Prédiction de liens et d'hyperliens : applications de l'apprentissage de représentations à la contextualisation de contenus textuels dans les bases de connaissances

Jean Dupuy

N° d'ordre NNT : 2022LYSE2043

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512

Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 6 mai 2022, par :

Jean DUPUY

Prédiction de liens et d'hyperliens.

Applications de l'apprentissage de représentations à la contextualisation de contenus textuels dans les bases de connaissances.

Devant le jury composé de :

Christophe GRAVIER, Professeur des universités, Université Jean Monnet, Président

Sandra BRINGAY, Professeure des universités, Université de Montpellier, Rapporteure

Peggy CELLIER, Maîtresse de conférences HDR, Université Rennes 1, Rapporteure

Julien JACQUES, Professeure des universités, Université Lumière Lyon 2, Co-Directeur de thèse

Adrien GUILLE, Maître de conférences, Université Lumière Lyon 2, Co-Directeur de thèse

Contrat de diffusion

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

Prédiction de liens et d'hyperliens : applications de l'apprentissage de représentations à la contextualisation de contenus textuels dans les bases de connaissances

Jean DUPUY

Devant le jury composé de :

Sandra BRINGAY	**Rapportrice**
Professeur des Universités, Université Montpellier 3	
Peggy CELLIER	**Rapportrice**
Maître de conférences, Université Rennes 1	
Christophe GRAVIER	**Examinateur**
Professeur des universités, Université Jean Monnet Saint-Etienne	
Julien JACQUES	**Directeur de thèse**
Professeur des Universités, Université Lumière Lyon 2	
Adrien GUILLE	**Directeur de thèse**
Maître de Conférences, Université Lumière Lyon 2	

Abstract

The work presented in this thesis was made in collaboration with MeetSYS, a Knowledge Management consulting company. We aim to develop tools for content recommendation in knowledge bases, which are mostly constituted of textual documents, by taking adventage of their graph structure.

We first review state of the art methods in representation learning, including textual information, graphs and documents network. Then, we present a study on using a simple translation method between document and non-attributed graph embeddings. We experimentally show that projecting a new document embedding in the graph embedding space improves performances on a link prediction task. The second contribution we describe, *Contextual Relational Topic Model* (CRTM), is a topic model built to predict which work would carry a hyperlink between two documents. We discuss different ways to take the context of apparition of a hyperlink into account, and assess CRTM's performances through experiments.

Keywords : Representation learning, document embedding, graph embedding, topic modeling, document networks, anchor prediction

Résumé

Les travaux présentés dans ce manuscrit de thèse ont été réalisés en collaboration avec l'entreprise MeetSYS, qui propose des solutions et du conseil en gestion des connaissances et en résolution de problèmes industriels. L'objectif est de développer des outils pour la recommandation de contenus dans des bases de connaissances, principalement constituées de documents textuels, en prenant en compte leur structure en réseaux.

Nous proposons en premier lieu un état de l'art revenant sur les principales méthodes en apprentissage de représentations de documents, de graphes et de réseaux documentaires. Nous présenterons ensuite une étude sur l'utilisation d'une méthode simple de traduction entre des représentations de textes et des représentations de sommets non attribués dans un graphe. Nous montrons expérimentalement qu'après avoir appris une matrice de projection entre les deux espaces de représentation, la projection d'un contenu textuel dans l'espaces des sommets améliore les performances en prédiction de liens. Nous discuterons ensuite de la seconde contribution de ce manuscrit de thèse, *Contextual Relational Topic Model*, une méthode de prédiction des mots portant les hyperliens entre deux documents. Nous détaillons au cours de cette section les différentes voies envisagées afin de prendre en compte le contexte d'apparition d'un hyperlien, et étudions expérimentalement la qualité des liens proposés.

Mots clefs : apprentissage de représentation, plongements de document, plongements de graphes, modélisation de thématiques, réseaux de documents, prédiction d'ancres

Remerciements

Si le doctorat porte encore parfois l'image d'un travail solitaire et égotiste, je tiens à m'inscrire en faux et plaide avec plaisir que cette production est le résultat d'un effort collectif tant au niveau scientifique qu'humain, et qu'il convient de remercier comme il se doit toutes les personnes qui y ont contribué, de (très) près ou de (plus) loin.

Je tiens en tout premier lieu à exprimer toute ma gratitude envers mes encadrants de thèse, Julien Jacques et Adrien Guille. Sans vos conseils, votre soutien et votre patience ce travail n'aurait jamais pu voir le jour, et je prends la pleine mesure de ce que vous m'avez apporté pendant ces trois années et demie.

Mes plus sincères remerciements vont également à Sandra Bringay et Peggy Cellier pour avoir accepté de rapporter cette thèse, ainsi qu'à Christophe Gravier d'avoir accepté d'être examinateur.

Je remercie chaleureusement l'ensemble de MeetSYS : Olivier Nerot, Manuel Balbo et Jérôme Laforcade d'avoir rendu cette thèse CIFRE possible ; Christophe Merle, qui, non content d'avoir toujours été un formidable camarade, m'a longuement épaulé dans les méandres administratifs sur lesquels il nous aura fallu naviguer ; toute l'équipe de Lyon, Pierre-Louis (*obrigado*), Nico, Xavier, Zuzana, Valentin, Antonin et Gabrielle, pour l'excellente ambiance de travail, toujours conviviale mais néanmoins studieuse ; et enfin l'équipe de Paris, pour tous les moments sympathiques passés ensemble lors des séminaires.

Je remercie également tous les membres d'ERIC pour leur accueil et la vie au sein du laboratoire, et tout particulièrement Habiba pour toute l'aide qu'elle a su m'apporter et les échanges que nous pûmes avoir au cours de ces trois années (et demie !), ainsi qu'à tous les camarades doctorant·es (mais pas que !) rencontré·es sur le chemin : (Dr) Antoine, Martial, Gaël, Enzo, Loic, (Dr) Margot, (Dr) Robin, (Dr) Clément, (Dr) Charles, Etienne, Rheda, (Dr) Nico, Camille et (Dr) Arwa ; merci pour la bonne humeur tant intra qu'extra recherche.

S'en suivra à partir de ces mots une longue litanie de remerciements aussi personnels que référencés, semblant inhérente à l'exercice, et qui ne prendra fin qu'au moment d'exprimer à ma famille mon infinie gratitude pour tout le soutien et la confiance qu'elle a su me témoigner depuis que le monde est monde (modulo une trentaine d'années), et sans qui je ne serais très certainement pas en train de rédiger ce message aujourd'hui. Merci donc à Mor-

gane pour m'avoir accompagné tout ce temps (*c'était facile !*) et d'avoir supporté angoisses et annulations de vacances avec un stoïcisme t'éloignant chaque jour davantage de Mitia. Merci Dimitri d'avoir été si souvent pourvoyeur de jeux, de nourriture et d'une fraternité fortement typée et fonctionnellement pure. Merci Clélia pour la discussion, les verres, les cookies (et la correction). Merci Amandine d'être égale à toi même, ça ne gâche vraiment rien. Merci Grincheux d'être égal à toi même, ce qui est parfois un peu gâcher ton talent. Merci Mathieu, pour tout (ton soutien, ton irréelle gentillesse et les brindibous). Merci à Estelle d'exister, tout simplement. Merci à Tom de m'avoir entraîné dans cette galère. Merci à Marcus & Trane pour les échanges, la production et la curiosité. Merci à Baptiste, Lionel et Rémi d'avoir été depuis si longtemps des camarades d'excellente facture, vous êtes beaux comme une startup qui coule. Merci à Paradoxycal d'avoir fait naître des docteurs au moins aussi bons musiciens qu'amis. Germain, les airs iodés des ballades irlandaises te vont si bien. Alexis, quel plaisir que ta tournée t'ait amené à Lyon. Antoine, j'espère que l'on pourra deviser à nouveau de toutes les musiques étranges. Tes étudiants ne mesurent pas leur chance. Merci à Roxanne de m'avoir introduit à l'éthologie en échange de menus cours de stats, et de m'avoir appris qu'Heidegger pouvait être une racine du véganisme. Merci à Florian pour les jeux, les maths, la bonne humeur, le milliard de lions, et plus généralement pour le formidable isomorphisme des graphes de nos intérêts. Merci à Émilie et Adrien d'être des amies de tant de valeur qu'aucune accumulation de signes arbitraires ne saura jamais l'exprimer pleinement. Merci à Katy Perry

• • • [1]

1. Afin de ne pas apporter de condition d'arrêt à ces remerciements, et qu'ils puissent continuer à tout jamais, je ne reviendrais pas sur l'étendue de ma reconnaissance à l'égard de ma famille, dont elle connaît déjà, je l'espère, toute l'ampleur.

Table des matières

Table des figures

Chapitre 1

Introduction

Au cours de ce chapitre d'introduction, nous commencerons par présenter le contexte dans lequel prennent place les travaux réalisés au cours de ce doctorat. Suite à cela, nous discuterons de axes de recherches propres aux problématiques rencontrées par l'entreprise commanditant cette thèse. Enfin, nous détaillerons les différentes contributions proposées dans ce manuscrit, et en présenterons le plan.

1.1 Contexte industriel

Le terme de big-data commence à apparaître au cours des années 90, sous l'impulsion d'informaticiens américains issus du secteur privé [1]. Ce terme désigne alors la somme de données de plus en plus importante à produire et stocker. Cette augmentation s'opère simultanément sur trois grands axes : le volume des données, leur variété, et la vélocité de leur production. Ainsi, on acquière et stocke de plus en plus de données, de nature différentes, et ce chaque jour un peu plus.

Des sites comme Wikipedia [2] sont emblématiques de la massification des volumes de données qui prend place dans le web depuis une vingtaine d'années, comme l'illustre la Figure 1.1 présentant l'évolution du volume d'articles anglophones depuis la création du site. Le 15 Janvier 2001, Jimmy Wales et Larry Sanger annonçaient officiellement la mise en ligne de l'encyclopédie en ligne. Vingt ans plus tard, la version anglophone du site compte plus de six millions de pages. Depuis les dix derrières années, un peu plus de 600 nouveaux articles anglophones voient le jour quotidiennement, et le volume de contribution dans les autres langues tend à suivre également cette tendance. De façon beaucoup plus générale, cette augmentation se transpose à tous les autres types de données produites et stockées dans le monde, principalement sous l'effet du développement des objets connectés, et l'on considère aujourd'hui que celle-ci suit une évolution exponentielle.

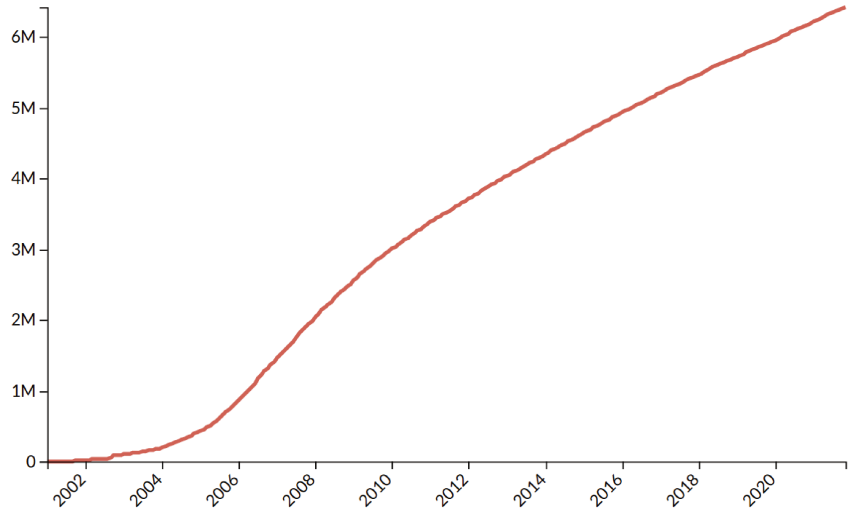


FIGURE 1.1 – Évolution du nombre d'articles présents dans la version anglophone de Wikipédia depuis sa création.

Cet immense volume de données textuelles, grandissant de jour en jour, couplé aux pro-

1. Steve Lohr (1 Février 2013). "The Origins of 'Big Data' : An Etymological Detective Story". The New York Times. Consulté le 15 Janvier 2022

2. https://wikipedia.org

grès continuels des capacités de calculs et de stockage, a fait naître un important besoin d'en automatiser le traitement. En effet, selon par exemple la typologie proposée par Ackoff, 1989, une donnée brute n'est d'aucune utilité en soi, et c'est par son traitement que l'on peut en tirer de l'information, qui sera, elle, exploitable. Si la lecture constitue un moyen efficace pour traiter un nombre limité de documents, le volume aujourd'hui disponible dépasse de loin nos capacités de lecture. Dans le cas des données textuelles qui nous intéressent ici, ce *Traitement Automatique de la langue* (TAL), plus souvent désigné par son appellation anglophone *Natural Language Processing* (NLP), est devenu un des champs les plus importants de la recherche en apprentissage automatique[3]. Les applications portent sur un large champ de tâches, couvrant des aspects syntaxiques (comme par exemple l'étiquetage morpho-syntaxique ou la lemmatisation), sémantiques (traduction ou résumé automatique) ou encore documentaires (recherche d'informations, classification de documents...). Cette information nécessite cependant qu'un utilisateur se l'approprie, en intégrant cette information et en la contextualisant avec le reste de ses connaissances. Là encore, le machine learning trouve une place dans ce processus, en permettant de faciliter cette mise en contexte.

