**96.06**	**96.32**	96.47	**96.82**	701s
$\frac{\tau-q+1}{\tau}$	1	75.69	78.47	79.57	79.92	80.94	94.75	95.63	96.01	96.36	96.70	642s
$\frac{1}{q}$	1	**76.01**	78.75	79.74	**80.26**	80.81	94.90	95.71	96.13	**96.49**	96.75	**625s**

des données. Enfin, les performances augmentent systématiquement avec le nombre de dimensions ρ, même si cette augmentation devient marginale au-delà de $\rho = 128$. De plus, nous avons testé différents schémas de pondération λ de la fenêtre proposés dans la littérature. Si les fonctions $\lambda = \frac{\tau-q+1}{\tau}$ et $\lambda = \frac{1}{q}$ produisent des performances similaires, cette dernière réduit fortement la complexité algorithmique en temps de GVNR.

2.7 Conclusions et perspectives

Dans ce chapitre, nous avons présenté GVNR, une méthode de plongement de réseau reposant sur la factorisation d'une matrice des comptages de co-occurrences des sommets dans des marches aléatoires. À la différence de GloVe, notre méthode considère pour la factorisation les paires de sommets qui ne co-occurrent pas tout en négligeant les faibles valeurs de co-occurrence. Cette formulation de la factorisation permet notamment d'améliorer les performances en terme de classification des sommets et de prédiction de liens mais aussi de réduire la complexité en temps de l'algorithme. Nous montrons expérimentalement que GVNR a une performance proche voire parfois meilleure que les algorithmes de l'état de l'art, pour une complexité moindre.

Pour compléter nos travaux, il nous semblerait nécessaire d'approfondir plusieurs aspects évoqués dans ce chapitre :

TABLE 2.13 – Effets de la pondération λ de la fenêtre et du filtrage $s_{\min}$ sur les performances de classification de GVNR sur Wikipedia

λ	$s_{\min}$	F1					AUC					Temps
		10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	
1	0	22.88	25.69	28.22	28.86	28.77	67.15	67.08	70.61	69.81	67.73	6031s
$\frac{\tau-q+1}{\tau}$	0	24.26	28.02	29.17	30.60	30.25	67.91	69.49	71.47	69.16	69.41	6048s
$\frac{1}{q}$	0	25.38	28.49	30.16	30.05	30.09	63.63	69.84	71.16	70.12	69.15	6165s
1	1	27.47	32.28	30.87	32.98	31.75	67.64	70.94	72.03	70.92	71.14	3768s
$\frac{\tau-q+1}{\tau}$	1	**31.95**	**34.79**	34.99	35.37	35.62	**72.75**	75.49	74.11	74.57	**74.01**	3409s
$\frac{1}{q}$	1	31.36	34.64	**35.71**	**36.51**	**35.64**	70.92	**76.06**	**75.24**	**74.91**	73.81	**2669s**

— **l'échantillonnage négatif :** nous avons vu que la plupart des méthodes de plongement utilisent un moyen d'optimiser la vraisemblance du modèle sans avoir à estimer les probabilités de l'intégralité des paires de sommets possibles. Skip-Gram avec échantillonage négatif simplifie notamment la méthode d'estimation contrastive bruitée, dont il est prouvé que sa solution converge vers la solution réelle lorsque n_k tend vers l'infini. GloVe réalise la factorisation de S en écartant les valeurs nulles de la matrice, mais comme nous l'indiquons en section 2.4.2, les valeurs s_{ij} inférieures à 1 remplissent un rôle similaire à un échantillonnage négatif dans SGNS. Enfin, notre méthode s'appuie sur le seuillage de S afin de réduire significativement le nombre d'échantillons négatifs nécessaires à la bonne optimisation des paramètres. Il serait intéressant d'approfondir à la fois la compréhension théorique de ces méthodes d'optimisation mais aussi d'identifier empiriquement les forces et faiblesses de celles-ci en fonction de la nature des données étudiées (structure du réseau, origine, etc.). Il faudrait notamment éclaircir les raisons pour lesquelles NCE et SGNS bénéficient toujours d'un nombre d'échantillonnages négatifs plus élevé tandis que GVNR semble atteindre une valeur optimale pour $n_k = 1$;
— **la similarité S :** les méthodes de plongement de réseau de la littérature opèrent leur apprentissage implicitement ou explicitement à partir d'une matrice capturant les similarités entre sommets en terme de communauté. La matrice laplacienne d'un graphe, les puissances de la matrice d'adjacence ainsi que les marches aléatoires permettent d'identifier les groupements de sommets qui sont particulièrement interconnectés. Deux méthodes de la littérature cependant proposent d'élargir le type de similarité capturée par l'algorithme : (1) Node2vec [Grover and Leskovec, 2016] grâce à ses marches biaisées favorisant l'exploration en largeur des sommets et Verse [Tsitsulin et al., 2018], qui propose notamment d'utiliser une matrice de similarité obtenue avec SimRank [Jeh and Widom, 2002], identifiant les sommets aux rôles structurels semblables. Si les communautés d'un réseau reflètent le phénomène d'homophilie souvent sous-jacent à leur construction, il est certain que d'autres phénomènes peuvent entrer en jeu et un algorithme de plongement pourrait alors bénéficier d'une similarité S prenant en compte la diversité des caractéristiques topologiques d'un graphe. Une étude analytique sur la corrélation entre les performances de classification et de prédiction de liens en fonction du choix de la similarité S approchée par les algorithmes de plongement nous semble être une piste de réflexion importante ;
— **la pondération de la fenêtre :** les marches aléatoires fournissent un cadre puissant pour l'exploration des grands réseaux. Cependant, le passage des marches à une similarité entre les sommets S se fait par l'extraction de co-occurrences dans une fenêtre glissante sur les séquences obtenues. À notre connaissance, il n'existe aucun cadre théorique permettant de comprendre l'impact de la taille et de la pondération de cette fenêtre. Il semble alors naturel de chercher à optimiser ces paramètres directement sur les données. C'est notamment l'approche choisie dans [Abu-El-Haija et al., 2018] où les facteurs λ pondérant les puissances de la matrice d'adjacence sont appris sur les données à travers un mécanisme d'attention. Néanmoins, cet apprentissage des λ introduit de nouveaux hyperparamètres qui ne sont pas nécessairement plus facile à déterminer. Nous pensons que les récentes avancées en traitement du langage naturel et les *graph neural networks* introduisent des pistes intéressantes avec notamment les vecteurs d'encodages de position utilisés dans le *Transformer* [Vaswani et al., 2017] et plus généralement avec les mécanismes d'attentions [Veličković et al., 2017] (voir section 3.3.1) permettant d'adapter le principe de convolution sur les données en réseau.

Les algorithmes de plongement de réseau permettent de représenter les sommets dans

un espace euclidien de faible dimension, propice à l'application de méthodes éprouvées de classification et de prédiction. Dans la suite de ce manuscrit, nous cherchons à étendre ces algorithmes afin de pouvoir intégrer les attributs liés au sommets dans l'apprentissage. Par exemple, un réseau de citations lie des articles scientifiques entre eux. L'information textuelle contenue dans ces articles a le potentiel d'améliorer la qualité des représentations apprises.

Chapitre 3

Apprentissage de Représentation de Documents en Réseau

Ce chapitre traite de l'apprentissage de représentation de documents organisés au sein d'un réseau, autrement appelé plongement de réseau de documents (*document network embedding*). Nous abordons en premier lieu l'état de l'art de l'apprentissage de représentation de documents pour ensuite décrire les récentes techniques de plongement de documents en réseau. Nous présentons alors trois contributions réalisées durant cette thèse : (1) GVNR-t, une extension de GVNR qui intègre des représentations des mots des documents dans la factorisation de la matrice des comptages de co-occurrences des sommets, (2) MATAN, un modèle qui représente les documents à partir d'un mécanisme d'attention mutuelle, mettant en lumière les mots expliquant les liens d'un réseau et (3) IDNE, un modèle d'apprentissage de représentation de documents mettant en œuvre un mécanisme d'attention thématique entraîné à partir des liens du réseau. De plus, nous présentons une méthode d'évaluation de ces algorithmes dite « inductive » qui mesure la capacité d'un modèle à représenter des documents encore jamais observés durant l'optimisation de ses paramètres. Enfin, nous présentons et analysons les résultats expérimentaux de nos modèles, quantitativement et qualitativement.

Sommaire

3.1 Introduction

Le chapitre précédent concerne l'apprentissage de représentation dans un réseau permettant, entre autres, de classer ses sommets et de prédire de nouveaux liens. Nombre de réseaux naturels sont en réalité associés à des données riches en information. Le réseau Internet relie des pages Web constituées de code *HTML* et de contenus multimédias et un réseau de citations relie des articles scientifiques comportant des informations textuelles. Les méthodes de plongement de réseau décrites dans le chapitre précédent ne prennent pas en compte l'information associée aux sommets. Dans ce chapitre, nous nous intéressons particulièrement aux réseau de documents textuels.

Analyser ce type de réseau consiste à identifier les facteurs implicites qui régissent non seulement la structure du réseau, mais aussi la distribution des attributs des sommets. Les théories des sciences sociales se sont efforcées de montrer que le contenu associé aux sommets d'un réseau social peut à la fois refléter et affecter la structure du réseau [McPherson et al., 2001, Marsden and Friedkin, 1993, Yang et al., 2010]. En outre, le phénomène d'homophilie stipule que des sommets au propriétés semblables ont nécessairement plus de chance de se connecter et inversement le phénomène de similarité stipule que les sommets connectés entre eux ont tendance à se ressembler de plus en plus. À cela s'ajoute que, ces deux types d'information (le graphe et le texte) étant fondamentalement différents, leur analyse conjointe repose nécessairement sur des techniques issues de deux domaines *a priori* éloignés : la théorie des graphes et le traitement automatique des langues. Pour aborder cette problématique, les techniques de plongement de documents en réseau étendent l'approche suivie par les techniques de plongement de réseau traditionnelles en cherchant à construire des représentations qui conservent dans un espace vectoriel de faible dimension les caractéristiques individuelles des documents au sein du graphe. Un approche simple consiste à apprendre indépendamment, via deux modèles distincts, des représentations des sommets du réseau et des documents puis de les concaténer. Seulement, l'interaction entre ces deux sources d'information [Romero et al., 2013] est alors ignorée. Par exemple, lorsque une publication scientifique cite une autre publication, leurs contenus textuels sont probablement liés. Il existe cependant une multitude de raisons différentes de citer une publication et il est nécessaire de considérer ces deux informations si l'on veut capturer les manifestations de leurs interactions. Certains travaux s'attachent par exemple à déterminer les différentes thématiques d'influences des auteurs ayant contribué à une publication en analysant leur réseau de collaboration [Tang et al., 2009]. D'autres encore cherchent à identifier les caractéristiques des sentiments exprimés dans un réseau social en ligne permettant d'expliquer les apparitions de nouveaux liens d'amitiés [Yuan et al., 2014].

Le plongement de réseau de documents réalise donc un apprentissage de représentation des sommets en tirant profit simultanément des liens du réseau et du texte des documents. L'approche choisie pour ce faire peut cependant varier. En fonction de la façon dont on considère l'information textuelle, on peut envisager différents objectifs, pas nécessairement disjoints, poursuivis par un modèle de plongement de document en réseau :

— l'enrichissement des représentations des sommets grâce à l'information textuelle des documents. Si l'on est capable de contraindre un modèle de plongement de réseau avec le contenu des documents, les représentations construites devraient être encore plus riches. De plus, en opposition à l'approche triviale consistant à concaténer des représentations des deux sources d'information, un modèle unifiant les deux sources devrait être plus efficace à entraîner, en terme de complexité algorithmique. C'est l'idée principale que nous suivons pour l'extension textuelle GVNR-t de notre contribution GVNR présentée dans le chapitre précédent ;

— l'apprentissage d'un modèle du langage. L'apprentissage non supervisée de repré-

sentations de documents est souvent réalisée à travers des tâches annexes telles que la prédiction de mots cachés ou l'identification de phrases qui se suivent dans un corpus [Devlin et al., 2018]. La structure du graphe peut aussi servir de biais d'apprentissage pour guider la construction de représentations du texte. L'une de nos contributions, IDNE, adopte cette approche en définissant une tâche de reconstruction des chemins dans un réseau reposant sur un modèle du langage. Cela permet de générer de manière inductive des représentations pour de nouveaux documents jusqu'alors jamais observés et ne faisant pas partie du réseau considéré ;

— l'interprétation, en langage naturel, des représentations des sommets d'un réseau. Si l'on est capable de représenter dans un même espace les sommets et les mots constituant les documents qui leur sont associés, il est alors possible de « mettre des mots » sur les représentations des sommets. Cette démarche, que nous suivons dans GVNR-t, est approfondie dans le mécanisme d'attention mis en œuvre avec IDNE. Nous étudions qualitativement cet aspect en section 3.6.3 ;

— l'explicabilité, en langage naturel, de la structure du réseau. On cherchera par exemple à comprendre les facteurs discriminants dans les textes d'une paire de documents qui expliquent l'existence d'un lien entre ceux-ci. Cette approche est mise en avant dans notre modèle d'attention mutuelle MATAN et est étudiée en section 3.6.4.

Avant d'aborder l'état de l'art autour de l'apprentissage de représentation de documents, nous définissons formellement le cadre du problème du plongement de documents en réseau dans la section suivante.

3.2 Définition du problème

On considère un réseau de documents décrit par un graphe pondéré, non orienté et attribué $G = (V, E, X)$ composé de n_v sommets, où $X \in \mathbb{R}^{n_d \times n_\omega}$ désigne la matrice des termes des n_d documents construite à partir d'un vocabulaire de n_ω mots. Notons que dans la mesure où les sommets du graphe sont des documents, nous avons toujours $n_v = n_d$ et nous choisirons généralement la seconde notation. De plus, la représentation des documents par sac de mots est un choix que nous faisons par soucis de formaliser au mieux les approches que nous décrirons. Il existe cependant des méthodes qui considèrent l'ordre des mots associés aux sommets. La matrice d'adjacence du graphe est notée $A \in \mathbb{R}^{n_d \times n_d}$ telle que $a_{ij} \geq 0$ si et seulement si il existe un lien entre les documents d_i et d_j. On cherche à représenter chaque document d_i du réseau par un vecteur u_i de dimension ρ_d tel que $\rho_d \ll n_d$ et $\rho_d \ll n_\omega$. On note $U \in \mathbb{R}^{n_d \times \rho_d}$ l'agencement en une matrice de ces représentations. On cherche à construire U de sorte à respecter les mêmes critères définis pour le plongement de réseau en section 2.2 en tirant profit de l'information textuelle supplémentaire décrite dans la matrice X.

Après un état de l'art, dans la section suivante, sur l'apprentissage de représentations de documents et de documents en réseau, nous présentons en section 3.4 trois modèles de plongement de documents en réseau réalisés durant cette thèse : GVNR-t, IDNE et MATAN. Nous détaillons ensuite dans les sections 3.5 et 3.6 les protocoles d'évaluation mis en œuvre et les résultats expérimentaux obtenus pour ces modèles.

3.3 État de l'art

Dans cet état de l'art, nous présentons en premier lieu les méthodes d'apprentissage de représentation du langage naturel, notamment les méthodes statistiques classiques reposant sur la description en sac de mots de documents puis les méthodes récentes

basées sur des réseaux de neurones profonds. Ensuite, nous abordons particulièrement les méthodes de plongement de réseau de documents, dans lesquelles l'interaction entre les deux sources d'information, le graphe et le texte, sont considérées simultanément au sein d'un unique modèle d'apprentissage.

3.3.1 Apprentissage de représentation de documents

Le traitement automatique du langage (TAL) est un ensemble de techniques algorithmiques visant à analyser de manière automatique le langage humain. Les travaux dans ce domaine couvrent la détection d'entité nommée [Nadeau and Sekine, 2007], la recherche d'information dans des corpus [Manning et al., 2008] ou encore la traduction automatique [Koehn, 2009] etc. Pour chacune de ces applications, la représentation du texte dans un espace mathématique adapté est un facteur déterminant. Comme vu dans le chapitre d'introduction (voir section 1.2.2), le texte peut être vu comme une séquence de symboles, tels que les caractères ou les mots. Il est par la suite possible de transformer ces séquences de multiples façons afin d'appréhender une tâche particulière de TAL.

Dans cet état de l'art, nous nous intéressons à la représentations de documents. Ainsi, nous abordons particulièrement les techniques de traitement du texte qui permettent de construire des représentations pour des séquences textuelles, qui peuvent être des phrases, des paragraphes ou des documents entiers. De plus, nous nous intéressons aux techniques de TAL pour la recherche d'information, dans lesquels la capacité à comparer des documents entre eux constitue l'objectif central. Nous n'aborderons alors que succinctement les tâches dites *seq2seq* (séquence à séquence) utilisées pour la traduction automatique et pour les modèles conversationnels, où une importance particulière est accordée à l'ordre des mots ainsi qu'à la génération de nouvelles séquences. Cependant, les récentes avancées dans ces domaines reposent sur des architectures prometteuses, tels que les mécanismes d'attention et les encodeur-décodeurs, qui ont inspiré certains de nos travaux.

Nous présentons en premier lieu dans cette section les méthodes classiques de représentation de documents reposant sur la description en sac de mots. Ensuite, nous abordons les avancées majeures insufflées par l'essor des techniques d'apprentissage profond. Enfin, nous détaillons les mécanismes d'attention, introduits récemment comme une amélioration des réseaux de neurones récurrents, puis utilisés comme brique de base de modèles *seq2seq* capables d'aborder un grand nombre de tâches en TAL.

Modèles statistiques sur les sacs de mots

Le traitement le plus commun pour représenter un corpus de document est la description par sac de mots. On représente le corpus par une matrice document-terme $X \in \mathbb{N}^{+n_d \times n_\omega}$ où chaque valeur x_{ij} correspond aux nombres d'occurrences du mot ω_j dans le document d_i, autrement appelé fréquence absolue de ω_j dans d_i. Dans ce manuscrit, nous faisons référence à la matrice des fréquences des termes pour désigner les fréquences relatives (notée TF pour *Term Frequency*) lorsque l'on normalise en ligne la matrice document-terme en divisant chaque comptage d'apparition d'un mot par la somme des comptages du document (correspondant à la norme 1). Une extension de cette représentation, le TF-IDF (*term frequency-inverse document frequency*), intègre des statistiques globales sur le corpus dans ces représentations. L'intérêt est de pondérer les mots par leurs fréquences d'apparition dans le corpus, donnant ainsi plus de poids aux mots faisant de rares apparitions et réduisant le poids des mots très courants. La composante *inverse document frequency* est calculée comme le logarithme de l'inverse des fréquences d'apparition d'un mot à travers les documents :

$$\mathrm{IDF} = \log \frac{n_d}{\mathrm{DF}}, \tag{3.1}$$

avec DF étant le nombre de documents dans lesquels le mot apparaît. La représentation des documents se calcule alors comme suit : $x_{ij} = \text{TF} \times \text{IDF}$.

Les représentations TF et TF-IDF décrivent les documents en termes d'occurrence des mots. Néanmoins, une autre source d'information importante réside dans les co-occurrences des mots. Le fait que plusieurs mots co-occurrent régulièrement constitue en soi un motif qui peut être intéressant de capturer. Par exemple, dans le tableau 3.1, la similarité entre les documents 1 et 3 n'est pas directement perceptible. Les termes de ces deux documents co-occurrent néanmoins dans le document 2, ce qui permet de déduire qu'ils sont liés. Une approche simple pour saisir cet aspect consiste à calculer $X^\top X$ donnant les comptages de co-occurrences entre chaque paire de mots dans le corpus. Néanmoins, cette matrice est généralement très dense car un nombre élevé de paires de mots occurrent dans les documents ce qui rend son utilisation coûteuse en calculs. Pour capturer ce genre de motif à moindre coût, on peut distinguer deux grandes approches : les techniques de réduction de dimension et les modèles génératifs probabilistes.

TABLE 3.1 – Hypothétique matrice document-terme. Les documents 1 et 2 sont proches car ils partagent les termes 1 et 2. Les documents 2 et 3 le sont aussi car ils partagent les termes 3 et 4. Mais, selon cette représentation, les documents 1 et 3 sont dissimilaires (en terme de la similarité cosinus par exemple). Seulement, par observation des co-occurrences, on peut penser que ces documents sont relativement proches puisque leurs termes respectifs apparaissent ensemble dans un document tiers (le document 2).

	mot 1	mot 2	mot 3	mot 4
document 1	1	1	0	0
document 2	1	1	1	1
document 3	0	0	1	1

L'analyse sémantique latente *LSA* (*Latent Semantic Analysis*) [Deerwester et al., 1990, Landauer et al., 1998] fait partie des techniques de réduction de dimension. Cette méthode consiste à plonger les représentations des documents dans un espace de faible dimension par décomposition en valeurs singulières de la matrice TF-IDF : $X = U\Sigma H$ où Σ a pour coefficients diagonaux les valeurs singulières par ordre décroissant de la matrice X et où U et H sont les vecteurs singuliers formant des bases orthonormées de $\mathbb{R}^{n_d}$ et $\mathbb{R}^{n_\omega}$ respectivement. L'espace de faible dimension est obtenu en supprimant les valeurs singulières les plus faibles de Σ et les vecteurs singuliers associés dans U et H. On réalise ainsi une compression de l'information depuis un espace de haute dimension (égale au nombre de mots n_ω dans le vocabulaire) à une dimension réduite $\rho_d \ll n_\omega$. La réduction est faite de manière linéairement optimale en gardant les valeurs propres les plus élevées, garantissant ainsi de reconstruire les axes orthogonaux de plus grandes variances dans X. De cette façon, les mots qui présentent les mêmes motifs d'occurrences à travers les documents du corpus ont tendance à être projetés sur les mêmes dimensions dans l'espace réduit.

Les méthodes de réduction de dimension présentent plusieurs avantages. Tout d'abord elles réduisent fortement la dimension des vecteurs documents, ce qui peut permettre de réduire les temps de calculs sur certains algorithmes. De plus, elles améliorent certains problèmes inhérents aux représentations par sac de mots dans lesquelles une forte majorité de valeurs sont nulles (représentations creuses). Par exemple, selon ces représentations, deux mots pourtant synonymes ont une similarité nulle puisqu'ils sont chacun associés à une dimension unique. Dans un espace réduit, les représentations de ces mots se retrouvent alors très proches. Enfin, les méthodes de réduction de dimensions permettent plus géné-

ralement de contrer le fléau de la dimension [Bellman, 1966] qui désigne le fait que des données en très haute dimension ont tendance à se retrouver isolées car le nombre d'observations nécessaire pour couvrir tout l'espace considéré devient en pratique impossible à obtenir [Giraud, 2014].

L'allocation de Dirichlet latente ou LDA (*Latent Dirichlet Allocation*) [Blei et al., 2003] est un modèle génératif probabiliste dont le modèle graphique est présenté en figure 3.1. L'objectif est d'expliquer les occurrences des mots en les affectant à des thématiques (*topics*), en supposant que les occurrences au sein d'un document sont conditionnées par un mélange d'un faible nombre de thématiques. Le modèle effectue l'hypothèse que les n_d documents $D = [d_1, ..., d_{n_d}]$ sont générés à partir de distributions $\theta_1, ..., \theta_{n_d}$ de n_t thématiques. Ces thématiques sont elles-mêmes définies comme des probabilités, de distributions $\phi_1, ..., \phi_{n_t}$, de générer les n_ω mots du vocabulaire. Ces distributions sont tirées suivant des lois de Dirichlet de paramètres respectifs $\alpha = (\alpha_1, ..., \alpha_{n_d})$ et $\beta = (\beta_1, ..., \beta_{n_t})$. Ainsi, *LDA* construit un modèle génératif reposant sur le processus suivant :

- générer $\theta_i \sim \mathrm{Dir}(\alpha)$;
- générer $\phi_k \sim \mathrm{Dir}(\beta)$;
- pour chaque mot ω_j à générer dans chaque document d_i :
  - choisir un thème k suivant les probabilités θ_i ;
  - choisir un mot ω_j suivant les probabilités ϕ_k.

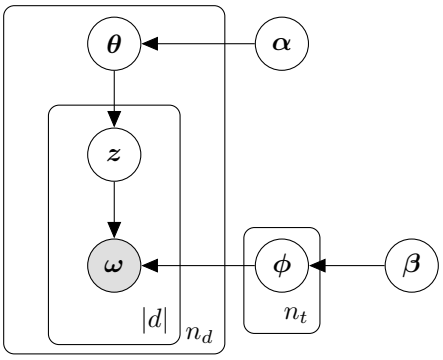


FIGURE 3.1 – Modèle graphique de LDA où $|d|$ représente le nombre de mots composant un document et z représente l'assignation d'une thématique à un mot ω du document.

LDA cherche à identifier les distributions θ et ϕ qui ont généré l'ensemble du corpus. Pour déterminer ces distributions, des méthodes d'inférence statistique sont mises en œuvre telles que l'échantillonnage de Gibbs [Porteous et al., 2008] ou les algorithmes de type *variational Bayes* [Hoffman et al., 2010].

Il existe une multitude de méthodes de modélisation de thématiques basées sur les techniques de réduction de dimension et sur des modèles génératifs probabilistes. On peut noter les approches de factorisation de matrices non négative (NMF) [Lee and Seung, 2001] présentant l'avantage de produire des dimensions interprétables ce qui est difficilement le cas avec LSA. pLSA (*probabilistic LSA*) est une approche probabiliste de LSA, à l'origine de LDA, qui effectue un mélange de décompositions issues de l'analyse des classes latentes. NMF et pLSA peuvent être unifiées [Gaussier and Goutte, 2005] dans le cadre des *multinomial PCA* [Buntine, 2002]. Si ces méthodes présentent des avantages divers tels que l'interprétabilité et la rapidité d'inférence, ces dernières années ont vu l'émergence de modèles d'apprentissage profonds qui ont fortement impactés le domaine du traitement du langage naturel.

Apprentissage profond pour le TAL

Les modèles d'apprentissage profond (*deep learning*) surpassent les approches présentées précédemment dans de nombreuses tâches de TAL [Collobert et al., 2011]. Ces modèles se basent sur des représentations denses des mots, telles que GloVe et Word2vec présentées en section 2.3.3, pour apprendre des représentations du texte. Ces représentations vectorielles distribuées des mots [Bengio et al., 2003] permettent notamment d'utiliser des méthodes éprouvées originellement utilisées dans d'autres disciplines telles que les réseaux neuronaux convolutifs (CNN pour *convolutional neural network*) [LeCun et al., 1998] pour la reconnaissance d'images et les réseaux de neurones récurrents (RNN pour *recurrent neural network*) avec notamment les LSTMs (*long short-term memory*) [Hochreiter and Schmidhuber, 1997] pour la reconnaissance vocale [Graves et al., 2013]. Nous retraçons ici les avancées majeures de l'apprentissage profond pour le TAL avant d'étudier plus en détail les mécanismes d'attention.

La qualité des représentations des mots est généralement jugée par leur capacité à encoder leurs propriétés syntaxiques et sémantiques. Les modèles traditionnels, tels que Word2vec et GloVe, capturent le contexte « moyen » des mots dans les phrases où ils apparaissent pour construire une représentation statique de ceux-ci. Cependant, un mot peut avoir des sens différents en fonction de son contexte. Par exemple, dans les phrases « Il y a beaucoup de *bouchons* ce matin. », « Le *bouchon* a sauté jusqu'au plafond. » et « J'ai mangé dans un *bouchon* hier soir. », le mot *bouchon* décrit des concepts très différents. Il peut être nécessaire de construire des représentations des mots qui soient contextuelles aux autres mots avec lesquels ils apparaissent, pour chaque instance de phrase possible. Ainsi, pour N apparitions d'un mot ω_i dans différents morceaux de texte $d_1, ..., d_N$, on le représentera par N représentations contextuelles $u(\omega_i|d_1), ..., u(\omega_i|d_N)$. C'est notamment l'approche choisie dans ELMo (*embedding from language model*) [Peters et al., 2018]. Étant donnée une séquence de ℓ mots $(\omega_1, ..., \omega_\ell)$, ELMo utilise deux modèles du langage pour calculer la probabilité $p(\omega_k|\omega_1, ..., \omega_{k-1}, \omega_{k+1}, ..., \omega_\ell)$ de chaque mot sachant son contexte. Ces deux modèles rendent compte d'une lecture de gauche à droite et d'une lecture de droite à gauche de la séquence, produisant ainsi deux contextes. On a alors :

$$p_{\rightarrow}(\omega_1, \omega_2, \ldots, \omega_\ell) = \prod_{k=1}^{\ell} p(\omega_k|\omega_1, \omega_2, \ldots, \omega_{k-1}) \tag{3.2}$$

$$p_{\leftarrow}(\omega_1, \omega_2, \ldots, \omega_\ell) = \prod_{k=1}^{\ell} p(\omega_k|\omega_{k+1}, \omega_{k+2}, \ldots, \omega_\ell) \tag{3.3}$$

$$\text{et finalement} \quad p(\omega_1, ..., \omega_\ell) = p_{\rightarrow}(\omega_1, \omega_2, \ldots, \omega_\ell)\, p_{\leftarrow}(\omega_1, \omega_2, \ldots, \omega_\ell) \tag{3.4}$$

Ces probabilités sont modélisées via deux réseaux neuronaux récurrents, architectures que nous abordons ci-après. Les paramètres de ELMo sont optimisés une première fois, de manière non supervisées en maximisant la vraisemblance de $p(\omega_1, ..., \omega_\ell)$. De plus, le modèle est par la suite ré-entraîné sur des tâches spécifiques de TAL, pour lesquelles la concaténation des représentations de la dernière couche du réseau neuronal est utilisée comme représentation de la phrase complète. ELMo est, à notre connaissance, l'une des première approches à suggérer un entraînement en deux étapes pour pour le traitement du langage naturel. Cette méthodologie est adoptée par plusieurs autres modèles de référence tels que BERT [Devlin et al., 2018] et GPT-2 [Radford et al., 2019].

Au-delà des représentations des mots, les modèles d'apprentissage profond réalisent d'excellentes performances sur des tâches définies sur des textes entiers. Les réseaux convolutifs (CNN), par leur capacité à apprendre des compositions complexes des caractéristiques des données d'entrées, sont appliqués avec succès sur des séquences de plongement de mots [Collobert and Weston, 2008]. Les CNN ont la capacité de construire des re-

présentations de N-Grams de mots qui seront à leur tour composées pour construire des représentations de morceaux de phrases etc. jusqu'au niveau d'un document entier [Collobert et al., 2011]. De cette façon, des représentations du texte peuvent être construites sur une variété de tâches supervisées de TAL telles que la classification de phrases [Kim, 2014], la génération de résumés [Denil et al., 2014] ou encore la recherche d'information [Shen et al., 2014].

Les réseaux neuronaux récurrents, par leur capacité à traiter des séquences de données, ont constitués l'autre approche majeure d'apprentissage profond pour le TAL. Ces modèles sont particulièrement utilisés pour les tâches dites *seq2seq* où l'on cherche à générer une séquence de texte en sortie étant donnée une séquence d'entrée. Les réseaux récurrents peuvent identifier des dépendances lointaines au sein de séquences de données. Plusieurs modélisations de neurones récurrents existent telles que le GRU (*gated recurrent unit*) [Cho et al., 2014] et le LSTM [Hochreiter and Schmidhuber, 1997]. Pour traiter du texte, cette dernière parcourt itérativement une séquence de ℓ plongements de mots $(x_1, ...x_\ell)$ où $x_i \in \mathbb{R}^{\rho_w}$, générant pour chacun d'eux une représentation h_i « cachée » qui représente l'ensemble de la séquence des i premiers mots $(x_1, ...x_i)$. Cette représentation est calculée à partir d'une représentation c_i « contextuelle » pour chaque mot. Cette dernière sert de pont à travers la séquence d'entrée puisqu'elle dépend directement de c_{i-1} et de h_{i-1} permettant ainsi au modèle de pondérer l'importance accordée à l'historique (les mots $x_1, ...x_{i-1}$), à l'entrée (le mot x_i) et à la sortie (ici h_i). Ce type de réseau de neurones réussit à contrôler la perte de propagation du gradient, inexorable lorsque les séquences de mots sont longues [Bengio et al., 1994].