La typologie d'Ackoff nous permet ainsi de mettre en évidence trois strates de gestion dans les organisations : la gestion des données, celle de l'information, et celle des connaissances. La gestion des données est une tâche d'ores et déjà bien connue, dans la mesure où elle est très tôt devenue un besoin primaire des organisations, et touche à des domaines de l'informatique très étudiés comme les bases de données, la collecte de données, etc.. La question de la gestion de l'information a elle aussi pris de l'importance à mesure que la quantité des données a augmenté. Il s'agit alors de pouvoir interpréter ces données, recueillies en masse de façon automatique, au moyen, par exemple, de méthodes de fouilles des données.

Les problématiques de gestion des connaissances, en revanche, ont mis plus de temps à s'imposer comme un enjeu crucial dans le domaine industriel [Ermine, 2003]. Les organisations devenant de plus en plus vastes, et les sujets abordés plus étendus et complexes, la nécessité de recueillir, conserver et rendre accessible la connaissance est allée croissante. Ce processus n'est cependant pas simple et peine à se mettre en place dans bon nombre d'organisations[4]. La gestion des connaissances, en tant que discipline formalisée et objet de recherche, a commencé à prendre forme durant la seconde moitié des années 90 [Ponzi, 2004], même si l'on peut déjà trouver des traces de ces questionnements dans certains cercles managériaux vers la fin des années 70.

Ce sont les travaux de Nonaka et al., 1995 qui deviennent la pierre angulaire de la gestion des connaissances durant les années 90. Ils y proposent une modélisation de la créa-

3. Par la suite, nous emploierons indifféremment ce terme et sa traduction anglaise *machine learning*.

4. https://www.lemonde.fr/archives/article/2000/10/10/la-gestion-des-connaissances-peine-a-trouver-sa
3715562_1819218.html

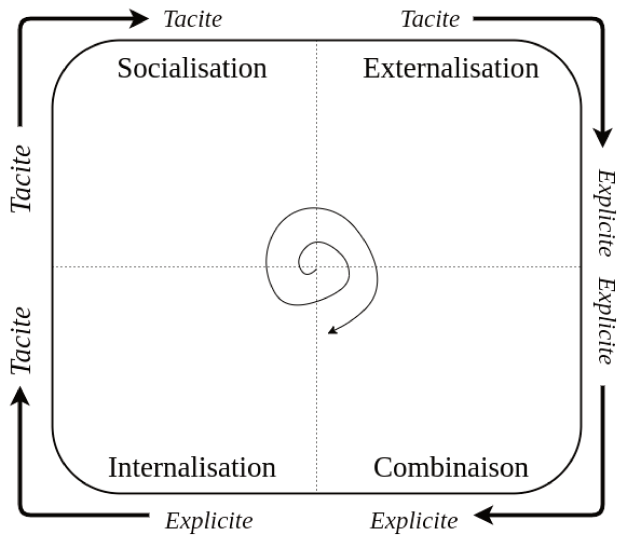


FIGURE 1.2 – Diagramme de transfert des connaissances, comme décrit par NONAKA et al., 1995.

tion de connaissances au sein des organisations, en identifiant deux facettes de celle-ci : la connaissance explicite, ce que l'on sait et que l'ont peut facilement formaliser (par exemple *Sur un vélo, pédaler permet d'avancer*) ; et la connaissance implicite, liée à l'expérience ou au savoir-faire, et qui s'acquière le plus souvent au travers d'une expérience sensible (pour continuer notre exemple, *savoir faire du vélo*). Les auteurs proposent ainsi quatre phases lors du partage d'une connaissance, chacune symbolisant la transmission d'un état vers un autre, et illustrée à la Figure 1.2 :

— La *Socialisation* correspond à la transmission du tacite au tacite. C'est une expérience sociale, reposant principalement sur le principe d'imitation et le partage ;
— L'*Externalisation* est une phase d'explicitation des connaissances tacites vers une connaissance implicite. Elle permet, en suivant un formalisme particulier, des images ou des métaphores, de transmettre à un public ne pouvant participer à l'expérience sensible de la socialisation. Elle peut donner lieu à un processus de sauvegarde dans un système d'information ;
— La *Combinaison* est l'étape de mise en relation des connaissances explicites entre elles, par le partage et l'organisation de documents.
— L'*Internalisation*, enfin, marque le passage d'une connaissance explicite à une connaissance implicite, par la consultation d'une connaissances explicite et l'application de celle-ci.

C'est dans ce cadre que MeetSYS [5], l'entreprise qui commandite cette thèse, se positionne. Fondée en 2013, elle propose une méthode d'interview et une plate-forme collaborative de

5. http://meetsys.com/

capitalisation et de partage des savoir-faire nommée i^2Kn[6]. Elle se place donc à la fois dans un processus d'externalisation et de combinaison des connaissances. Les données issues des entretiens avec les experts industriels sont formalisées et retranscrites dans la base de connaissances, dont l'architecture s'inspire des wiki. Là où chacune des pages traite d'un problème industriel documenté, d'un phénomène physique ou d'une méthode de production, celles-ci sont toutes liées entre elles soit par des liens globaux, soit par des hyperliens.

Le choix de cette architecture présente plusieurs avantages pour la modélisation des connaissances et la résolution de problèmes. Premièrement, la création d'hyperliens permet une mise en relation des contenus sans hiérarchisation. Ensuite, cette organisation des contenus est "plate", en ce sens qu'elle ne hiérarchise pas les pages et les concepts. Cette organisation présente un certain intérêt lorsqu'il s'agit de construire des « ponts » entre les différentes pages. En ne les assignant pas explicitement à une (ou plusieurs) catégories, et sans notion d'ordre entre elles, on laisse plus de latitude au lecteur (ou à un algorithme de recommandation).

La forme prise par *i2Kn* peut être fondamentalement vue comme un wiki cherchant à favoriser la navigation entre les pages et l'interaction avec les utilisateurs, tout en respectant les besoins des organisations en matière d'accès aux données et de gestion des rôles des utilisateurs. En effet, selon les règles en vigueur dans certaines organisations, les utilisateurs n'ont pas toujours la possibilité de consulter ou contribuer aux bases de secteurs différents. Si cet état de fait permet de maintenir une certaine norme en matière de confidentialité, cela impose des contraintes techniques, dont celle de pouvoir garantir une certaine indépendance au sein des bases. Cette architecture est illustrée à la Figure 1.3. L'utilisateur, une fois enregistré, ne se voit capable d'accéder qu'aux bases 1 et 2, ainsi qu'à une autre base appartenant à une autre organisation (par exemple, une base publique comme celle proposée par MeetSYS[7]).

La Figure 1.4 donne un exemple de l'interface d'une base de connaissance, que l'on peut découper en trois parties. La première, numéroté (1), rassemble les différents outils de navigation. La partie centrale, (2), rassemble le contenu à proprement parler. On peut y retrouver divers types de contenu, et également plusieurs types d'hyperliens. Les liens intra-bases sont symbolisés en bleu, et les liens inter-base en jaune. Enfin, la partie tout à droite de l'interface, notée (3), regroupe les autres éléments de navigation contextuelles. On y trouve les recommandations entres pages, la liste des contributeurs, les recommandations d'experts et les fichiers associés.

6. http://i2kn.com/
7. https://i4kn.meetsys.i2kn.com/

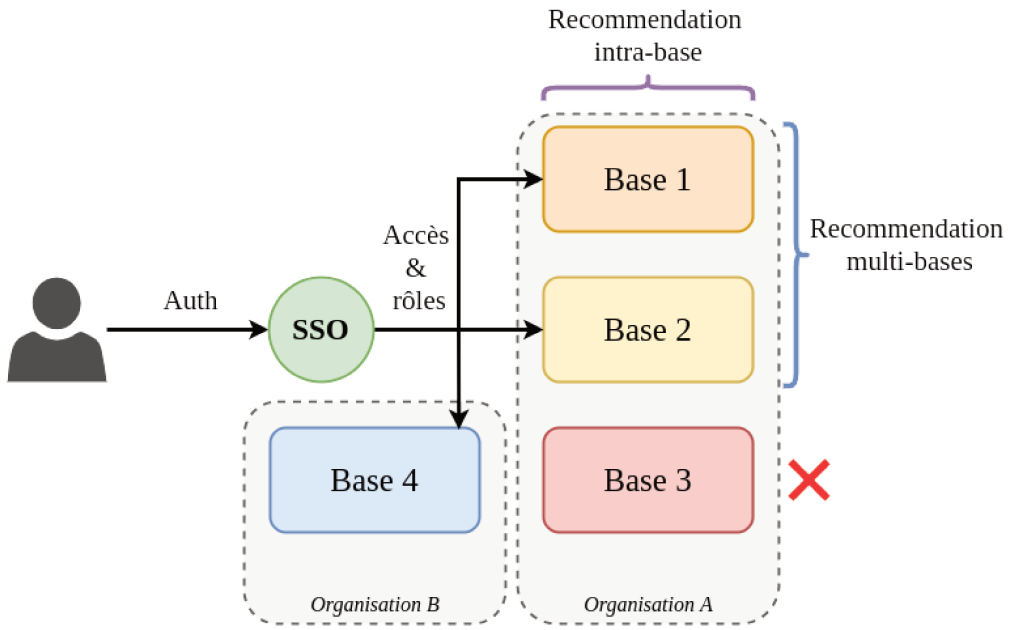


FIGURE 1.3 – Schémas de la gestion des droits et des accès. L'utilisateur, après s'est connecté à l'outil, n'a accès qu'aux bases 1 et 2 de l'organisation B, ainsi que la base 4 de la A. Bien qu'elle fasse partie de son organisation (B), il ne peux accéder à la base 3, car il n'en possède pas les droits. Les recommandations multi-bases ne devront alors pas prendre en compte les documents de cette base.

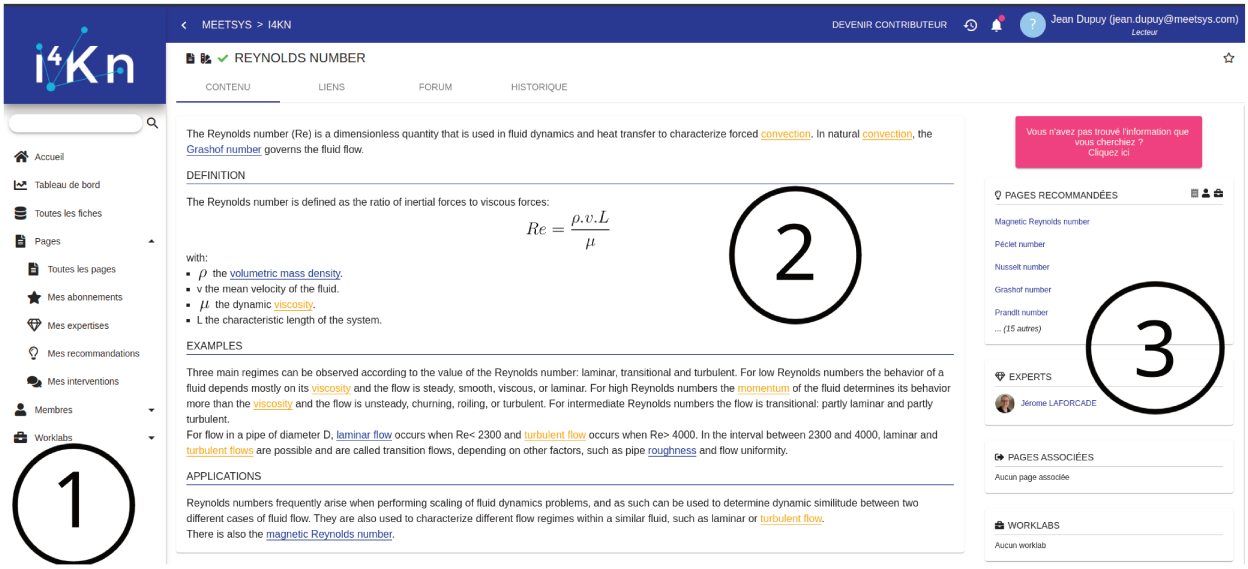


FIGURE 1.4 – Capture d'écran d'une base de connaissances.

1.2 Apprentissage de représentations

Afin de pouvoir traiter les documents textuels, les algorithmes d'intelligence artificielle ne peuvent se satisfaire de données symboliques ; ils nécessitent des *représentations* qui leur sont intelligibles ; cet enjeu étant en effet crucial pour les performances des méthodes d'appren-

tissage artificiel. Ces représentations peuvent être construites manuellement en sélectionnant les caractéristiques jugées les plus intéressantes. Cette façon de faire s'avère néanmoins complexe et fastidieuse, et la nécessité de construire des représentations moins dépendantes de la sélection des caractéristiques s'est faite de plus en plus pressante.

On définit l'apprentissage de représentation comme « *l'apprentissage des représentations des données qui rendent plus simple l'extraction d'informations lors de la construction d'un classifieur ou de tout autre prédicteur* » [Bengio, Courville et al., 2013]. Les représentations ne sont dès lors plus construites mais apprises, et sont spécifiées non plus les caractéristiques à conservées, mais les propriétés que l'on souhaite préserver. Les méthodes d'apprentissages peuvent être supervisées, ou non. Un autre intérêt se trouve également dans la réduction de la dimension des données, permettant d'obtenir des représentations denses et compactes, autrement dit des représentations de dimensions très inférieures à celles des données d'origine, très peu creuses. Cette réduction de dimensions permet de capter certains invariants ou régularités entre les données. On cherche également des représentations continue, telles que $f(x + \epsilon) \approx f(x)$, pour ϵ petit. Cette propriété permet aux algorithmes de machine learning de pouvoir généraliser à partir des données d'entraînement à des exemples qui leurs sont proches dans l'espace initial où elles prennent place.

Dans la suite de ce manuscrit, nous verrons que l'apprentissage de représentations de graphes partage énormément avec celui de mots et de documents. Cela s'explique par le fait que beaucoup de méthodes adaptées aux graphes s'appuient sur des méthodes originellement développées pour le texte, et utilisent des marches aléatoires, dont les suites de sommets peuvent être assimilées à des phrases, autrement dit des suites de mots. Le cas de l'apprentissage de représentations de réseaux de documents est quand à lui un peu différent, en ceci que l'information portée par le réseau joue souvent un rôle de « lissage » sur les représentations ; l'hypothèse de modélisation le plus souvent retenue étant que les documents liés dans le réseaux sont également proches par le sens.

1.3 Cas d'application

Les travaux présentés au long de ce manuscrit tenteront de s'inscrire dans une recherche de liens entre informations textuelles et structure des réseaux. C'est en effet la mise en lien des différents contenus entre eux qui permet la mise en place d'une base de connaissances utilisable et vivante.

Un élément constitutif des bases de connaissances peut être vu dans la génération des hyperliens au sein des textes, pointant vers les pages associées. La création de ces hyperliens requiert une intervention humaine. Ils peuvent être insérés lors de la rédaction de la page en question, comme cela se fait par exemple sur Wikipedia, ou alors générés à la volée, en cher-

chant dans chaque document l'apparition des mots constituants le titre d'une autre page. Ces deux méthodes se révèlent contraignantes, chacune à leur façon, pour les contributeurs. Dans le premier cas, l'auteur doit les spécifier manuellement, opération requérant de connaître le titre exact de la page pointée[8]. Dans le second, il devra, soit faire apparaître explicitement le titre d'une page lors de la rédaction, ou tout du moins ajouter un alias, ce qui rajoute une charge importante aux contributeur. Il devient alors intéressant de proposer une méthode permettant de proposer automatiquement les mots ou groupes de mots possiblement à l'origine d'hyperliens entre documents.

1.4 Contributions et plan du manuscrit

Au cours de ce manuscrit nous nous intéressons aux liens entre entre les contenus textuels et leurs organisation sous la forme de réseaux.

Le **chapitre 2** sera extensivement consacrée à un état de l'art sur les différentes méthodes existantes permettant l'apprentissage de représentations de corpus de documents, de réseaux, et de réseaux de documents textuels. Un travail préliminaire sur le sujet a été publié pour les rencontres jeunes chercheurs, en marge de la *Conférence en Recherche d'Information et Applications 2019* (CORIA19) [Dupuy, 2019].

Le **chapitre 3** portera sur les premiers travaux publiés lors de ce travail de thèse, à la conférence *Symposium on Applied Computing 2020* (SAC20) [Dupuy, 2020]. On se propose ici d'employer une méthode déjà utilisée en traduction automatique, les procrustes orthogonaux [Schönemann, 1966], afin de déterminer une transformation linéaire permettant le passage d'un espace de représentation des documents en réseau à celui de leurs contenus.

Le **chapitre 4** présentera la principale contribution de ce manuscrit, *Contextuel Relational Topic Model* (CRTM). Nous décrirons ici une méthode de modélisation de thématiques adaptée à la prédiction d'hyperliens au sein de corpus en réseau, en nous basant sur les travaux de Chang et al., 2009. Ce travail a été publié et présenté lors de la conférence *Extraction et Gestion des Connaissances 2021* (EGC21) [Dupuy et al., 2021]. Nous présenterons également une version plus aboutie de ce travail, apportant plusieurs améliorations, accepté à l'atelier Wiki Workshop 2022, adossé à la conférence The Web Conf 22.

Nous conclurons enfin ce manuscrit en résumant les différentes contributions, puis en développant les différentes perspectives de recherches qui émanent du travail présenté.

8. C'est par exemple le choix fait Wikipedia, où les hyperliens sont spécifiés de la façon suivante : `[[ ... | ... ]]`. La partie gauche correspond au texte qui sera affiché et que nous appellerons l'ancre de l'hyperlien. L'élément à droite de la barre verticale correspond lui au titre de la page pointée.

Chapitre 2

État de l'art

Dans cet état de l'art nous nous attacherons à proposer un état des lieux des différentes méthodes existantes permettant l'apprentissage de représentations de documents textuels, de réseaux, puis, à la confluence des deux, de corpus de documents structurés en réseaux. On définira tout au long de ce chapitre l'apprentissage de représentations comme une tâche visant à apprendre des représentations denses d'objets, de telle façon que la dimension de cette représentation soit d'une dimension très inférieure à celle des données d'origine (la taille du vocabulaire pour un document, ou le nombre de sommets pour un graphe), tout en cherchant à conserver diverses propriétés. Cette tâche est devenue cruciale aujourd'hui, tant le traitement de l'immense quantité de données générées au cours des dernières décennies est un enjeu important. Les méthodes présentées par la suite peuvent être apparentées à trois grandes familles d'algorithmes : méthodes génératives et apprentissage de thématiques, méthodes de réductions de dimensions et factorisation de matrice, et enfin, les méthodes reposant sur des architectures de réseaux de neurones profonds.

2.1 Plongement de mots et de documents

2.1.1 Méthodes statistiques

Une des façons les plus simples de représenter des documents au sein d'un corpus est l'utilisation d'une matrice de comptage termes-documents. On suppose alors l'ordre des mots comme une information non importante, et seule la présence des mots est prise en compte dans la représentation. On pose $X \in \mathbb{N}^{D \times V}$ une matrice de comptage termes-documents, où chacun des $x_{i,j}$ représente le nombre d'occurrence du mot $v_j \in \mathcal{V}$, avec $\mathcal{V}$ un vocabulaire de V mots, dans le document $d_i \in \mathcal{D}$, avec $\mathcal{D}$ un corpus de D documents. Cette représentation des documents, se basant uniquement sur des comptages et ne prenant pas en compte l'ordre d'apparition des mots est appelé *Sac de mots* (*Bag of words*).

Luhn, 1957 propose de se baser sur la fréquence (*Term Frequency*) d'apparition des mots, en faisant le postulat que l'importance d'un terme est proportionnel à sa fréquence d'apparition. En prenant en compte le nombre d'occurrences f_{d,w_j} d'un mot w_j dans d, on calcule :

$$tf(w_j, d) = \frac{f_{d,w_j}}{\sum_{k \in d} f_{d,w_k}} \tag{2.1}$$

Un document d peut ainsi être représenté comme un vecteur de taille V, où chaque coefficient sera la valeur de f_{d,w_j}, pour chaque $w_j \in \mathcal{V}$.

Cependant, tout corpus contient une importante proportion de mots vides (prépositions, articles, pronoms...), ou simplement non pertinents. La fréquence d'apparition de ces termes sera donc importante dans tous les documents, leur donnant ainsi plus de poids que des termes plus rares, mais plus représentatifs du contenu. Il a donc été proposé par Jones, 1972 de prendre en compte également l'importance d'un mot dans le corpus, la rareté d'un terme dans tout le corpus $\mathcal{D}$. Un mot rare sera jugé plus discriminant qu'un mot commun. Cette mesure est la *Fréquence inverse de document* (*Inverse Document Frequency*), notée $idf(w_i, \mathcal{D})$, telle que :

$$idf(w_j, \mathcal{D}) = \log \frac{D}{|\{d_i : w_j \in d_i\}|} \tag{2.2}$$

avec au numérateur le nombre de documents dans $\mathcal{D}$, et au dénominateur le nombre de documents $d_j \in \mathcal{D}$ où apparaît le terme w_i.

En combinant ces deux notions on peut représenter un mot (et *de facto* un document) en utilisant tf-idf, tel que proposé par Salton et al., 1975 :

$$tf\text{-}idf(w_j, d_i, \mathcal{D}) = tf(w_j, d) \times idf(w_j, \mathcal{D}) \tag{2.3}$$

On peut donc assigner à chaque valeur $x_{i,j}$ de X la valeur de $tf\text{-}idf(w_j, d_i, \mathcal{D})$, afin d'obtenir une matrice terme-documents.

Cette matrice s'avère cependant être large, grandissant avec le nombre de documents et la taille du vocabulaire, et très creuse. Il devient donc intéressant de réduire la dimension de cette matrice afin de la rendre plus exploitable. *Latent Semantic Analysis* [Deerwester et al., 1990] propose de remédier à ce problème au moyen d'une décomposition en valeurs singulières (*singular value decomposition*, SVD) tronquée, afin de plonger linéairement les représentations des documents dans un espace de dimension K, tel que $K \ll V$.

Une décomposition en valeurs singulières permet de décomposer la matrice $X \in \mathbb{R}^{D \times V}$ tel quel $X = U\Sigma R^\top$, avec $U \in \mathbb{R}^{D^2}$ et $R \in \mathbb{R}^{V^2}$ des matrices unitaires, R^* la matrice transposée de R et $\Sigma \in \mathbb{R}^{D \times V}$ une matrice diagonale des valeurs singulières de X. Obtenir $\tilde{X}$, la meilleure approximation de rang K de X, revient, en vertu du théorème de Eckart–Young [Eckart et al., 1936], à minimiser la distance, au sens de la norme de Frobenius, entre ces deux matrices. $\tilde{X}$, la matrice approximant X, est de la forme $\tilde{X} = U\tilde{\Sigma}R^\top$, avec $\tilde{\Sigma}$ la matrice diagonale ne contenant plus que les K plus grandes valeurs singulières (les autres étant mises à 0). En ne gardant ainsi que les valeurs singulières les plus grandes, on garantit de conserver les axes de X avec les variances les plus importantes.

Un lien assez naturel peut être fait avec l'*Analyse en Composantes Principales* (ACP) [Hotelling, 1933], puisque SVD et ACP sont deux méthodes linéaires voisines lorsque les données sont centrées. Si la matrice termes-documents X n'est pas centrée, elle est tout de même par nature très creuse (peu de mots du vocabulaire occurrent dans chaque document), amenant les moyennes des colonnes à être proches de 0. *LSA* peut ainsi être rapproché d'une recherche des K composantes principales de la matrice de variance-covariance de X.

pLSA [Hofmann, 1999] reprend le concept développé avec LSA, en proposant cette fois-ci une approche non plus algébrique, mais probabiliste. Là où LSA cherche à obtenir une approximation de X de rang K optimale selon la norme de Frobenius, pLSA propose un objectif différent. Le modèle considère trois ensembles de variables :

- Les documents $d \in \mathcal{D} = \{d_1, \cdots, d_D\}$, des variables observées. Ceux-ci sont représentés par leur index dans le corpus.
- Les mots $w \in \mathcal{V} = \{w_1, \cdots, w_V\}$, également observés.
- Les thématiques $z \in \mathcal{Z} = \{z_1, \ldots, z_K\}$, des variables latentes.

On définit les thématiques dans pLSA, comme dans les autres méthodes de modélisation de thématiques, comme un ensemble de mots susceptibles de porter sur les mêmes sujets.

On représente la probabilité qu'un mot occurre dans un document comme :

$$p(d, w) = \sum_{z \in \mathcal{Z}} p(z)p(d|z)p(w|z) \tag{2.4}$$

pLSA est un modèle ayant pour paramètres K distributions multinomiales de taille V et D mélanges de K thématiques, soit $K(V \times D)$ paramètres au total. Le nombre total de

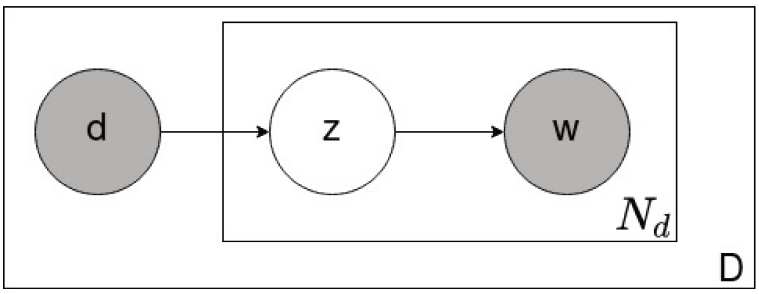


FIGURE 2.1 – Diagramme en plaques de pLSA (en gris les observations)

paramètres du modèle grossit linéairement avec la taille du corpus D, le rendant plus sensible à des problèmes de sur apprentissages.