L'avantage de ces méthodes sur les CNN est leur capacité à modéliser des séquences de longueurs variables. Parmi les nombreuses applications de ces méthodes en TAL, on peut noter la traduction automatique [Cho et al., 2014] et les chatbots [Sutskever et al., 2014]. Récemment, les mécanismes d'attention ont été introduits comme des composantes améliorant la capacité de mémorisation des réseaux récurrents. Ces mécanismes permettent au modèle de sélectionner plus finement quelle information utiliser comme contexte dans l'historique des éléments de la séquence d'entrée. Par la suite, les mécanismes d'attention multi-têtes, tels que le *Transformer*, ont été proposés comme des solutions s'affranchissant de composantes de type CNN et RNN. Dans la section suivante, nous détaillons ces méthodes que nous avons adaptées pour le plongement de documents en réseau.

Les mécanismes d'attention

Pour cerner l'intérêt des mécanismes d'attention, prenons l'exemple de la traduction automatique, particulièrement telle qu'elle est abordée dans un modèle de type encodeur-décodeur neuronal [Bahdanau et al., 2014]. La tâche consiste à produire en sortie une séquence de plongements de mots $(y_1, ..., y_{\ell_y})$ étant donnée une séquence de plongements de mots d'entrée $(x_1, ..., x_{\ell_x})$. Les mots en sortie correspondent par exemple à de l'anglais alors que les mots en entrée correspondent à du français. Le processus est auto-régressif, ce qui signifie que la séquence de sortie est générée mot par mot. À chaque étape i, l'encodeur utilise la séquence d'entrée $(x_1, ..., x_{\ell_x})$ et le morceau de séquence de sortie précédemment généré $(y_1, ..., y_{i-1})$ afin de prédire le mot suivant y_i. Notons que l'on définit le premier vecteur de sortie comme une constante $y_1 = y_{\text{start}}$ ce qui permet de définir la récurrence pour la première itération. Le problème auquel cette approche fait face est la gestion de la diversité d'information présente en entrée pour prédire chaque sortie. En traduction automatique, prédire un mot ne dépend souvent que d'une infime proportion des vecteurs présentés à l'encodeur-décodeur $(x_1, ..., x_{\ell_x}, y_1, ..., y_{i-1})$. Lorsque cette séquence est longue, il devient difficile pour le modèle d'en faire le tri. C'est précisément pour faciliter ce tri que les mécanismes d'attention sont utilisés. Ils vont permettre au modèle de ne sélectionner

qu'un sous-ensemble précis de la séquence d'entrée afin de prédire le prochain mot.

Le premier mécanisme d'attention pour les réseaux profonds est présenté dans [Xu et al., 2015] pour la génération automatique de légendes pour images. Le modèle est décrit comme un encodeur-décodeur. L'encodeur extrait un ensemble de N représentations vectorielles $(a_1, ..., a_N)$ d'une image via un CNN, appelées vecteurs d'annotation. Ces vecteurs sont générés de sorte à capturer les différentes composantes de l'image. Le décodeur, un LSTM, produit de manière auto-régressive une séquence de mots en sortie. Pour ce faire, il prend en entrée les mots précédemment générés $(y_1, ..., y_{i-1})$. L'attention est intégrée pour conditionner cette génération au vecteurs d'annotation. Le vecteur contexte c_i du LSTM est calculé non plus comme une fonction de c_{i-1} et de h_{i-1} mais comme une moyenne pondérée des vecteurs d'annotation issus de l'image $c_i = \sum_{k=1}^{N} \alpha_k a_k$. Les poids α_k sont calculés par produit scalaire entre le vecteur caché h_{i-1} de la précédente étape (capturant l'historique des mots générés $(y_1, ..., y_{i-1})$) avec chacun des vecteurs d'annotation a_k. En somme, chaque mot est généré en confrontant les caractéristiques de l'image et l'historique des mots précédemment générés en légende. Ceci permet au modèle de se concentrer alternativement sur les différents éléments de l'image afin de construire une description en langage naturelle de celle-ci. De plus, en étudiant les poids α_k, on peut facilement identifier la région de l'image ayant motivé le choix du mot généré par le modèle.

Les mécanismes d'attention ont été largement utilisés pour d'autres tâches. En traduction automatique, ils ont permis d'améliorer la qualité des traductions et les temps d'entraînement des modèles [Luong et al., 2015]. Ils ont montré d'intéressantes capacités pour apprendre des représentations hiérarchiques de documents pour la classification [Yang et al., 2016] ainsi que pour construire des représentations interprétables de phrases [Lin et al., 2017].

Le *Transformer* [Vaswani et al., 2017] est le premier modèle de traduction automatique reposant uniquement sur un mécanisme d'attention, sans RNN ni CNN. L'aspect séquentiel de l'entraînement d'un RNN (prédire chaque mot l'un après l'autre) constitue le véritable point faible de ces architectures car il rend difficile leur parallélisation. Le *Transformer* reprend l'architecture encodeur-décodeur couramment utilisée en traduction automatique et introduit un mécanisme d'attention, le *scaled dot-product attention* (SDPA), parallélisable et reposant principalement sur des opérations matricielles. Dans sa version la plus simple, SDPA transforme une séquence de vecteurs d'entrée $(x_1, ..., x_\ell)$ en une séquence de vecteurs de sortie $(y_1, ..., y_\ell)$ dont chaque représentation y_i représente x_i conditionnellement à l'ensemble des vecteurs $(x_1, ..., x_\ell)$. Pour ce faire, en notant X et Y les agencements matriciels des séquences d'entrée et de sortie, SDPA construit trois représentations distinctes des x_i par projections linéaires : les requêtes $Q = XW_q$, les clefs $K = XW_k$ et les valeurs $V = XW_v$. En notant ρ_w la dimension de toutes les représentations X, Y, Q, K et V, le mécanisme d'attention s'écrit :

$$Y = \text{softmax}(\frac{QK^\top}{\sqrt{\rho_w}})V, \tag{3.5}$$

où $QK^\top$ est une matrice dont chaque valeur contient le produit scalaire entre une requête q_i et une clef k_j. Diviser par $\sqrt{\rho_w}$ permet de réduire le problème de fuite du gradient qui peut se produire lorsque le softmax prend des valeurs extrêmes, en limitant la magnitude du produit scalaire dont l'étendue des valeurs augmente naturellement avec la dimension. Le softmax, opéré sur chaque ligne de la matrice, permet d'obtenir une pondération composée de valeurs positives sommant à un, similaire à des probabilités associées aux n mots d'entrée, pour chaque requête q_i. Les poids d'attention s'écrivent $\alpha = \text{softmax}(\frac{QK^\top}{\sqrt{\rho_w}})$ et l'on a $\sum_{j=1}^{\ell} \alpha_{ij} = 1, \forall i \in [1, \ell]$. La multiplication matricielle entre ces poids d'attention et

V consiste à réaliser des moyennes pondérées des v_j telles que $y_i = \sum_{j=1}^{\ell} \alpha_{ij} v_j$. La figure 3.2 détaille ce calcul sur un exemple de phrase hypothétique de trois mots.

Phrase	self	attention	mechanism
Plongements	$x_1 = (0.7, 1.1)$	$x_2 = (-0.4, 0.3)$	$x_3 = (0.3, -0.5)$
Requêtes	$q_1 = x_1 W^q$	$q_2 = x_2 W^q$	$q_3 = x_3 W^q$
Clefs	$k_1 = x_1 W^k$	$k_2 = x_2 W^k$	$k_3 = x_3 W^k$
Valeurs	$v_1 = x_1 W^v$	$v_2 = x_2 W^v$	$v_3 = x_3 W^v$
Scores	$q_1 \cdot k_1 = 29.61$	$q_1 \cdot k_2 = 31.02$	$q_1 \cdot k_3 = 26.79$
Divisions ($\sqrt{\rho_w}$)	20.94	21.93	18.94
Poids (softmax)	$\alpha_{12} = 0.26$	$\alpha_{12} = 0.70$	$\alpha_{13} = 0.04$
Pondérations	$0.26 \times v_1$	$0.70 \times v_2$	$0.04 \times v_3$
Sommes	$y_1 = \sum_{i=1}^{3} \alpha_{1i} v_i$	$y_2 = \sum_{i=1}^{3} \alpha_{2i} v_i$	$y_3 = \sum_{i=1}^{3} \alpha_{3i} v_i$

FIGURE 3.2 – Illustration du mécanisme d'attention introduit dans le *Transformer* [Vaswani et al., 2017], ici pour le mot « Self ». Le produit scalaire entre requêtes et clefs génère des poids d'attention qui permettent de pondérer les valeurs, de sorte à représenter ce mot contextuellement aux autres mots de la phrase. Ici, on voit que la représentation contextuelle y_1 de « self » est fortement liée au mot « attention » ($\alpha_{12} = 0.70$).

L'intuition derrière cette formule est que si α_{ij} est proche de 1, le vecteur y_i sera fortement influencé par la valeur v_j (et donc à une projection linéaire près par x_j). Inversement, si α_{ij} est proche de 0, c'est que y_i n'est pas lié à x_j. Les seuls paramètres de ce mécanisme sont W_q, W_k et W_v de dimensions $\rho_w \times \rho_w$ et leur rôle est de projeter la séquence d'entrée dans trois espaces Q, K et V de sorte à capturer les dépendances entre les mots. Les concepts de « clef », « valeur » et « requête » proviennent de systèmes de stockage des données. Par exemple, lorsque l'on saisit une requête pour rechercher un document dans une base de données, un moteur de recherche associe cette requête à un ensemble de clefs enregistrés dans la base (titre du document, contenu, date, etc.), puis il présente les meilleures correspondances de documents (valeurs).

Dans le *Transformer*, le mécanisme d'attention est utilisé de deux façons différentes : (1) dans des mécanismes d'auto-attention (*self-attention*) de sorte à construire des représentations contextuelles X^c des mots de la séquence d'entrée et Y^c des mots de la séquence de sortie puis (2) dans le décodeur, où les requêtes sont construites à partir des sorties contextuelles Y^c et où les clefs et valeurs sont construites à partir des entrées contextuelles X^c. Le décodeur fabrique ainsi des représentations de la séquence de sortie capturant ses dépendances avec la séquence d'entrée, facilitant la prédiction du prochain mot. Enfin, le modèle dans son intégralité est en réalité composé de multiples mécanismes d'attention opérés en parallèles, concaténés puis réutilisés sur de multiples couches. L'architecture entière fait intervenir un grand nombre de perceptrons à plusieurs couches augmentant significativement le nombres de paramètres et rendant non trivial l'entraînement du modèle.

Le *Transformer*, plus précisément son encodeur avec ses mécanismes d'auto-attention, constitue la brique de base de BERT [Devlin et al., 2018], un modèle de représentation du langage naturel. L'idée principale est de pré-entraîner ce mécanisme d'attention sur des tâches très générales ne requérant pas d'annotation particulière avant d'affiner les paramètres du modèle sur des tâches spécifiques, telles que la détection d'entité nommée et l'analyse de sentiment. Pour le pré-entraînement, deux tâches sont proposées : la prédiction de mots masqués et la prédiction de phrases successives. Dans la première tâche, on présente au modèle des phrases dont 15% des mots ont été remplacés par un vecteur spécial de masque et on optimise les paramètres du modèle de sorte à ce que les vecteurs de sortie associés aux masques soient les plus proches possible des vecteurs des mots cachés. Pour la seconde tâche, on tire aléatoirement 50% de paires de phrases qui se suivent dans un corpus et 50% de paires de phrases qui ne se suivent pas et on entraîne un classifieur à prédire si ces phrases se suivent à partir d'une représentation de sortie produite à partir d'un vecteur spécial de classification ajouté aux phrases d'entrée.

Les modèles d'apprentissage dédiés au TAL sont conçus pour capturer les caractéristiques du langage naturel. Dans les cas non supervisés, leur entraînement se fait à partir d'hypothèses sur la structure du texte traduites en tâches à résoudre telles que la conservation optimale de la variance des occurrences des mots dans les documents pour LSA ou la prédiction de mots masqués pour BERT. De nombreuses tâches non supervisées sont proposées dans la littératures et ce choix impacte fortement la qualité des représentations apprises comme le montrent les extensions de BERT, ALBERT [Lan et al., 2019] et RoBERTa [Liu et al., 2019]. Dans la section suivante, nous verrons que lorsque le corpus de texte est structuré en réseau, on peut se servir de la structure du réseau pour construire une nouvelle tâche non supervisée.

3.3.2 Plongement de réseau de documents

Les réseaux de documents peuvent être vus comme des graphes attribués $G = (V, E, X)$ dont les attributs X sont des descriptions des textes. Nous avons présenté en section 2.3.3 l'état de l'art du plongement de réseau qui vise à construire des représentations des sommets telles que les proximités d'ordre faible observées dans leurs liens soient conservées à travers les distances dans l'espace de représentation. De plus, nous avons vu dans la section précédente des éléments de l'état de l'art en traitement automatique du langage. Pour représenter des documents en réseau, une méthode triviale consiste à construire de manière indépendante des représentations avec deux modèles, issus respectivement de ces deux domaines de l'apprentissage automatique. On construit par exemple des représentations u_i des sommets du réseau $G = (V, E)$ en utilisant DeepWalk, puis on construit des représentations h_i des documents X en utilisant LSA. Les représentations finales peuvent être la concaténation des deux vecteurs $u_i \oplus h_i$.

Néanmoins, apprendre des représentations des documents en tenant en compte simultanément du graphe et du texte peut s'avérer bénéfique. En effet, ces deux sources d'informations, bien qu'étant souvent corrélées, peuvent aussi se compléter. Si les thématiques des documents peuvent refléter les communautés dans le réseau, elles peuvent aussi fournir un niveau de détail plus fin que celui du réseau, permettant ainsi d'améliorer son analyse. De plus, le graphe et le texte peuvent avoir des interactions sophistiquées. Il existe par exemple de multiples raisons de citer un article scientifique et ces raisons peuvent être explicitées en analysant les relations entre les thématiques des publications et la répartition des liens dans le graphe de citations. Pour ces raisons, plusieurs approches d'apprentissage de représentations de documents en réseau ont été proposées.

TADW (*Text-Associated DeepWalk*) [Yang et al., 2015] constitue le modèle de référence en plongement de documents en réseau. En montrant, similairement à NetMF (c.f. section

2.3.3), que DeepWalk est équivalent à une factorisation de matrice, ce modèle consiste à incorporer le contenu textuel des documents dans l'apprentissage de représentation des sommets. DeepWalk réalise implicitement la factorisation d'une matrice M dont les entrées m_{ij} sont les logarithmes de la probabilité moyenne qu'une marche aléatoire partant du sommet v_i s'arrête sur le sommet v_j. Les auteurs proposent alors une extension de cette factorisation de M en incorporant des représentations des documents X obtenues par analyse sémantique latente (LSA, c.f. section 3.3.1). Formellement, considérant que DeepWalk est équivalent à la décomposition de rang faible de la matrice M en deux matrices U et H, dont les valeurs peuvent être déterminées par un problème de minimisation par moindres carrés avec pénalité sur la norme de Frobenius des paramètres de facteur $\lambda \in \mathbb{R}$, on peut réécrire le problème d'optimisation comme suit :

$$\min_{U,H} \left\| M - U^\top H \right\|_F^2 + \frac{\lambda}{2} \left(\|U\|_F^2 + \|H\|_F^2 \right). \tag{3.6}$$

Pour intégrer les composantes textuelles des documents du réseau dans cette optimisation, TADW emploie une méthode de tri-factorisation, appelée complétion inductive de matrice [Natarajan and Dhillon, 2014]. La factorisation s'écrit alors :

$$\min_{U,H} \left\| M - U^\top HX \right\|_F^2 + \frac{\lambda}{2} \left(\|U\|_F^2 + \|H\|_F^2 \right). \tag{3.7}$$

La minimisation est réalisée par descente de gradient en optimisant alternativement U et H. Les représentations finales des documents sont obtenues par concaténation des matrices U et HX où U peut être considérée comme une composante « réseau » et HX comme une composante « textuelle ». Dans les faits, TADW ne factorise pas la même matrice que DeepWalk car calculer de manière exacte M nécessite de calculer les τ premières puissances de la matrice d'adjacence, où τ est la taille de la fenêtre choisie. TADW réalise un compromis en choisissant $\tau = 2$ très inférieure aux valeurs généralement choisies pour DeepWalk. De plus, au lieu de factoriser le logarithme des probabilités des co-occurrences des sommets, TADW factorise directement la matrice $M = \frac{A+A^2}{2}$. Ceci permet de garder un grand nombre de valeurs nulles dans la factorisation de M, valeurs qui ne sont pas prises en compte dans la minimisation des moindres carrés, réduisant fortement la complexité en temps de l'algorithme.

Graph2gauss (*Deep Gaussian Embedding of Graphs*) [Bojchevski and Günnemann, 2018] est une approche qui plonge un réseau via un encodeur prenant en entrée les attributs des sommets. Cette approche considère les représentations des sommets comme des vecteurs aléatoires gaussiens et vise à modéliser leurs densité de probabilité. Chaque sommet est alors non seulement représenté par un point dans un espace vectoriel mais aussi par une variance, donnant une notion d'incertitude sur sa représentation. Enfin, les représentations n'étant dépendantes que des attributs des sommets (par exemple du texte), Graph2gauss est capable d'induire de nouvelles représentations pour des sommets jamais observés pendant l'entraînement. Le modèle est optimisé selon une procédure en deux étapes : (1) les attributs des sommets x_i sont transformés par un encodeur, un perceptron multi-couches, pour construire des représentations $\mu_\theta(x_i)$ et $\Sigma_\theta(x_i)$ puis (2) ces représentations sont optimisées via une fonction de perte de type *energy-based* [LeCun et al., 2006] afin d'approcher le classement des paires de sommets selon leurs distances Δ dans le graphe :

$$\Delta(h_i, h_{k_1}) < \Delta(h_i, h_{k_2}) < \cdots < \Delta(h_i, h_{k_K}) \quad \forall k_1 \in N_{i1}, \forall k_2 \in N_{i2}, \ldots, \forall k_K \in N_{iK}, \tag{3.8}$$

où $h_i = \mathcal{N}(\mu_i, \Sigma_i), \mu_i \in \mathbb{R}^\rho, \Sigma_i \in \mathbb{R}^{\rho \times \rho}$ sont les paramètres de la gaussienne décrivant le sommet v_i avec $\rho \ll n_v$ et où N_{ik} est l'ensemble des sommets de distance géodésique k

au sommet v_i. On fait ainsi en sorte que les voisins directs de v_i aient des représentations plus proches que les voisins de second ordre, eux même plus proches que les voisins de troisième ordre etc. Pour un réseau de documents, les attributs des sommets X sont les représentations document-terme de ceux-ci. Les représentations utilisées pour résoudre une tâche de classification ou de prédiction sont généralement les moyennes des gaussiennes μ_i, les variances n'apportant pas de bénéfice notable.

RLE (*Regularized Linear Embedding*) [Gourru et al., 2020] est un modèle qui projette une collection de documents en réseau dans un espace de représentation des mots préalablement appris. Étant données une matrice de transition P du réseau de documents, une matrice TF du contenu textuel des documents X ainsi qu'une matrice W de plongements des mots du vocabulaire pré-calculée (par exemple avec GloVe sur le corpus), RLE cherche à construire des représentations U des documents, dans le même espace que celui des plongements des mots, qui tiennent compte des liens du réseau. Chaque représentation s'écrit $u_i = \alpha_i W$ où α_i est un vecteur de pondération sur les mots du vocabulaire. Pour déterminer ces pondérations, RLE construit en premier lieu une matrice de similarité des sommets comme suit : $S = \frac{P+P^2}{2}$. Ensuite, une matrice de lissage B est définie de la façon suivante :

$$b_i = \frac{1}{\sum_j s_{ij}} \sum_j s_{ij} x_j. \tag{3.9}$$

Ainsi, chaque ligne de B est une moyenne pondérée de chaque ligne de la matrice des fréquences des termes X, dont les poids sont les similarités contenues dans $s_{i.}$ issues du graphe. Finalement, les représentations des documents sont calculées grâce aux poids $\alpha = (1-\lambda)X + \lambda B$ où $\lambda \in [0,1]$ est un hyperparamètre contrôlant l'intensité du lissage. De la sorte, chaque représentation des documents est une moyenne pondérée des vecteurs mots W, dont la pondération est un ratio entre la fréquence des mots du document et de celles des mots des documents proches dans le graphe.

CANE (*Context-Aware Network Embedding*) [Tu et al., 2017] est une méthode de plongement de réseau de documents qui construit autant de représentations de chaque sommet qu'il a de voisins directs dans le graphe. Ces représentations capturent alors les caractéristiques textuelles des documents qui expliquent les liens entre ceux-ci. Cela est fait à travers un mécanisme d'attention mutuelle entre paires de documents. Formellement, CANE cherche les paramètres optimaux U^s, H^s, W et Q de sorte à (1) minimiser les distances des représentations « structurelles » U^s et H^s des sommets proches dans le réseau, (2) minimiser la distance entre les représentations « textuelles » mutuelles des paires de documents u_i^t et h_j^t, calculés à partir des paramètres W et Q et (3) minimiser les distances entre les représentations structurelles et textuelles des documents proches dans le réseau. CANE opère sur un graphe dirigé de documents, où chaque sommet v_i est associé à une séquence de mots $x_i = (\omega_1, ..., \omega_{\ell_i})$. Le modèle minimise la fonction de perte globale :

$$L = \sum_{(v_i,v_j)\in A} L_s(v_i, v_j) + L_t(v_i, v_j), \tag{3.10}$$

où L_s est un fonction de perte dite structurelle basée sur le réseau et L_t une fonction de perte dite textuelle basée sur le contenu des documents. La représentation finale d'un sommet est la concaténation de ses plongements structurels et textuels $u_i = u_i^s \oplus u_i^t$. La fonction basée sur la structure est définie de la sorte :

$$L_s(v_i, v_j) = a_{ij} \log p_{s\to s}(v_j|v_i), \tag{3.11}$$

$$\text{avec} \quad p_{s\to s}(v_j|v_i) = \frac{e^{u_i^s \cdot h_j^s}}{\sum_{k\in\{l|a_{il}>0\}} e^{u_i^s \cdot h_k^s}}. \tag{3.12}$$

L'objectif textuel est choisi pour rapprocher les deux types de plongements (structure et texte) dans le même espace de représentation :

$$L_t(v_i, v_j) = \alpha L_{tt}(v_i, v_j) + \beta L_{ts}(v_i, v_j) + \gamma L_{st}(v_i, v_j) \tag{3.13}$$

$$= a_{ij}\big(\alpha \log p_{t\to t}(v_j|v_i) + \beta \log p_{t\to s}(v_j|v_i) + \gamma \log p_{s\to t}(v_j|v_i)\big). \tag{3.14}$$

Les probabilités $p_{t\to t}, p_{t\to s}$ et $p_{s\to t}$ sont calculées avec une fonction softmax similairement à l'équation 3.12 en remplaçant u_i^s et h_i^s par leur représentations textuelles u_i^t et h_i^t lorsque nécessaire. Ces dernières sont générées par un CNN suivi d'un mécanisme d'attention mutuelle. Tout d'abord, les mots $x_i = (\omega_1, ..., \omega_{l_i})$ sont transformés en séquence de vecteurs $(w_1, ..., w_{l_i})$ à partir d'une matrice de plongement de mots W, puis cette séquence est filtrée par plusieurs convolutions pour générer N représentations intermédiaires que l'on agence de manière matricielle dans C^i. L'attention mutuelle intervient sur ces ensembles de vecteurs. Pour une paire de documents (v_i, v_j) données, une matrice de corrélation F est calculée telle que $F = \tanh({C^i}^\top Q C^j)$. Chaque élément f_{kl} représente un score de corrélation, paramétré par la matrice Q, entre la paire de vecteurs ${c^i}_k$ et ${c^j}_l$. Les poids d'attention α_i et α_j sont calculés en effectuant les moyennes en ligne de F pour C_i et en colonne pour C_j, suivies d'un softmax. Finalement, les plongements textuels sont calculés comme des moyennes pondérés de leurs représentations textuelles intermédiaires : $u_i^t(v_j) = C^i\alpha_i$ et $h_j^t(v_i) = C^j\alpha_j$.

D'autres méthodes de plongement de réseau de documents sont proposées. STNE (*self-translation network embedding*) [Liu et al., 2018] approche ce problème comme une tâche de traduction entre le contenu des documents et la structure du réseau à travers un encodeur basé sur un LSTM bi-directionnel. AANE (*Accelerated Attributed Network Embedding*) [Huang et al., 2017] propose un cadre général pour le plongement de réseau attribué et propose une méthode de construction de représentations en décomposant le problème en plusieurs sous-problèmes indépendants et donc parallélisables. RTM [Chang and Blei, 2009] est une approche probabiliste qui modélise les liens d'un réseau de documents conditionnellement à leur contenu. Enfin, les GNN (*Graph Neural Networks*) permettent de construire des représentations de données structurées dans des graphes. On peut citer VGAE (*Variational Graph Auto-Encoders*) [Kipf and Welling, 2016], GraphSage [Hamilton et al., 2017b] ou encore GAT [Veličković et al., 2017] qui apprennent à agréger le voisinage des sommets grâce à un réseau de neurones profond. Ces approches ont montrés d'excellents résultats lorsque l'apprentissage s'effectue de manière supervisée mais, à notre connaissance, il n'est pas prouvé qu'ils réalisent de meilleures performances dans un cadre non supervisé comme le nôtre. De plus, les GNN constituent des techniques très générales d'apprentissage de représentations dans les graphes pour lesquelles un nombre important de choix sont à prendre. Par exemple, GraphSage vient avec un large choix de fonctions d'agrégation du voisinage des sommets et nécessite de précalculer des représentations des documents qui influent fortement sur les performances. Dans ce sens, il est difficile de sélectionner une configuration d'application de ces approches pour le plongement de documents en réseau qui soit équitable. À l'inverse, l'approche suivie dans cette thèse consiste à construire des modèles « tout en main », opérant sur des données brutes pour aborder des tâches de recommandation. Pour ces raisons, nous ne nous comparons pas aux méthodes de GNN lors de nos expérimentations.

3.4 Méthodes proposées

Dans cette section, nous présentons nos trois contributions aux algorithmes de plongement de réseau de documents. Tout d'abord, nous étendons GVNR, présenté dans le chapitre précédent, afin d'incorporer l'information textuelle d'un réseau de documents

dans la formulation de la factorisation. Ensuite, nous détaillons MATAN (*Mutual Attention for Text-Attributed Network*), un modèle qui représente un document d'un réseau contextuellement à ses documents voisins, similaire à CANE mais reposant sur un nombre très inférieur de paramètres. Enfin, nous présentons IDNE (*Inductive Document Network Embedding*), un algorithme qui entraîne un mécanisme d'attention à identifier des thématiques associées aux mots des documents de façon à modéliser la structure du réseau qui les lie.

3.4.1 GVNR-t

GVNR est un algorithme de plongement de réseau qui construit des représentations U et H de dimension $n_v \times \rho$ en factorisant une matrice des comptages de co-occurrences S des sommets dans des marches aléatoires. Pour intégrer l'information textuelle associée aux documents d'un réseau, on considère la matrice $X \in \mathbb{N}^{+^{n_d \times n_\omega}}$ des fréquences des mots des documents. De plus, on associe des représentations W de dimension $n_\omega \times \rho$ aux mots du vocabulaire. Nous faisons l'hypothèse que le sens d'un document peut être capturé par la moyenne des vecteurs mots le représentant. Cette hypothèse est raisonnable lorsque la taille des documents est limitée [Arora et al., 2016]. Pour un document d_i, cette moyenne s'obtient par la formule $x_i W$. Au lieu de définir le vecteur contexte d'un sommet par la matrice H dans l'équation 2.24, on le remplace par la moyenne des vecteurs mots composant le document associé à ce sommet, c'est-à-dire $h_i = x_i W$. GVNR-t, l'extension de GVNR pour les réseaux de documents cherche alors les paramètres idéaux de U, W b^u et b^h de sorte à minimiser l'équation suivante :

$$\underset{U,W,b^u,b^h}{\operatorname{argmin}} \sum_{i=1}^{n_d} \sum_{j=1}^{n_d} \delta(s_{ij}) \big(u_i \cdot x_j W + b_i^u + b_j^h - \log(1 + s_{ij})\big)^2, \tag{3.15}$$

où la fonction δ est définie par les équations 2.25 et 2.26 de façon à sous-échantillonner les entrées nulles de S :

$$\delta(s_{ij}) = \begin{cases} 1 & \text{si } s_{ij} > 0, \\ m_i & \text{sinon, où } m_i \sim \text{Bernoulli}(\alpha_i), \end{cases} \tag{3.16}$$

avec

$$\alpha_i = \begin{cases} n_k \times \frac{p_i}{1-p_i} & \text{si } p_i \leq (n_k + 1)^{-1}, \\ 1 & \text{sinon.} \end{cases} \tag{3.17}$$

De cette manière, le modèle apprend simultanément des représentations des sommets U et des mots W qui permettent alors de représenter les documents. La représentation finale d'un document est la concaténation des deux composantes $u_i \oplus x_i W$. Le nombre de paramètres de cette extension est similaire à celui de l'algorithme original puisque le nombre de sommets et le nombre de mots dans le vocabulaire sont souvent d'un ordre de grandeur similaire (la matrice H est remplacée par la matrice W). Cependant, la représentation h_i, qui dans GVNR est directement extraite de la matrice H, fait intervenir dans cette extension une moyenne de plusieurs lignes de W. La rétropropagation du gradient, pour toute paire de documents, se fait alors sur une nombre plus important de paramètres. En somme, pour l'apprentissage des paramètres, GVNR-t possède une complexité algorithmique en temps plus grande que GVNR, proportionnellement au nombre moyen de mots contenus dans les documents du réseau. Enfin GVNR-t est semblable à TADW dans ce sens où le modèle possède une composante « réseau » U et une composante « textuelle » XW. Cependant il ne repose pas sur le calcul potentiellement coûteux de représentations

LSA des documents, mais fait directement intervenir un apprentissage de représentation des mots, spécifique au réseau.

Par la suite nous présentons MATAN, notre première contribution pour le plongement de documents en réseau reposant sur un mécanisme d'attention. Ce modèle réalise une attention dite mutuelle, c'est-à-dire qu'elle s'applique à des paires de documents en cherchant à identifier les facteurs du premier document contextuellement au second qui expliquent l'existence ou non d'un lien entre ceux-ci.

3.4.2 MATAN

CANE, décrit en section 3.3.2, est un modèle de plongement qui génère des représentations pour des paires de documents dans un réseau. La représentation d'un document est conditionnée au contenu d'un autre document. Ce faisant, l'algorithme modélise la structure du réseau de manière à mettre en lumière les caractéristiques textuelles qui expliquent les liens. Cependant, CANE utilise un nombre élevé de paramètres (plongements de mots, CNN et matrice de corrélation) et son optimisation fait intervenir de multiples fonctions de pertes ce qui rend sa complexité algorithmique élevée.

Nous présentons dans cette section MATAN (*Mutual Attention for Text-Attributed Network*), un modèle qui génère des représentations mutuelles de paires de documents, où un document possède plusieurs représentations en fonction de ses voisins dans le réseau. Soit $G = (V, E, X)$ un réseau constitué de n_d documents dont les liens sont décrits par une matrice d'adjacence A et dont les textes associés sont décrits par une matrice document-terme X. Nous décrivons en premier le mécanisme d'attention f_Θ, paramétré par Θ, qui génère des représentations mutuelles d'une paire de documents étant donnée une matrice de plongements de n_ω mots de dimension ρ telle que $W \in \mathbb{R}^{n_\omega \times \rho}$ et des paramètres Θ connus. Nous détaillons ensuite l'optimisation de ces paramètres grâce au réseau.

Attention mutuelle

L'attention mutuelle est définie comme une fonction non-symétrique $f_\Theta(W^i, W^j)$ où W^i est le sac de plongement de mots du document d_i, correspondant à une version compacte de la matrice $\text{diag}(x_i)W$ dont les lignes nulles sont supprimées. Cette fonction, paramétrée par trois matrices de projections $\Theta = \{P^Q, P^K, P^V\}$ de dimensions $\rho \times \rho$, prend une paire de documents en entrée et génère une représentation du premier document $u(d_i|d_j)$ contextuellement au second document. On peut générer une représentation du second document contextuellement au premier en inversant les indices $u(d_j|d_i) = f_\Theta(W^j, W^i)$. Suivant le mécanisme d'attention du *Transformer* [Vaswani et al., 2017] présenté en section 3.3.1, nous construisons des vecteurs requêtes Q_i par projection des vecteurs mots du premier document ainsi que des vecteurs clefs et des vecteurs valeurs par projections des vecteurs mots du second document tel que :

$$Q^i = W^i P^Q, \tag{3.18}$$

$$K^j = W^j P^K, \tag{3.19}$$

$$V^j = W^j P^V. \tag{3.20}$$

Similairement à la formule 3.5, chaque mot du premier document est représenté comme une combinaison des valeurs, générées à partir du second document :

$$U(d_i|d_j) = \text{softmax}(\frac{Q^i {K^j}^\top}{\sqrt{\rho}})V^j. \tag{3.21}$$

Une différence notable avec le *Transformer* réside dans le fait que nous appliquons le softmax sur l'intégralité des valeurs de la matrice et non sur chaque ligne. Ceci permet

au modèle de ne pas nécessairement attribuer de poids important à chaque mot, ce qui est le cas lorsque l'on applique le softmax en ligne puisque alors la somme des poids de chaque mot somme à 1. Les représentations $U(d_i|d_j)$ ont comme dimension $\ell_i \times \rho$ où ℓ_i est le nombre de mots composant le document d_i. Pour obtenir la représentation finale du document d_i contextuellement au document d_j, on moyenne les lignes de $U(d_i|d_j)$ tel que $u(d_i|d_j) = \sum_{k=1}^{\ell_i} U(d_i|d_j)_k$.

Optimisation du modèle

Pour optimiser les paramètres W et $\Theta = \{P^Q, P^K, P^V\}$, nous calculons une matrice binaire $S \in \{0,1\}^{n_d \times n_d}$ à partir de la matrice d'adjacence du réseau où s_{ij} indique si deux documents sont considérés comme liés. De plus, nous pré-entraînons les vecteurs mots W avec Word2vec (SGNS) afin de favoriser l'apprentissage des matrices de projection Θ. On maximise alors la log-vraisemblance suivante :

$$L(W,\Theta) = \sum_{i=1}^{n_d}\sum_{j=1}^{n_d} s_{ij} \log \sigma\big(u(d_i|d_j) \cdot u(d_j|d_i)\big) + (1 - s_{ij}) \log \sigma\big(- u(d_i|d_j) \cdot u(d_j|d_i)\big). \tag{3.22}$$

Nous cherchons ainsi à rapprocher les représentations mutuelles de paires de documents qui sont proches dans le réseau et à éloigner celles qui ne le sont pas. L'objectif est que les paramètres W et $\Theta = \{P^Q, P^K, P^V\}$ capturent les affinités entre les mots des documents du réseau de sorte à expliquer les liens dans le réseau. Pour visualiser ces liens, nous pourrons nous intéresser aux poids d'attentions $\alpha = \text{softmax}(\frac{Q^i {K^j}^\top}{\sqrt{\rho}})$ dans l'équation 3.21, ce que nous faisons en section 3.6.3.

L'équation 3.22 se faisant sur l'ensemble des paires de documents possibles, dont le nombre d'élève à n_d^2, celle-ci est en pratique coûteuse en calcul. Le choix de la construction de la matrice S et son sous-échantillonage permettent de réduire significativement la complexité algorithme de l'optimisation des paramètres. MATAN cherchant à identifier les facteurs textuels expliquant les liens entre documents dans le réseau, nous utilisons pour nos expérimentations la version binarisée de la matrice d'adjacence telle que $s_{ij} = 1$ si et seulement si il existe un lien entre les documents d_i et d_j dans le réseau. De plus, nous réduisons la complexité en temps de l'optimisation de la log-vraisemblance en sous-échantillonnant les paires de documents (d_i, d_j). Nous faisons ceci en sélectionnant des nombres égaux et très inférieurs à n_d^2 de paires positives ($s_{ij} = 1$) et de paires négatives ($s_{ij} = 0$). L'optimisation des paramètres du modèle est réalisé avec l'algorithme du gradient stochastique ADAM [Kingma and Ba, 2014].

Nous détaillons dans la section suivante notre seconde contribution (IDNE) dans les mécanismes d'attention pour le plongement de documents reposant sur un nouveau mécanisme : l'attention thématique.

3.4.3 IDNE

Lorsque l'on cherche à représenter des documents en réseau, l'approche classique consiste à apprendre conjointement et de manière transductive des vecteurs qui capturent les caractéristiques du réseau et du texte des documents. Le terme « transduction » réfère au fait que l'intégralité des données que l'on veut représenter doit être accessible durant l'entraînement des paramètres. Par opposition, le terme « induction » réfère à la capacité d'un modèle de représenter des données qu'il n'a pu observer durant l'apprentissage. À titre d'exemples, TADW et GVNR-t sont des modèles transductifs, dans la mesure où la construction d'une représentation d'un nouveau document n'est pas triviale. Pour

ce dernier, si x_i est le vecteur TF d'un document qui n'appartient pas au réseau qui a servi de données d'entraînement au modèle, la seule façon de pouvoir le représenter est de construire la moyenne de ces mots x_iW (composante textuelle). Cependant, il est impossible d'obtenir une représentation équivalente à celles contenues dans U (composante réseau), à moins de réentraîner le modèle sur un nouveau réseau contenant le document en question ou, *a minima*, de mettre en place une méthode incrémentale de factorisation telle que décrite dans [Zhao and Tan, 2016]. Cette dernière approche, quoique moins coûteuse qu'un réapprentissage complet, n'est pas pratique puisqu'elle repose sur des méthodes d'optimisation en ligne complexes dont la convergence n'est pas toujours garantie.

Pour de nombreuses applications, l'induction est une propriété importante d'un modèle. Par exemple, lorsqu'un nouvel article scientifique est publié, on voudrait induire directement une représentation de celui-ci de façon à pouvoir le comparer avec les articles d'un réseau scientifique que l'on a préalablement plongés. Sur un site de questions et réponses, lorsqu'un utilisateur pose une nouvelle question, on voudrait pouvoir la comparer avec l'ensemble des questions posées par le passé, pour voir si elle n'a pas déjà été posée. Il est donc intéressant de construire un modèle de représentation de documents en réseau qui permette d'induire des représentations pour des documents qui ne font pas (encore) partie du réseau. IDNE, que nous détaillons dans cette section, répond précisément à ce besoin en considérant le problème du plongement de réseau de documents comme un problème d'apprentissage de représentation du texte guidé par la structure d'un réseau. À notre connaissance, Graph2gauss [Bojchevski and Günnemann, 2018] est le premier modèle à avoir proposé un modèle inductif. Notons que, en plongement de réseau, la notion d'induction peut aussi s'appliquer à un problème différent, où l'on cherche à représenter de nouveaux sommets dans un graphe en se basant uniquement sur les nouveaux liens associés à ce sommet. Ici nous n'abordons que le problème de l'induction de nouvelles représentations à partir des attributs, en particulier du texte.

Ajoutons que l'interprétabilité [Goebel et al., 2018] devient un enjeu important pour de nombreux systèmes de recherche d'information. Au-delà de la qualité des représentations apprises, la capacité de comprendre ces représentations est souhaitable pour plusieurs raisons : (1) cela permet d'expliciter les forces et les faiblesses d'un modèle et peut, par exemple, servir de guide dans la sélection des hyperparamètres optimaux et (2) cela peut avoir un impact bénéfique sur la confiance accordée par un utilisateur aux recommandations issues d'un tel modèle. Ce dernier point relève aussi de la notion d'équité dans les systèmes d'information. Les problèmes de biais dans les systèmes de recommandation, dus à de multiples facteurs [Buckley et al., 2007, Craswell et al., 2008, Bellogín et al., 2017], peuvent en effet être corrigés, ou du moins appréhendés, par les utilisateurs eux-mêmes à condition de leur exposer de manière claire les raisons des recommandations qu'ils reçoivent. Dans ce sens, IDNE met en œvre un mécanisme d'attention qui modélise des thématiques associées aux documents. Il est alors possible d'étudier les mots qui définissent ces thématiques afin d'interpréter facilement les caractéristiques du réseau de documents détectées par l'algorithme.

Pour répondre à ces problématiques, IDNE construit des représentations interprétables de documents en réseaux et permet aussi de générer des représentations de manière inductive, pour des documents non observés durant l'apprentissage.

Méthode générale

Nous cherchons à construire, dans un espace vectoriel de dimension faible, des représentations d'un ensemble de n_d documents organisés dans un réseau. Ceux-ci sont décrits par une matrice document-terme $X \in \mathbb{N}^{n_d \times n_\omega}$, où n_ω est la taille du vocabulaire. La méthode que nous proposons, IDNE (*Inductive Document Network Embedding*), ap-

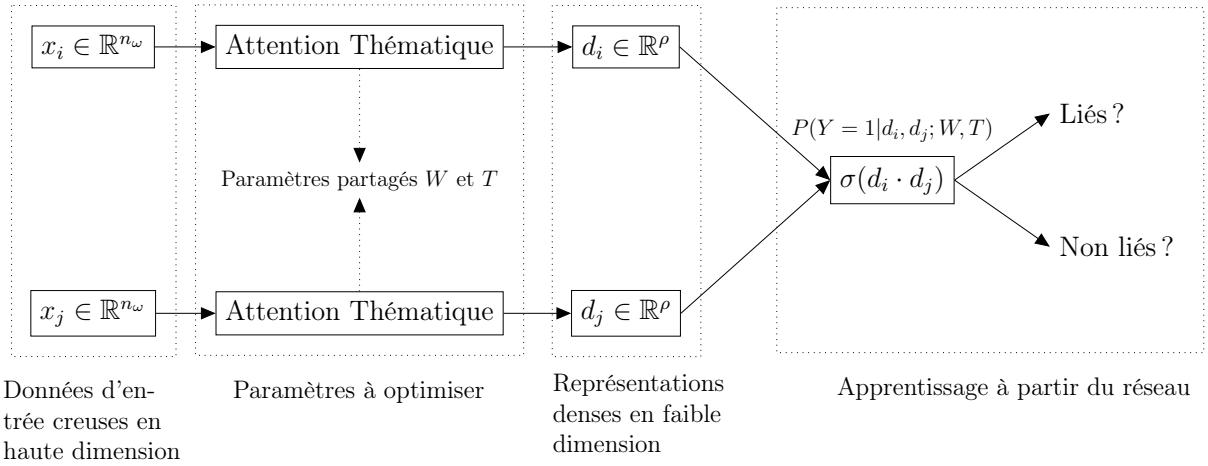


FIGURE 3.3 – Vue d'ensemble de IDNE. Des paires de représentations creuses de documents x_i and x_j sont échantillonnées et transformées en représentations denses de faible dimension d_i and d_j par le mécanisme d'attention thématique. Les paramètres de ce mécanisme sont optimisés en utilisant la structure du réseau connectant les documents et en définissant une probabilité selon le modèle que ces paires échantillonnées soient liés.

prend à représenter les mots et les thématiques sous-jacentes au corpus dans un unique espace vectoriel. Les représentions des documents sont calculées en combinant les mots et les thématiques à l'aide d'un mécanisme d'attention. Nous présentons tout d'abord comment nous construisons les représentations des documents à partir de vecteurs mots et de vecteurs thématiques calculés par un nouveau mécanisme d'attention, l'Attention Thématique (AT). Ensuite, nous décrivons comment nous estimons les paramètres de ces vecteurs mots et vecteurs thématiques, en guidant l'apprentissage par la structure du réseau de documents. La figure 3.3 montre une vue d'ensemble du modèle et de son entraînement.

Mécanisme d'attention thématique

Nous définissons un espace vectoriel de dimension ρ dans lequel les mots et les thématiques sont représentés. On note $W \in \mathbb{R}^{n_\omega \times \rho}$ la matrice qui contient les n_ω vecteurs mots et $T \in \mathbb{R}^{n_t \times \rho}$ la matrice des n_t vecteurs thématiques. Nous cherchons à contextualiser ces représentations globales des thématiques du corpus selon les mots composant chaque documents du réseau. Ceci est fait à travers un mécanisme d'attention, dont le calcul matriciel des poids est présenté en figure 3.4.

Attention thématique Inspirés par le *Set Tranformer* [Lee et al., 2019], dans lequel le mécanisme d'attention original du *Transformer* est modifié en remplaçant les vecteurs requêtes par des « points inducteurs » globaux, nous proposons un nouveau mécanisme d'attention faisant intervenir n_t requêtes thématiques globales. Cette approche permet notamment de réduire la complexité en temps de l'attention, puisque le nombre de thématiques est généralement inférieur aux nombre de mots ℓ_i du document sur lequel elle est appliquée. Formellement, étant donné un document d_i et sa représentation document-terme $x_i \in \mathbb{N}^{+n_\omega}$, nous mesurons les poids d'attention entre les thématiques et les mots, $Z^i \in \mathbb{R}^{n_t \times n_\omega}$, comme suit :

$$Z^i = \mathrm{g}\big(TW^\intercal \mathrm{diag}(x_i)\big). \tag{3.23}$$

La fonction d'activation g doit satisfaire deux exigences : (1) tous les poids sont non négatifs et (2) les colonnes de Z^i somment à un. L'intuition derrière la première exigence

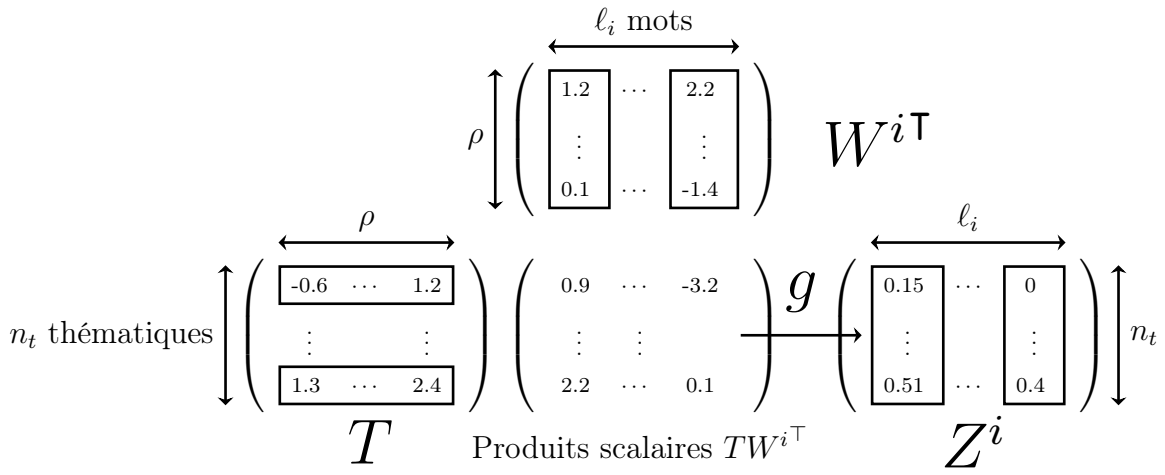


FIGURE 3.4 – Calcul matriciel des poids de l'attention thématique. Ici W^i est la version compacte de $\text{diag}(x_i)W$ où les lignes nulles sont supprimées car elles n'ont pas d'impact sur le résultat. ℓ_i indique le nombre de mots distincts dans le document d_i. Chaque élément z_{jk} de Z^i est le produit scalaire rectifié et normalisé en colonne par la fonction g entre le vecteur thématique t_j et le vecteur mot ${w^i}_k$ représentant la force d'association entre la thématique et le mot en question dans le document d_i. La représentation finale du document est alors la somme des représentations spécifiques aux thématiques $u_i = \sum_k^{n_t} \left(\frac{Z^i W^i}{|x_i|_1}\right)_k$.

est que l'application de la positivité devrait conduire à des vecteur creux interprétables. La deuxième exigence assure que les poids associés à chaque mots soient analogues à des probabilités sur les thématiques, similairement à ce qui est fait dans les modèles thématiques neuronaux [Srivastava and Sutton, 2017]. Un choix évident serait d'appliquer la fonction softmax en colonne, cependant, nous constatons empiriquement que la fonction $\text{ReLU}(x) = max(0, x)$ suivie d'une normalisation en colonne produit de meilleurs résultats.

Représentation des documents Étant donné Z^i, nous pouvons calculer des représentations spécifiques aux thématiques pour chaque document i. Pour une thématique k donnée, la représentation de dimension ρ du document d_i est :

$$u(i|k) = \frac{z_k^i \text{diag}(x_i) W}{|x_i|_1}, \tag{3.24}$$

où z_k^i désigne la $k^{\text{ème}}$ ligne de la matrice Z^i. De manière similaire à l'équation 3.5, chaque vecteur thématique, jouant le rôle d'un vecteur requête, porte son attention sur les vecteurs mots, jouant le rôle des vecteurs clefs, pour générer Z^i. Les représentations spécifiques à chaque thématique sont alors la somme des valeurs, également jouées par les vecteurs mots, pondérée par les poids d'attention. La représentation finale d'un document est obtenue par simple somme de toutes les représentations spécifiques aux thématiques, ce qui se traduit par $u_i = \sum_k^{n_t} u(i|k)$. La normalisation par $|x_i|_1$ dans l'équation 3.24 garantit que les représentations des documents aient le même ordre de grandeur que les vecteurs mots et stabilise ainsi l'apprentissage des paramètres.

Apprentissage guidé par le réseau

Le corpus de documents étant organisé en réseau, nous proposons d'estimer les paramètres W et T du mécanisme d'attention thématique en exploitant les liens entre les

documents. Nous postulons que les représentations des documents reliés par un court chemin dans le réseau devraient être plus similaires dans l'espace vectoriel que celles qui sont éloignées. Ainsi, nous apprenons W et T de manière supervisée à travers l'apprentissage d'un modèle discriminatif.

Soit $S \in \{0,1\}^{n_d \times n_d}$ une matrice binaire, de sorte que $s_{ij} = 1$ si le document j est accessible à partir du document i et $s_{ij} = 0$ sinon. Nous modélisons la probabilité qu'une paire de documents soit connectée, compte tenu de leurs représentations, en termes de la sigmoïde du produit scalaire entre u_i et u_j :

$$P(Y = 1|u_i, u_j; W, T) = \sigma(u_i \cdot u_j). \tag{3.25}$$

En supposant que les représentations du document sont i.i.d, nous pouvons exprimer la log-vraisemblance de S étant donnés W et T :

$$\begin{aligned} L(W,T) &= \sum_{i=1}^{n_d} \sum_{j=1}^{n_d} \log p(Y = \delta_{ij}|u_i, u_j; W, T), \\ &= \sum_{i=1}^{n_d} \sum_{j=1}^{n_d} s_{ij} \log \sigma(u_i \cdot u_j) + (1 - \delta_{ij}) \log \sigma(-u_i \cdot u_j). \end{aligned} \tag{3.26}$$

En cherchant à maximiser la log-vraisemblance par un algorithme de descente du gradient, nous apprenons des vecteurs mots et des vecteurs thématiques produisant des représentations des documents capables de reconstruire au mieux la matrice S. La représentation d'un document u_i est directement obtenue en appliquant le mécanisme d'attention thématique sur son texte x_i, indépendemment des informations liées au réseau. Une fois le modèle entraîné, on peut donc représenter tout document dont les mots font partie du vocabulaire d'apprentissage de manière inductive, sans nécessairement que le document fasse partie du réseau. De plus, on peut analyser les valeurs des poids d'attention Z^i afin d'interpréter les thématiques liées au document. Nous évaluons qualitativement ces thématiques en section 3.6.3 en étudiant la distribution des produits scalaires entre les vecteurs thématiques et les vecteurs mots.

Comme pour MATAN, IDNE est défini à partir d'une matrice de binaire indiquant si les paires de documents sont considérés comme liés dans le réseau. Le choix de cette matrice est libre, mais doit permettre d'identifier les connexions du réseau de sorte à guider l'apprentissage du mécanisme d'attention thématique. Lors de nos expérimentations, nous avons utilisé la matrice suivante :

$$s_{ij} = \begin{cases} 1 & \text{si } (A + A^2)_{ij} > 0, \\ 0 & \text{sinon.} \end{cases} \tag{3.27}$$

Cette matrice indique que deux documents sont considérés comme liés si et seulement si ils sont directement voisins dans le réseau où s'ils ont au moins un voisin en commun. Ce choix empirique pourrait différer selon le type de similarité que l'on souhaite capturer entre les documents dans le réseau, ce dont nous discutions plus en détail dans la conclusion de ce chapitre. De plus, la log-vraisemblance est maximisée en sous-échantillonnant le nombre d'exemples observés, comme pour MATAN, en sélectionnant des nombres égaux et très inférieurs à n_d^2 de paires positives ($s_{ij} = 1$) et de paires négatives ($s_{ij} = 0$). Les paires positives de documents sont tirées en fonction du nombre de chemins de longueur 1 ou 2 les reliant et les échantillons négatifs sont uniformément tirés. L'optimisation des paramètres du modèle est réalisé avec l'algorithme du gradient stochastique ADAM [Kingma and Ba, 2014].

Avant de présenter nos résultats d'expérimentation, nous détaillons dans la section suivante les protocoles d'évaluation que nous avons mis en œuvre, notamment pour évaluer la capacité d'induction des algorithmes de plongement de documents en réseau.

3.5 Méthodologies d'évaluation

L'évaluation d'un algorithme de plongement de documents en réseau peut se faire de la même façon qu'en section 2.5, c'est-à-dire en mesurant la performance d'un classifieur linéaire sur les représentations des documents apprises et en prédisant des liens cachés durant l'apprentissage. Cependant, ces évaluations s'appliquent sur des modèles transductifs. En effet, l'intégralité des documents qui servent de données de test sont observées durant l'apprentissage du modèle. L'évaluation de la capacité d'induction d'un algorithme nécessite des protocoles expérimentaux différents, dans lesquels les documents testés ne sont pas observés durant l'apprentissage du modèle. Pour évaluer GVNR-t, IDNE et MATAN, nous nous intéressons alors aux tâches de classification de documents et de prédiction de liens abordées de manière transductive et de manière inductive, que nous décrivons ci-après.

3.5.1 Prédiction inductive de liens

La prédiction inductive de liens est une tâche qui sert à évaluer la capacité d'un algorithme de plongement de documents en réseau à reconstruire la structure du réseau à partir du contenu textuel seulement. À la différence de la version transductive, les documents pour lesquels l'existence d'un lien est prédite n'ont pas été observés au sein du réseau durant l'apprentissage. Ainsi, on cherche à mesurer si l'algorithme est effectivement capable de retranscrire les interactions dans le graphe en analysant uniquement le contenu des documents.

Étant donnés un algorithme de plongement de documents en réseau et un réseau de documents de matrice d'adjacence $A \in \mathbb{R}^{n_d \times n_d}$, le protocole d'évaluation commence par sélectionner un certain sous-ensemble de $m_d < n_d$ documents et extrait le sous-graphe correspondant de matrice d'adjacence $A' \in \mathbb{R}^{m_d \times m_d}$. L'algorithme de plongement est entraîné sur ce sous-graphe. Les représentations des n_d documents sont ensuite générées de manière inductive en utilisant leurs contenus textuels seulement. La prédiction de lien est alors effectuée sur des paires de documents aléatoirement tirés où au moins l'un des documents ne fut pas observé durant l'apprentissage et de sorte à ce que 50% des paires soient liées dans le réseau original et que 50% ne le soient pas. Nous utilisons l'aire sous la courbe ROC comme mesure de la qualité des prédictions. La figure 3.5 illustre ce protocole d'évaluation. Pour réduire l'impact du tirage aléatoire des m_d documents sur l'évaluation, on répète plusieurs fois cet algorithme (souvent 10) et on calcule la moyenne des scores obtenus.

3.5.2 Classification inductive des sommets

La classification inductive consiste à évaluer la capacité d'un algorithme de plongement documents en réseau à classer des documents qui ne font pas partie du réseau servant à l'apprentissage, en utilisant comme information leur contenu textuel seulement. À la différence de la version transductive, le documents pour lequel l'appartenance à une classe est prédite n'est pas observé au sein du réseau durant l'apprentissage. Ainsi, on cherche à mesurer si l'algorithme est capable de construire des représentations efficaces en dehors des données qui lui ont servies d'entraînement.

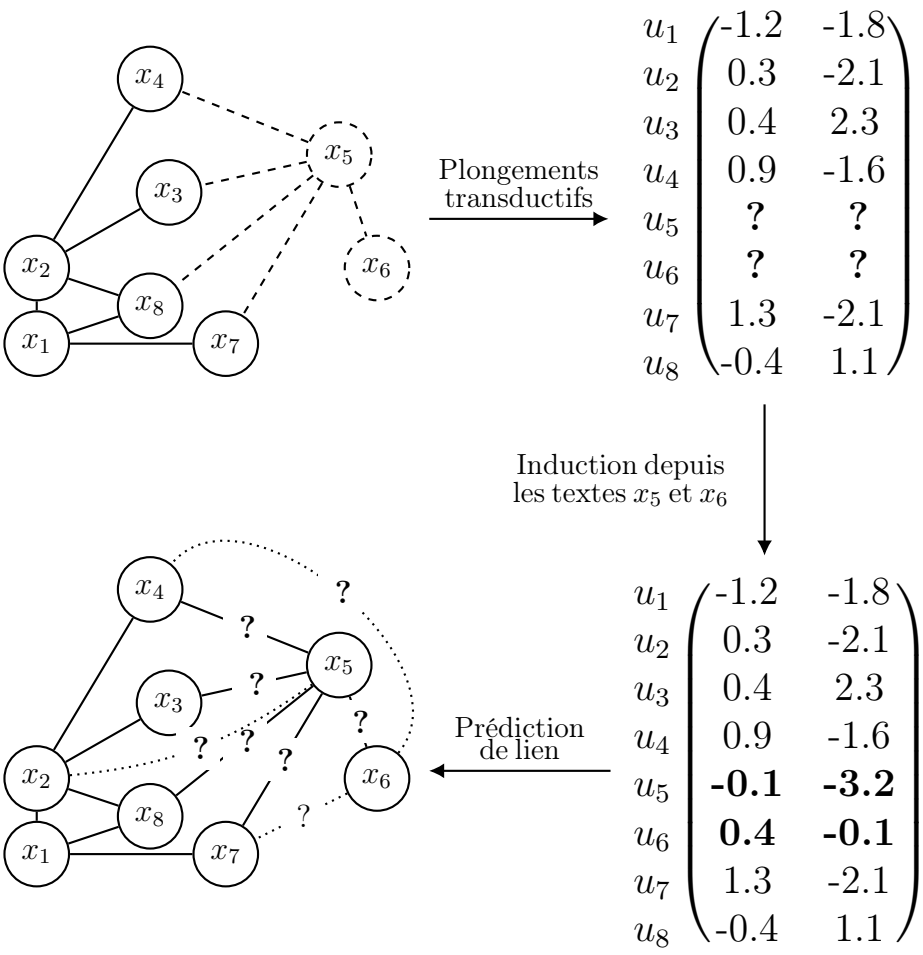


FIGURE 3.5 – Évaluation des plongements de réseau de documents pour la prédiction inductive de liens.

Étant donnés un algorithme de plongement de documents en réseau et un réseau de documents de matrice d'adjacence $A \in \mathbb{R}^{n_d \times n_d}$, l'évaluation commence par sélectionner certain sous-ensemble de $m_d < n_d$ documents et extrait le sous-graphe correspondant de matrice d'adjacence $A' \in \mathbb{R}^{m_d \times m_d}$. L'algorithme de plongement est entraîné sur ce sous-graphe. Un classifieur linéaire est entraîné sur les représentations des m_d documents observés durant l'apprentissage de l'algorithme. La classification des documents non observés durant l'apprentissage se fait alors après avoir généré leurs représentations de manière inductive, utilisant uniquement leurs contenus textuels. Pour évaluer la qualité des prédictions, nous utilisons les mêmes mesures que pour la version transductive, à savoir le score F_1 et l'AUC. La figure 3.6 illustre ce protocole d'évaluation.

Pour réduire l'impact du tirage aléatoire des m_d documents sur l'évaluation, on répète plusieurs fois cet algorithme (souvent 10) et on calcule la moyenne des scores obtenus. De plus, on peut utiliser différents pourcentages r de documents observés pour avoir une idée plus fine de la robustesse de l'algorithme dans l'apprentissage des représentations vis-à-vis de la quantité d'information disponible. Enfin, cette évaluation repose sur un tirage aléatoire d'une sous-partie des documents. Ce tirage suit les mêmes précautions détaillées en section 2.5, à savoir, les documents conservés pour l'apprentissage sont tirés de manière à maintenir des proportions de documents appartenant aux classes équivalentes aux proportions originales, en utilisant l'algorithme itératif de stratification [Sechidis et al., 2011].