L'estimation de z par maximum de vraisemblance se fait au moyen d'un algorithme *Expectation-Maximisation* (EM) [DEMPSTER et al., 1977], qui alterne entre deux étapes. Lors de la première, E, on calcule la distribution a posteriori pour les variables latentes $p(z|d, w)$, puis on passe à une phase dite de *maximisation* (M) où l'on calcule les paramètres afin de maximiser la log-vraisemblance. Cette procédure d'estimation permet d'approcher un maximum local de la log-vraisemblance observée.

Il est possible de dresser un parallèle entre *pLSA* et *LSA*. Comme illustré Figure 2.2, on peut aisément rapprocher la formulation de *pLSA* à l'Équation 2.4 et celle de *LSA* : $\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{R}^T$, et retrouver une équivalence entre $\tilde{U}$ et $p(d|z)$ (où chaque ligne représente la probabilité sur un document), $\tilde{V}$ et $p(w|z)$ (où chaque ligne représente la probabilité de générer un mot selon chaque thématique) et $\tilde{\Sigma}$ et $diag(p(z))$ (la probabilité d'apparition de chaque thématique).

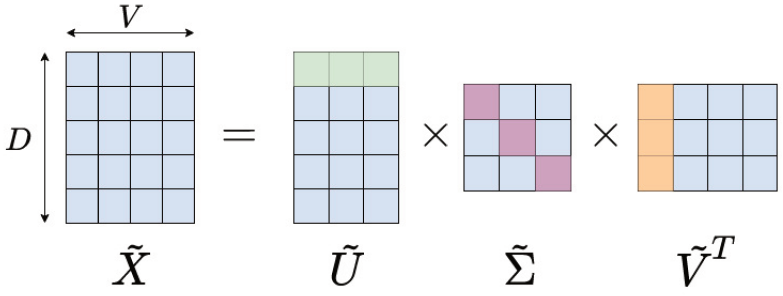


FIGURE 2.2 – Représentation alternative de pLSA, comme décomposition de matrice. La ligne verte représente $p(d|z)$, la diagonale rouge $p(z)$ et la colonne orange $p(w|z)$

2.1.2 Modélisation thématique : LDA

Une des principales limitations de *pLSA* se trouve dans son incapacité à pouvoir plonger de nouveaux documents, puisque ceux-ci sont représentés par leur index dans le corpus d'entraînement. Pour pallier ce problème, *Latent Dirichlet Allocation* LDA ; [Blei, Ng et al., 2003] étend *pLSA* à un cadre bayésien, en ajoutant un a priori aux distributions document-thématiques $p(d|z)$, sous la forme d'une distribution de Dirichlet. En fixant un nombre K de thématiques latentes, on cherche a apprendre lesquelles sont représentées dans chaque document, et quels mots leurs sont associées. Ceci sous-tend plusieurs choses. Premièrement, une thématique explique l'occurrence d'un mot dans un document. Ensuite, les mots du vocabulaire appartiennent aux différentes thématiques avec différentes probabilités (par exemple le terme *truffe* peut être lié à une thématique relative aux animaux et à une autre relative aux champignons). Enfin, les documents ne sont expliqués que par un petit nombre de thématiques.

LDA étant une base importante du travail présenté le long de ce manuscrit, nous nous attarderons sur son fonctionnement, puis sur sa procédure d'inférence.

2.1.2.1 Modèle génératif

Soit $\alpha \in \mathbb{R}^K$ un hyperparamètre contrôlant la distribution des thématiques des documents. On considère alors $\theta_d \in \mathbb{R}^K$ le paramètre représentant le mélange de thématiques du document d, généré selon une loi de Dirichlet de paramètre α. Le choix de la loi de Dirichlet s'explique par le fait qu'elle est la loi conjuguée de la loi multinomiale, permettant ainsi de réaliser les calculs analytiquement. Pour chaque mot observé $w_{d,i}$ du document d, on lui attribue une thématique $z_{d,i} \in \{1, \cdots, K\}$ tirée selon une loi multinomiale de paramètre θ_d. Enfin, on tire le mot $w_{d,n} \in \mathcal{V}$ selon une loi multinomiale de paramètre $\beta_{z_{d,n}}$, avec $\beta \in \mathbb{R}^{K \times V}$ l'a priori sur la distribution des mots par thématiques.

Algorithme 1 : Modèle génératif de LDA

foreach $d \in \mathcal{D}$ **do**
  $\theta_d|\alpha \sim Dir(\alpha)$
  foreach $w_{d,i} \in d$ **do**
    $z_{d,i}|\theta_d \sim Multi(\theta_d)$
    $w_{d,i}|z_{d,i} \sim Multi(\beta_{z_{d,i}})$

2.1.2.2 Estimation

L'estimation bayésienne des paramètres θ, β et z de LDA requiert le calcul de la distribution a posteriori des variables latentes sachant un document, tel que :

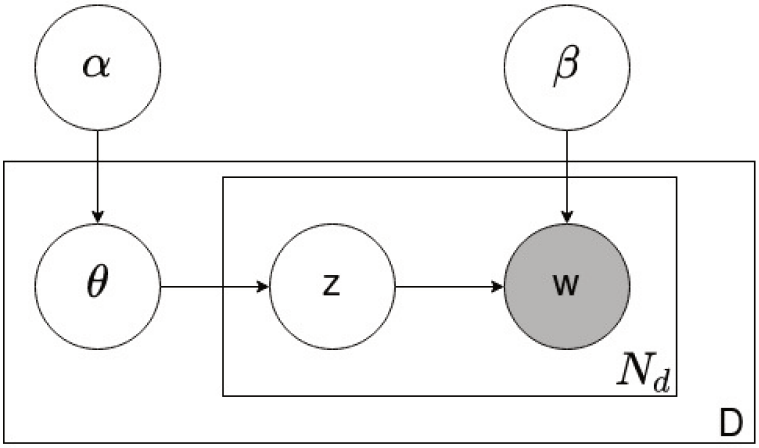


FIGURE 2.3 – Modèle graphique de LDA (en gris les observations)

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{2.5}$$

avec :

$$p(w|\alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_{i=1}^{K} \int (\theta_i^{\alpha_i - 1}) \left(\prod_{n=1}^{N} \sum_{i=1}^{K} \prod_{j=1}^{V} (\theta_i \beta_{i,j})^{w_n^j} \right) d\theta \tag{2.6}$$

On remarque cependant que le dénominateur (Équation 2.6) devient rapidement incalculable, car il nécessite de marginaliser sur θ et z. Les auteurs proposent d'utiliser l'inférence variationnelle [BISHOP, 2006] afin d'estimer les paramètres (bien que d'autres méthodes puissent être envisagées, telle que l'approximation de Lagrange ou l'échantillonnage de Gibbs [WEI et al., 2006]).

2.1.2.3 Approximation en champ moyen

Le problème posé précédemment peut être contourné en approximant la distribution postérieure $p(w|\alpha, \beta)$ au moyen d'une famille de distributions q, afin de simplifier les calculs. On introduit deux variables (dites *paramètres variationnels*) γ et ϕ, respectivement liées à θ et z, afin de les découpler, comme montré à la Figure 2.4. Cette hypothèse d'indépendance entre les variables latentes est connue sous le nom de *mean field approximation* [PARISI, 1988], et permet de calculer efficacement les règles de mise à jour des paramètres indépendamment. L'objectif est donc maintenant d'approcher au mieux $p(\theta, z|w, \alpha, \beta)$, c'est à dire trouver les valeurs du couple (γ^*, ϕ^*) minimisant la divergence de Kullback-Leiber (D_{KL}) entre cette distribution $q(\theta, z|\gamma, \phi)$ et la véritable postérieure, comme présenté à l'Équation 2.7.

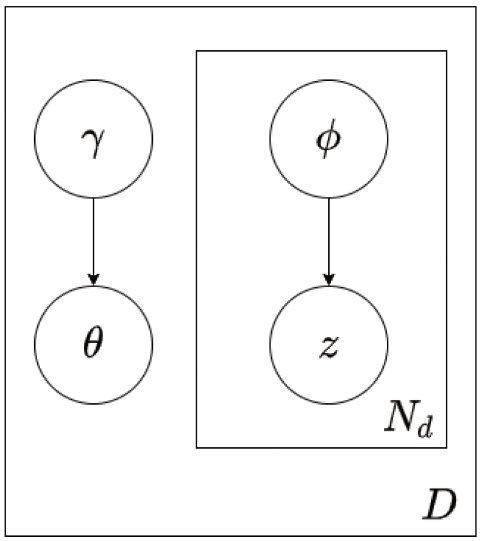


FIGURE 2.4 – Modèle graphique de LDA avec les paramètres variationnels

$$(\gamma^*, \phi^*) = argmin_{(\gamma,\phi)} D_{KL}\left(q(\theta, z|\gamma, \phi) \| p(\theta, z|w, \alpha, \beta)\right) \tag{2.7}$$

Pour cela nous exprimons la log-probabilité marginale de nos observations de façon à la borner (Équations 2.8). Le passage de 2.8b à 2.8c est permis par l'inégalité de Jensen. [1], du fait de la stricte concavité de la fonction logarithme. On nomme la borne inférieure donnée Équation 2.9, l'opposé de l'entropie, *Evidence Lower Bound* (ELBO), notée $\mathcal{L}(\gamma, \phi; \alpha, \beta)$. Après réécriture, on note que maximiser cette ELBO revient à minimiser la divergence KL entre la distribution postérieur et distribution variationnelle.

$$\begin{aligned}
\log p(w|\alpha, \beta) &= \log \int \sum_Z p(\theta, z, w|\alpha, \beta) d\theta && \text{(2.8a)}\\
&= \log \int \sum_Z \frac{p(\theta, z, w|\alpha, \beta) q(\theta, z)}{q(\theta, z|\gamma, \phi)} d\theta && \text{(2.8b)}\\
&\geq \log \int q(\theta, z) p(\theta, z, w|\alpha, \beta) d\theta - \log \int q(\theta, z) \log q(\theta, z|\gamma, \phi) d\theta && \text{(2.8c)}\\
&= \underbrace{\mathbb{E}_q\left[\log p(\theta, z, w|\alpha, \beta)\right]}_{D_{KL}} - \underbrace{\mathbb{E}_q\left[\log q(\theta, z|\gamma, \phi)\right]}_{Entropie} && \text{(2.8d)}\\
&= \underbrace{\mathcal{L}(\gamma, \phi; \alpha, \beta)}_{ELBO}
\end{aligned}$$

1. **Inégalité de Jensen** : Soit f une fonction concave sur un intervalle réel I, et X une variable aléatoire à valeurs dans I, dont l'espérance $\mathbb{E}(f(X))$ existe. Alors, $f(\mathbb{E}(X)) \geq \mathbb{E}[f(X)]$.

2.1.2.4 Algorithme EM

Les paramètres γ_i, le paramètre variationnel correspondant à θ_i, $\phi_{n,i}$, le paramètre variationnel correspondant à $z_{n,i}$, et β peuvent maintenant être estimés au moyen de multiplicateurs de Lagrange. On détaillera la méthode pour l'estimation de $\phi_{n,i}$, avant de donner les résultats pour les autres paramètres.

Dans un premier temps on peut réécrire la ELBO sous la forme :

$$\begin{aligned}\mathcal{L}(\gamma,\phi;\alpha,\beta) = &\mathbb{E}_q[\log p(\theta|\alpha)] + \mathbb{E}_q[\log p(z|\theta)] + \mathbb{E}_q[\log p(w|z,\beta)] \\ &- \mathbb{E}_q[\log q(\theta)] - \mathbb{E}_q[\log q(z)]\end{aligned} \tag{2.9}$$

Soit,

$$\begin{aligned}\mathcal{L}(\gamma,\phi;\alpha,\beta) = &\log\Gamma(\sum_{i=1}^{K}\alpha_i) - \sum_{i=1}^{K}\log\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i-1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j)) \\ &+ \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{n,i}(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j)) \\ &+ \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V}\phi_{n,i}w_n^j\log\beta_{i,j} \\ &- \log\Gamma(\sum_{j=1}^{K}\gamma_j) + \sum_{i=1}^{K}\log\Gamma(\gamma_i) - \sum_{i=1}^{K}(\gamma_i-1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j)) \\ &- \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{n,i}\log\phi_{n,i},\end{aligned} \tag{2.10}$$

avec Ψ la fonction diagamma, tel que $\Psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$, où Γ est la fonction gamma. On commence par former le lagrangien en ne gardant que les termes relatifs à $\phi_{n,i}$, et on peut écrire :

$$\mathcal{L}_{\phi_{n,i}} = \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{n,i}(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j)) + \phi_{n,i}\log\beta_{i,v} - \phi_{n,i}\log\phi_{n,i} + \lambda_n(\sum_{j=1}^{K}\phi_j - 1)$$

avec $\beta_{i,v} = p(w_n^v = 1|z^i = 1)$

En résolvant $\frac{\partial\mathcal{L}}{\partial\phi_{n,i}} = 0$ on arrive alors à :

$$\phi_{n,i} = \beta_{i,v}\exp\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j) - 1 + \lambda_\phi\right) \tag{2.11}$$

$$\propto \beta_{i,v}\exp\left(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{K}\gamma_j)\right), \tag{2.12}$$

avec λ_ϕ le coefficient de Lagrange.

En suivant cette procédure, on détermine également l'expression du paramètre variationnel γ et du paramètre β, respectivement aux équations 2.13 et 2.14. On alterne finalement l'estimation des paramètres variationnels et de β, de la même façon qu'un algorithme EM classique. La procédure complète d'estimation a une complexité en temps de $\mathcal{O}(D\tilde{N}K)$, avec $\tilde{N}$ le nombre moyen de mots dans un document. Puisque le nombre de thématiques K est fixé et que la taille des documents d'un corpus est souvent homogène, on peut avancer que la complexité en temps de LDA dépend linéairement du nombre de documents du corpus.

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{n,i} \tag{2.13}$$

$$\beta i, j = \sum_{d=1}^{D} \sum_{n=1}^{N} \phi_{d,n,i} w_{d,n}^{j} \tag{2.14}$$

2.1.2.5 Extensions

LDA est un modèle qui a été particulièrement utilisé dans de nombreuses tâches de NLP. C'est donc tout naturellement qu'il a été continuellement étendu au fil du temps et adapté à une grande variété de cas d'usage. Certaines de ces extensions proposent de nouvelles procédure d'inférence, d'autres étendent LDA à de nouveaux types de données et d'autres adaptent LDA à de nouveaux cas d'usage.

2.1.2.5.1 Autres procédures d'estimation : Blei, Ng et al., 2003 proposent dans leur article d'estimer les paramètres de LDA en utilisant une méthode variationnelle. Cette approche sera reprise dans de nombreux travaux suivants la publications de LDA, comme dans [Chang et al., 2009] ou [Guo et al., 2009]. D'autres travaux ont cependant proposé d'effectuer cette estimation au moyen d'un échantillonnage de Gibbs, comme [Griffiths et al., 2004] ; [Wallach et al., 2009] ; [Jagarlamudi et al., 2010]. Ces méthodes, basées sur les *Monte Carlo Markov Chains* [Robert et al., 2011], semblent aujourd'hui les plus populaires parmi les procédures d'estimation des méthodes basées sur LDA. Chacune des ces méthodes présentent ses propres avantages et inconvénients : le Gibbs sampling propose une garantie de tirer des échantillons asymptotiquement exacts de la distribution cible, contrairement à l'inférence variationnelle, mais requiert beaucoup plus de calcul, le rendant en général plus lent que les méthodes variationnelles et donc moins adapté aux larges jeux de données (la continuelle amélioration des capacités de calcul tend cependant à effacer de plus en plus cette limitation). On trouve également d'autres propositions permettant de surmonter la limitation de LDA quand à sa mise à jour en présence de documents non initialement présents dans

le corpus d'entraînement. C'est ce que propose par exemple *Online LDA* [Hoffman et al., 2010], en adaptant l'inférence variationnelle de LDA

2.1.2.5.2 Autres cas d'usage : LDA repose sur l'hypothèse *sac de mots*, c'est à dire que l'ordre d'apparition des mots n'est pas pris en compte lors de la modélisation des documents (les mots sont indépendants les uns des autres) et les motifs de co-occurrence des mots ne sont qu'implicitement modélisés, et seulement au niveau du document. Plusieurs travaux proposent donc de dépasser la formulation *sac de mots* et de réintroduire une certaine dépendance entre les termes. Yan et al., 2013 propose par exemple avec *Biterm Topic Model* de s'intéresser aux paires de mots qui co-occurrent dans un corpus. Les thématiques ne sont dès lors plus assignées à un mot, mais à une paire de mots, permettant la production de thématiques de meilleure qualité lorsque les textes sont très courts. On peut également noter *Hidden Topic Markov Model* (HTMM) [Gruber, Weiss et al., 2007], un modèle proche de LDA, mais qui abandonne l'hypothèse d'indépendance entre les thématiques latentes. Au lieu de présumer de l'indépendance conditionnelle de la distribution de thématique au sein d'un document, HTMM modélise explicitement cette dépendance au moyen d'une chaîne de Markov, faisant dépendre les thématiques des mots d'une phrase de celles des mots de la phrase précédente. Cette approche permet une nette amélioration en désambiguïsation et améliore la capacité du modèle à correctement assigner une thématique aux termes non observés lors de l'entraînement .

On trouve également de nombreux travaux cherchant à enrichir le cadre de LDA avec des données additionnelles. *Dynamic Topic Model* [Blei et Lafferty, 2006] prend en compte, en plus des documents d'un corpus, leur date de publication. Le corpus est découpé en tranches (une année dans l'article original) et les documents de chaque tranche sont considérés comme étant générés par un ensemble de thématiques venant de la tranche précédente. On peut également citer *Author Topic Model* [Rosen-Zvi et al., 2004] où s'ajoute au corpus de documents une information sur les auteurs. On modélise qu'un groupe d'auteurs écrit un document, et l'on choisit pour chaque mot un auteur, selon une loi uniforme. On tire ensuite la thématique de ce mot selon la distribution de thématiques propre à son auteur, avant de le générer à partir de la thématique assignée. On estime donc conjointement la proportion de thématiques de chaque document, ainsi que celles de ses auteurs, traduisant leurs thématiques de prédilection. Chang et al., 2009, enfin, proposent de prendre en compte, avec *Relational Topic Model* (RTM), les liens existant dans des réseaux de documents afin de permettre une meilleure représentation des documents et de s'adapter à une tâche de prédiction de liens. Cet article étant une base importante des travaux présentés dans ce manuscrit, il sera couvert en détail dans la partie **??** de ce chapitre.

Enfin LDA a été adapté à une grande variété de tâches, et ce, dans tous les domaines,

comme par exemple :

— Géographie [L. Zhang et al., 2015] ; [H. Tang et al., 2012]
— Sciences politiques [Song et al., 2014] ; [Balasubramanyan et al., 2012]
— Linguistique [Heintz et al., 2013] ; [Vulic et al., 2011]
— Biologie [Perina et al., 2010] ; [Millar et al., 2009]
— Développement informatique [Lukins et al., 2010] ; [X. Yang et al., 2017]
— Analyse de réseaux sociaux [Hu et al., 2012] ; [Tan et al., 2013]

La modélisation de thématique n'est cependant pas la seule approche pour la représentation de documents, et de nombreuses méthodes ont émergé lors de la dernière décennie, se focalisant sur l'apprentissage de représentations distributionnelles. Dans la mesure où cette catégorie de méthodes repose énormément sur l'apprentissage de représentations de mots, nous dresserons un rapide état de l'art des différents modèles destinés à apprendre des représentations distributionnelles de mots.

2.1.3 Méthodes d'apprentissage de représentations distributionnelles

L'apprentissage de représentations de mots est une tâche centrale, et les méthodes développées pour y répondre trouvent des applications bien au delà des frontières du TAL. Au cours de la dernière décennie de nombreux modèles ont été proposés, dont l'hypothèse principale trouve sa source dans la linguistique de corpus. Nous présenterons dans cette parties des contributions majeures permettant d'apprendre des représentations distributionnelles de mots.

Cette nouvelle façon de représenter les mots a émergé en prenant comme paradigme l'*hypothèse distributionnelle* [Harris, 1954], théorie née dans la seconde moitié des années 50 dans la communauté des linguistes américains, et qui marque les débuts de la linguistique de corpus. Firth, 1957 résume cette hypothèse par la formule "You shall know a word by the company it keeps", autrement dit les mots apparaissant dans des contextes similaires tendent à être également proches par le sens. L'une des premières méthodes visant à apprendre une représentation distribuée des mots est attribuée à Bengio, Ducharme et al., 2003. Cette représentation sera par la suite appelée *plongement de mots* (*word embedding* en anglais), et prend la forme d'un vecteur de taille ρ dans $\mathbb{R}^\rho$, de telle façon que $\rho << V$.

2.1.3.1 Word2Vec

Les travaux ayant eu le plus d'influence dans l'apprentissage de plongement de mots se retrouvent dans Word2Vec. Dans Word2Vec le contexte, c'est-à-dire les motifs de co-occurrence des mots entre eux, est modélisé par une fenêtre de taille $l = 2m + 1$ centrée

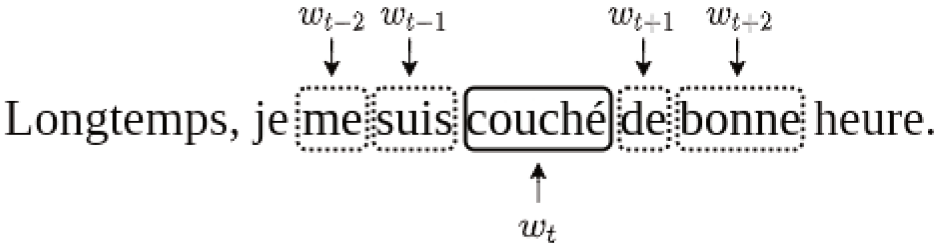


FIGURE 2.5 – Exemple de fenêtre glissante pour $m = 2$. Ici la fenêtre est centrée sur le mot *couché*, produisant les paires de mots suivantes : (*me*, *couché*), (*suis*, *couché*), (*de*, *couché*) et (*bonne*, *couché*).

autour d'un mot, comme représenté à la Figure 2.5. Cette fenêtre est dites "glissante" car elle se déplace tout du long des documents du corpus afin de construire les paires de mots. On note $\mathcal{C}$ l'ensemble des paires ainsi construite.

Deux architectures différentes coexistent au sein de Word2Vec, et ont des bases communes : *Continuous bag of words* (CBOW) et *Skip-gram* (SG) [MIKOLOV, K. CHEN et al., 2013]. Ces deux méthodes se rejoignent en ceci qu'elles reposent sur le contexte d'apparition des mots. Elles prennent également la forme de réseaux de neurones à deux couches qui visent à apprendre pour chaque mot un vecteur lui correspondant dans un espace de représentation. Elles diffèrent cependant sur leur objectif d'optimisation, puisque CBOW vise à prédire un mot suivant les mots qui l'entourent alors que Skip-gram cherche lui à prédire un contexte en prenant en entrée un mot, comme illustré à la Figure 2.6.

2.1.3.1.1 Skip-gram Le modèle Skip-gram (SG) cherche à apprendre la représentation $u_i \in \mathbb{R}^\rho$ d'un mot w_i qui permette de prédire les représentations $(c_{i-m}, \cdots, c_{i+m})$, elles aussi des vecteurs dans $\mathbb{R}^\rho$, des mots $w_j \in \{w_{i-m}, \cdots, w_{i+m}\}$ qui l'entourent dans un document. Par la suite, nous utiliserons également les notations $U \in \mathbb{R}^{D\times\rho}$ et $C \in \mathbb{R}^{D\times\rho}$, la matrice dont la i$^{\text{ième}}$ correspond à u_i et c_i respectivement. Skip-Gram modélise la probabilité d'observer un mot w_j appartenant au contexte de w_i sachant w_i. On cherche alors à maximiser la log-vraisemblance de l'ensemble $\mathcal{C}$ des paires de mots qui co-occurrent dans le corpus telles que :

$$\sum_{(w_i,w_j)\in\mathcal{C}} \log p(w_j|w_i), \tag{2.15}$$

$$\Leftrightarrow \sum_{(w_i,w_j)\in\mathcal{C}} \log \frac{e^{s(w_i,w_j)}}{\sum_{k=1}^{V} e^{s(w_k,w_j)}}, \tag{2.16}$$

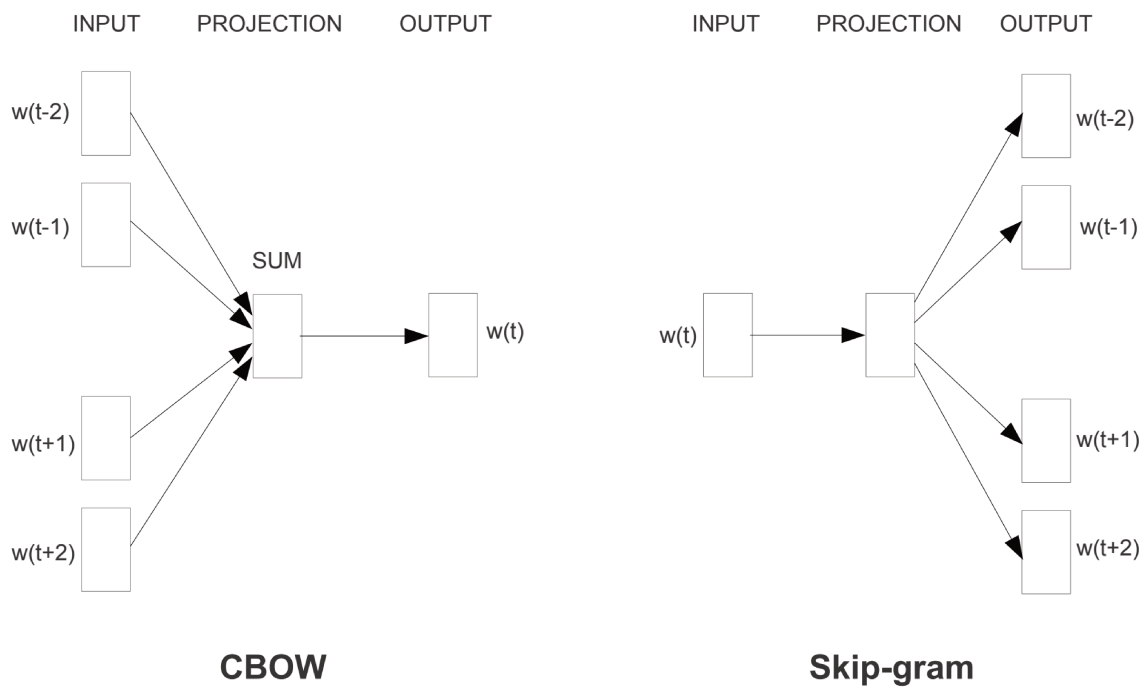


FIGURE 2.6 – Représentations graphiques des modèles CBOW et Skip-gram, tirées de [MIKOLOV, Q. LE et al., 2013]

où $s(w_i, w_j) = u_i^\top \cdot c_j$ est appelé fonction de score, avec u_i est le plongement de w_i, dit vecteur cible, et c_j un vecteur de plongement du mot w_j, dit vecteur contexte.