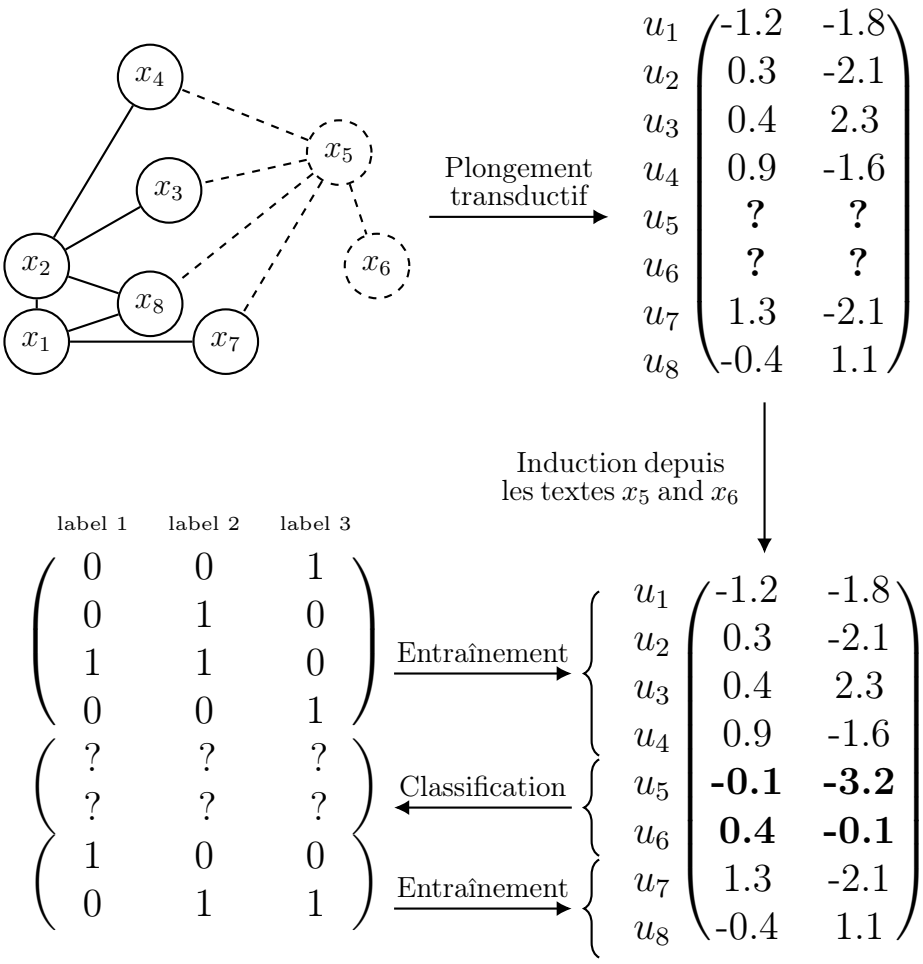


FIGURE 3.6 – Évaluation des plongements de réseau de documents pour la classification inductive des sommets.

3.6 Expérimentations

Nous présentons ici les résultats des expérimentations mettant en valeur les performances de GVNR-t MATAN et IDNE vis-à-vis des algorithmes de l'état de l'art. Nous montrons en premier lieu les résultats quantitatifs obtenus sur les tâches transductives et inductives de classification des sommets et de prédiction de liens. Nous montrons ensuite l'effet du choix des hyperparamètres sur IDNE et étudions qualitativement les représentations des thématiques et des mots apprises par ce modèle avant d'étudier les poids d'attention obtenus avec MATAN.

3.6.1 Résultats quantitatifs

Nous présentons dans cette section les résultats expérimentaux obtenus sur la classification et la prédiction de liens de manières transductive et inductive. Nous détaillons tout d'abord les paramètres des évaluations et présentons les algorithmes et implémentations utilisées pour conduire les expériences en discutant notamment les avantages et inconvénients de ces différentes méthodes. Nous analysons ensuite les résultats.

Évaluations

Comme pour le chapitre précédent, nous évaluons les algorithmes sur les tâches transductives de classification des sommets et de prédiction de liens, toutes deux détaillées en section 2.5. Nous réalisons de plus une évaluation sur les tâches inductives détaillées en section 3.5. Ces évaluations sont réalisées sur les jeux de données présentés en introduction de ce manuscrit en section 1.4.2 qui représentent des réseaux de documents (tous sauf Wikipedia et PPI).

Pour la classification et la prédiction transductive, nous suivons précisément les mêmes protocoles d'évaluation décrits en section 2.6.1. Pour les tâches inductives, nous enlevons 10% des sommets du réseau et les liens les connectant, puis réalisons un apprentissage transductifs sur les 90% de sommets non cachés en utilisant leurs contenus textuels et les liens du sous-réseau résultant. Nous construisons ensuite des représentations pour les sommets cachés en utilisant exclusivement leurs contenus textuels. Pour la classification, un modèle de régression logistique est entraîné sur les représentations de 90% de sommets observés durant l'apprentissage et les *labels* des 10% de sommets dont les représentations ont été générées inductivement sont prédits. Nous rapportons le score F_1 et l'aire sous la courbe ROC de ces prédictions vis-à-vis des classes vérité terrain. Pour la prédiction, nous calculons les probabilités des 10% des sommets cachés d'être connectés avec des sommets des 90% de sommets non cachés par similarité cosinus entre leurs représentations.

Algorithmes

Pour nos expérimentations, nous considérons 3 représentations n'utilisant que les données textuelles (TF, TF-IDF et LSA), la concaténation d'un algorithme de représentation des documents et d'un algorithme de représentation du réseau (DW+LSA), deux algorithmes de représentation de documents en réseau de la littérature (TADW et G2G) et enfin nos trois propositions décrites dans la section 3.4 (GVNR-t, MATAN et IDNE). De plus, nous montrons pour comparaison les performances obtenues par un algorithme aléatoire. La dimension de toute représentation dense (autres que TF et TF-IDF) est fixée à $\rho = 256$. Lorsqu'une méthode repose sur des marches aléatoires, nous réalisons systématiquement $\mu = 80$ marches de longueurs $\ell = 40$ et appliquons une fenêtre de taille $\tau = 5$. Voici les détails des implémentations que nous avons utilisées :

- Aléatoire : génère des représentations aléatoires ;
- TF : représentations document-terme normalisées des documents. Pour les calculer, nous enlevons les mots vides, gardons que les caractères alphanumériques et les réduisons en minuscule pour construire un vocabulaire de mots. Ce vocabulaire est ensuite réduit en excluant tout mot occurrant moins de 5 fois dans le corpus ou apparaissant dans plus d'un quart des documents ;
- TF-IDF : représentations des documents obtenues à partir des représentations TF en pondérant par le logarithme des fréquences inverses des occurrences dans les documents ;
- LSA : représentations obtenues par analyse sémantique latente calculées par décomposition en valeurs singulières tronquée des représentations TF-IDF ;
- DW+LSA : concaténation de représentations des sommets du réseau obtenues avec DeepWalk (DW) tel que décrit dans le chapitre précédent, en dimension 128, et de représentations LSA en dimension 128. Cette méthode ne peut induire de nouvelles représentations car DeepWalk n'est pas inductif ;
- TADW : nous utilisons l'implémentation fournie pas les auteurs en en utilisant les représentations LSA précédemment présentées et un facteur pénalité $\lambda = 0.2$. Pour induire de nouvelles représentations à partir du texte d'un document, nous calculons sa représentation LSA x_i et nous calculons le vecteur Hx_i où H est la

composante « textuelle » de TADW dans l'équation 3.7 ;
— G2G : nous utilisons l'implémentation fournie par les auteurs et utilisons les représentations document-terme comme attributs associés aux sommets du réseau ;
— GVNR-t : nous utilisons $n_k = 1$ échantillon négatif par échantillon positif avec un seuillage $s_{\min} = 1$ réalisons 4 époques avec un taux d'apprentissage initial de 0.001. Pour induire une représentation d'un nouveau document, nous calculons sa représentation x_i obtenues par TF, puis calculons le vecteur $x_i W$ représentant la moyenne des vecteurs mots appris par GVNR-t en équation 3.15 ;
— MATAN : nous utilisons $n_k = 1$ échantillons négatifs, réalisons 5000 itérations constituées de 32 échantillons équilibrés de paires positives et négatives avec un taux d'apprentissage initial de 0.001. Nous pré-entraînons les représentations des mots en appliquant SGNS sur les documents avec 15 échantillons négatifs, une fenêtre de taille 10 et en opérant 20 itérations sur les corpus. L'induction d'une nouvelle représentation d'un document d_i se fait en passant la paire de représentations document-terme (x_i, x_i) ;
— IDNE : nous utilisons $n_k = 1$, $n_t = 32$ vecteurs thématiques et réalisons 5000 itérations constituées de 32 échantillons équilibrés de paires positives et négatives avec un taux d'apprentissage initial de 0.001.

Comme nous le verrons par la suite, les représentations TF et TF-IDF permettent d'établir de bons scores de référence pour les tâches d'induction puisque ces représentations, basées uniquement sur le texte, vont nous permettre de déterminer à quel point la structure du graphe permet à un algorithme d'apprentissage de représentation de documents en réseau d'apprendre des représentations du texte. Cependant il est nécessaire de noter que ces représentations sont en haute dimension (égale à n_ω, le nombre de mots dans le vocabulaire) ce qui impacte négativement la complexité en temps du modèle de régression logistique appliqué pour la classification. À titre d'exemple, l'évaluation en classification transductive pour les représentations TF sur Academia SE a duré 20 549 secondes, contre en moyenne 12 000 ± 1 000 secondes pour IDNE, Graph2gauss, GVNR-t et MATAN. LSA permet de palier ce problème de la dimension, à condition d'être correctement dimensionnée. Comme nous le montrons dans nos résultats, il faudrait utiliser une dimension plus élevée que celle avec laquelle nous comparons les algorithmes de plongement ($\rho_v = 256$). Nous rapportons ses performances essentiellement pour comparaison à dimensions égales d'une méthode de représentation basée uniquement sur le texte.

La méthode DW+LSA permet de déterminer si la prise en compte au sein d'un unique modèle du graphe et du texte permettent effectivement d'améliorer les performances. Cette méthode ne permet néanmoins pas de générer de nouvelles représentations pour des documents autrement qu'en utilisant uniquement sa composante LSA, ce que nous rapportons déjà dans les résultats. Enfin, MATAN est l'unique modèle reposant sur des représentations des mots pré-entraînées, lui procurant un avantage certain. C'est pourquoi, en terme de performances de classification et de prédiction, nous étudions particulièrement les scores obtenus par GVNR-t et IDNE vis-à-vis de TADW et Graph2gauss, seuls modèles constituant réellement des plongements de documents en réseau comparables. Les tableaux 3.2 à 3.8 montrent les résultats des évaluations. Nous les analysons dans les sections suivantes.

Classification transductive

Nous observons plusieurs résultats intéressants sur la tâche de classification inductive. Premièrement, GVNR-t et IDNE sont tous deux les meilleurs modèles sur les réseaux de citations (Cora et CiteSeer) et sur les réseaux de questions et réponses en terme d'AUC. TADW réalise cependant de meilleures performances sur ces derniers réseaux en terme

TABLE 3.2 – Résultats expérimentaux obtenus sur le réseau Cora

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	13.85	13.70	12.66	12.83	12.53	51.51	51.96	51.60	51.22	51.22	11.80	48.84	49.87	50.27
TF	68.84	74.69	77.20	78.90	80.05	92.13	94.63	95.64	96.06	96.55	81.98	97.16	83.44	84.88
TF-IDF	72.56	76.99	80.13	80.66	81.84	93.11	95.24	96.38	96.74	97.20	**83.24**	**97.59**	85.17	85.02
LSA	72.46	77.22	79.92	80.56	81.16	93.65	95.47	96.63	96.95	97.17	81.26	97.35	87.17	88.63
DW+LSA	77.52	80.70	83.40	83.12	85.21	95.31	96.47	97.24	97.23	97.79	-	-	82.67	-
TADW	61.77	67.02	70.84	71.62	73.13	89.01	91.25	93.22	93.37	94.20	79.55	96.35	81.59	84.71
G2G	79.38	82.19	83.57	84.08	85.03	96.14	97.39	97.65	97.89	98.13	71.98	94.57	86.36	74.72
MATAN	75.48	76.65	77.58	79.20	78.30	95.19	95.79	96.23	96.41	96.69	76.13	94.91	82.72	71.47
GVNR-t	**82.20**	83.49	**85.26**	**85.82**	**86.67**	97.10	97.53	98.00	98.08	**98.48**	79.91	97.21	**94.31**	**92.44**
IDNE	80.41	**83.83**	84.79	84.67	86.06	**97.27**	**97.79**	**98.28**	**98.23**	98.45	82.25	97.57	92.90	88.56

TABLE 3.3 – Résultats expérimentaux obtenus sur le réseau New York Time

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	24.78	22.26	23.64	19.60	20.72	57.53	55.74	55.90	54.40	53.94	19.96	52.70	49.47	51.99
TF	72.76	75.93	76.21	76.94	77.41	91.97	92.87	93.35	93.24	93.67	77.59	93.80	54.64	54.06
TF-IDF	71.81	75.61	76.83	77.34	**78.44**	90.65	93.02	93.58	93.98	94.67	**79.42**	**94.92**	54.29	54.60
LSA	72.73	73.23	73.69	73.68	74.56	91.44	91.74	92.09	92.11	92.49	74.75	93.36	55.44	60.25
DW+LSA	**74.74**	**76.82**	**77.20**	**77.62**	78.29	**92.99**	**94.32**	**94.58**	**94.75**	**95.08**	-	-	96.28	-
TADW	74.61	74.98	74.78	74.44	74.00	93.25	93.52	93.43	93.41	93.20	69.49	90.49	53.85	65.55
G2G	69.07	71.34	71.25	71.71	71.92	90.30	91.72	91.61	91.85	92.02	65.25	88.07	**99.94**	68.30
MATAN	67.83	70.47	69.71	69.78	70.05	89.12	90.91	89.78	90.23	90.98	70.43	90.89	96.43	71.06
GVNR-t	65.74	66.32	66.18	66.30	66.81	88.26	89.44	89.25	89.58	89.75	57.12	81.89	99.84	66.51
IDNE	73.07	72.60	73.76	73.79	73.19	92.13	91.94	92.25	92.35	92.07	71.13	90.77	88.29	**79.04**

TABLE 3.4 – Résultats expérimentaux obtenus sur le réseau Gaming SE

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	5.96	5.48	5.47	5.36	5.32	58.98	56.86	54.96	53.57	52.42	4.93	51.74	49.91	50.66
TF	49.84	58.70	63.97	67.45	69.91	73.88	82.80	87.55	89.29	90.43	75.72	93.62	57.00	57.70
TF-IDF	44.45	56.61	64.99	69.18	72.44	77.46	75.35	83.40	84.18	87.97	**78.30**	91.60	56.61	58.60
LSA	58.62	59.47	61.00	61.72	62.06	85.64	89.02	89.28	90.21	90.62	63.72	90.80	59.25	58.31
DW+LSA	53.68	64.70	69.83	72.67	**75.07**	82.07	87.20	90.31	92.87	94.16	-	-	98.17	-
TADW	**69.44**	**70.44**	**72.20**	**73.58**	73.25	88.16	90.85	92.29	93.07	93.44	54.40	92.77	56.65	57.27
G2G	45.92	48.44	49.24	50.88	51.67	91.63	93.30	94.09	**95.01**	**95.67**	40.42	89.46	97.86	58.32
MATAN	47.03	48.83	50.30	51.47	52.12	91.70	93.59	**94.61**	94.89	95.48	55.08	**96.40**	70.59	60.43
GVNR-t	35.01	43.03	48.23	51.38	54.20	74.92	80.15	84.66	87.40	88.49	44.32	88.55	**98.94**	58.94
IDNE	57.70	59.48	59.42	59.16	59.74	**92.15**	**93.85**	94.37	94.68	94.85	59.94	95.35	68.33	**65.07**

TABLE 3.5 – Résultats expérimentaux obtenus sur le réseau Travel SE

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	7.37	6.67	6.82	6.69	6.72	57.76	53.78	54.04	52.96	52.70	6.16	51.30	49.66	50.30
TF	38.64	48.44	50.78	52.95	53.76	72.31	79.33	80.14	83.68	84.58	56.83	86.11	56.82	57.02
TF-IDF	29.79	45.05	49.75	**53.01**	**54.21**	69.99	74.99	74.08	76.18	80.47	**59.48**	87.31	57.46	60.72
LSA	48.58	51.13	50.16	51.25	50.20	80.93	84.16	82.19	84.20	84.41	53.10	85.72	58.56	56.90
DW+LSA	35.78	46.17	49.11	50.88	51.14	69.14	78.57	79.62	81.45	83.28	-	-	99.00	-
TADW	**49.32**	**52.21**	**51.01**	52.08	51.34	79.03	85.04	84.59	84.02	83.93	43.63	90.47	55.22	55.83
G2G	22.10	26.19	27.96	28.64	28.86	76.94	82.00	83.63	84.13	85.22	24.88	79.63	98.78	58.90
MATAN	28.67	33.73	34.86	36.25	36.76	82.96	86.56	**87.91**	**88.86**	**89.60**	40.00	**92.08**	72.80	**66.46**
GVNR-t	14.77	19.91	22.02	24.21	26.03	64.13	68.54	71.60	73.13	75.35	31.06	80.48	**99.14**	60.26
IDNE	39.09	40.96	40.51	41.77	40.93	**86.50**	**87.72**	87.77	88.58	88.64	42.28	90.28	62.90	59.16

TABLE 3.6 – Résultats expérimentaux obtenus sur le réseau CiteSeer

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	15.24	16.17	15.06	15.11	15.57	49.97	49.67	49.14	49.67	49.22	16.02	50.41	49.12	51.95
TF	66.42	68.69	69.80	71.03	71.45	89.70	90.94	91.52	92.25	92.72	71.20	93.06	88.52	86.08
TF-IDF	66.80	70.86	**72.73**	**73.25**	**73.57**	88.91	91.42	92.54	93.03	93.24	**74.70**	**93.96**	89.06	89.51
LSA	67.11	69.97	71.88	72.56	72.23	89.81	91.64	92.27	92.48	92.80	74.40	93.41	89.88	91.49
DW+LSA	60.04	65.53	67.81	68.84	70.27	85.47	88.82	90.48	90.91	91.87	-	-	81.19	-
TADW	54.50	59.40	61.43	62.75	63.04	83.12	85.79	87.44	88.36	88.57	70.18	92.70	84.70	86.39
G2G	67.91	69.38	70.94	71.85	71.61	90.42	91.73	92.48	92.73	92.93	63.25	89.56	89.77	79.53
MATAN	65.75	66.94	68.18	70.02	70.59	90.69	91.22	91.81	92.32	92.94	60.84	88.74	83.26	79.53
GVNR-t	70.37	**71.24**	71.30	72.47	72.39	92.25	92.73	92.66	93.41	93.46	69.52	92.63	82.83	90.21
IDNE	**70.49**	71.17	72.07	72.80	72.68	**92.49**	**93.22**	**92.92**	**93.71**	**93.67**	72.89	93.62	**93.85**	**92.96**

TABLE 3.7 – Résultats expérimentaux obtenus sur le réseau Stats SE

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	7.15	6.57	6.34	6.33	6.27	60.08	55.65	53.05	52.58	53.09	6.11	52.05	50.28	49.85
TF	44.73	51.09	53.49	54.63	56.33	79.06	83.43	87.04	87.16	87.74	58.04	88.81	67.13	67.31
TF-IDF	38.29	50.49	53.70	54.90	**57.74**	72.11	74.10	79.20	80.82	83.28	**61.18**	86.24	69.61	67.71
LSA	51.05	52.10	51.57	51.84	51.42	82.19	85.88	85.62	85.07	84.54	53.54	86.02	68.10	67.04
DW+LSA	41.11	48.78	50.72	51.98	53.89	70.68	76.90	81.23	81.34	82.60	-	-	98.00	-
TADW	**54.50**	**55.11**	**55.95**	**55.19**	54.88	82.09	85.17	86.39	85.18	85.69	44.79	89.88	60.46	62.59
G2G	29.58	32.11	33.70	35.06	35.71	83.59	86.87	87.85	88.41	89.16	32.23	82.69	97.79	68.78
MATAN	34.38	36.67	37.32	39.00	38.46	87.26	89.15	90.24	**90.58**	91.09	41.17	**93.22**	80.28	**72.57**
GVNR-t	19.51	23.15	25.55	28.04	29.54	68.53	70.72	75.49	73.65	77.37	32.89	81.08	**98.94**	69.18
IDNE	43.26	43.96	44.85	44.57	45.03	**89.20**	**90.09**	**90.73**	**90.58**	**91.16**	44.49	92.29	74.72	70.80

TABLE 3.8 – Résultats expérimentaux obtenus sur le réseau Academia SE

	TC										IC		TP	IP
	F1					AUC					F1	AUC	AUC	AUC
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	90%	90%	50%	90%
Aléatoire	8.11	7.52	7.38	7.36	7.36	59.38	56.01	53.77	53.03	53.24	7.71	51.83	49.86	49.62
TF	41.80	47.76	50.15	50.94	52.40	73.26	81.38	84.21	84.80	85.44	54.13	88.23	57.97	57.25
TF-IDF	38.17	47.87	49.74	**52.47**	**53.08**	65.94	74.05	76.33	80.08	82.45	**55.26**	87.27	58.48	58.39
LSA	48.48	49.09	48.96	48.05	48.23	82.29	82.92	85.38	84.24	85.47	49.64	87.45	57.16	**59.27**
DW+LSA	36.12	45.66	48.31	49.58	50.63	65.00	77.18	79.31	80.88	82.10	-	-	**99.30**	-
TADW	**49.24**	**50.89**	**50.20**	49.99	50.10	78.49	82.16	83.42	83.38	84.08	44.78	91.55	53.62	56.00
G2G	26.58	29.78	30.28	32.55	33.12	81.52	84.45	85.88	86.60	87.77	27.33	80.46	97.88	56.23
MATAN	32.54	35.37	35.10	37.28	37.58	86.19	88.22	**89.39**	**90.27**	**90.61**	40.27	**92.49**	60.94	58.84
GVNR-t	12.67	14.91	16.30	19.74	21.41	59.58	65.94	66.58	67.85	71.03	28.02	80.68	98.96	57.07
IDNE	38.15	40.82	39.89	40.91	40.71	**88.20**	**88.89**	89.28	89.68	90.14	43.51	91.75	59.32	57.70

de score F_1, ce qui signifie que ses représentations sont plus à même à engendrer des prédictions exactes même si les classements des *labels* produits par les probabilités estimées par la régression logistique sont mieux ordonnés avec les représentations de GVNR-t et IDNE. En particulier, à l'exception du score F_1 sur les réseaux de Q&A, IDNE réalise des scores toujours aussi bons ou meilleurs que les autres méthodes de plongement de réseau de documents. Deuxièmement, les représentations TF-IDF réalisent des bonnes et même souvent les meilleures performances lorsque le pourcentage de données d'entraînement est élevé, surtout en terme de score F_1, laissant supposer que le réseau permet de compenser la sémantique manquante lorsque peu d'annotations sont disponibles. Ce comportement est observé sur tous les jeux de données à l'exception de Cora et de Gaming SE. Enfin, MATAN réalise généralement de moins bons scores que les autres algorithmes de plongement de documents en réseau même si ce modèle se démarque sur les réseaux de Q&A, notamment en terme d'AUC.

On remarque donc que IDNE et GVNR-t sont de meilleures alternatives que TADW sauf en terme de score F_1 sur les réseaux de questions et réponses. D'une manière générale, si le pourcentage de données d'entraînement est élevé, les représentations TF-IDF constituent la meilleure approche. Sur NYT, un réseau dense de documents courts, la combinaison DW+LSA et TADW dominent largement les méthodes de plongement et de peu les méthodes basées uniquement sur le texte. Ceci fait écho aux résultats obtenus dans le chapitre précédent, où DeepWalk obtient très largement les meilleurs scores sur ce jeu de données. TADW étant une extension de DeepWalk, ceci confirme la capacité de ce dernier à capturer les caractéristiques particulières du réseau NYT, même si TADW diffère nettement en limitant la fenêtre à une taille de 2.

Classification inductive

En classification inductive, TF-IDF constitue la meilleure représentation, particulièrement en terme de score F_1. MATAN et IDNE sont généralement les meilleures selon l'AUC. Notamment, IDNE réalise d'aussi bonnes prédictions que TF-IDF sur les réseaux de citations. Ces résultats sont peu surprenants. Premièrement, TF-IDF est déjà une bonne représentation pour la classification transductive, lorsque les pourcentages d'entraînement sont élevés. Ici, le pourcentage est de 90% ce qui permet d'estimer des fréquences de termes proches de celles de l'ensemble du jeu de données. Les bons résultats observés en transduction s'observent alors en induction. Ensuite, IDNE et MATAN sont deux modèles de plongement de réseau de documents dont l'apprentissage est propice à l'induction, puisque si l'entraînement de leurs paramètres se fait grâce au graphe de connexions, la construction d'une représentation d'un document ne se fait qu'à partir du contenu textuel. Ces deux modèles se démarquent alors particulièrement des autres méthodes de plongement. Un résultat étonnant néanmoins est la faible performance de Graph2gauss. En effet, ce modèle est, comme IDNE et MATAN, pensé pour l'induction. Nous pensons que l'absence de représentations denses des mots dans ce modèle limite ses capacités de généralisation.

Prédiction transductive

La tâche de prédiction de liens transductive permet d'évaluer la capacité des algorithmes à reconstruire le réseau lorsqu'une partie des liens sont cachés. On observe que GVNR-t et Graph2gauss sont les deux algorithmes les plus performants sur cette tâche. IDNE réalise des bons scores sur les réseaux de citations alors que DW+LSA est performant sur les réseaux de questions et réponses. Les algorithmes basés uniquement sur le texte réalisent de nettement moins bonnes performances sur tous les jeux de données.

Prédiction inductive

La tâche de prédiction de liens inductive permet d'évaluer la capacité des algorithmes à induire la structure du réseau depuis le texte des documents. Sur cette tâche, IDNE réalise particulièrement de bonnes performances, réussissant toujours aussi bien et souvent nettement mieux que les autres approches. MATAN réalise les meilleurs scores sur les réseaux de questions et réponses à l'exception de Gaming SE où IDNE est meilleur. GVNR-t est encore un fois une bon algorithme pour les réseaux de citation.

Résumé des résultats quantitatifs

Pour résumer les résultats, nous rassemblons les observations suivantes :

— GVNR-t et IDNE réalisent de bonnes performances en classification transductive même si (1) TADW montre sur les réseaux de questions et réponses de meilleures performance en terme de score F_1 et (2) TF-IDF est une représentation difficile à battre sur les hauts pourcentages de données d'entraînement ;

— IDNE est la méthode la plus constante à travers les différentes tâches, réalisant souvent les meilleurs scores ou en étant proche lorsque ce n'est pas le cas ;

— la tâche de classification inductive semble ne pas bénéficier de l'information du réseau en terme de score F_1, où TF-IDF est systématiquement la meilleure approche. IDNE et MATAN sont cependant les meilleures approches selon l'AUC. Obtenir des probabilités bien calibrées (correspondant à une bonne AUC) est utile en recherche d'information ;

— en prédiction de lien transductive, GVNR-t et Graph2gauss sont les plus performants suivis par IDNE, qui est cependant la meilleure approche en prédiction inductive.

Dans la suite de cette section, nous allons étudier les hyperparamètres de IDNE avant de présenter des analyses qualitatives sur les mécanismes d'attention de IDNE et MATAN.

3.6.2 Étude des hyperparamètres de IDNE

Nous étudions ici le comportement de IDNE en fonction de ses hyperparamètres. Nous rapportons en figure 3.7 les performances obtenues par l'algorithme sur Cora en classification transductive en faisant varier le nombre d'itérations, le nombre d'échantillons négatifs, le nombre de thématiques n_t et la dimension des représentations ρ. Pour toutes les expérimentations, nous utilisons les hyperparamètres suivants (lorsqu'ils ne sont pas étudiés) : $n_k = 1$, $n_t = 32$, $\rho = 256$ et réalisons 5 000 itérations.

Nous observons que les représentations s'améliorent jusqu'à 5 000 itérations, après quoi les performances se stabilisent pour 10 000 et semblent même se détériorer au-delà. Concernant le nombre d'échantillons négatifs, nous observons le même effet que pour GVNR. Le choix $n_k = 1$ permet d'obtenir les meilleurs scores avec une légère baisse au delà. Le choix $n_k = 0$ produit de très mauvais scores. Plus le nombre de thématiques n_t est grand, meilleurs sont les résultats. Sur ce jeu de données, un plateau est atteint pour $n_t = 16$. Ce plateau varie grandement en fonction des jeux de données, notamment en fonction de la richesse du vocabulaire qui leur est associé. Pour nos expériences, nous avons empiriquement déterminé que $n_t = 32$ permettait de couvrir tous les jeux de données. Enfin, les performances s'améliorent systématiquement avec le nombre de dimensions.

3.6.3 Interprétabilité des thématiques de IDNE

Dans cette section, nous étudions la possibilité d'interpréter les représentations apprises par IDNE. Pour rappel, IDNE construit des représentations d_i des documents à

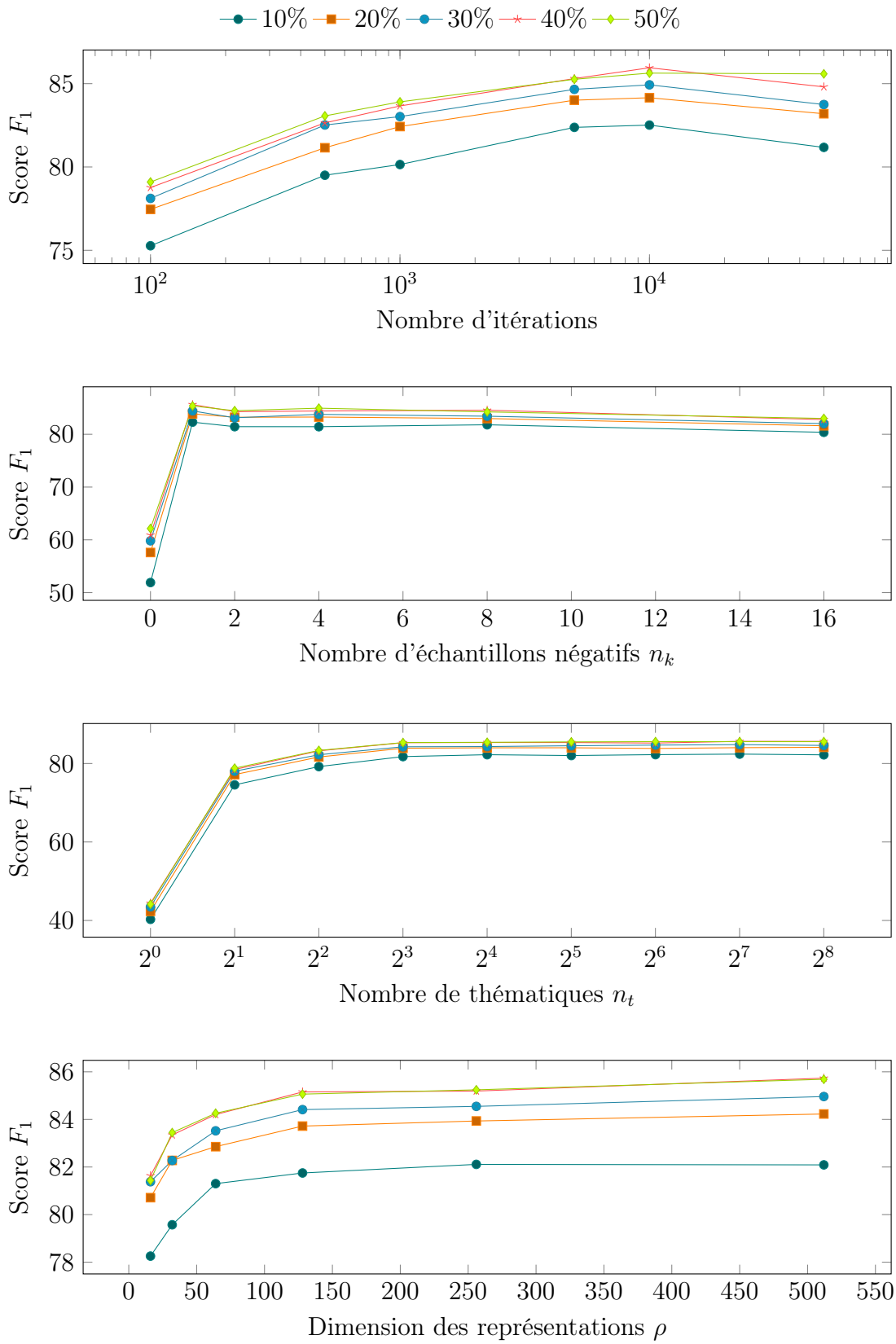


FIGURE 3.7 – Effets du nombre d'itérations et des hyperparamètres n_k, n_t et ρ sur les performances de IDNE pour la classification transductive des sommets du jeu de données Cora selon plusieurs pourcentages de données d'entraînement.

partir d'un ensemble de n_t vecteurs thématiques t_j et de n_ω vecteurs mots w_k. Un mot peut être considéré comme associé à une thématique si le produit scalaire de son vecteur avec le vecteur thématique est élevé, car selon le mécanisme d'attention thématique présenté figure 3.4, c'est ce produit scalaire qui détermine le poids d'attention de la thématique sur ce mot. Une fois IDNE entraîné sur un réseau de documents, on peut donc s'intéresser aux positions relatives, dans un même espace, des vecteurs documents D, des vecteurs thématiques T et des vecteurs mots W.

Pour ce faire, nous avons appliqué IDNE au jeu de données Cora avec un nombre de thématiques égal au nombre de *labels* associés aux sommets, c'est-à-dire $n_t = 7$ et en utilisant les mêmes hyperparamètres que dans les expérimentations précédentes. Le tableau 3.9a montre les 20 mots les plus proches des 7 thématiques. On voit que certaines thématiques semblent se concentrer sur des mots que l'on associerait à des *labels* telles que la thématique 7 décrivant le domaine de l'apprentissage par renforcement (*Reinforcement Learning*). Le tableau 3.9b montre les normes des mots appris par IDNE. On voit que les mots les plus discriminants en terme de communauté dans le réseau ont des normes particulièrement élevées, leurs permettant de générer plus facilement des poids d'attention élevés. Inversement, des mots communément utilisés à travers tout le réseau ont une norme deux ordres de grandeur en dessous, tels que « epochs » qui constitue un terme employé dans toutes les disciplines représentées dans ce réseau de citation.

Pour approfondir notre analyse, nous avons effectué une réduction de dimension conjointe de D, T et W afin de pouvoir les visualiser dans le plan. Pour cela, nous avons utilisé l'algorithme UMAP (*Uniform Manifold Approximation and Projection for Dimension Reduction*) [McInnes et al., 2018] sur les représentations apprises en dimension $\rho = 256$ considérant un nombre de voisins par individus de 30 et une distance minimale de 0.1. La figure 3.8 montre le résultat. Les points correspondent aux vecteurs documents D, colorés selon les classes auxquels ils appartiennent. Les grandes croix correspondent aux vecteurs thématiques et les petites croix correspondent aux mots les plus proches de chaque thématiques, tels que rapportés dans le tableau 3.9a.

Un premier constat est que les documents sont bien séparés selon leurs classes d'appartenances, même si les *labels* « Case Based », « Rule Learning » et « Theory » semblent plus difficiles à différencier. On observe les fortes corrélations entre la thématique 7 et la classe « Reinforcement Learning », la thématique 4 et la classe « Case Based » ainsi que la thématique 5 et la classe « Probabilistic Methods ». Les thématiques 1 et 3 sont toutes deux proches de la classe « Neural Networks » mais couvrent néanmoins deux régions bien distinctes, ce que se vérifie dans le tableau 3.9a, la thématique 3 étant axée sur les problèmes de séparation des sources et la 1 sur des problèmes plus généraux de classification. Cette dernière recouvre d'ailleurs aussi les *labels* plus théoriques « Rule Learning » et « Theory ». Enfin, on observe que la thématique 2, dont les mots décrivent les arbres de décision (« tree, decision, selection ») et les séries temporelles (« markov, radar, lband, seasonality »), forme principalement un îlot isolé. Seuls quelques uns des mots qui lui sont associés semblent être liés aux domaines « Genetic Algorithms » et aux domaines théoriques.

TABLE 3.9 – Étude des vecteurs thématiques et des vecteurs mots appris sur Cora avec IDNE en utilisant $n_t = 7$ thématiques.

(a) Thématiques et mots les plus proches produits par IDNE sur Cora. les *labels* dans ce jeu de données sont : « Case Based », « Genetic Algorithms », « Neural Networks », « Probabilistic Methods », « Reinforcement Learning », « Rule Learning » et « Theory ».

Thématique	Mots les plus proches
1	pattern, features, classification, regulate, machine, limitations, domains, datasets, methods, patterndirected, dataset, connectionist, backpropagation, pythia, accuracy, classifier, realworld, exemplar, symbolic, links
2	tree, decision, selection, trees, markov, radar, subsurface, moisture, cband, ers, cedar, sar, lband, polarized, minnesota, lter, ersjers, creek, raco, seasonality
3	saturation, separation, stabilization, gradient, blind, asymptotic, universal, matrix, square, signals, necessarily, sign, projection, approximated, bandpass, descent, norm, cubic, subproblems, speakers
4	design, casebased, pac, sme, analogical, mcmc, structuremapping, planning, wellunderstood, retrieval, reasoning, mistakes, case, reuse, retrieving, instance, chain, analogy, convolution, solving
5	mcmc, execution, speculation, ga, parallelism, genetic, instruction, aimed, dirichlet, instructions, chain, consensus, substantive, sequences, macroscopic, issue, processor, recipient, analogy, nerve
6	accuracy, learning, experiments, machine, ilp, datasets, comprehensibility, sufficed, sbc, inductive, accurate, decompose, warehouses, mining, induction, dataset, gratefully, attribute, assign, challenges
7	reinforcement, mdp, mdps, qlearning, policy, crosses, actions, value, reward, policies, brigade, rl, macros, relax, hinders, dynamic, satisficingoptimizing, action, functionings, barto

(b) Mots dont les représentations produites par IDNE ont les normes les plus élevées et les plus basses sur le jeu de données Cora.

Plus élevées	genetic (13.72), reinforcement (13.32), evolutionary (11.08), mdp (10.36), bayesian (10.17), mcmc (9.87)
Plus basses	counterexamples (0.19), lecun (0.19), it (0.19), epochs (0.19), shavlik (0.19), cdc (0.18)

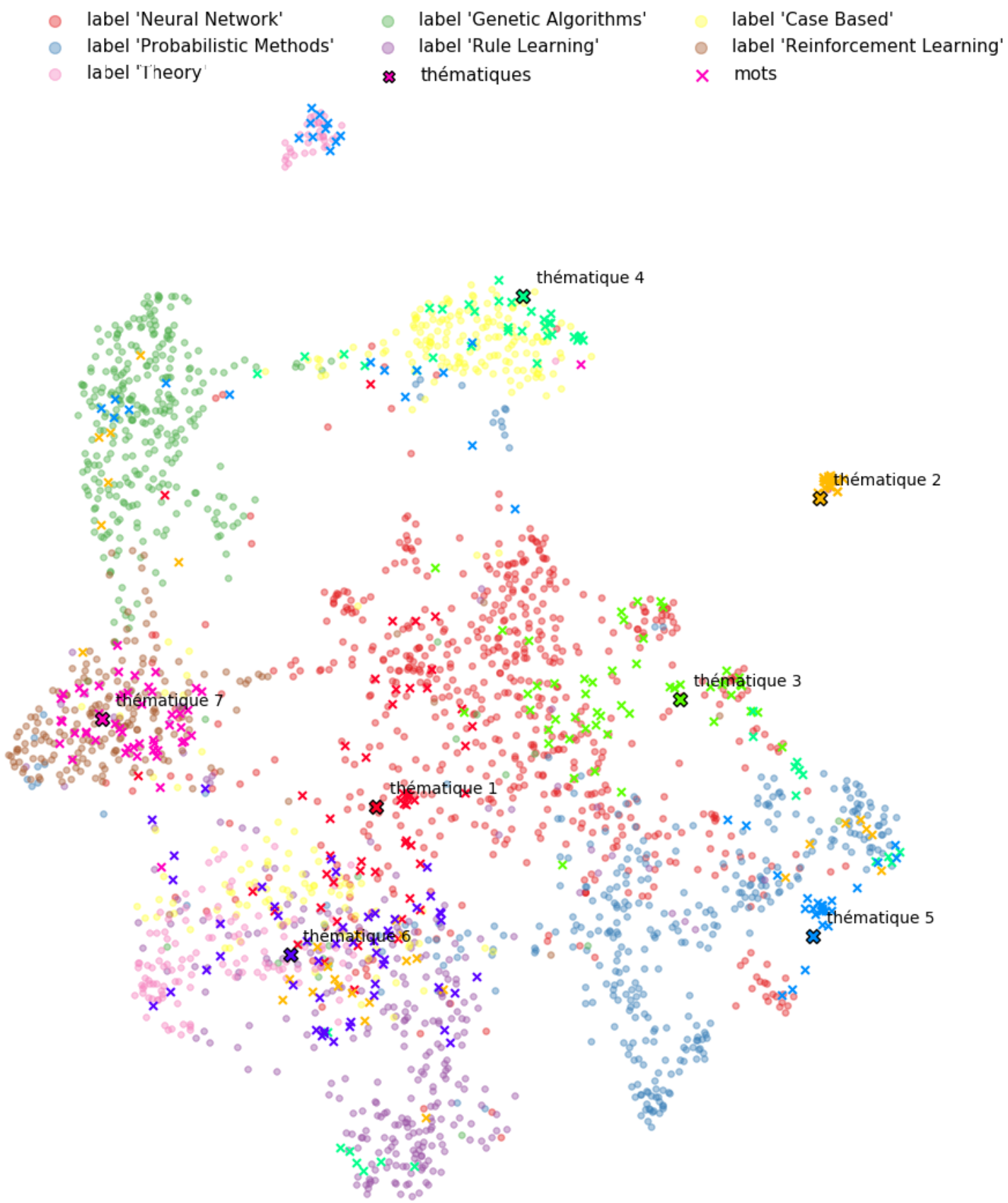


FIGURE 3.8 – Visualisation en deux dimensions des représentions des documents, des thématiques et des mots obtenues par IDNE sur Cora avec les *labels* associés.

3.6.4 Explicabilité avec MATAN

Dans cette section, nous cherchons à expliquer les liens d'un réseau en utilisant le modèle MATAN. Pour rappel, ce modèle génère des représentations de paires de documents, où la représentation d'un document d_i est obtenue par projection $w_j P^V$ puis moyenne pondérée des vecteurs mots de l'autre document. Cette pondération est effectuée grâce à un mécanisme d'attention générant des poids selon la formule :

$$\alpha^i = \text{softmax}(\frac{Q^i {K^j}^\top}{\sqrt{\rho_d}}), \tag{3.28}$$

où Q^i sont les représentations requêtes des mots du premier document, K^j les représentations clefs des mots du second document et ρ_d la dimension de toutes ces représentations. Il faut noter que cette fonction d'attention n'est pas symétrique, signifiant que les poids ne sont pas les mêmes si l'on considère plutôt le document d_j contextuellement à d_i :

$$\alpha^j = \text{softmax}(\frac{Q^j {K^i}^\top}{\sqrt{\rho_d}}). \tag{3.29}$$

En notant ℓ_i et ℓ_j les longueurs respectives des documents d_i et d_j, on a alors $\alpha^i \in \mathbb{R}^{\ell_i \times \ell_j}$ et $\alpha^j \in \mathbb{R}^{\ell_j \times \ell_i}$ où chaque valeur α^i_{kl} indique la force d'association entre le mot ω_k du premier document et le mot ω_l du second document, spécifiquement à cette paire de document. Pour expliquer la force du lien entre deux documents, on peut alors s'intéresser à la somme de ces poids d'attention $\alpha = \alpha_i + {\alpha_j}^\top$. De la sorte, chaque élément α_{kl} représente à quel point la paire de mots ω_k et ω_l a contribué à la construction mutuelle des représentations de u_i et u_j des deux documents.

Les figures 3.9 et 3.10 montrent les valeurs de ces poids d'attention obtenu avec un même document d_i contextuellement à deux documents différents d_j. Par soucis de lisibilité, nous n'avons gardé que les 30 premiers mots de chaque document afin de calculer les poids d'attention α. De plus, nous montrons ces 30 premiers mots en les colorant selon la somme de leurs contributions, définie comme la somme des valeurs leurs étant associées dans α.

Dans la première figure, on observe que ce sont principalement les mots « decision », « lookup » et « balance » qui produisent de forts poids d'attention avec les mots « processing », « neural » et « network » du second document. La paire de mots « decision | neural » est notamment celle qui induit le plus fort poids, montrant que l'aspect d'apprentissage est est le facteur qui explique le mieux le lien entre ces documents. Pour la seconde paire de documents, c'est la paire « balance | mixed » qui induit le plus fort poids, suivi des paires « decision | neural » et « neural | accuracy ». Pour cette seconde paire de documents, c'est donc plutôt l'aspect de balancement des données qui est mis en avant. Les poids d'attention permettent ainsi de rapidement identifier les axes expliquant les différents liens d'un document selon les mots qui le composent.

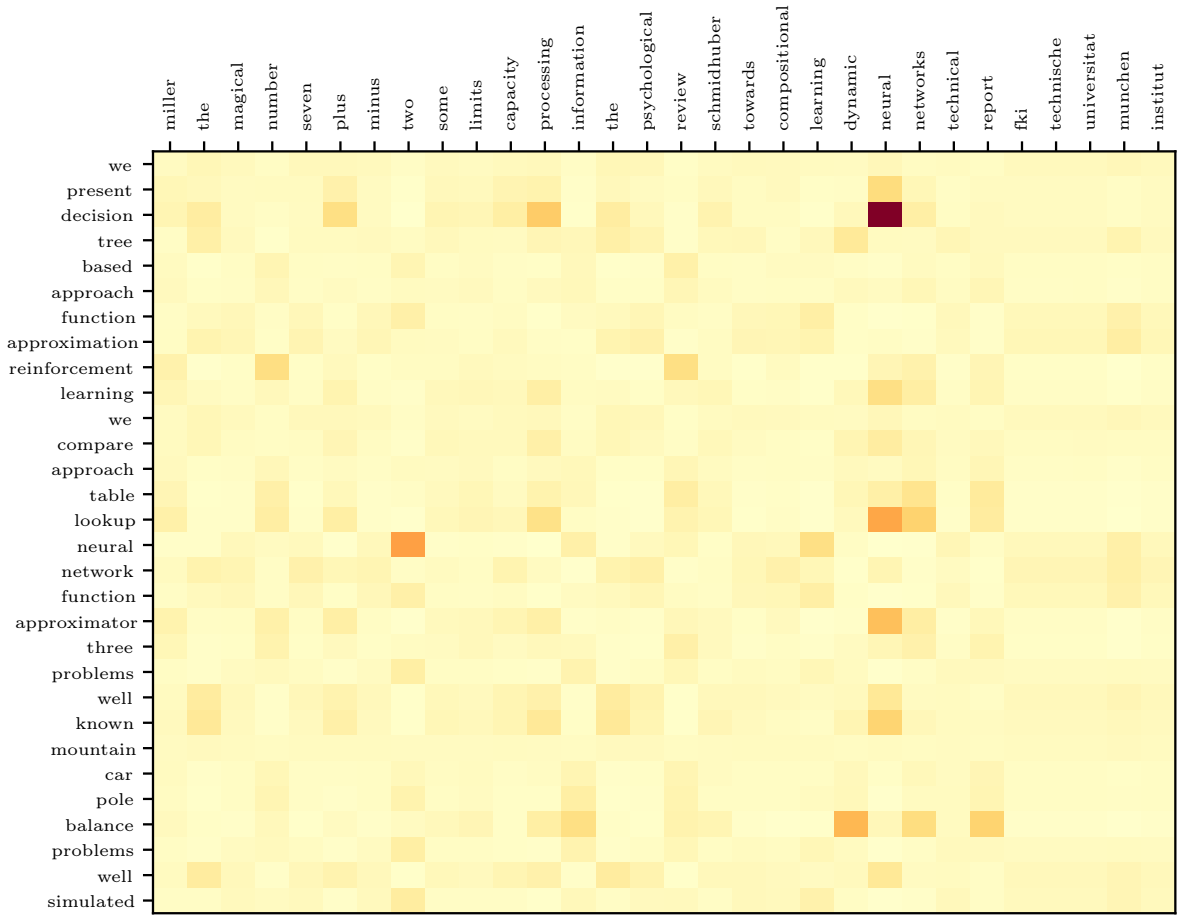
3.6.5 Résumé des résultats expérimentaux

Nous avons vu que, malgré la simplicité de cette extension, GVNR-t constitue une approche efficace de plongement de réseau de documents, sauf pour la classification inductive. IDNE est le seul modèle qui présente une constance dans ses performances, réalisant de bons scores sur toutes les tâches et sur tous les jeux de données. MATAN n'est pas une bonne méthode pour une approche transductive mais montre de bonnes performances en induction, aussi bien en classification de sommets qu'en prédiction de liens. De plus, nous avons qualitativement montré que IDNE produit des représentations interprétables

we present decision tree based approach function approximation reinforcement learning we compare approach table lookup neural network function approximator three problems well known mountain car pole balance problems well simulated

miller the magical number seven plus minus two some limits capacity processing information the psychological review schmidhuber towards compositional learning dynamic neural networks technical report fki technische universitat munchen institut

(a) Contributions globales des mots des documents dans la construction de leurs représentations.



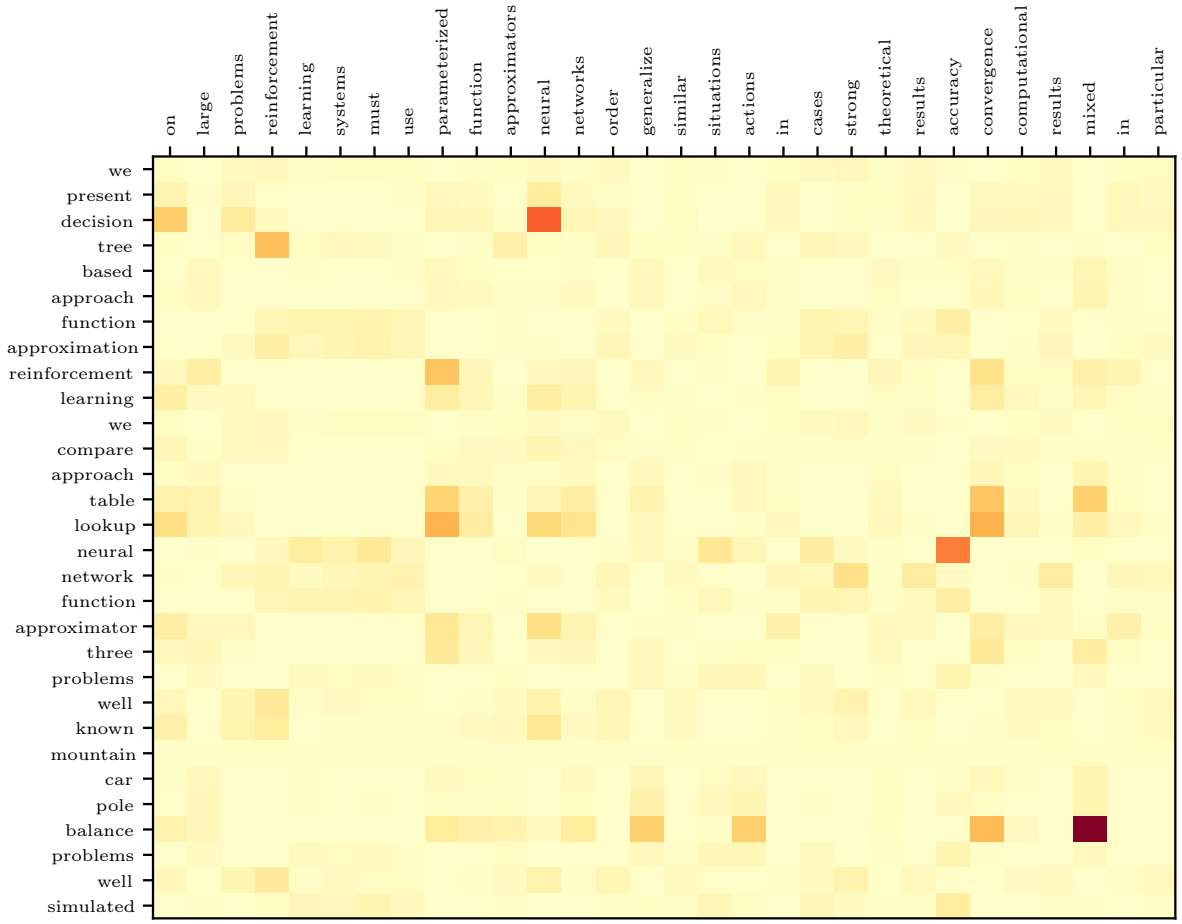
(b) Poids entre chaque paire de mots des deux documents.

FIGURE 3.9 – Visualisation des poids d'attention mutuelle générés par MATAN sur une paire de documents extraite du jeu de données Cora (1).

we present decision tree based approach function approximation reinforcement learning we compare approach table lookup neural network function approximator three problems well known mountain car pole balance problems well simulated

on large problems reinforcement learning systems must use parameterized function approximators neural networks order generalize similar situations actions in cases strong theoretical results accuracy convergence computational results mixed in particular

(a) Contributions globales des mots des documents dans la construction de leurs représentations.



(b) Poids entre chaque paire de mots des deux documents.

FIGURE 3.10 – Visualisation des poids d'attention mutuelle générés par MATAN sur une paire de documents extraite du jeu de données Cora (2).

des mots, thématiques et documents ce qui peut faciliter l'exploration d'un réseau de documents. Enfin, la lecture des poids d'attention produits par MATAN permettent de facilement expliquer les facteurs, en terme d'associations de mots, qui ont induit l'existence de liens dans un réseau de documents.

3.7 Conclusions et perspectives

Dans ce chapitre, nous avons présenté trois contributions dans le domaine du plongement de documents en réseau : (1) GVNR-t est une extension de notre modèle de plongement de réseau qui intègre des représentations des mots des documents dans sa factorisation, (2) MATAN s'appuie sur un mécanisme d'attention mutuelle permettant d'identifier les mots d'un document qui expliquent l'existence d'un lien avec un second document et (3) IDNE met en œuvre un nouveau mécanisme d'attention thématique sur le texte et construit des représentations des mots et des thématiques associés à un corpus de documents en tirant profit de la structure du réseau qui les lie.

Nous avons évalué ces modèles sur des tâches de classification de sommets et de prédiction de liens dans des configurations transductives et inductives. Ceux-ci sont compétitifs vis-à-vis des méthodes de l'état de l'art et IDNE montre des performances constantes sur toutes les tâches et tous les jeux de données. Nous avons aussi étudié qualitativement les mécanismes d'attention mis en œuvre dans IDNE et MATAN et avons montré leurs capacités d'interprétation et d'explicabilité.

Pour approfondir nos travaux, plusieurs aspects de nos contributions auraient besoin d'être explorés :

— **l'attention thématique :** nous proposons ce mécanisme comme un moyen efficace d'intégrer l'information textuelle dans un modèle de plongement de documents en réseau. Néanmoins, ce mécanisme pourrait être mis en œuvre pour aborder d'autres tâches, notamment en traitement automatique du langage. On peut en outre imaginer d'utiliser ce mécanisme en conjonction du modèle BERT [Devlin et al., 2018] pour une tâche de classification supervisée de documents. Il serait alors intéressant d'étudier les thématiques apprises comme nous l'avons fait en section 3.6.3 ;

— **l'impact de la similarité :** comme nous l'avons signalé en conclusion du chapitre précédent, le choix de la fonction de similarité approchée par les méthodes de plongement de réseau peut avoir de nombreux effets. Dans le cas de IDNE, nous avons vu que les thématiques apprises reflètent cette similarité qui, dans notre cas, reflète elle-même les communautés du réseau. Un axe à explorer serait de tester d'autres formes de similarité. Par exemple, dans un réseau de citation, peut-on identifier les thématiques reflétant le positionnement d'un article scientifique vis-à-vis des autres articles ? Au sein d'une communauté bien spécifique, s'agit-il d'un article théorique, d'une analyse empirique, d'une extension d'un modèle ou d'une étude de reproductibilité ? Pour ce faire, il peut être intéressant de calculer la matrice S avec une mesure de similarité qui capture les rôles, tel que SimRank, ou en utilisant des marches aléatoires biaisées comme avec Node2vec ;

— **l'interprétabilité :** l'un des succès des mécanismes d'attention, outre leur efficacité dans divers domaines, est leur lisibilité. En effet, comme pour IDNE et MATAN, le calcul explicite de poids d'attentions guidant la construction des représentations permet d'interpréter ces dernières. Dans le *Transformer*, ces poids sont construits de manière dense à cause de l'utilisation de la fonction softmax, c'est-à-dire que toutes les combinaisons possibles de paires de mots génèrent des poids non nul. Dans IDNE, même si l'utilisation de la fonction ReLU engendre des poids nuls, rien ne favorise particulièrement ce phénomène. Ceci engendre deux pro-

blèmes. Premièrement, la complexité en temps de l'apprentissage des paramètres est grande puisque qu'il faut rétropropager le gradient pour l'ensemble de ces paires. De plus, l'interprétation de ces poids en est plus difficile. Avec IDNE, l'utilisation de $n_t = 32$ thématiques rend la distribution des poids d'attention pour chaque document difficilement lisible. Pour contrer ces problèmes, de récents travaux proposent des alternatives aux fonctions ReLU et softmax, telle que sparsemax [Martins and Astudillo, 2016] généralisée dans [Niculae and Blondel, 2017]. Ces techniques résolvent les deux problématiques à la fois en générant des poids d'attention creux, et en ne propageant le gradient que pour les poids non nuls.

Les algorithmes de plongement de réseau de documents permettent d'aborder efficacement les tâches de classification des documents et de prédiction des liens. Dans le chapitre suivant, nous nous intéressons à l'application de ces méthodes pour une tâche particulière, la recherche d'experts.

Chapitre 4

Cas d'Application : la Recherche d'Experts

Dans ce chapitre, nous explorons une tâche particulière : la recherche d'experts. Nos travaux réalisent une première ébauche vers la mise au point d'un nouveau protocole d'évaluation pour cette tâche et étudient l'applicabilité des méthodes de plongement de réseau de documents pour celle-ci. Nous abordons en premier lieu l'état de l'art de ce domaine, notamment les méthodes d'évaluation et les algorithmes proposés dans la littérature. Nous présentons alors une nouvelle méthode d'évaluation pour laquelle nous fournissons quatre jeux de données que nous comparons à une approche traditionnelle. Nous présentons aussi nos travaux sur l'application des méthodes de plongement de réseau de documents (DNE) pour la recherche d'experts. Nous rapportons ensuite les résultats d'expériences permettant de (1) juger la pertinence de notre méthode d'évaluation, (2) identifier les meilleurs algorithmes selon différentes méthodologies et (3) étudier les performances des algorithmes de DNE selon différentes façons de les appliquer. Enfin, nous concluons ce chapitre en identifiant les axes de recherche possibles pour combler l'écart entre les techniques de DNE et les systèmes de recherche d'experts.

Sommaire

4.1 Introduction

Nous abordons dans ce chapitre un cas d'application particulier de la recherche d'information : la recherche d'experts. Le choix de cette tâche s'explique de deux façons. Premièrement, la thèse décrite dans ce manuscrit s'inscrit en collaboration avec une entreprise, DSRT, dont l'un des produits qu'elle a développés est un système d'aide à la recherche de relecteurs pour la publication scientifique. Dans ce cadre, nous avons proposé une solution accessible au grand public [Brochier et al., 2018b], dénommée *Peerus Review*, reposant sur un modèle de la littérature. Deuxièmement, la recherche d'experts s'appuie très souvent sur des données composées de réseaux de documents. Il est ainsi naturel de chercher à utiliser des techniques de plongement de réseau de documents pour améliorer voire construire des nouveaux algorithmes abordant cette tâche. Les travaux présentés dans ce chapitre concernent deux aspects de la recherche d'experts : (1) les données et les méthodes d'évaluations de cette tâche et (2) l'applicabilité des méthodes de plongement de réseau de documents pour sa résolution.

L'expertise est un concept flou, difficile à formaliser. De nombreux travaux cherchent à construire des algorithmes de recherche d'experts dans de grandes bases de données pour divers domaines d'applications. Il est commun de considérer l'expertise comme un savoir latent, que des personnes portent et partagent de plusieurs façons. La recherche d'experts consiste alors à identifier ce savoir à travers des artefacts manifestes, tels que les communications, les actions et les interactions mises en œuvre entre ces personnes.

À titre d'exemple, l'évaluation par les relecteurs est un processus scientifique par lequel les experts d'une discipline vérifient la qualité du travail de leurs pairs. L'examen et la validation des travaux scientifiques est une pierre angulaire de la recherche scientifique. Le nombre croissant de publications journalières dans un contexte académique de plus en plus compétitif (publier ou périr) et la digitalisation du monde de l'édition légitiment le développement d'outils informatiques d'aide au processus de relecture. Lorsqu'un chercheur propose un article à un éditeur, ce dernier peut être en charge de trouver un certain nombre de relecteurs. Le temps nécessaire pour trouver ces relecteurs constitue le principal goulot d'étranglement de l'édition scientifique, retardant parfois la date de publication d'un article de plusieurs mois. Pour accélérer ce processus, un système d'information peut s'appuyer sur le vaste réseau de publications scientifiques, partiellement accessible sur le Web, afin de trouver les chercheurs travaillant sur la même thématique de recherche que l'article en relecture. Il existe de nombreux autres cas d'application pour les algorithmes de recherche d'experts. Parmi eux, les sites de question et réponses (Q&A) rassemblent des utilisateurs autour de plateformes collaboratives traitant de thèmes spécifiques. Pour les développeurs informatique, *Stack Overflow*[1] est devenu un outils de travail presque indispensable. Lorsqu'un utilisateur pose une question, le reste de la communauté y répond et les meilleures réponses sont identifiées grâce à un système de vote collaboratif. Seulement, face à l'étendue des thématiques abordées et au nombre élevé de questions posées, il est nécessaire de présenter aux utilisateurs en priorité les questions auxquelles ils sont susceptibles de répondre, selon leurs domaines d'expertises.

Formellement, un système de recherche d'experts prend en entrée une requête textuelle et produit en sortie un classement de candidats. Pour ce faire, il s'appuie sur un réseau candidat-document en utilisant les documents liés aux candidats comme expressions de l'expertise de ces derniers. L'évaluation d'un algorithme de recherche d'experts nécessite d'avoir un ensemble vérité terrain de candidats liés à plusieurs domaines d'expertises puis de confronter cet ensemble aux classements produits par l'algorithme pour différentes requêtes. La plupart des travaux réalisés dans la recherche d'experts utilisent, comme requête, l'intitulé du domaine d'expertise. Ainsi, si un jeu de données possède certains

1. `https://stackoverflow.com/`

experts dans le domaine du *data mining*, l'évaluation du classement des candidats produit par l'algorithme est réalisé selon la requête textuelle « data mining ». Cependant, ces requêtes ne sont pas représentatives des applications réelles citées précédemment pour deux raisons majeures :

— il arrive souvent que différents utilisateurs proposent différentes formulations même s'ils cherchent la même chose. Le terme « data mining », couramment utilisé en français, possède plusieurs synonymes tels que « analyse de données » ou « fouille de données ». Cet aspect de synonymie affecte la qualité de l'algorithme de recherche d'experts qui n'est pas capturé par les méthodes d'évaluations classiques ;

— on cherche rarement des experts dans des domaines aussi larges. Si une entreprise cherche un nouveau collaborateur, il est plus probable que le recruteur fournisse une description détaillée du poste à pourvoir plutôt qu'un unique terme générique tel que « développeur ». La granularité des thématiques d'expertises est un aspect important pour la bonne évaluation d'un algorithme.