Le dénominateur de l'Équation 2.16 est néanmoins coûteux à calculer dans la plupart des cas, le calcul étant en temps $\mathcal{O}(V)$, et les vocabulaires de tailles importantes. Deux méthodes ont été proposées par les auteurs pour pallier ce problème.

Hierarchical Softmax : Cette méthode propose de représenter le softmax sous la forme d'un arbre binaire, chaque mot devenant une feuille de l'arbre. Il n'est alors plus nécessaire que de suivre le chemin de la racine de l'arbre au noeud correspondant, sans prendre en compte les autres noeuds. On représente Figure 2.7 un exemple d'arbre binaire pour Hierachical Softmax.

Le calcul peut être effectué en temps $\mathcal{O}(\log(V))$. $p(w_j|w_i)$ est alors approché de la façon suivante :

$$p(w_j|w_i) = \prod_{k=1}^{L(w_i)} p(n_{(w_i,k)}|w_j) \tag{2.17}$$

$$= \prod_{k=1}^{L(w_i)} \sigma(\gamma_{n_{(w_i,k)}} \cdot c_j), \tag{2.18}$$

où $L(w_i)$ est la longueur du chemin pour atteindre le noeud correspondant au mot w_j, σ la fonction sigmoïde tel que $\sigma(x) = \frac{1}{1+\exp^{-x}}$, et $\gamma_{n_{(w_i,k)}}$ un vecteur servant de coefficient, lié au

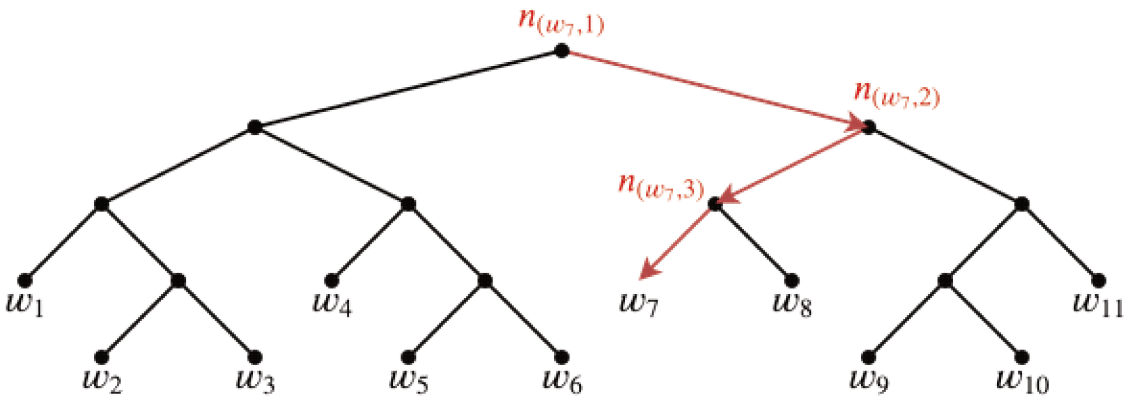


FIGURE 2.7 – Exemple d'arbre binaire, pour un vocabulaire de 11 mots w_i. En rouge $L(w_7)$, le chemin de la racine à la feuille correspondant au mot w_7. $n_{(w_7,j)}$ représente le $j^{\text{ième}}$ noeud de $L(w_7)$.

longtemps	2
je	9
me	8
bleu	4
de	14
heure	3
cheval	1

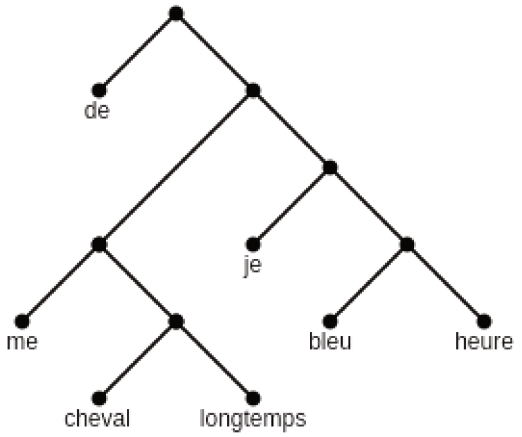


FIGURE 2.8 – Exemple d'arbre Huffman. À gauche le tableau donnant le nombre d'occurrence de chaque mots dans le corpus, et à droite l'arbre correspondant.

noeud $n_{(w_i,k)}$. Si plusieurs façons de construire l'arbre sont envisageables [GOODMAN, 2001] ; [MORIN et al., 2005] ; [MNIH et al., 2008], les auteurs proposent de construire un arbre de Huffman. Comme illustré Figure 2.8, la longueur du chemin de la racine de l'arbre à la feuille correspondant à un mot est inversement proportionnelle au nombre d'occurrences de ce mot dans le corpus, ce qui minimise le nombre de calculs nécessaires à l'Equation 2.18.

Échantillonnage négatif : Si Hierarchical Softmax permet un gain de performance par rapport au softmax classique, il tend à perdre en efficacité sur les corpus volumineux, où même $\log(V)$ devient grand. MIKOLOV, Q. LE et al., 2013 proposent avec *Skip-gram with negative sampling* (SGNG), une nouvelle méthode d'estimation, basée sur la Noise Contrastive Estimation présentée dans [GUTMANN et al., 2010]. Puisque l'objectif premier de Skip-gram est de pouvoir différentier des paires de mots co-occurant réellement dans un corpus des

paires de mots aléatoires, les auteurs réécrivent la fonction objectif en deux parties afin de s'affranchir du calcul du dénominateur du softmax de l'Equation 2.18. D'abord la probabilité conditionnelle du mot contexte en fonction du mot cible devient :

$$p(w_j|w_i) = \sigma(s(w_i, w_j)), \tag{2.19}$$

avec σ la fonction sigmoïde et $s(w_i, w_j) = u_i \cdot c_j$.

Ensuite, la maximisation de la log-vraisemblance est rapportée à un problème de classification binaire cherchant à différencier les mots apparaissant effectivement dans les mêmes contextes, les paires positives, du bruit, autrement dit des paires négatives ne co-occurrant pas ou presque jamais. L'objectif à maximiser devient alors :

$$\sum_{(w_j,w_i)\in\mathcal{C}} \left(\log \sigma(s(w_i, w_j)) + \sum_{k=1}^{K} \mathbb{E}_{w_k\sim P_n(w)} \left[\log \sigma(-s(w_i, w_k))\right] \right), \tag{2.20}$$

avec K le nombre d'exemples négatifs tiré selon une loi $P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=1}^{V} f(w_j)^{\frac{3}{4}}}$, où $f(w_i)$ est la fréquence d'apparition du mot w_i dans tout le corpus. L'exposant $\frac{3}{4}$ a été trouvé empiriquement, et permet d'effectuer un lissage donnant plus de poids aux mots les moins fréquents.

En maximisant l'Équation 2.20 les mots sémantiquement proches auront tendance à se rapprocher dans l'espace de représentation, et les mots dissimilaires à s'éloigner. La qualité des plongements de mots obtenus dépend cependant du nombre d'exemples négatifs fixés lors de l'entraînement, et il est nécessaire de déterminer la valeur de K offrant le meilleur compromis entre qualité et temps de calcul. Les auteurs indiquent que pour de larges corpus, un nombre d'échantillons négatifs K entre 2 et 5 permet d'atteindre de bons résultats. La complexité de SGNG est de $\mathcal{O}(K|\mathcal{C}|)$.

2.1.3.1.2 Liens avec la factorisation de matrices Plusieurs travaux ont cherché à faire le lien entre l'objectif de SGNS et la factorisation de matrice. Levy et al., 2014 montrent que l'on peut ramener SGNS une factorisation implicite d'une matrice M, dont chaque coefficient $m_{i,j}$ correspond à la Pointwise Mutual Information (PMI) entre w_i et w_j. Cependant cette méthode repose sur une hypothèse forte, à savoir que $K \geq V$. La PMI entre deux mots représente une mesure de l'association entre eux. Plus formellement, elle s'intéresse au ratio entre la probabilité $p(w_i, w_j)$ que ces mots co-occurrent et la probabilité $p(w_i)p(w_j)$ qu'ils co-occurrent s'ils étaient indépendants. Elle est notée $PMI(w_i, w_j) = \log \frac{p(w_i,w_j)}{p(w_i)p(w_j)}$. Plus précisément, les auteurs utilisent comme mesure la *Shifted Positive* PMI (SPPMI), telle que $SPPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0) - \log(K)$, avec K le nombre d'exemples négatifs. Il serait donc possible, en théorie, d'atteindre les performances de SGNS en construisant la matrice de SPPMI. Les expériences menées par les auteurs ne permettent pas d'atteindre

les mêmes performances que celles obtenues par Mikolov, Sutskever et al., 2013, particulièrement en résolution d'analogie. Ces résultats semblent indiquer que la factorisation de la SPPMI et que le choix de la SVD comme outil de factorisation ne constituent pas les meilleures approches. L'approche retenue par Y. Li et al., 2015 repose, elle, sur la factorisation de la matrice de co-occurrences, et ramène SGNS à une factorisation explicite de cette matrice. Les auteurs proposent ainsi une formulation strictement équivalente à SGNS. Les expérimentations menées valident également cette approche, les performances obtenues étant égales, voire parfois légèrement meilleures, à celles obtenues avec la formulation originale.

2.1.3.1.3 Continuous bag of words Continuous bag of words (CBOW) vise, quant à lui, à prédire un mot à partir des mots de son contexte, comme illustré à la Figure 2.6. CBOW défini le vecteur contexte $\mathbf{c}$ d'un mot comme la moyenne des vecteurs de plongement des mots qui l'entourent dans la fenêtre glissante. On note $\mathbf{C}$ l'ensemble des plongement des mots du contexte, et $\mathbf{c} = \frac{1}{|\mathbf{C}|}\sum_{c_i \in \mathbf{C}} c_i$. En utilisant l'échantillonnage négatif, l'objectif à maximiser sera le même que celui présenté à l'Équation 2.20, mais la fonction de score sera cette fois $s(w_i, \mathbf{C}) = u_i \cdot \mathbf{c}$.

Le choix de l'une ou l'autre des variantes de Word2Vec dépend grandement des besoins exprimés par leur utilisateur. Le temps nécessaire à l'entraînement de CBOW est nettement moins important que celui de n'importe quelle variante de Skip-gram. Cependant, la quantité de données sur lesquelles les modèles seront entraînés joue aussi un rôle important dans ce choix. Skip-gram, et plus particulièrement sa variante avec échantillonnage négatif, tend à produire des plongements de mots d'une meilleure qualité que CBOW lorsque les corpus disponibles sont de petites tailles [Mikolov, K. Chen et al., 2013]. Néanmoins, lorsque les corpus disponibles sont importants, et que la représentation des mots les plus rares ne constitue pas un enjeu primordial, CBOW, de part le lissage sur les mots du contexte, permet d'obtenir de meilleurs plongements des mots les plus fréquents. Skip-gram semble cependant avoir été la variante majoritaire durant de nombreuses années, et ce succès peut en partie s'expliquer par le fait que le modèle exhibe une particularité intéressante dans la géométrique de ses représentations. Il semble en effet que SGNS puisse capter les liens sémantiques entre certains termes, permettant ainsi la construction d'analogies. Un des exemples les plus connu illustrant cette propriété montre que le vecteur de plongement du mot `reine`, u_{reine}, vérifie la relation $u_{reine} = u_{roi} - u_{homme} + u_{femme}$, indiquant que le modèle a su capter la relation entre les concepts de royauté et de genre.

2.1.3.2 FastText

Bojanowski et al., 2017 adapte la formulation de SGNS pour apprendre des représentations de mots, mais les représente comme la somme des représentations des n-grams qui les

composent. Le mot `temps` sera ainsi décomposé en `<te, tem, emp, mps, ps>` pour $n = 3$. Les caractères `<` et `>` permettent de marquer le début et la fin des mots, afin par exemple que le n-gram `<te` ne soit pas assimilé au mot `<te>`. Les auteurs introduisent une nouvelle fonction de score dans l'objectif de SGNS. En lieu et place du produit scalaire entre le vecteur cible et le vecteur contexte donné à l'Équation 2.19, ce score est maintenant obtenu en sommant les produits scalaires des n-grams composant le mot cible et le vecteur contexte. L'objectif à maximiser reste :

$$\sum_{(w_j,w_i)\in\mathcal{C}} \left[\sigma(s(w_i,w_j)) + \sum_{k=1}^{K} \mathbb{E}_{w_k\sim P_n(w)} \left[\sigma(-s(w_i,w_k)\right] \right],$$

avec K le nombre d'exemples négatifs désirés, mais

$$s(w_i,w_j) = \sum_{g\in\mathcal{G}_{w_i}} z_g c_j, \tag{2.21}$$

où $\mathcal{G}_{w_i}$ correspond à l'ensemble des n-grams qui composent le mot w_i, z_g la représentation de son $g^{\text{ième}}$ n-gram, et c_j la représentation contexte du mot w_j. Cette formulation permet de partager entre tous les mots la représentation des n-grams et ainsi améliorer la représentation des mots les plus rares. Elle présente également un avantage important dans l'apprentissage de représentation de mots dans le contexte d'une langue agglutinante, comme peuvent l'être par exemple le finnois, le turc ou le hongrois [Kuyumcu et al., 2019].

2.1.3.3 GloVe

Un autre modèle de plongement de mots a été proposé comme alternative à Word2vec : GloVe [Pennington et al., 2014]. Ce modèle repose cette fois sur une factorisation explicite de la matrice des co-occurences. En la factorisant et en la reconstruisant selon l'objectif décrit à l'Équation 2.22, le modèle apprend les représentations cibles u_i de w_i et contexte c_j de w_j, ainsi que les biais b_{u_i} et b_{c_j}. On définit également la matrice de co-occurrence S, où chaque coefficient s_{ij} représente le nombre de co-occurrences entre les mots w_i et w_j.

$$\sum_{i=1}^{V}\sum_{j=1}^{V} f(s_{ij}) \left(u_i \cdot c_j + b_u + b_c - \log s_{ij}\right)^2, \tag{2.22}$$

avec $f(s_{ij})$ une fonction de pondération permettant de limiter le poids donné aux co-occurrences les moins courantes et de limiter l'importance des co-occurrences trop fréquentes. Cette fonction est définie tel que :

$$f(s_{ij}) = \begin{cases} \left(\frac{s_{ij}}{s_{max}}\right)^{\frac{3}{4}} & \text{si } s_{ij} < s_{max} \\ 1 \text{ sinon} \end{cases}, \tag{2.23}$$

avec s_{max} un seuil défini a priori.

GloVe et Skip-Gram fonctionnent de façons similaires : modéliser la probabilité qu'un mot w_j apparaisse à proximité d'un mot w_i, comme détaillé aux Équations 2.15 et 2.16. Puisqu'il est bien souvent trop coûteux de calculer le dénominateur de la dernière équation, Skip-Gram proposait plusieurs méthodes pour s'affranchir de cette limite. GloVe en propose une nouvelle en introduisant dans l'expression de la log-vraisemblance le nombre de co-occurrences s_{ij} entre chaque termes, tel que :

$$\sum_{i=1}^{V}\sum_{j=1}^{V} s_{ij} \log p(w_j|w_i) \tag{2.24}$$

Une autre différence entre SG et GloVe se trouve dans le choix de la fonction d'erreur utilisée. Le premier cherche à minimiser l'entropie croisée quand le second choisit la méthode des moindres carrés. Les auteurs justifient ce choix par la propriété de l'entropie croisée à mal modéliser les distributions avec une longue traîne, et ainsi donner un poids trop important aux évènements rares. La matrice de co-occurrences S est construite au moyen d'une fenêtre glissante identique à celle utilisée dans Word2Vec. Si cette méthode permet un gain en termes de temps de calcul, la construction de la matrice a cependant un coût en mémoire qui peut s'avérer important.

2.1.3.4 Limites de la représentation distributionnelle

Ces différentes méthodes ne sont cependant pas exemptes de problèmes et de limitations, et captent de nombreux biais inhérents aux corpus sur lesquels ils sont entraînés. En premier lieu, nous sommes moins enclins, lorsque nous communiquons, à mentionner des éléments qui nous semblent évident, et inversement, nous insisterons plus sur des informations nouvelles ou des liens rares. En termes de plongement de mots, cela pourra se traduire par une co-occurrence moins importante entre deux mots dont le lien est pourtant évident. Un cygne noir sera mentionné comme *un cygne noir*, alors qu'un cygne blanc ne sera mentionné que comme *un cygne*. Une autre limitation dans ces représentations est leur incapacité à gérer le caractère polysémique de certains mots. Le terme *Opéra* peut aussi bien désigner un bâtiment, un art ou une pâtisserie et apparaître dans des contextes radicalement différents, mais ne disposera que d'une seule représentation, inadaptée.

Enfin, et d'une façon plus générale, toutes les méthodes basées sur l'apprentissage à partir de larges corpus tendent à refléter les biais intrinsèques des personnes qui les écrivent. Caliskan et al., 2017 montrent que l'on retrouve dans les plongements de mots certains stéréotypes raciaux ou liés au genre. Plusieurs méthodes ont cependant été proposées depuis, afin de tenter de mitiger ces biais [Bolukbasi et al., 2016] ; [T. Wang et al., 2020].

2.1.4 Méthodes de plongement de documents basées sur l'hypothèse distributionnelle

Dans la lignée des méthodes de plongement de mots définies précédemment plusieurs méthodes permettant de représenter des documents ont été proposées. Si la méthode consistant à moyenner les représentations des mots qui composent un document semble la plus évidente, et qu'elle permet d'obtenir des résultats intéressants [Gershman et al., 2015], elle ne représente pas le meilleur moyen d'obtenir des plongements de documents exploitables pour la vaste majorité des tâches de TAL. Pour y parvenir, les méthodes envisagées peuvent, soit reposer sur l'apprentissage direct de représentations des documents conjointement à celle des mots, ou bien sur l'apprentissage de paramètres permettant de combiner efficacement des plongements de mots pré-appris.

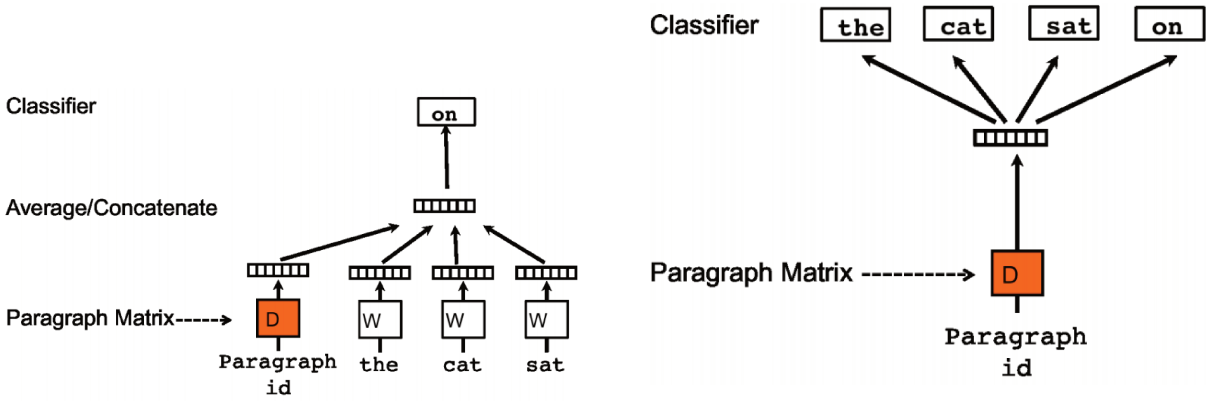
La plus connue des méthodes d'apprentissage de représentation de documents est probablement Paragraph2Vec [Q. Le et al., 2014] (souvent dénommé Doc2Vec, selon l'implémentation proposée dans la bibliothèque python Gensim [2]). Ce modèle connaît deux variantes, chacune reprenant une variante de Word2Vec. La première, *Paragraph Vectors - Distributed Memory* (PV-DM) repose sur CBOW et *Paragraph Vectors - Distributed Bag of Words* (PV-DBOW) sur Skip-Gram. La Figure 2.9 présente leurs représentations graphiques. Dans ces deux variantes, chaque document est représenté par un identifient unique, qui est ajouté à tous les contextes présents au sein de celui-ci. On fait ainsi co-occurrer la représentation du document avec chaque mot qui le compose. Dans PV-DM on cherche a prédire le mot au centre de la fenêtre en sommant les représentations des mots du contexte et la représentation du document. Dans PV-DBOW, le modèle cherche à prédire le contenu des fenêtres glissantes à partir de la représentation du document. Les auteurs remarquent que si PV-DM semble fournir des représentations de meilleure qualité que PV-DBOW, c'est la concaténation des représentations des documents obtenues par les deux modèles qui semble la plus efficace.

Thongtan et al., 2019 reprend la formulation de PV-DBOW pour l'adapter à une tâche de classification en analyse de sentiment et propose de remplacer la fonction de score utilisé à l'Equation 2.19 par un cosinus de l'angle entre les deux vecteurs, tel que :

$$s(w_d, w_j) = \frac{w_d^\top w_j}{\|w_d\|\|w_j\|}, \tag{2.25}$$

avec w_d le token représentant le document d, et w_j un mot de d. Cette modification permet d'obtenir de meilleurs résultats que l'implémentation classique de PV-DBOW sur la tâche considérée par les auteurs. Les auteurs avancent que l'utilisation du cosinus en temps que fonction de score permet de réduire le sur-apprentissage sur la tâche d'apprentissage de représentation, ce qui permettrait de produire de meilleurs plongement de documents.

2. https://radimrehurek.com/gensim/models/doc2vec.html



(a) PV-DM (b) PV-DBOW

FIGURE 2.9 – Représentations des modèles PV-DM (a) et PV-DBOW (b), tirées de [Q. LE et al., 2014].

KIROS et al., 2015 reprend avec Skip-Thought l'idée de la contextualisation en l'appliquant cette fois, non plus aux mots, mais directement aux phrases, dans l'optique de produire des plongements de celles-ci. Le modèle proposé cherche ainsi à apprendre des représentations des phrases, puis à prédire la phrase précédente, et la suivante. Pour ce faire, leur architecture se compose d'un encodeur, un réseau de neurones récurrent de type GRU (*Gated recurrent unit*) [CHO, GULCEHRE et al., 2014], prenant en entrée une séquence de mots, et produisant un vecteur dense. Ce plongement de phrase est alors passé en entrée de deux décodeurs distincts (eux aussi des réseaux de neurones récurrents), cherchant à prédire respectivement la phrase précédente et la phrase suivante.

D'autres méthodes de représentation de documents se basent plus explicitement sur des plongements de mots pré-appris. Par exemple, WU, I. E.-H. YEN et al., 2018 propose avec Word Mover's Embedding d'apprendre des plongements de documents en utilisant la Word Mover's Distance [KUSNER et al., 2015] :WMD. La WMD présente la mesure de distance entre deux documents (c'est à dire entre deux ensembles de mots) comme un problème de transport optimal, en cherchant la quantité minimum de trajet nécessaire pour transporter les mots d'un document vers un autre dans l'espace de représentation. En construisant un noyau défini positif basé sur cette distance [WU, I. E. YEN et al., 2018], le modèle apprend de façon non supervisée des représentations des documents à partir de plongements de mots pré-appris, et ce, dans l'espace de représentation des mots.

2.1.5 Deep learning pour l'apprentissage de représentations de documents

Les récentes avancées en matière de deep learning, couplées à l'évolution des moyens de calcul et de la quantité de données disponible, ont également touché les problématiques de traitement du langage. Les premiers travaux adaptent avec succès des architectures ayant déjà fait leurs preuves en dehors du champ du traitement automatique du langage. Par exemple les *Convolutional Neural Networks* (CNN) [Lecun et al., 1998], originellement développés sur des tâches de reconnaissance d'images, sont employés par Collobert et al., 2008 pour différentes tâches de TAL, allant de la détection d'entités nommées à la modélisation de langage. De la même manière, les *Long Short-Term Memory* (LSTM) [Hochreiter et al., 1997], initialement utilisées en reconnaissance vocale ou en reconnaissance d'image, ont su trouver leur place en TAL. Cette tendance est également suivie pour les Gate Reccurent Units GRU, [Cho, Gulcehre et al., 2014], et plus généralement, pour les Réseaux de Neurones Réccurents (RNN). Cette tendance semble cependant s'être inversée au cours des dernières années avec l'arrivée en traitement du langage des mécanismes d'attention, dont le représentant le plus connu est aujourd'hui le *Transformer* [Vaswani et al., 2017], qui ont été adaptés à d'autres domaines d'application tels que l'image [Dosovitskiy et al., 2021] ou la parole. Dans cette partie, nous reviendrons sur le fonctionnement et les principales applications des LSTM et des CNN, avant de nous pencher plus particulièrement sur les mécanismes d'attention.