Pour palier ces problèmes, nous proposons en section 4.3.1 un nouveau protocole d'évaluation reposant sur un ensemble de requêtes directement extraites du réseau candidat-document, fournissant ainsi une large diversité de requêtes pour chaque domaine d'expertise. De plus, nous proposons l'utilisation de nouveaux jeux de données issus de plateformes de Q&A possédant un riche ensemble de thématiques d'expertise ainsi qu'un jeu de données issu d'un réseau de citations scientifiques que nous avons manuellement annoté.

Enfin, les travaux présentés dans le chapitre précédent abordent les méthodes d'apprentissage de représentation pour les documents en réseau. Leur application directe à la recherche d'experts n'est pas triviale puisqu'elles ne sont pas conçues pour des réseaux hétérogènes, en particulier les réseaux bipartis de type candidat-document. Nous proposons en section 4.3.2 d'utiliser ces techniques comme représentations des documents pour des algorithmes de l'état de l'art en recherche d'experts ainsi que d'étudier deux méthodes de traitement du réseau candidat-document pour appliquer ces techniques à la recherche d'experts de manière *ad hoc*. Dans la prochaine section, nous présentons tout d'abord l'état de l'art de l'évaluation et des algorithmes en recherche d'experts.

4.2 Travaux connexes

Nous présentons tout d'abord dans cet état de l'art les protocoles d'évaluation mis en œuvre pour la recherche automatique d'experts. La définition de cette tâche et les données qui lui sont associées pouvant fortement varier dans la littérature, il est nécessaire de définir formellement le cadre dans lequel s'inscrivent nos travaux. Nous présentons ensuite les travaux majeurs dans la résolution de cette tâche.

4.2.1 Protocoles d'évaluation

Le concept de recherche d'experts peut couvrir un large éventail de tâches. Le principe fondamental de la recherche d'expertise est la recherche de candidats à partir d'une requête. Pour faire correspondre ces deux objets, un algorithme s'appuie sur des données afin de relier l'espace de sortie, un classement de candidats, avec l'espace d'entrée qui est souvent un contenu textuel. Cependant, de nombreux types de données peuvent être envisagés pour résoudre cette tâche. Afin de comparer équitablement des algorithmes, il faut choisir une structure fixe pour ces données qui puisse refléter des cas d'application réels. Si les méthodes supervisées bénéficient de domaines d'expertise associés aux candidats, elles sortent du cadre de nos travaux qui se concentrent uniquement sur les

méthodes non supervisées. Notre objectif est de comparer des méthodes qui ne nécessitent pas d'annotations parfois coûteuses.

Les premiers travaux en recherche d'experts [Craswell et al., 2005] considèrent généralement un petit ensemble de requêtes thématiques. Les dénominations de ces thématiques (telles que « data mining ») sont directement utilisées pour rechercher une liste d'experts parmi les candidats en tirant parti d'une collection de documents qu'ils ont publiés (par exemple des courriels ou des articles scientifiques). Ce type d'évaluation est utilisé dans plusieurs jeux de données publics [Zhang et al., 2007b, Macdonald et al., 2007, Mislevy and Riconscente, 2011]. La figure 4.1 présente un jeu de données hypothétique représentant les candidats C, les documents D, les requêtes thématiques L et les annotations que l'on retrouve dans ce type d'évaluation.

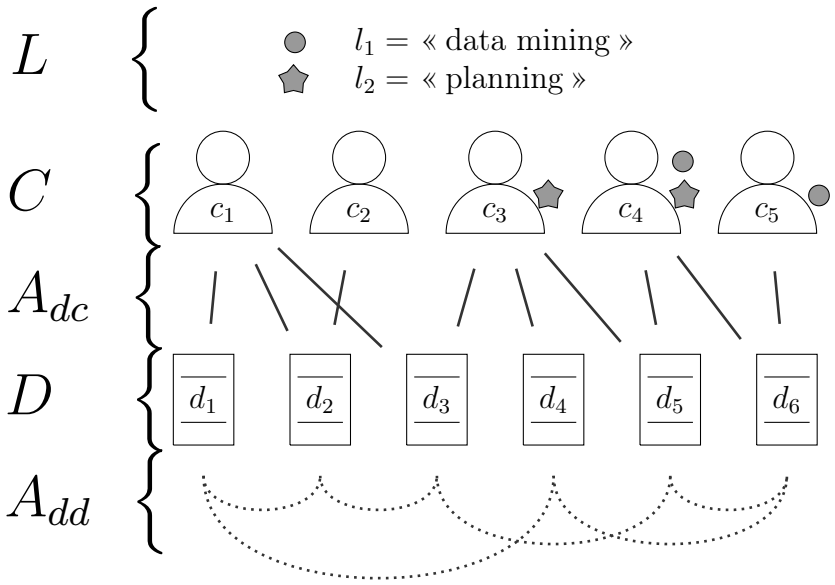


FIGURE 4.1 – Exemple hypothétique d'un jeu de données d'évaluation de recherche d'experts. Cinq candidats sont auteurs de six documents. Les 6 documents sont reliés les uns aux autres par citations dans un corpus de publications scientifiques. Parmi les candidats, c_3 et c_4 sont experts en « planning » (étoiles) et c_4 et c_5 sont experts en « data mining » (cercle). Dans la méthode traditionnelle d'évaluation, avec requêtes thématiques, un algorithme produit un classement de candidats pour chacune des requêtes l_1 et l_2. Par exemple, un algorithme parfait pourrait générer les classements $l_1 \mapsto c_4c_5c_1c_2c_3$ et $l_2 \mapsto c_4c_3c_1c_5c_2$.

Les données sur lesquelles s'appuie un algorithme sont constituées dans cet exemple des ensembles C et D ainsi que des matrices d'adjacence A_{dc} et A_{dd} décrivant leurs interactions. Les thématiques L sont utilisées pour évaluer les algorithmes dont on espère qu'ils produiront de meilleurs scores pour les candidats associés à ces thématiques (les experts) qu'aux autres candidats. La mesure de la qualité des classements produits par les algorithmes se fait généralement en calculant l'aire sous la courbe ROC (AUC), la précision moyenne (AP) ou la précision jusqu'à un certain rang (P@rang).

Plus récemment, le concept de recherche d'experts a fusionné avec le concept plus large de recherche d'entités [Balog et al., 2010]. Alors que des données de plus en plus complexes et interconnectées sont produites sur le Web, la recherche d'experts devient une application particulière de la recherche d'entités. Dans le même temps, les plateformes de Q&A génèrent et rendent accessibles au public un grand nombre de questions avec des réponses d'experts, collaborativement évaluées par leurs utilisateurs. Plusieurs travaux traitent de la recherche d'experts dans ces sites Web [Riahi et al., 2012, Zhao et al., 2014]. Souvent, la tâche considérée consiste à trouver la liste exacte des utilisateurs qui

ont répondu à une question spécifique ou à classer des réponses en fonction des votes des utilisateurs. Dans le premier cas, l'évolution des utilisateurs dans le temps est à considérer et, dans le second cas, la qualité intrinsèque d'une réponse devient un facteur déterminant. Ces deux aspects dépassent le cadre des nos travaux. Pour plus de détails, [Yuan et al., 2020] examine plusieurs modèles de recherche d'experts dans les sites Web Q&A et décrit les récents progrès accomplis.

Par la suite, nous introduirons une nouvelle méthodologie d'évaluation de recherche d'experts [Brochier et al., 2018a] dans laquelle les requêtes sont des documents associés aux thématiques. Une requête est donc l'un des documents D (par exemple un article scientifique) pour lesquels nous visons à retrouver certains experts des sujets qui y sont décrits. Cette configuration reflète de nombreux scénarios réels tels que la recherche automatique de relecteurs scientifiques, la recommandation d'utilisateurs experts sur les sites de Q&A ou encore l'identification de profils intéressants pour des offres d'emploi. Dans la section suivante, nous décrivons quelques travaux importants concernant les algorithmes de recherche d'experts.

4.2.2 Algorithmes de recherche d'experts

La recherche automatique d'experts est un domaine abordé très tôt par la communauté scientifique, en particulier en recherche d'information, dès l'émergence du Web dans les années 2000. Nous retraçons ici certains des travaux majeurs dans ce domaine.

Premiers modèles

L'automatisation de la recherche d'expertise est apparue avec la croissance de bases de données liées aux grandes organisations. La digitalisation des ressources dans les librairies et entreprises ont amené la communauté scientifique à s'intéresser au problème de la recherche d'experts à travers un ensemble de documents numériques indexés. L'un des premiers systèmes mis au point pour répondre à cette problématique est P@noptic [Craswell et al., 2001]. Il consiste en un service web reposant sur les données de l'intranet de *CSIRO Mathematical and Information science (CMIS)* capable de recommander des experts en fonction d'une requête composée de plusieurs termes. L'algorithme sous-jacent crée un méta-document pour chaque employé de l'organisation en agrégeant l'ensemble des documents qui lui sont liés. La recherche d'experts devient alors un problème de recherche d'information classique, classant les candidats selon la similarité de leurs méta-documents avec une requête donnée.

Selon Balog *et al.* [Balog et al., 2012], c'est en 2005 que la communauté scientifique s'intéresse en profondeur à la recherche d'experts avec l'apparition de *TREC-2005 Enterprise Track, Expert search task*. La conférence *Text REtrieval* a proposé un jeu de données issu du *World Wide Web Consortium (W3C)* composé de 331 037 documents de plusieurs types, majoritairement des contenus d'emails. Un ensemble de 50 requêtes pour lesquelles 1 092 candidats sont annotés comme experts est fourni ainsi que les codes sources pour l'évaluation des résultats. Plusieurs équipes proposent alors différentes approches à la recherche d'experts et une formalisation plus précise du problème émerge [Craswell et al., 2005].

Modèles génératifs

Une grande partie des approches proposées dans la littérature repose sur des modèles génératifs qui cherchent à caractériser le processus selon lequel une requête est générée par un candidat via un ensemble support de documents. Classer des candidats selon une requête revient à estimer la probabilité $p(c|q)$ d'un candidat c d'être expert de la requête

q déterminant ainsi le classement des candidats. La difficulté de la recherche d'experts consiste à estimer précisément cette probabilité, à laquelle on peut appliquer le théorème de Bayes :

$$p(c|q) = \frac{p(q|c)p(c)}{p(q)} \overset{\text{rang}}{=} p(q|c)p(c), \tag{4.1}$$

où $p(q|c)$ est la probabilité que la requête q ait été générée par le candidat c, $p(c)$ est la probabilité *a priori* que c soit un expert et $p(q)$ est la probabilité de la requête, constante et donc n'influençant pas le rang d'un candidat.

Dans [Balog et al., 2006], deux modèles sont proposés pour estimer $p(q|c)$. Le premier, appelé *modèle-candidat*, représente un candidat par une distribution multinomiale $p(\omega|\theta_c)$ sur le vocabulaire des mots ω contenus dans les documents qui lui sont associés. Ce modèle ressemble à l'approche proposée avec *P@noptic* en créant directement un modèle θ_c pour chaque candidat. En supposant l'indépendance entre les mots de la requête, il calcule la probabilité de la requête :

$$p(q|c) = \prod_{\omega \in q} p(\omega|\theta_c)^{n(\omega,q)}, \tag{4.2}$$

$n(\omega, q)$ étant le nombre d'apparitions du mot ω dans q. Le second modèle, appelé modèle-document, calcule indirectement à travers les documents la probabilité d'un expert d'avoir généré la requête telle que $p(q|c) = \sum_d p(q|d, c)p(d|c)$ qui peut être réécrit, sous l'hypothèse que la requête et les candidats sont indépendants étant donné un document :

$$p(q|c) = \sum_d p(q|d)p(d|c), \tag{4.3}$$

où $p(q|d) = \prod_{\omega \in q} p(\omega|d)^{n(\omega,q)}$ est une mesure de similarité entre la requête et le document et où $p(d|c)$ indique la force avec laquelle le document d représente c (ou la force d'association entre d et c). Ce dernier terme peut être calculé de différentes façons à partir d'un ensemble d'associations candidat-document définies entre 0 (aucun lien) et 1 (lien très fort), notamment en normalisant selon les documents (document-centré) ou selon les candidats (candidat-centré).

Le modèle-document présente quelques avantages sur son concurrent. Tout d'abord, il peut être utilisé avec n'importe quelles représentations des documents préalablement construites rendant le calcul de $p(q|d)$ flexible. De plus il est propice aux extensions puisque l'estimation de $p(d|c)$ repose sur des choix spécifiques aux données. Enfin, les résultats rapportés par Balog *et al.* montrent qu'il présente une forte robustesse au choix des paramètres des techniques de représentation des documents choisies.

Modèles de votes

Les modèles de votes sont très proches du modèle-document précédemment décrit. L'approche consiste à attribuer un score à chaque document étant donnée une requête et de faire voter les documents pour classer les candidats qui leur sont associés. Ceci implique d'utiliser des techniques de fusion de données. Les auteurs font l'hypothèse que trois caractéristiques peuvent influencer le vote :

— le nombre de documents associés à chaque candidat ;
— le score de chaque document ;
— le rang de chaque document une fois le corpus classé selon une requête.

Pour chaque candidat, son score est calculé à partir des documents auxquels il est lié selon des techniques de fusion dont certaines ont été confrontées dans [Macdonald and Ounis,

2006]. On peut citer les trois méthodes de fusions suivantes :

$$\mathrm{RR}(c) = \sum_{d \in D(c|q)} \frac{1}{r_d}, \tag{4.4}$$

$$\mathrm{CombSUM}(c) = \sum_{d \in D(c|q)} s_d, \tag{4.5}$$

$$\mathrm{CombMNZ}(c) = |D(c|q)| \times \mathrm{CombSUM}(c), \tag{4.6}$$

où $D(c|q)$ est l'ensemble des documents d liés au candidat c, classés selon la requête q et où s_d est le score du document et r_d son rang associé. Notons que les documents peuvent être partiellement classés, ne gardant que les meilleurs scores et évitant ainsi l'impact du nombre de documents associés à chaque candidat dans le calcul des scores. Cette approche est similaire au modèle-document où les candidats sont classés à travers leurs associations aux documents $p(q|c) \overset{\text{rang}}{=} \sum_d p(q|d)p(d|c)p(c)$. Cette dernière formule peut être adaptée en variant les mesures de $p(q|d)$ et $p(c)$. Dans le cas de CombMNZ(c), $p(q|d)$ correspond au score de similarité sémantique entre la requête et le document et $p(c) = |D(c|q)|$ correspond au nombre de documents liés à ce candidat (plus il est grand, plus sa probabilité *a priori* d'être expert est grande).

Modèles discriminatifs

La surreprésentation des modèles génératifs pour modéliser les relations entre requêtes et candidats a tardivement motivé l'exploration de modèles discriminatifs. Si les modèles génératifs cherchent à caractériser le processus selon lequel une requête est générée par un candidat via un ensemble support de documents, les modèles discriminatifs proposent d'inférer directement la probabilité qu'un candidat soit expert d'une requête à travers un système paramétrique et un ensemble d'exemples. Dans [Fang et al., 2010], la recherche est définie comme un problème de classification binaire qui, à un ensemble de paires requête-candidat, associe un ensemble de décisions binaires $\mathscr{E}$ sur l'expertise du candidat $p_\theta(e|q,c) \in \{0,1\}$ correspondant à *non-expert* et *expert*. L'optimisation des paramètres θ se fait par maximisation de la vraisemblance :

$$L = \prod_q \prod_c p_\theta(\mathscr{E} = 1|q,c)^{e_{cq}} p_\theta(\mathscr{E} = 0|q,c)^{1-e_{cq}}, \tag{4.7}$$

où $e_{cq} \in \{0,1\}$ est la vérité terrain sur l'association (q,c) (si le candidat c est expert de la requête q).

Une application directe de cette approche est proposée par les auteurs. Influencés par le succès du modèle-document, qui collecte les indices d'expertise à travers les documents, ceux-ci déterminent l'expertise de c sur q à partir de deux facteurs : (1) la pertinence de q vis-à-vis de chaque document d et (2) la force d'association entre le candidat c et chaque document d. L'expertise finale d'un candidat sur une requête est calculée en moyennant sur tous les documents : $p_\theta(\mathscr{E} = 1|q,c) = \sum_d p(\mathscr{E}_1 = 1|q,d)p(\mathscr{E}_2 = 1|d,c)p(d)$ où $p(\mathscr{E}_1 = 1|q,d)$ représente à quel point d est lié à la requête q et $p(\mathscr{E}_2 = 1|d,c)$ représente à quel point le document d représente c. La probabilité *a priori* d'un document $p(d)$ est supposée uniforme. Ces deux probabilités sont modélisées par des fonctions logistiques σ

sur des combinaisons linéaires de caractéristiques f et g, on a alors :

$$p(\mathscr{E}_1 = 1|q, d) = \sigma(\sum_i \alpha_i f_i(q, d)), \tag{4.8}$$

$$p(\mathscr{E}_2 = 1|d, c) = \sigma(\sum_i \beta_i g_i(d, c)), \tag{4.9}$$

$$\text{avec} \quad \sigma(x) = \frac{1}{1 + e^{-x}}. \tag{4.10}$$

Les paramètres α_i et β_i sont optimisés par la méthode de Quasi-Newton à partir de la log-vraisemblance (comme une régression logistique). On voit que pour le modèle-document, la pertinence des documents vis-à-vis de la requête est calculée selon le modèle génératif sous-jacent et la force d'association entre candidats et documents est calculée à partir de critères heuristiques tandis que pour ce modèle, ceux-ci sont estimés à partir d'exemples. Le choix des caractéristiques f et g est libre ce qui rend cette approche flexible. Les expérimentations menées par Fang *et al.* montrent que ce modèle discriminatif réalise de meilleurs résultats que le modèle-document et montre une forte robustesse au choix des techniques de représentation des documents utilisées pour générer les caractéristiques.

Utiliser une vérité terrain sur l'expertise des candidats peut être une approche efficace. Malheureusement, dans beaucoup d'applications de recherche d'experts, il est difficile d'obtenir des données d'entraînement conséquentes. Les modèles génératifs non supervisés tirent pleinement profit du nombre de documents non labellisés disponibles. Cependant, ils reposent parfois sur des hypothèses trop fortes pour saisir correctement le processus qu'ils cherchent à estimer.

Marches aléatoires dans le graphe des candidats

Deux aspects essentiels de la recherche d'experts présentés jusqu'ici sont l'étude du contenu sémantique des documents à travers la relation requête-document ainsi que l'exploitation des liens candidat-document. Une autre source d'information réside dans les interactions entre candidats. Une hypothèse couramment admise est qu'un expert est nécessairement influent dans son entourage social. L'exploration des caractéristiques topologiques du graphe social des candidats peut permettre d'établir un score, appelé autorité, estimant la force d'influence d'un candidat sur ses pairs. Deux algorithmes fondateurs, *HITS* [Kleinberg, 1999] et *PageRank* [Page et al., 1999], ont introduit la notion d'autorité dans les graphes pour les moteurs de recherche de pages Web. Dans le cas du monde scientifique, le graphe sur lequel ces algorithmes opèrent est construit en attribuant un sommet a chaque candidat, et en établissant des liens de co-auteurs (lorsque deux candidats ont signé le même document) et de citation (lorsqu'un document cite un autre document, on connecte leurs auteurs). Sur un tel graphe, les algorithmes basés sur les marches aléatoires peuvent permettre d'identifier les experts parmi les candidats.

Dans [Zhang et al., 2007a], la possibilité d'identifier des experts uniquement sur la base des interconnexions entre candidats est examinée sur un forum Web de questions et réponses dans le domaine de la programmation (réseau communautaire d'expertises). Les données issues de ce forum sont transformées en établissant un graphe pondéré où chaque sommet représente un utilisateur et les liens connectent chaque utilisateur ayant posé une question à tous les utilisateurs y ayant répondu. Les auteurs confrontent plusieurs algorithmes de recherche d'experts basés uniquement sur ce graphe, présentés en tableau 4.1, allant de simples indicateurs statistiques à l'implémentation de *HITS* et de *PageRank* .

Les résultats indiquent que tous ces algorithmes donnent des résultats similaires, malgré la simplicité des trois premiers algorithmes comparée à PageRank et HITS. Ils utilisent

TABLE 4.1 – Algorithmes utilisés dans [Zhang et al., 2007a] sur un graphe communautaire de candidats.

Nom	Description
Nombre de réponses	Le score est le simple compte de réponses données par un candidat.
Nombre de utilisateurs	Le score est le compte du nombre d'utilisateurs différents ayant reçu une réponse du candidat.
Z-score	Le score est la variation du rapport de réponses données sur le nombre de questions posées par rapport à l'utilisateur moyen.
ExpertiseRank	Adaptation de PageRank sur ce graphe d'experts.
HITS	Implémentation de HITS sur ce graphe d'experts.

alors une simulation pour déterminer les processus génératifs du graphe qui influent sur les résultats de ces différents algorithmes. Ils comparent principalement deux simulations : la première où les meilleurs experts répondent en priorité, correspondant au fonctionnement du graphe communautaire de candidats, et la seconde où n'importe quel utilisateur un peu plus expert que celui qui pose la question peut répondre, correspondant à l'environnement d'une entreprise où l'entraide n'est pas explicitement récompensée. Ils observent alors que *PageRank* est l'algorithme le plus stable et le plus à même d'identifier les utilisateurs influents, même si le nombre de réponses qu'ils donnent est très inférieur à ceux des candidats moyens, ce qui est le cas dans la seconde simulation.

Les avantages d'un algorithme tel que PageRank, ou plus généralement des marches aléatoires, sont sa capacité à intégrer (1) des contraintes et (2) différents types de liens. Cependant, l'hypothèse que l'expertise est équivalente à l'autorité n'est pas toujours vraie (un scientifique très spécialisé peut être expert dans son domaine mais avoir peu d'autorité). De plus, il peut être difficile de régler correctement les paramètres de PageRank dans un graphe possédant de plusieurs communautés où l'autorité est distribuée de manière différentes dans chacune d'elles, ce qui peut être le cas dans un jeu de données académique multidisciplinaire (les scientifiques ne collaborent pas de la même façon selon leurs domaines de recherche). Il est alors possible d'étendre ce modèle de façon à intégrer dans la propagation l'aspect sémantique de la requête. Cette extension peut se faire naturellement dans PageRank à travers la distribution initiale π_0 des scores attribués aux sommets (PPR), à condition de travailler sur un graphe biparti, où les documents sont aussi associés à des sommets.

Modèles de propagation dans le graphe biparti candidat-document

Dans [Serdyukov et al., 2008], les auteurs étudient la propagation de la pertinence des documents vis-à-vis de la requête à travers le réseau candidat-document. Ils présentent plusieurs types de marches aléatoires dans un graphe, étant donné un ensemble de documents dont les scores de pertinence vis-à-vis de la requête sont précalculés. On peut construire un graphe biparti dans lequel documents et candidats sont des sommets connectés selon les liens d'auteurs et de citations. On peut alors construire une matrice de transition M entre ces sommets ainsi qu'un vecteur π_0 contenant des scores initiaux associés aux candidats et documents, même si l'on ne s'intéresse au final qu'aux scores des candidats après quelques itérations ℓ dans π_ℓ. π_0 peut-être sélectionné selon différentes politiques en fonction notamment des similarités sémantiques entre la requête et les documents. Les algorithmes proposés par les auteurs visent à modéliser le comportement d'un humain recherchant des experts à travers le graphe candidat-document. A tout moment,

celui-ci peut choisir aléatoirement un document ou un candidat. Après avoir consulté l'un de ces deux types de sommet, il suit les liens du réseau pour consulter un autre document ou un autre candidat. Les scores sont calculés selon la formule du PageRank personnalisé à partir de la matrice de transition normalisée P, en définissant une matrice de transition M avec rebond γ et un nombre d'itérations à réaliser ℓ :

$$M = \gamma P + (1 - \gamma) I_{n_v} m_0, \tag{4.11}$$

$$\pi_\ell = \pi_0 M^\ell. \tag{4.12}$$

De la sorte, le vecteur m_0 constitue la probabilité de revenir sur le sommet initial à chaque itération. Les auteurs explorent alors 2 types de marches, modélisant des comportements d'exploration différents selon les choix de ℓ, m_0 et π_0 :

— marches aléatoires finies : le nombre de marches est fixé à l'avance. La distribution π_0 est choisie selon la similarité sémantique entre les documents et la requête et donne aux candidats une valeur nulle. L'utilisation d'un facteur de rebond est inutile car aucune stationnarité n'est souhaitée, le nombre d'étapes étant limité. On a donc $\gamma = 1$ et m_0 n'a pas d'effet ;

— marches aléatoires infinies : le nombre de marches est infini et la procédure est stoppée lorsque les scores des experts atteignent un état stationnaire. L'initialisation suit la même règle que pour la version finie (π_0 est calculée par similarité requête-document et est nulle pour les candidats), mais cette fois le facteur de rebond est utilisé pour garantir la stationnarité du système. La distribution m_0 est calculée selon la similarité sémantique pour les documents, tandis que pour les candidats, elle est égale à la probabilité de le trouver parmi les auteurs des K meilleurs documents selon la requête, donnant ainsi plus de poids à ces candidats.

D'après les expériences menées avec ces modèles, les marches finies et infinies permettent d'améliorer la qualité des classements par rapport aux modèles de votes et au modèle-document de Balog *et al.* avec associations *documents-centré*. Ce dernier est par ailleurs un cas spécial d'une marche aléatoire finie où la longueur des marches est fixé à $\ell = 1$. En effet, lors de l'étape d'initialisation π_0 le marcheur aléatoire sélectionne un document selon les scores de similarité sémantique avec la requête, équivalents aux probabilités $p(q|d)$ de l'équation (4.3) puis les somme à travers les associations document-candidat, représentant la probabilité $p(d|c)$.

Autres approches

WISER [Cifariello et al., 2019] modélise chaque candidat comme un sous-graphe pondéré extrait de la base de connaissance Wikipedia (*Knowledge Graph*). Les informations dérivées de cette base et les techniques traditionnelles de représentation des documents sont combinées pour identifier les experts vis-à-vis d'une requête. Notons que les méthodes utilisant des données externes sortent du cadre de cette thèse. *LT Expertfinder* [Fischer et al., 2019] est une suite logicielle d'évaluation pour la recherche d'experts basé sur un outil interactif. Elle intègre divers algorithmes existants (tels que [Balog et al., 2010]) et permet à des utilisateurs de facilement évaluer les experts identifiés par ceux-ci. Le corpus sous-jacent utilisé par cet outil est le réseau *ACL Anthology*. Cependant, il n'inclut pas une vérité terrain bien établie pour évaluer qui sont les experts. En effet, l'évaluation est purement réalisée en ligne puisque l'utilisateur doit évaluer le degré d'expertise des candidats en fonction de plusieurs fonctionnalités, telles que les citations de l'auteur, le *h-index*, les mots-clefs, etc. Des travaux récents proposent des techniques de plongement. [Van Gysel et al., 2016] présente un modèle qui plonge les mots et les experts du réseau candidat-document de sorte à estimer directement $p(c|q)$. [Dargahi Nobari et al., 2017] suggère un modèle de traduction pour identifier les experts sur la plateforme *Stack Overlfow*. L'idée

centrale est de traduire les mots-clefs d'expertise (tels que « python ») en ensemble de mots définissant ce domaine d'expertise (par exemple « programming, language, numpy, scipy, PyCharm, etc. »). Ensuite, les candidats ayant répondus à une question sont ordonnés selon la similarité de ces mots liés aux mots-clefs avec les mots contenus dans les réponses. Notons qu'ici la tâche considérée diffère de celle adoptée dans nos travaux puisqu'il s'agit de ne classer que quelques candidats en fonction de leurs réponses et non de retrouver ceux-ci parmi la liste intégrale des candidats.

4.3 Contributions

Notre contribution dans le domaine de la recherche d'experts se divise en deux aspects. Premièrement nous proposons une méthode d'évaluation des algorithmes différente de la méthode traditionnellement adoptée où les requêtes ne sont plus les dénominations des thématiques d'expertise mais sont des documents associés à ces thématiques. Nous détaillons en section 4.3.1 cette approche et décrivons quatre jeux de données que nous avons utilisés pour la mettre en œuvre. Ensuite, nous étudions l'applicabilité des techniques de plongement de réseau de documents à la tâche de recherche d'experts. Nous présentons en section 4.3.2 les méthodes d'applications de telles techniques aussi bien pour enrichir les méthodes existantes que pour les utiliser de manière *ad hoc*.

4.3.1 Les requêtes-documents

Nous présentons dans cette section notre méthode d'évaluation introduite dans [Brochier et al., 2018a] et employée dans [Brochier et al., 2020a]. Nous décrivons ensuite les jeux de données nous permettant de la mettre en œuvre.

Protocole d'évaluation

La recherche d'experts est une tâche complexe qui peut être formalisée de plusieurs façons. Les premiers travaux définissent cette tâche comme un problème de classement où les dénominations des thématiques d'expertise sont directement utilisées comme requêtes pour rechercher les candidats experts. Cependant, dans de nombreuses applications réelles, un utilisateur est invité à fournir une requête spécifique et détaillée. Dans un site de questions et réponses par exemple, un utilisateur expose généralement le problème auquel il est confronté en détail et ne connaît pas nécessairement la dénomination exacte des domaines d'expertise nécessaires pour résoudre son problème. De plus, l'évaluation d'un algorithme avec un petit ensemble de requêtes thématiques peut conduire à de mauvaises mesures en raison du nombre généralement faible de domaines d'expertise associés au jeu de données. Enfin, la différence entre le riche contenu textuel des documents et la souvent très courte dénomination des domaines d'expertise ajoute un biais à l'évaluation. Pour ces raisons, nous proposons une nouvelle méthodologie d'évaluation avec des requêtes orientées documents et nous construisons quatre jeux de données pour lesquels l'ensemble des requêtes est annoté manuellement.

Dans ce protocole d'évaluation, la tâche de recherche d'expert est un problème de classement. Étant donné un document d_i étiqueté avec un ou plusieurs de domaines d'expertise, un algorithme est interrogé pour classer les candidats, parmi lesquels un sous-ensemble d'experts est associé aux mêmes domaines d'expertise que le document. Les données fournies aux algorithmes consistent en un corpus de n_d documents D, n_c candidats C, un réseau candidat-document avec pour matrice d'adjacence $A_{dc} \in \mathbb{N}^{n_d \times n_c}$ et un réseau document-document avec pour matrice d'adjacence $A_{dd} \in \mathbb{N}^{n_d \times n_d}$. La figure 4.2 montre un jeu de données hypothétique reflétant cette méthode d'évaluation.

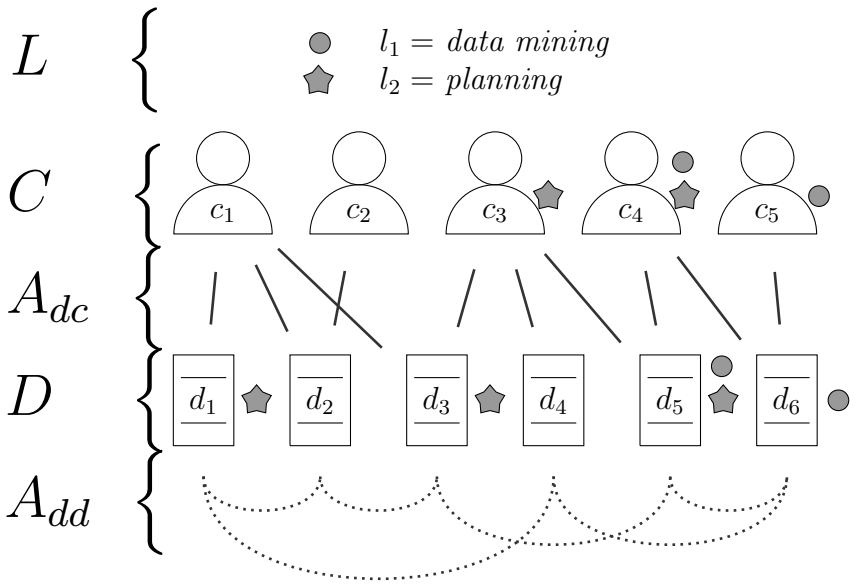


FIGURE 4.2 – Exemple hypothétique d'un jeu de données de recherche d'experts reflétant notre méthodologie d'évaluation. Parmi les candidats, c_4 et c_5 sont experts en *data mining* (étoiles) et c_3 et c_4 sont experts en *planning*. À la différence de la méthode traditionnelle illustrée en figure 4.1, les documents sont aussi associés aux domaines d'expertise. Ici notamment, d_1, d_3 et d_5 sont associés à *data mining* et les documents d_5 et d_6 sont associés à *planning*. Dans notre méthodologie d'évaluation, nous interrogeons un algorithme avec ces 4 documents et attendons un classement des candidats qui correspond aux domaines d'expertise de chaque document. Par exemple, un algorithme parfait pourrait générer les classements $d_1 \mapsto c_3c_4c_5c_1c_2$ et $d_6 \mapsto c_4c_5c_3c_2c_1$. Notons qu'au lieu d'avoir 2 requêtes thématiques nous avons 4 requêtes documents (les documents d_2 et d_4 ne sont ici pas annotés).

Le classement est effectué dans un cadre non supervisé, c'est-à-dire qu'aucun étiquetage d'expertise n'est perçu par les algorithmes. L'ensemble des documents étiquetés (les requêtes) peut être de taille inférieure à n_d et l'ensemble des candidats étiquetés (les experts) peut être de taille inférieur à n_c (c'est-à-dire que tous les documents et candidats ne sont pas nécessairement liés aux domaines d'expertise).

Un exemple de cas réel représenté par notre méthode d'évaluation est la recherche automatique de relecteurs. Dans le cadre de l'évaluation traditionnelle illustrée en figure 4.1, un éditeur chargé de trouver des relecteurs pour juger la qualité d'un article serait censé fournir au système de recherche d'experts une requête thématique, c'est-à-dire l'intitulé du domaine scientifique de l'article (ex : *data mining*). Notre méthodologie, illustrée en figure 4.2, considère le fait que l'éditeur fournisse directement le contenu de l'article au système de recherche d'experts. Nous pensons que ce scénario est plus représentatif de la tâche de recherche d'experts pour différents environnements, dont celui des sites de questions et réponses, où l'utilisateur décrit en détail sa requête sans nécessairement connaître l'intitulé exacte de la thématique qui se rapporte à sa question.

Jeux de données

Pour mettre en œuvre notre méthode d'évaluation, nous considérons quatre jeux de données. Le premier est un extrait de DBLP [Tang et al., 2008] dans lequel une liste de

199 experts dans 7 domaines sont annotés [Zhang et al., 2007b] par jugements humains [2]. La version de ce jeu de données ne prend en compte que les experts annotés et les autres candidats proches dans le réseau de coauteurs, ce qui explique la taille relativement petite de notre réseau par rapport à l'original. En plus des annotations d'experts, notre méthode d'évaluation nécessite des annotations des documents. Nous avons demandé à deux doctorants en informatique d'associer indépendamment 20 documents tirés au sort par domaine d'expertise (140 au total). Ensuite, seules les annotations sur lesquelles les deux annotateurs se sont mis d'accord ont été conservées, produisant 114 documents étiquetés. Le coefficient moyen κ de Cohen sur l'ensemble des annotations est $\kappa = 0,718$. Un avantage de notre méthodologie est que nous pouvons évaluer les algorithmes sur plus de requêtes (ici 114 documents) que la méthode traditionnelle (7 expertises). Cela nous permet d'évaluer la robustesse des algorithmes en fonction de la dispersion des mesures de qualité des classements à travers les requêtes. Cependant, on peut suggérer que ces 7 domaines ne reflètent pas un ensemble représentatif d'expertises car ils couvrent des spectres de recherche relativement larges. Pour cette raison, nous recherchons une plus grande granularité d'expertises par l'utilisation d'un réseau de sites reconnus de questions et réponses.

Si les réseaux de publications scientifiques sont faciles à trouver sur le Web, les annotations d'expertise scientifique sont rarement disponibles pour les auteurs et les publications. Nous utilisons les données téléchargées en juin 2019 à partir de *Stack Exchange* recueillies auprès de trois communautés étroitement liées au monde de la recherche : Academia SE, Math Overflow et Stats SE. Pour chaque jeu de données, nous conservons d'abord les questions qui ont obtenues au moins 10 votes utilisateurs et qui ont au moins une réponse avec 10 votes ou plus. Nous construisons les réseaux en reliant les questions à leurs réponses et en reliant les questions/réponses aux utilisateurs qui les ont publiées. Les domaines d'expertise sont les mots-clefs (ou *tags*) associées aux questions. Seuls les mots-clefs qui sont utilisés au moins 50 fois par la communauté sont considérés. Nous annotons un candidat comme expert d'un certain mot-clef si sa réponse à une question associée à ce mot-clef a reçu au moins 10 votes. Si les mots-clefs sont d'abord fournis par les utilisateurs qui posent les questions, ils sont par la suite presque systématiquement vérifiés par des utilisateurs expérimentés.

Les propriétés générales de ces 4 jeux de données sont présentées dans le tableau 4.2. Les annotations et les jeux de données prétraités sont rendus publics [3].

TABLE 4.2 – Propriétés générales des jeux de données pour la recherche d'experts.

	# candidats	# documents	# expertises	# experts	# requêtes	exemple d'expertise
DBLP	707	1641	7	199	114	'information_extraction'
Stats SE	5765	14834	59	5765	3966	'maximum-likelihood'
Academia SE	6030	20799	55	6030	4214	'recommendation-letter'
Math Overflow	7382	38532	98	7382	10614	'galois-representations'

4.3.2 Les plongements pour la recherche d'experts

Les techniques de plongement de réseau de documents fonctionnent dans des réseaux homogènes qui ne possèdent pas de sommets correspondant aux candidats dans les réseaux hétérogènes considérés en recherche d'experts. Pour les appliquer dans le cadre de la recherche d'experts, nous étudions deux utilisations : (1) comme représentations des documents pour un modèle de recherche d'experts et (2) comme modèles *ad hoc* de re-

2. https://lfs.aminer.cn/lab-datasets/expertfinding/#expert-list

3. https://github.com/brochier/expert_finding

cherche d'experts en transformant le réseau hétérogène document-candidat en un réseau de documents sur lequel ces techniques peuvent être appliquées.

DNE pour représenter les documents

Comme vu dans la section 4.2, de nombreux algorithmes de l'état de l'art s'appuient sur des représentations préalablement construites des documents, permettant de calculer des scores de similarité avec les requêtes. Parce qu'ils prennent en compte à la fois le contenu textuel et la structure du réseau, les méthodes de DNE ont le potentiel de fournir de meilleures représentations des documents que les techniques de représentations classiques, basées uniquement sur l'information textuelle. Pour ce faire, nous pouvons construire un réseau de documents à partir d'un réseau hétérogène document-candidat en construisant une matrice d'adjacence A_d entre documents telle que $A_d = A_{dc}A_{dc}^\top + A_{dd}$. De la sorte, il existe un lien entre deux documents s'il existe un lien direct dans le réseau hétérogène et/ou s'il existe un lien indirect, c'est-à-dire si deux documents partagent un même auteur. La matrice A_d et les contenus textuels des documents étant utilisés comme données d'entrées avec un algorithme de DNE, on construit ainsi des représentations $X \in \mathbb{R}$ des documents pouvant être utilisées par différents modèles de recherche d'experts, tels que les modèles de votes ou de propagation.

DNE comme modèle *ad hoc* de recherche d'experts

Une autre approche que nous considérons pour utiliser les algorithmes de DNE pour la recherche d'experts consiste cette fois à adapter le réseaux hétérogène candidat-document de sorte à représenter les candidats comme des documents. Ceci peut se faire de deux façon, soit avant l'entraînement du modèle de DNE en agrégeant les contenus des documents associés à un candidat (pré-agrégation), soit après l'entraînement en agrégeant les représentations des documents associés à un candidat (post-agrégation). Voici les détails de ces deux solutions simples :

— pré-agrégation : comme dans le modèle P@noptic, des méta-documents sont générés en agrégeant les documents produits par chaque candidat. Nous construisons un réseau de candidats $A_c = A_{dc}^\top A_{dc}$ et un réseau de documents $A_d = A_{dc}A_{dc}^\top + A_{dd}$. Nous construisons alors le méta-réseau $A = \begin{pmatrix} A_d & A_{dc} \\ A_{dc}^\top & A_c \end{pmatrix}$. Les représentations des candidats et des documents sont ensuite générées en traitant ce méta-réseau et la concaténation des documents et méta-documents comme une instance ordinaire de réseau de documents. A partir de ce méta-réseau, nous générons des représentations avec les algorithmes de DNE. Les scores des candidats sont calculés par similarité cosinus entre la représentation d'un document requête et les représentations des méta-documents candidats ;

— post-agrégation : dans cette approche, nous entraînons d'abord l'algorithme DNE sur le réseau de documents défini par $A_d = A_{dc}A_{dc}^\top + A_{dd}$. Une fois les représentations générées pour tous les documents, une représentation d'un candidat est calculée en faisant la moyenne des vecteurs de tous les documents qui lui sont associés. Les scores sont ensuite calculés par similarité cosinus.

4.4 Expérimentations

Nous présentons dans cette section les résultats de nos expérimentations. Nous comparons notre méthode d'évaluation (requêtes documents) avec la méthode traditionnelle

(requêtes thématiques) et nous analysons les performances obtenues par les différents algorithmes.

4.4.1 Algorithmes

Nous décrivons ici les algorithmes de la littérature que nous utilisons dans nos expérimentations. Nous présentons d'abord les algorithmes classiques puis les algorithmes de DNE employés.

Modèles de recherche d'experts

Nous menons les expériences avec 3 modèles de la littérature et un modèle aléatoire servant de référence. Par défaut, nous utilisons les représentations TF-IDF pour représenter les documents et calculons la similarité entre les requêtes et les documents par similarité cosinus :

- Aléatoire : des scores de 0 à 1 sont tirés aléatoirement pour chaque candidat quelle que soit la requête ;
- P@noptic [Craswell et al., 2001] : nous concaténons les documents produits par chaque candidat au sein de méta-documents et utilisons les représentations TF-IDF de ces derniers pour calculer leurs similarités cosinus avec la requête ;
- Vote [Macdonald and Ounis, 2006] : nous utilisons le rang réciproque (RR) donné en équation 4.4 comme technique de fusion pour calculer les scores des candidats à partir de ceux des documents ;
- Propagation [Serdyukov et al., 2008] : nous concaténons les matrices d'adjacences A_{dc} et A_{dd} pour construire une matrice de transition entre candidats et documents de sorte que $A = \begin{pmatrix} A_{dd} & A_{dc} \\ A_{dc}^\top & 0 \end{pmatrix}$. Les scores d'initialisation π_0 sont les similarités cosinus normalisées entre la requête et les documents. Nous utilisons le modèle avec marches aléatoires infinies avec rebond $\gamma = 0.5$ et arrêtons la propagation lorsque les scores des candidats ont convergé.

De plus, nous étudions les performances de ces mêmes modèles en remplaçant les représentations TF-IDF par des représentations apprises avec des modèles de DNE que nous décrivons ci-dessous. Enfin, nous appliquons les deux schémas d'agrégation décrits dans la section précédente (Pré-agr et Post-agr) en utilisant des représentations issues de DNE.

Modèles de plongement de documents en réseau

Nous menons les expériences avec quatre algorithmes de plongement de réseau de documents en utilisant les implémentations des auteurs. Pour toutes les méthodes, la dimension des représentations est fixée à $\rho_d = 256$:

- TADW [Yang et al., 2015] : nous réutilisons l'implémentation des auteurs en appliquant 20 itérations et un facteur de régularisation $\lambda = 0.2$;
- GVNR-t [Brochier et al., 2019a] : nous réalisons $\mu = 10$ marches aléatoires par sommet de longueurs $\ell = 40$, une taille de fenêtre $\tau = 5$ et un seuil $x_{\min} = 2$ avec 4 itérations ;
- Graph2gauss (G2G) [Bojchevski and Günnemann, 2018] : nous nous assurons de la convergence de la fonction de perte pour chaque jeu de données ;
- IDNE [Brochier et al., 2020b] : nous utilisons $n_t = 32$ vecteurs thématiques avec 5000 itérations de 16 échantillons positifs et 16 échantillons négatifs.

4.4.2 Résultats quantitatifs

Nous présentons ici les résultats de nos expérimentations. Les tableaux 4.3 et 4.4 donnent les scores pour les jeux de données DBLP et Stats SE obtenus par les algorithmes traditionnels de recherche d'experts selon la méthode classique d'évaluation avec des requêtes thématiques. Les tableaux 4.5 à 4.8 donnent les résultats obtenus par tous les algorithmes sur tous les jeux de données avec notre méthode d'évaluation basée sur des requêtes documents. Nous analysons dans les prochaines sections ces résultats. Notons que l'implémentation de TADW n'a pu passer à l'échelle sur le jeu de données Math Overflow.

TABLE 4.3 – Scores moyens et leurs écarts-types sur DBLP suivant la méthode d'évaluation avec requêtes thématiques.

	AUC	P@10	AP
Aléatoire	48.44 (08.59)	02.86 (04.52)	06.15 (01.56)
P@noptic	85.37 (11.60)	31.43 (11.25)	28.12 (12.72)
Vote	85.77 (16.08)	40.00 (16.90)	41.85 (14.55)
Propagation	**86.14** (18.03)	**47.14** (18.29)	**49.47** (19.96)

TABLE 4.4 – Scores moyens et leurs écarts-types sur Stats SE suivant la méthode d'évaluation avec requêtes thématiques.

	AUC	P@10	AP
Aléatoire	50.18 (03.00)	01.19 (03.23)	01.80 (01.24)
P@noptic	80.64 (08.96)	10.00 (10.89)	09.88 (05.05)
Vote	90.20 (04.81)	43.90 (14.50)	27.09 (06.77)
Propagation	**92.32** (07.40)	**76.78** (19.08)	**38.76** (11.35)

TABLE 4.5 – Scores moyens et leurs écarts-types sur DBLP suivant la méthode d'évaluation avec requêtes documents.

	AUC	P@10	AP
Aléatoire	49.47 (09.80)	05.00 (06.66)	07.09 (03.81)
P@noptic (TF-IDF)	74.06(12.94)	22.37 (16.35)	23.24 (12.55)
Vote (TF-IDF)	78.60 (11.97)	26.05 (15.76)	28.24 (13.92)
Propagation (TF-IDF)	79.26 (13.09)	33.07 (19.61)	34.66 (18.21)
Pré-agr TADW	65.84 (12.94)	15.61 (11.63)	17.26 (08.78)
Pré-agr GVNR-t	76.90 (11.46)	19.04 (11.70)	21.39 (09.61)
Pré-agr G2G	72.87 (12.75)	15.70 (11.62)	18.53 (09.37)
Pré-agr IDNE	78.08 (11.27)	20.18 (11.85)	22.00 (09.87)
Post-agr TADW	68.01 (13.37)	16.32 (11.57)	18.01 (08.97
Post-agr GVNR-t	73.91 (13.93)	18.86 (12.19)	20.57 (10.33)
Post-agr G2G	68.94 (15.23)	16.23 (12.02)	18.21 (09.76)
Post-agr IDNE	76.87 (13.36)	19.04 (14.57)	21.57 (10.96)
Vote (IDNE)	82.23 (11.08)	34.82 (18.46)	37.27 (16.16)
Propagation (IDNE)	**82.44** (16.14)	**44.47** (22.91)	**47.01** (22.06)

TABLE 4.6 – Scores moyens et leurs écarts-types sur Stats SE suivant la méthode d'évaluation avec requêtes documents.

	AUC	P@10	AP
Aléatoire	50.01 (02.24)	04.52 (07.02)	04.96 (02.81)
P@noptic (TF-IDF)	79.47 (06.22)	13.45 (13.39)	15.22 (05.62)
Vote (TF-IDF)	84.96 (05.22)	52.53 (16.13)	31.01 (06.58)
Propagation (TF-IDF)	86.33 (05.64)	**91.53** (13.44)	**44.09** (07.70
Pré-agr TADW	63.07 (07.70)	11.42 (12.34)	08.45 (03.87)
Pré-agr GVNR-t	70.67 (09.49)	21.12 (20.99)	12.43 (07.30
Pré-agr G2G	63.63 (07.62)	12.93 (12.06)	07.81 (04.15
Pré-agr IDNE	65.07 (09.05)	13.37 (13.48)	09.40 (05.19)
Post-agr TADW	68.74 (07.02)	13.67 (12.59)	09.99 (04.37)
Post-agr GVNR-t	66.56 (08.61)	22.47 (15.92)	10.75 (05.42
Post-agr G2G	62.53 (07.44)	11.95 (11.86)	07.48 (04.13)
Post-agr IDNE	65.63 (08.57)	13.34 (13.13)	09.38 (04.94
Vote (IDNE)	**86.94** (04.91)	53.91 (18.06)	32.18 (08.33)
Propagation (IDNE)	67.62 (10.11)	90.43 (15.20)	33.07 (08.93)

TABLE 4.7 – Scores moyens et leurs écarts-types sur Academia SE suivant la méthode d'évaluation avec requêtes documents.

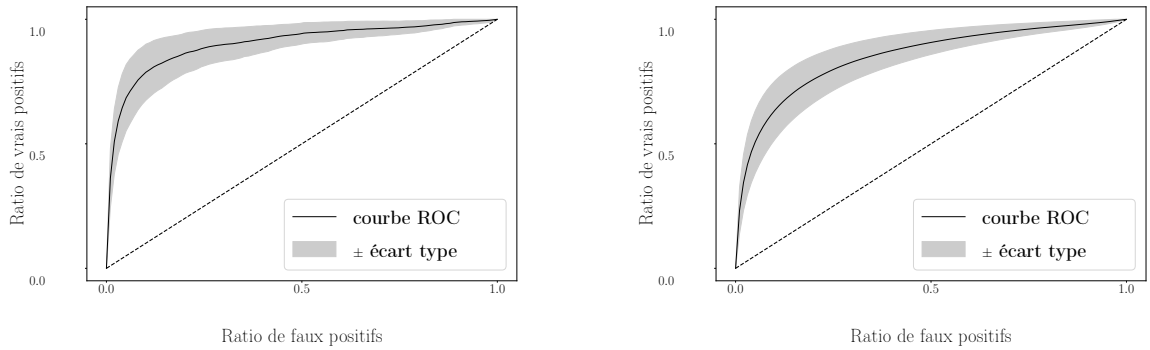
	AUC	P@10	AP
Aléatoire	50.02 (01.78)	05.93 (08.07)	06.09 (02.72)
P@noptic (TF-IDF)	81.54 (04.36)	18.35 (18.76)	22.93 (07.14)
Vote (TF-IDF)	85.88 (03.47)	57.99 (15.87)	37.66 (05.83
Propagation (TF-IDF)	**88.02** (03.32)	**99.01** (03.57)	**54.04** (05.44)
Pré-agr TADW	61.47 (06.16)	11.09 (12.04)	09.29 (03.53)
Pré-agr GVNR-t	64.22 (09.69)	25.67 (23.27)	13.07 (07.54)
Pré-agr G2G	61.54 (05.38)	14.30 (12.91)	08.74 (03.69)
Pré-agr IDNE	58.74 (07.49)	10.21 (11.58)	08.41 (03.99
Post-agr TADW	71.94 (04.63)	14.44 (12.87)	12.68 (04.37
Post-agr GVNR-t	61.22 (06.24)	20.70 (14.59)	10.19 (04.21)
Post-agr G2G	58.87 (05.79)	12.80 (12.06)	08.12 (03.67)
Post-agr IDNE	59.97 (07.40)	10.61 (11.19)	08.76 (04.17))
Vote (IDNE)	86.79 (03.90)	55.81 (17.35)	37.13 (07.58)
Propagation (IDNE)	61.35 (08.56)	95.02 (10.15)	31.27 (08.21)

TABLE 4.8 – Scores moyens et leurs écarts-types sur Math Overflow suivant la méthode d'évaluation avec requêtes documents.

	AUC	P@10	AP
Aléatoire	49.98 (01.62)	06.44 (08.28)	06.53 (03.06)
P@noptic (TF-IDF)	81.87 (04.46)	21.95 (19.15)	22.95 (07.54)
Vote (TF-IDF)	86.80 (03.23)	61.11 (18.68)	40.10 (08.27)
Propagation (TF-IDF)	88.08 (03.38)	**93.68** (12.16)	**49.58** (08.90)
Pré-agr TADW	-	-	-
Pré-agr GVNR-t	65.34 (09.22)	44.02 (28.31)	16.88 (08.55)
Pré-agr G2G	66.84 (08.99)	22.95 (17.81)	12.49 (05.70)
Pré-agr IDNE	67.01 (09.26)	22.96 (17.84)	13.40 (06.02)
Post-agr TADW	-	-	-
Post-agr GVNR-t	63.84 (07.59)	41.81 (22.68)	14.96 (06.25)
Post-agr G2G	65.06 (09.09)	22.43 (16.94)	11.78 (05.51)
Post-agr IDNE	66.74 (09.10)	21.92 (17.21)	13.11 (05.87)
Vote (IDNE)	**88.71** (03.76)	68.46 (18.53)	43.53 (09.90)
Propagation (IDNE)	69.38 (09.65)	92.35 (13.88)	39.62 (09.89)

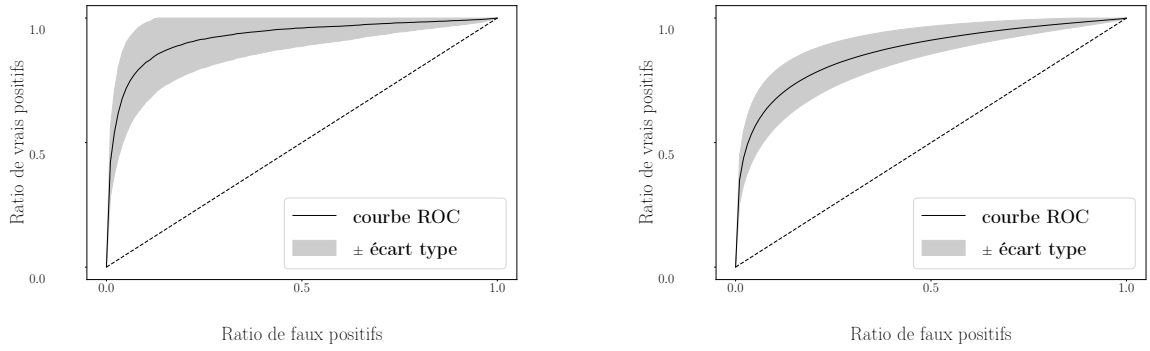
4.4.3 Comparaison des méthodologies d'évaluation

Pour comparer les méthodes d'évaluations, nous pouvons observer les tableaux 4.3 et 4.5 concernant DBLP ainsi que les tableaux 4.4 et 4.6 concernant Stats SE. Le classement des algorithmes est le même selon les deux méthodes d'évaluation, le modèle de propagation obtenant les meilleurs scores, suivi de près par le modèle de vote en terme d'AUC. Pour analyser de plus près, nous traçons en figures 4.3 et 4.4 les courbes ROC obtenues par le modèle de vote et le modèle de propagation avec représentations TF-IDF respectivement sur le jeu de données Stats SE selon les deux méthodes. Un premier avantage de notre méthode réside dans la stabilité des courbes ROC obtenues, due au nombre nettement plus élevé de requêtes soumises pour la construire. De plus, on voit que l'algorithme de propagation montre un écart-type important pour l'évaluation avec requêtes thématique (figure 4.4a), qui est gommé lors de notre évaluation (figure 4.4b), ce qui se traduit par l'étroitesse de l'intervalle de confiance. Ceci peut s'expliquer par le trop faible nombre de mots dans les requêtes thématiques, que l'on ne retrouve pas dans les requêtes documents.



(a) Évaluation avec requêtes thématiques. (b) Évaluation avec requêtes documents.

FIGURE 4.3 – Comparaison des courbes ROC du modèle de vote obtenues sur Stats SE selon deux types d'évaluation.



(a) Évaluation avec requêtes thématiques. (b) Évaluation avec requêtes documents.

FIGURE 4.4 – Comparaison des courbes ROC du modèle de propagation obtenues sur Stats SE selon deux types d'évaluation.

Pour la suite de notre analyse, nous nous concentrons sur la méthode d'évaluation avec requêtes documents.

4.4.4 Performances des algorithmes

Pour tous les jeux de données, le modèle de propagation fonctionne généralement mieux que les autres algorithmes, notamment en termes de précision. Les deux schémas d'agrégation réalisent de moins bonnes performances. La méthode de post-agrégation semble légèrement meilleure que celle de pré-agrégation. GVNR-t est le meilleur algorithme parmi les modèles d'agrégation intégrant des plongements de réseau de documents. Si les algorithmes DNE sont efficaces pour l'apprentissage de représentation de documents en réseau, leur application à la tâche de recherche d'experts via nos solutions simples d'agrégation ne leur permet pas de rivaliser avec les algorithmes traditionnels, même s'ils s'en approchent sur le réseau de publications scientifiques DBLP. Nous pensons que ceci est dû à la structure du réseau qui change considérablement entre un réseau homogène et un réseau hétérogène. De plus, les algorithmes de recherche d'experts bénéficient souvent d'informations sur la centralité des candidats et des documents. Ni les algorithmes DNE ni nos schémas d'agrégation ne capturent particulièrement cet aspect. Une piste future consisterait donc à étudier les capacités des algorithmes de DNE à (1) intégrer l'hétérogénéité du réseau et (2) capturer la centralité des sommets dans les représentations. Nous discutons de ces aspects plus en détail dans la conclusion de ce chapitre.

4.4.5 Impact des plongements comme représentation des documents

Les algorithmes de votes et de propagation étant performants, nous étudions la possibilité d'utiliser des algorithmes de DNE pour représenter les documents. Nous rapportons uniquement les résultats avec les représentations calculées avec IDNE mais nous observons les mêmes comportements avec les autres algorithmes de DNE. Tout d'abord, ces représentations améliorent constamment le modèle de vote, qui obtient les meilleurs résultats en termes d'AUC sur Stats SE et Math Overflow. Ensuite, l'effet le plus surprenant est la baisse significative des performances du modèle de propagation. Si la précision sur les candidats les mieux classés n'est pas affectée (P@10), le score AUC diminue considérablement pour les trois jeux de données issus de sites de questions et réponses. Nous pensons que les plongements de réseau de documents capturent des dépendances longues distances entre les documents du réseau, qui sont ensuite exagérées par la propagation.

4.5 Conclusions et perspectives

Dans ce chapitre, nous présentons une nouvelle méthodologie d'évaluation pour la recherche d'experts à l'aide de quatre jeux de données annotés et rapportons des résultats d'expériences réalisées avec plusieurs algorithmes traditionnels de la littérature. De plus, nous étudions la capacité des méthodes de plongement de réseau de documents à résoudre cette tâche. Nous montrons que les algorithmes DNE ne peuvent être adaptés de manière triviale pour obtenir des scores suffisants. Cependant, nous montrons que l'intégration de représentations apprises par un algorithme de DNE améliorent un modèle de vote mais peuvent altérer un modèle de propagation.

Ces résultats constituent une première étape vers la réalisation d'un algorithme de recherche d'experts reposant sur les récentes avancées en apprentissage de représentation de documents en réseau. Deux axes à explorer nous semblent être les facteurs clefs pour la réussite d'un tel algorithme :

— **l'hétérogénéité :** dans le cadre des expérimentations que nous avons menées, les données utilisées pour la recherche d'experts sont des réseaux bipartis de candidats et documents. Nombreux travaux [Sun and Han, 2012, Sun et al., 2013, Dong et al., 2015] ont montrés que la prise en compte de cette hétérogénéité des sommets dans le modèle d'apprentissage peut significativement améliorer la qualité des représentations. Dans Metapath2vec [Dong et al., 2017], cette prise en compte se fait à travers des marches aléatoires suivant des méta-chemins (*Meta-Path-Based Random Walks*) [Sun et al., 2011]. Un méta-chemin est un schéma de marche indiquant un ordre précis de types de sommets devant être parcourus. Un schéma type pour la recherche d'expert serait « c-d-c », signifiant que les marches aléatoires doivent systématiquement sélectionner un candidat c, puis un document d, puis un candidat c. Ce schéma décrit notamment la sémantique du coauteur, c'est-à-dire les relations de collaboration traditionnelles dans la publication scientifique. On peut ainsi définir plusieurs méta-chemins afin de capturer différentes sémantiques de relations. Metapath2vec utilise ensuite les chemins générés en adaptant Skip-Gram et son échantillonnage négatif pour prendre en compte la diversité des sommets parcourus. Dans ce sens, adapter GVNR-t et IDNE pour les graphes hétérogènes grâce aux méta-chemins nous semble être une piste prometteuse ;
— **la centralité :** le modèle de propagation est capable de capturer la notion de centralité des sommets. En effet, plus un candidat est connecté (par exemple lorsqu'il a écrit un nombre élevé de documents cosignés par un nombre élevé de candidats) celui-ci a plus de chance d'obtenir un score élevé. L'expertise est fortement corrélée avec la centralité des candidats dans le réseau puisque ceux-ci font, de fait, autorité dans leur communauté. Ceci explique le succès des méthodes reposant sur les mesures de centralité telles que PageRank [Page et al., 1999] et HITS [Kleinberg, 1999]. À l'inverse, la centralité n'est pas une propriété capturée par la plupart des méthodes de plongements de réseau. En effet, DeepWalk, GVNR ou IDNE représentent les sommets dans un espace qui préservent principalement leurs communautés. Certains travaux, tels que Node2vec [Grover and Leskovec, 2016] et Struc2vec [Ribeiro et al., 2017] proposent de capturer les rôles structurels associés aux sommets. Néanmoins, la recherche d'experts nécessite de capturer à la fois les communautés et les rôles des sommets. Nous ne connaissons aucune méthode de plongement ayant étudié la possibilité de capturer ces deux propriétés à la fois, de manière explicite. Node2vec permet par exemple de contrôler ces deux aspects via ses marches aléatoires biaisées. Il faudrait alors mesurer à quel point le poids accordé à l'un se fait au détriment de l'autre.

Dans des travaux futurs, nous aimerions intégrer ces aspects d'hétérogénéité et de centralité dans un algorithme de plongement de réseau pour la recherche d'experts. Ceci permettrait alors de recommander des experts de manière précise et efficace en temps de calcul des candidats dans de grandes bases de données. Dans le chapitre suivant, nous concluons ce manuscrit en résumant nos contributions et en présentant nos perspectives de recherche.

Chapitre 5

Conclusion

Pour conclure ce manuscrit de thèse, nous faisons en premier lieu un résumé des travaux que nous avons présentés, puis nous abordons quelques perspectives de recherche qui s'inscrivent dans la continuité de ces travaux.

Sommaire

5.1 Résumé de la thèse

Dans cette thèse, nous avons étudié l'apprentissage de représentation dans les réseaux, notamment les réseaux de documents, avec pour objectif de proposer des solutions de recherche d'information.

Tout d'abord, nous avons présenté GVNR, un algorithme de plongement de sommets dans les réseaux inspiré de GloVe, une méthode de plongement de mots. Il se distingue des autres approches de l'état de l'art par le calcul explicite d'une matrice de comptages des co-occurrences des sommets dans des marches aléatoires. Ceci permet de modifier cette matrice afin d'en filtrer des valeurs faibles, qui ne contiennent pas d'information utile. Ce filtrage apporte deux avantages : (1) la réduction de la complexité en temps de l'algorithme et (2) l'amélioration des représentations apprises en terme de classification des sommets. Enfin, nous avons étendu ce modèle avec GVNR-t afin de prendre en compte le texte associé aux documents dans un corpus structuré. Ceci se fait par l'apprentissage de représentations des mots en faisant l'hypothèse que la moyenne des vecteurs mots d'un document permet d'en représenter le sens.

Ensuite nous avons exploré les mécanismes d'attention afin d'améliorer la prise en compte du texte dans les réseaux de documents. Nous avons présenté IDNE, un modèle d'apprentissage de représentation de documents dont l'entraînement est réalisé à partir de la structure d'un réseau. Pour ce faire nous avons introduit un nouveau mécanisme, l'attention thématique, qui exprime les mots d'un document comme des mixtures de thématiques. Comme les paramètres de ce modèle sont appris à partir du réseau, les thématiques apprises représentent les différentes tendances au sein des communautés de ce réseau. Nous avons observé ce phénomène en étudiant les distances entre les vecteurs thématiques et les vecteurs mots appris. De plus, nous avons présenté MATAN, qui reprend la formulation du mécanisme d'attention du *Transformer*, afin de proposer un nouveau mécanisme mutuel sur des paires de documents identifiant les mots qui expliquent les liens dans un réseau.

Nous avons étudié expérimentalement la qualité des représentations construites avec GVNR, GVNR-t, IDNE et MATAN sur des tâches de classification des sommets et de prédictions de liens. Nous avons appuyé l'importance de tester ces algorithmes dans différentes configurations : (1) de manière transductive lorsque l'ensemble des sommets est observé durant l'apprentissage et (2) de manière inductive lorsque l'on cherche à induire de nouvelles représentations de documents pour lesquels nous n'observons aucun lien avec le réseau d'apprentissage. Nous avons montré que GVNR constitue un algorithme performant de plongement de réseau et que GVNR-t, MATAN et particulièrement IDNE réussissent mieux que les méthodes de l'état de l'art sur la plupart des jeux de données et dans les deux configurations.

Enfin, nous nous sommes intéressés à l'application de ces méthodes de plongement pour la recherche d'experts, une tâche de recherche d'information. Nous avons présenté une nouvelle méthodologie d'évaluation de cette tâche reposant non plus sur des requêtes thématiques mais sur des requêtes documents. Nous pensons que cette approche est plus représentative de cas d'applications réels de recherche d'experts. Nous avons pour cela construit quatre jeux de données issus d'un réseau de collaborations scientifiques et de forums de questions et réponses. Nous avons montré la pertinence de notre approche en comparant différents algorithmes traditionnels de la littérature sur les deux méthodologies d'évaluation. De plus, nous avons proposé différents schémas d'application des algorithmes de plongement de documents en réseau pour la recherche d'experts. Nous avons montré que si les représentations des documents qu'ils produisent peuvent améliorer les résultats obtenus avec des approches traditionnelles, nos méthodes d'agrégations du réseau hétérogène candidat-document ne permettent pas toujours d'aborder efficacement cette tâche.

Nous suggérons qu'il est nécessaire d'étudier plus en profondeur la capacité de ces algorithmes à capturer la centralité des sommets ainsi que la prise en compte de l'hétérogénéité de l'information dans les réseaux.

Pour la suite, nous présentons nos perspectives de travaux qui s'inscrivent dans la continuité de cette thèse.

5.2 Perspectives

Dans les conclusions des chapitres 2, 3 et 4, nous avons soulevé plusieurs axes de travaux qui nous semblent importants pour approfondir nos contributions. L'un de ces axes concerne le choix des propriétés que l'on souhaite capturer dans les représentations apprises sur un réseau. Nous revenons en premier lieu sur ce point en l'inscrivant dans le cadre plus général des techniques dites d'apprentissage auto-supervisé (*self-supervised learning*). Ensuite, nous abordons un aspect jusqu'ici non abordé dans ce manuscrit : la prise en compte de la dynamique des réseaux naturels. Enfin, nous discutons de l'utilisation des méthodes d'apprentissage de représentations dans les réseaux et de l'impact qu'elles peuvent avoir dans la conception des systèmes de recommandation.

5.2.1 L'apprentissage auto-supervisé dans les réseaux

L'apprentissage auto-supervisé réfère à l'apprentissage de représentations de données sans supervision externe, c'est-à-dire où les données elles-mêmes fournissent la supervision. C'est une notion similaire à l'apprentissage non supervisé mais néanmoins plus précise. En général, cette forme d'apprentissage consiste à cacher une partie des données et à prédire celles-ci à partir des données non cachées. Cette idée n'est pas nouvelle [de Sa, 1994], mais elle connaît aujourd'hui une forte popularité due aux améliorations significatives des performances de certains algorithmes qu'elle permet dans différents domaines. L'apprentissage dans ces domaines se fait généralement en deux étapes : une première auto-supervisée, où les représentations des données sont apprises à travers des tâches définies de manière heuristique, et une seconde où l'on affine (*fine-tuning*) les paramètres pour une tâche spécifique pour laquelle nous possédons des informations de supervision, tels que des *labels* en classification.

Un exemple de domaine où l'apprentissage auto-supervisé est devenu la norme est la reconnaissance d'images. Le pré-entraînement classique des réseaux de neurones convolutifs par auto-encodeurs [Masci et al., 2011] a été remplacé par l'entraînement de ces modèles sur une multitudes de tâche auto-supervisées. Certains travaux pré-entraînent ces modèles à retrouver le bon positionnement de petites parties extraites d'images [Noroozi and Favaro, 2016], d'autres les entraînent à retrouver les vraies couleurs d'une image en niveau de gris. Ces tâches, qui n'ont pas d'origine théorique particulière, sont choisies de sorte à contraindre les modèles entraînés à capturer certaines structures des données qui les aideront par la suite à se spécialiser.

Nous avons présenté dans le chapitre 3 en section 3.3.1 des modèles de traitement du langage qui suivent cette même approche. BERT [Lan et al., 2019] et ses variantes [Lan et al., 2019, Liu et al., 2019], GPT-2 [Radford et al., 2019] ou encore ULMFIT [Howard and Ruder, 2018] proposent une multitude de tâches auto-supervisées permettant d'apprendre des représentations du texte. Celles-ci sont, par exemple pour BERT, la prédiction de mots cachés et la classification binaire de paires de phrases, déterminant si elles se suivent ou non. Là encore, ces tâches sont choisies à partir d'hypothèses issues de la connaissance que l'on a de la structure des données. L'idée est que si un modèle est capable de prédire un mot manquant dans une phrase, c'est qu'il capture la structure de cette dernière et il sera donc plus à même d'apprendre à résoudre une tâche spécifique.

Qu'en est-il des réseaux ? On peut considérer que les méthodes de plongement de réseau réalisent un apprentissage auto-supervisé des sommets à travers une tâche définie sur la structure du graphe étudié. Souvent, cette tâche consiste à retrouver des co-occurrences de paires de sommets issues soit directement de la matrice d'adjacence, comme dans LINE [Tang et al., 2015], soit d'une puissance de cette matrice, comme dans TADW [Yang et al., 2015] et IDNE, soit même dans une approximation d'une puissance plus élevée de cette matrice, comme dans les méthodes basées sur des marches aléatoires. Les GNN reprennent ces approches, par exemple avec GraphSage [Hamilton et al., 2017b] utilisant des marches aléatoires pour échantillonner des paires de sommets. Aussi, comme nous l'avons vu, ces tâches ont pour objectif de capturer les communautés dans ces réseaux, car cette propriété corrèle fortement avec les classes associées aux réseaux étudiés et avec l'existence de liens entre les sommets. Cependant, il existe de nombreuses autres propriétés permettant de caractériser un réseau (centralité, équivalence structurelle, etc.). Peu de travaux, à notre connaissance, ont étudié l'établissement de nouvelles tâches d'auto-supervision dans les réseaux. De telles tâches ne sont pas aussi « naturelles » à imaginer que pour l'image et le texte car le graphe est une structure complexe. Néanmoins, la théorie des graphes fournit un vaste ensemble de techniques de caractérisation des réseaux qui, il nous semble, ont le potentiel d'inspirer des méthodes d'auto-supervision pour l'apprentissage de représentation dans les réseaux qui pourraient apporter les mêmes bénéfices observés dans d'autres domaines.

5.2.2 La dynamique des réseaux

Dans cette thèse, nous avons étudié le plongement de réseaux statiques, où les sommets et leurs liens n'évoluent pas au cours du temps. Seulement, la plupart des réseaux naturels sont en réalité dynamiques. Un réseau de citations est en constante évolution, des nouveaux articles étant publiés quotidiennement, faisant référence à des articles publiés par le passé. Un réseau social est en perpétuel mouvement, ses utilisateurs se lient et se partagent des contenus tous les jours. Cette évolution a un impact certain sur la compréhension d'un réseau. Par exemple, un chercheur change régulièrement ses axes d'intérêt, en s'adaptant à l'état de la recherche mais aussi en s'intéressant à de nouveaux domaines. Ces changements de direction peuvent permettre d'affiner la représentation d'un sommet d'un réseau. En capturant non seulement son positionnement à un instant précis (avec qui ce sommet interagit) et son dynamisme (comment ont évolué ses interactions passées), on peut potentiellement mieux prédire ses liens futurs.

La construction de représentations dynamiques ont été abordées pour l'étude de l'évolution sémantique des mots [Bamler and Mandt, 2017, Yao et al., 2018, Rudolph and Blei, 2018]. Ces méthodes ont été adaptées à l'étude du dynamisme dans les réseaux [Li et al., 2017, Zhou et al., 2018, Du et al., 2018]. Celles-ci proposent majoritairement de découper le réseau en tranches temporelles, c'est-à-dire à construire une succession de réseaux statiques représentant chacun l'état du réseau dynamique pour différents horodatages. Le dynamisme est alors décrit par un ensemble discret de représentations, chacune d'elles décrivant un sommet pour une tranche temporelle t. Ces représentations sont apprises en contraignant des modèles traditionnels de plongement de réseau de sorte à ce que la représentation d'un sommet à l'instant t soit proche de celle de l'instant $t-1$. Cette approche peut être relativement coûteuse en calcul puisqu'il faut construire autant de représentations par sommet qu'il y a de tranches temporelles.

Une approche que nous souhaitons aborder dans des travaux futurs consiste à adapter les techniques traditionnelles utilisées en traitement du signal pour encoder des séries temporelles dans des espaces vectoriels de dimension faible. Les séries de Fourrier et les B-Splines fournissent des bases vectorielles qui peuvent être utilisées dans les modèles

additifs généralisés [Hastie and Tibshirani, 1990]. Ceux-ci permettent d'approcher une série temporelle en estimant par régression logistique des coefficients de pondération de ces bases vectorielles. Nous pensons que ces méthodes peuvent s'intégrer aux récents développements en plongement de réseau. L'idée serait alors de construire une unique représentation de chaque sommet qui, combinée à une base vectorielle de fonctions temporelles, permettrait d'inférer une représentation d'un sommet évoluant dans le temps, sans passer par un coûteux découpage en tranches temporelles.

5.2.3 Réseaux et systèmes de recommandation

L'analyse des réseaux et les systèmes de recommandation sont longtemps restés deux domaines distants. Ces derniers ont connu un développement constant ces dernières années, de nombreuses innovations étant insufflées par des initiatives venant de l'industrie, comme autour du filtrage collaboratif [Koren and Bell, 2015]. Récemment, l'avènement des *Graph Neural Networks* a inspiré de nombreux travaux autour des systèmes de recommandation [Monti et al., 2017, Fan et al., 2019]. Le caractère générique de ces approches permettent d'aborder des tâches de recommandation sous l'angle de l'apprentissage de représentations [Ying et al., 2018]. Les différentes entités dans ces systèmes (candidats et documents pour la recherche d'experts ou utilisateurs et produits pour la vente en ligne) et leurs interactions (qui écrit quoi ou qui achète quoi) peuvent naturellement être considérés comme des instances de réseaux hétérogènes potentiellement attribués. Apprendre des représentations de ces entités peut permettre d'aborder diverses tâches de recommandations à partir d'uniques représentations préalablement apprises. Pour cette raison, nous souhaiterions explorer ces modèles pour continuer nos travaux sur la recherche d'experts. Cependant, de nombreux aspects concernant la recommandation sont encore à étudier dans le cadre de ces approches, tels que leur interprétabilité, aspect de plus en plus important des systèmes de recommandation, ou leurs capacité à induire de nouvelles représentations, pouvant rencontrer les problèmes de démarrage à froid.

Bibliographie

[Abu-El-Haija et al., 2018] Abu-El-Haija, S., Perozzi, B., Al-Rfou, R., and Alemi, A. A. (2018). Watch your step : Learning node embeddings via graph attention. In *Advances in Neural Information Processing Systems*, pages 9180–9190.

[Andersen et al., 2006] Andersen, R., Chung, F., and Lang, K. (2006). Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE.

[Apostel et al., 1957] Apostel, L., Mandelbrot, B. B., and Morf, A. (1957). Logique, language et théorie de l'information.

[Arora et al., 2016] Arora, S., Liang, Y., and Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*.

[Balog et al., 2006] Balog, K., Azzopardi, L., and De Rijke, M. (2006). Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50. ACM.

[Balog et al., 2012] Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., Si, L., et al. (2012). Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3) :127–256.

[Balog et al., 2010] Balog, K., Serdyukov, P., and De Vries, A. P. (2010). Overview of the trec 2010 entity track. Technical report, Norwegian Univ of Science and Technology Trondheim.

[Bamler and Mandt, 2017] Bamler, R. and Mandt, S. (2017). Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 380–389. JMLR. org.

[Barabási, 2009] Barabási, A.-L. (2009). Scale-free networks : a decade and beyond. *science*, 325(5939) :412–413.

[Belkin and Niyogi, 2002] Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591.

[Bellman, 1966] Bellman, R. (1966). Dynamic programming. *Science*, 153(3731) :34–37.

[Bellogín et al., 2017] Bellogín, A., Castells, P., and Cantador, I. (2017). Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20(6) :606–634.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb) :1137–1155.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan) :993–1022.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5 :135–146.

[Bojchevski and Günnemann, 2018] Bojchevski, A. and Günnemann, S. (2018). Deep gaussian embedding of graphs : Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*.

[Bouma, 2009] Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

[Breitkreutz et al., 2007] Breitkreutz, B.-J., Stark, C., Reguly, T., Boucher, L., Breitkreutz, A., Livstone, M., Oughtred, R., Lackner, D. H., Bähler, J., Wood, V., et al. (2007). The biogrid interaction database : 2008 update. *Nucleic acids research*, 36(suppl_1) :D637–D640.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine.

[Brochier, 2019a] Brochier, R. (2019a). Apprentissage de représentation appliqué à la recommandation pour la littérature scientifique. In *In Conférence jointe CORIA-EARIA 2019*.

[Brochier, 2019b] Brochier, R. (2019b). Representation learning for recommender systems with application to the scientific literature. In *Companion Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Steering Committee.

[Brochier et al., 2020a] Brochier, R., Gourru, A., Guille, A., and Velcin, J. (2020a). New datasets and a benchmark of document network embedding methods for scientific expert finding. In *Bibliometric-enhanced Information Retrieval : 10th International BIR Workshop at ECIR*.

[Brochier et al., 2018a] Brochier, R., Guille, A., Rothan, B., and Velcin, J. (2018a). Impact of the query set on the evaluation of expert finding systems. In *3rd Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL) at SIGIR*.

[Brochier et al., 2019a] Brochier, R., Guille, A., and Velcin, J. (2019a). Global vectors for node representations. In *Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Steering Committee.

[Brochier et al., 2019b] Brochier, R., Guille, A., and Velcin, J. (2019b). Link prediction with mutual attention for text-attributed networks. In *Companion Proceedings of The 2019 World Wide Web Conference*. International World Wide Web Conferences Steering Committee.

[Brochier et al., 2020b] Brochier, R., Guille, A., and Velcin, J. (2020b). Inductive document network embedding with topic-word attention. In *Proceedings of the 42nd European Conference on Information Retrieval Research*. Springer.

[Brochier et al., 2018b] Brochier, R., Guille, A., Velcin, J., Rothan, B., and Cioccio, D. (2018b). Peerus review : a tool for scientific experts finding. In *Extraction et Gestion des Connaissances (EGC)*.

[Buckley et al., 2007] Buckley, C., Dimmick, D., Soboroff, I., and Voorhees, E. (2007). Bias and the limits of pooling for large collections. *Information retrieval*, 10(6) :491–508.

[Buntine, 2002] Buntine, W. (2002). Variational extensions to em and multinomial pca. In *European Conference on Machine Learning*, pages 23–34. Springer.

[Cao et al., 2015] Cao, S., Lu, W., and Xu, Q. (2015). Grarep : Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900.

[Chang and Blei, 2009] Chang, J. and Blei, D. (2009). Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv :1406.1078*.

[Cifariello et al., 2019] Cifariello, P., Ferragina, P., and Ponza, M. (2019). Wiser : A semantic approach for expert finding in academia based on entity linking. *Inf. Syst.*, 82 :1–16.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug) :2493–2537.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. A. (2000). *Multidimensional scaling*. Chapman and hall/CRC.

[Craswell et al., 2005] Craswell, N., de Vries, A. P., and Soboroff, I. (2005). Overview of the trec 2005 enterprise track. In *Trec*, volume 5, pages 199–205.

[Craswell et al., 2001] Craswell, N., Hawking, D., Vercoustre, A.-M., and Wilkins, P. (2001). P@ noptic expert : Searching for experts not just for documents. In *Ausweb Poster Proceedings, Queensland, Australia*, volume 15, page 17.

[Craswell et al., 2008] Craswell, N., Zoeter, O., Taylor, M., and Ramsey, B. (2008). An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94.

[Dargahi Nobari et al., 2017] Dargahi Nobari, A., Sotudeh Gharebagh, S., and Neshati, M. (2017). Skill translation models in expert finding. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1057–1060. ACM.

[de Sa, 1994] de Sa, V. R. (1994). Learning classification with unlabeled data. In *Advances in neural information processing systems*, pages 112–119.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391–407.

[Denil et al., 2014] Denil, M., Demiraj, A., Kalchbrenner, N., Blunsom, P., and de Freitas, N. (2014). Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv :1406.3830*.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

[Dong et al., 2017] Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec : Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144.

[Dong et al., 2015] Dong, Y., Zhang, J., Tang, J., Chawla, N. V., and Wang, B. (2015). Coupledlp : Link prediction in coupled networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 199–208.

[Du et al., 2018] Du, L., Wang, Y., Song, G., Lu, Z., and Wang, J. (2018). Dynamic network embedding : An extended approach for skip-gram based network embedding. In *IJCAI*, pages 2086–2092.

[Egghe and Rousseau, 1990] Egghe, L. and Rousseau, R. (1990). *Introduction to informetrics : Quantitative methods in library, documentation and information science*. Elsevier Science Publishers.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear : A library for large linear classification. *Journal of machine learning research*, 9(Aug) :1871–1874.

[Fan et al., 2019] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019). Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426.

[Fang et al., 2010] Fang, Y., Si, L., and Mathur, A. P. (2010). Discriminative models of integrating document evidence and document-candidate associations for expert search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 683–690. ACM.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8) :861–874.

[Fischer et al., 2019] Fischer, T., Remus, S., and Biemann, C. (2019). Lt expertfinder : An evaluation framework for expert finding methods. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 98–104.

[Fouss et al., 2007] Fouss, F., Pirotte, A., Renders, J.-M., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3) :355–369.

[Fruchterman and Reingold, 1991] Fruchterman, T. M. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software : Practice and experience*, 21(11) :1129–1164.

[Gaussier and Goutte, 2005] Gaussier, E. and Goutte, C. (2005). Relation between plsa and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602.

[Giles et al., 1998] Giles, C. L., Bollacker, K., and CiteSeer, S. L. (1998). An automatic citation indexing system. In *Digital Libraries*, pages 89–98.

[Giraud, 2014] Giraud, C. (2014). *Introduction to high-dimensional statistics*, volume 138. CRC Press.

[Goebel et al., 2018] Goebel, R., Chander, A., Holzinger, K., Lecue, F., Akata, Z., Stumpf, S., Kieseberg, P., and Holzinger, A. (2018). Explainable ai : the new 42 ? In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 295–303. Springer.

[Gourru et al., 2020] Gourru, A., Guille, A., Velcin, J., and Jacques, J. (2020). Document network projection in pretrained word embedding space. In *European Conference on Information Retrieval*, pages 150–157. Springer.

[Goyal and Ferrara, 2018] Goyal, P. and Ferrara, E. (2018). Graph embedding techniques, applications, and performance : A survey. *Knowledge-Based Systems*, 151 :78–94.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation : A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

[Hamilton et al., 2017a] Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

[Hamilton et al., 2017b] Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs : Methods and applications. *arXiv preprint arXiv :1709.05584*.

[Hastie and Tibshirani, 1990] Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*, volume 43. CRC press.

[He and Niyogi, 2004] He, X. and Niyogi, P. (2004). Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8) :1735–1780.

[Hoffman et al., 2010] Hoffman, M., Bach, F. R., and Blei, D. M. (2010). Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv :1801.06146*.

[Huang et al., 2017] Huang, X., Li, J., and Hu, X. (2017). Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 633–641. SIAM.

[Jeh and Widom, 2002] Jeh, G. and Widom, J. (2002). Simrank : a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543.

[Johnson et al., 2018] Johnson, R., Watkinson, A., and Mabe, M. (2018). The stm report. *An overview of scientific and scholarly publishing. 5th edition October*.

[Kas, 2011] Kas, M. (2011). Structures and statistics of citation networks. Technical report, Carnegie Mellon's Department of Electrical and Computer Engineering.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv :1408.5882*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam : A method for stochastic optimization. *3rd International Conference for Learning Representations (ICLR), San Diego, 2015*.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv :1611.07308*.

[Kleinbaum et al., 2002] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic regression*. Springer.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5) :604–632.

[Koehn, 2009] Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

[Koren and Bell, 2015] Koren, Y. and Bell, R. (2015). Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer.

[Kudo, 2018] Kudo, T. (2018). Subword regularization : Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv :1804.10959*.

[Lan et al., 2019] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert : A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

[Landauer et al., 1998] Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3) :259–284.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

[LeCun et al., 2006] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0).

[Lee and Seung, 2001] Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

[Lee et al., 2019] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer : A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753.

[Levy and Goldberg, 2014] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

[Levy et al., 2015] Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3 :211–225.

[Li et al., 2017] Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., and Liu, H. (2017). Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396.

[Liben-Nowell and Kleinberg, 2007] Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7) :1019–1031.

[Liberzon et al., 2011] Liberzon, A., Subramanian, A., Pinchback, R., Thorvaldsdóttir, H., Tamayo, P., and Mesirov, J. P. (2011). Molecular signatures database (msigdb) 3.0. *Bioinformatics*, 27(12) :1739–1740.

[Lin et al., 2017] Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv :1703.03130*.

[Liu et al., 2018] Liu, J., He, Z., Wei, L., and Huang, Y. (2018). Content to node : Self-translation network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1794–1802.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*.

[Lü and Zhou, 2011] Lü, L. and Zhou, T. (2011). Link prediction in complex networks : A survey. *Physica A : statistical mechanics and its applications*, 390(6) :1150–1170.

[Lu and Getoor, 2003] Lu, Q. and Getoor, L. (2003). Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv :1508.04025*.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov) :2579–2605.

[Macdonald and Ounis, 2006] Macdonald, C. and Ounis, I. (2006). Voting for candidates : adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396. ACM.

[Macdonald et al., 2007] Macdonald, C., Ounis, I., and Soboroff, I. (2007). Overview of the trec 2007 blog track. In *TREC*, volume 7, pages 31–43.

[Mahoney, 2011] Mahoney, M. (2011). Large text compression benchmark. *URL : http ://www. mattmahoney. net/text/text. html*.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

[Marsden and Friedkin, 1993] Marsden, P. V. and Friedkin, N. E. (1993). Network studies of social influence. *Sociological Methods & Research*, 22(1) :127–151.

[Martins and Astudillo, 2016] Martins, A. and Astudillo, R. (2016). From softmax to sparsemax : A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

[Masci et al., 2011] Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer.

[McCallum et al., 2000] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2) :127–163.

[McInnes et al., 2018] McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). Umap : Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29).

[McPherson et al., 2001] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather : Homophily in social networks. *Annual review of sociology*, 27(1) :415–444.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mimno and Thompson, 2017] Mimno, D. and Thompson, L. (2017). The strange geometry of skip-gram with negative sampling. In *Empirical Methods in Natural Language Processing*.

[Mislevy and Riconscente, 2011] Mislevy, R. J. and Riconscente, M. M. (2011). Evidence-centered assessment design. In *Handbook of test development*, pages 75–104. Routledge.

[Mnih and Hinton, 2009] Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

[Monti et al., 2017] Monti, F., Bronstein, M., and Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707.

[Morin and Bengio, 2005] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1) :3–26.

[Natarajan and Dhillon, 2014] Natarajan, N. and Dhillon, I. S. (2014). Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12) :i60–i68.

[Newman, 2003] Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, 45(2) :167–256.

[Newman and Girvan, 2004] Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113.

[Ng et al., 2002] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering : Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

[Nickel and Kiela, 2017] Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

[Niculae and Blondel, 2017] Niculae, V. and Blondel, M. (2017). A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, pages 3338–3348.

[Noroozi and Favaro, 2016] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

[Orduña-Malea et al., 2015] Orduña-Malea, E., Ayllón, J. M., Martín-Martín, A., and López-Cózar, E. D. (2015). Methods for estimating the size of google scholar. *Scientometrics*, 104(3) :931–949.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking : Bringing order to the web. Technical report, Stanford InfoLab.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv :1802.05365*.

[Porteous et al., 2008] Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., and Welling, M. (2008). Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577.

[Qiu et al., 2018] Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. (2018). Network embedding as matrix factorization : Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8) :9.

[Riahi et al., 2012] Riahi, F., Zolaktaf, Z., Shafiei, M., and Milios, E. (2012). Finding expert users in community question answering. In *Proceedings of the 21st International Conference on World Wide Web*, pages 791–798. ACM.

[Ribeiro et al., 2017] Ribeiro, L. F., Saverese, P. H., and Figueiredo, D. R. (2017). struc2vec : Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394.

[Romero et al., 2013] Romero, D. M., Tan, C., and Ugander, J. (2013). On the interplay between social and topical structure. In *Seventh International AAAI Conference on Weblogs and Social Media*.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500) :2323–2326.

[Rudolph and Blei, 2018] Rudolph, M. and Blei, D. (2018). Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference*, pages 1003–1011.

[Schlichtkrull et al., 2018] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

[Sechidis et al., 2011] Sechidis, K., Tsoumakas, G., and Vlahavas, I. (2011). On the stratification of multi-label data. *Machine Learning and Knowledge Discovery in Databases*, pages 145–158.

[Sen et al., 2008] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3) :93–93.

[Sennrich et al., 2015] Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv :1508.07909*.

[Serdyukov et al., 2008] Serdyukov, P., Rode, H., and Hiemstra, D. (2008). Modeling multi-step relevance propagation for expert finding. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1133–1142. ACM.

[Shen et al., 2014] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 101–110.

[Srivastava and Sutton, 2017] Srivastava, A. and Sutton, C. (2017). Autoencoding variational inference for topic models. *Proceedings of International Conference on Learning Representations (ICLR)*.

[Sun and Han, 2012] Sun, Y. and Han, J. (2012). Mining heterogeneous information networks : principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2) :1–159.

[Sun et al., 2011] Sun, Y., Han, J., Yan, X., Yu, P. S., and Wu, T. (2011). Pathsim : Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11) :992–1003.

[Sun et al., 2013] Sun, Y., Norick, B., Han, J., Yan, X., Yu, P. S., and Yu, X. (2013). Pathselclus : Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3) :1–23.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Tang et al., 2015] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line : Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077.

[Tang et al., 2009] Tang, J., Sun, J., Wang, C., and Yang, Z. (2009). Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816.

[Tang et al., 2008] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer : extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM.

[Tenenbaum et al., 2000] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500) :2319–2323.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics.

[Tsitsulin et al., 2018] Tsitsulin, A., Mottin, D., Karras, P., and Müller, E. (2018). Verse : Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, pages 539–548.

[Tu et al., 2017] Tu, C., Liu, H., Liu, Z., and Sun, M. (2017). Cane : Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1722–1731.

[Van Gysel et al., 2016] Van Gysel, C., de Rijke, M., and Worring, M. (2016). Unsupervised, efficient and semantic expertise retrieval. In *WWW*, volume 2016, pages 1069–1079. The International World Wide Web Conferences Steering Committee.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv :1710.10903*.

[Von Luxburg, 2007] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416.

[Wang et al., 2016] Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234.

[Wold et al., 1987] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3) :37–52.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell : Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

[Yan et al., 2005] Yan, S., Xu, D., Zhang, B., and Zhang, H.-J. (2005). Graph embedding : A general framework for dimensionality reduction. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 830–837. IEEE.

[Yang et al., 2015] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. (2015). Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[Yang et al., 2010] Yang, Z., Guo, J., Cai, K., Tang, J., Li, J., Zhang, L., and Su, Z. (2010). Understanding retweeting behaviors in social networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1633–1636.

[Yang et al., 2016] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics : human language technologies*, pages 1480–1489.

[Yao et al., 2018] Yao, Z., Sun, Y., Ding, W., Rao, N., and Xiong, H. (2018). Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681.

[Ying et al., 2018] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983.

[Yuan et al., 2014] Yuan, G., Murukannaiah, P. K., Zhang, Z., and Singh, M. P. (2014). Exploiting sentiment homophily for link prediction. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 17–24.

[Yuan et al., 2020] Yuan, S., Zhang, Y., Tang, J., Hall, W., and Cabotà, J. B. (2020). Expert finding in community question answering : a review. *Artificial Intelligence Review*, 53(2) :843–874.

[Zhang et al., 2007a] Zhang, J., Ackerman, M. S., and Adamic, L. (2007a). Expertise networks in online communities : structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, pages 221–230. ACM.

[Zhang et al., 2007b] Zhang, J., Tang, J., and Li, J. (2007b). Expert finding in a social network. In *International Conference on Database Systems for Advanced Applications*, pages 1066–1069. Springer.

[Zhao and Tan, 2016] Zhao, R. and Tan, V. Y. (2016). Online nonnegative matrix factorization with outliers. *IEEE Transactions on Signal Processing*, 65(3) :555–570.

[Zhao et al., 2014] Zhao, Z., Zhang, L., He, X., and Ng, W. (2014). Expert finding for question answering via graph regularized matrix completion. *IEEE Transactions on Knowledge and Data Engineering*, 27(4) :993–1004.

[Zhou et al., 2018] Zhou, L., Yang, Y., Ren, X., Wu, F., and Zhuang, Y. (2018). Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*.