En concluant la Section 2.1.3.3 nous pointions les difficultés des modèles comme Word2Vec ou GloVe à prendre en compte la polysémie. Dans le but de remédier à ce problème, un nouveau paradigme de représentation a émergé, basé sur la *contextualisation*. Quand les modèles précédents cherchaient à apprendre une représentation fixe pour chaque mot, les modèles contextuels assignent à chaque mots une représentation qui est fonction de la totalité de la phrase ou du document dans lequel il se trouve. Un même mot aura ainsi autant de représentations différentes que d'occurrences dans un corpus.

2.1.5.1 Embeddings from Language Models (ELMo)

ELMo [Peters et al., 2018] propose de produire des plongements de mots contextualisés en se basant sur une architecture bi-LSTM à L couches, intégrant deux modèles de langue, prenant respectivement en compte les mots de gauche à droite (forward) et de droite à gauche (backward) du terme considéré.

En partant d'une séquence de N mots $(w_1, \cdots, w_N)$, le premier modèle de langue modélise la probabilité de la séquence à partir des probabilités du $n^{\text{ième}}$ mot de la séquence en fonction des mots précédents $(w_1, \cdots, w_{n-1})$, c'est à dire de gauche à droite de la phrase, tel que :

$$p_{\rightarrow}(w_1, \cdots, w_N) = \prod_{n=1}^{N} p(w_n | w_1, \cdots, w_{n-1}). \tag{2.26}$$

Le second modèle, quant à lui, modélisera la séquence de droite à gauche, tel que :

$$p_{\leftarrow}(w_1, \cdots, w_N) = \prod_{n=1}^{N} p(w_n | w_{n+1}, \cdots, w_N). \tag{2.27}$$

ELMo optimise alors les paramètres une première fois en maximisant la log-vraisemblance de la séquence observée, le produit des deux équations précédentes :

$$\log \left(p_{\rightarrow}(w_1, \cdots, w_N)\, p_{\leftarrow}(w_1, \cdots, w_N) \right). \tag{2.28}$$

Le représentation intermédiaire à la $l^{\text{ième}}$ couche du mots w_n, $\mathbf{h}_n^l$, est construite en concaténant les représentations intermédiaires de réseaux traitant les séquences de gauche à droite (forward) et de droite à gauche (backward).

À l'issue de cette première optimisation non supervisée, les plongements de mots ne sont cependant pas encore exploitables, et le modèle est ensuite "raffiné", optimisé une seconde fois de manière supervisée, pour être adapté à une tâche de traitement du langage particulière (notée *task* par la suite). Lors de ce second entraînement on apprend les poids s_l^{task} et le facteur d'échelle γ^{task} afin de produire le vecteur de représentation $ELMo_n^{task}$ de w_n tel que :

$$ELMo_n^{task} = \gamma^{task} \sum_{l=0}^{L} s_l^{task} \mathbf{h}_n^l \tag{2.29}$$

Ces représentations contextuelles peuvent être utilisées dans une grande variété de tâches, comme l'analyse de sentiments ou la reconnaissance d'entités nommées. Certaines limitations demeurent cependant avec cette typologie d'architecture, dont la principale est la difficulté à paralléliser la procédure d'apprentissage à cause de leur nature séquentielle. Vaswani et al., 2017 sera le premier à répondre à cette limite en remplaçant le LSTM par une composante d'attention.

2.1.5.2 Les mécanismes d'attention

Les mécanismes d'attention sont un ensemble de techniques développées originellement pour les réseaux de neurones. Leur objectif est de mettre en avant une portion des données jugée pertinente tout en réduisant la contribution des autres. Les premières traces d'un mécanisme d'attention peuvent être trouvées dans Bahdanau et al., 2015, qui l'applique à la traduction automatique.

Ce modèle suit une architecture encodeur-décodeur, et vise à produire une séquence $(y_1, \cdots, y_T)$ de vecteurs de plongement de mots à partir d'une autre séquence de vecteurs

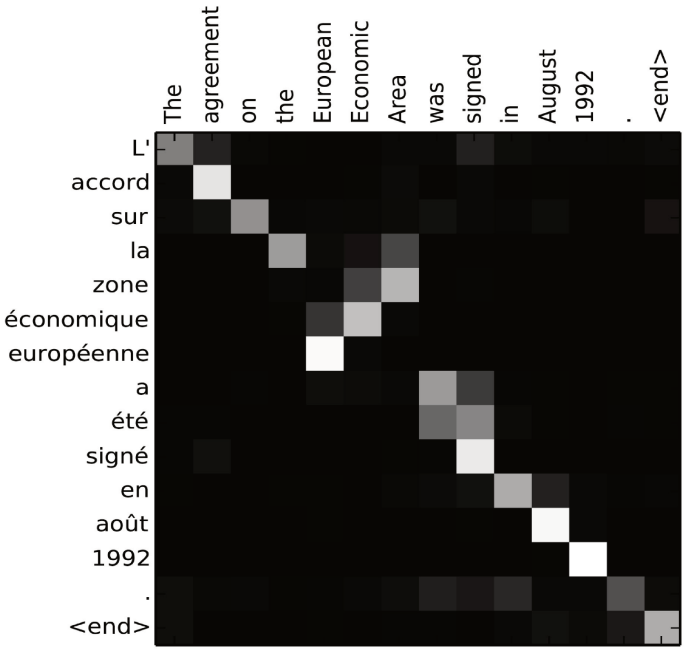


FIGURE 2.10 – Matrice des scores d'alignement entre la phrase "*L'accord sur l'Espace économique européen a été signé en août 1992* et sa traduction anglaise générée, tirée de BAHDANAU et al., 2015.

$(x_1, \cdots, x_N)$. L'encodeur reçoit donc en entrée une séquence $(x_1, \cdots, x_N)$ de plongement de mots et produit en sortie une séquence de vecteurs $(h_1, \cdots, h_N)$, dits "vecteurs d'annotations". Le décodeur est, quant à lui, constitué d'un modèle d'alignement suivit d'un LSTM. À chaque itération t, les vecteurs d'annotation sont combinés au sein du modèle d'alignement afin de produire un vecteur de contexte c_t, encodant la contribution de chaque annotation h_n. Le LSTM calcule ensuite son état caché s_t en fonction son état précédent s_{t-1}, le mot précédemment généré y_{t-1} et un vecteur contexte c_m. La Figure 2.11 récapitule le fonctionnement de cette architecture.

Ce mécanisme permet de ne plus prendre en compte qu'une petite partie de la séquence d'entrée. Cette méthode est cohérente avec notre intuition que, lors d'une traduction, nous ne prenons pas en compte l'ensemble de la phrase pour traduire un mot, mais seulement une portion du texte original. L'influence de ces poids d'alignement est visible à la Figure 2.10, et l'on remarque que les mots anglais produits en sortie ne profitent que d'une partie extrêmement réduite de l'influence de la séquence d'entrée en français. Le mécanisme d'attention permet aussi de s'affranchir de la nécessité des RNN d'encoder l'intégralité de la séquence d'entrée comme un vecteur de taille fixe, qui limite leur capacité à gérer de longues phrases CHO, MERRIËNBOER et al., 2014. Cette méthode a également été adaptée par [XU et al., 2015] pour la génération de légendes d'image, mais également par d'autres architectures pensées pour la traduction automatique [LUONG et al., 2015] ; [BRITZ et al., 2017], qui proposent

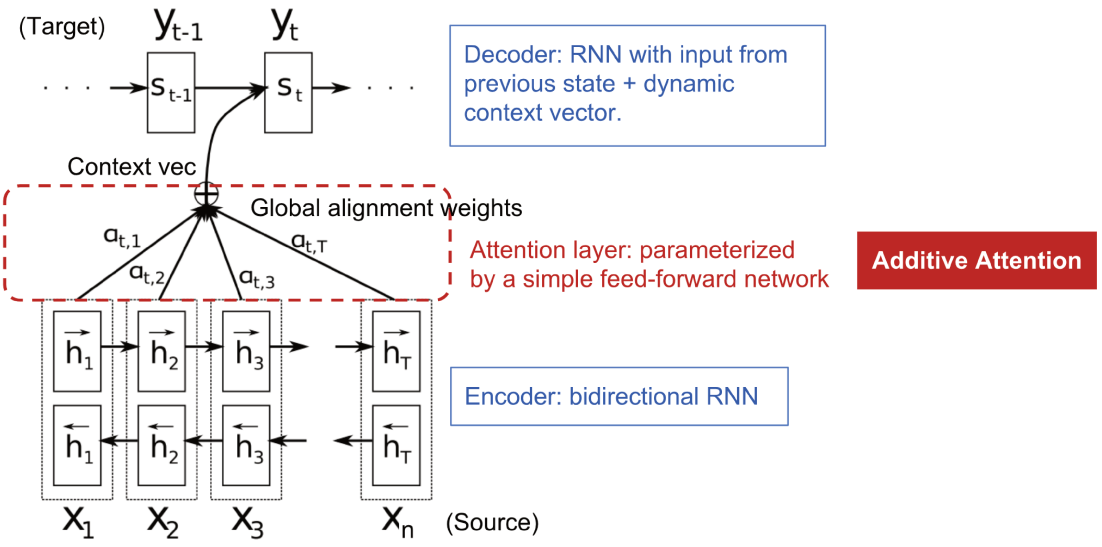


FIGURE 2.11 – Illustration du mécanisme d'attention comme défini par BAHDANAU et al., 2015.

d'autres formulations de l'attention.

Les modèles présentés précédemment reposent tous sur l'utilisation des réseaux de neurones convolutifs ou récurrents. VASWANI et al., 2017 propose avec le Transformer, un modèle conçu à l'origine pour des tâches de traduction automatique, s'affranchissant de ces architectures en les remplaçant par des mécanismes d'auto-attention et d'attention croisée plus simples que ses prédécesseurs, permettant la parallélisation : le *scaled dot-product*.

On considère en entrée une séquence de mots $(w_1, \cdots, w_N)$ et leurs plongements associés $(x_1, \cdots, x_N)$, et l'on cherche à produire une séquence de représentations contextualisées $(y_1, \cdots, y_N)$ de dimension ρ_{model}, avec $y_n \in \mathbb{R}^{\rho_{model}}$. La première étape du calcul de l'auto-attention consiste à créer pour chaque x_n trois vecteurs : $q_n \in \mathbb{R}^{\rho_q}$ un vecteur dit *requête* (en anglais *query*), $k_n \in \mathbb{R}^{\rho_k}$ un vecteur *clef* (*key*) et $v_n \in \mathbb{R}^{\rho_v}$ un vecteur *valeur* (*value*). Ces vecteurs sont respectivement calculés en multipliant les vecteurs x_n par les matrices W_Q, W_K et W_V. On peut alors calculer le vecteur d'attention propre au mot w_n comme $s_n = softmax\left(\frac{q_n \cdot K^T}{\sqrt{d_k}}\right)$. On utilise un softmax afin d'obtenir un vecteur sommant à 1. L'utilisation de $\sqrt{\rho_k}$ s'explique par la nécessité de limiter le produit scalaire en magnitude pour ne pas arriver dans les régions limites du softmax, évitant ainsi le problème d'évanescence du gradient. Une requête et une clef auront un produit scalaire d'autant plus important qu'ils sont plus similaires. Par souci d'efficacité, ce calcul peut être effectué sous forme matricielle. On note alors Q, K et V les matrices dont les lignes sont respectivement les vecteurs $q_{1:N}$,

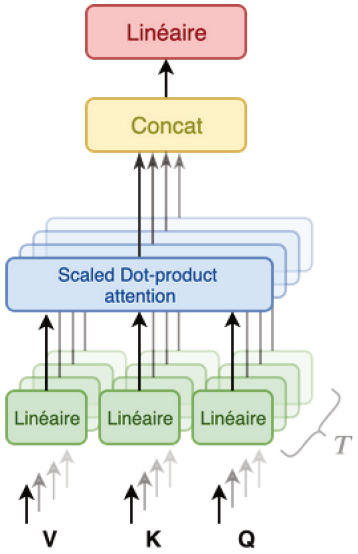


FIGURE 2.12 – Illustration de mécanisme d'attention multi-têtes. Les vecteurs valeurs, clef et requêtes sont passés dans T couches d'attentions différentes avant concaténation.

$k_{1:N}$ et $v_{1:N}$. Le coefficient d'attention est défini tel que :

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{\rho_k}}\right) V \tag{2.30}$$

Le produit $Q \cdot K^T$ est une matrice dont la $n^{\text{ième}}$ ligne est composée des produits scalaires entre la requête q_n et des vecteurs clefs k_{1_N}. Une fois ces scores d'attentions obtenus, le produit scalaire avec la matrice des valeurs permet d'obtenir une représentation contextualisée de w_n tel que $y_n = \sum_{i=1}^{N} s_{n,i} \cdot v_i$. Les scores d'attention permettent alors de donner plus de poids à certaines représentations des mots de la séquence en fonction de leur proximité avec le mot représenté.

Le Transformer introduit également une architecture appelée *Multi-head attention*, consistant en T tête d'attention en parallèle, dont les sorties sont concaténées. Les dimensions ρ_q, ρ_k et ρ_v sont alors égales à $\frac{\rho_{model}}{T}$. On assignera alors à chaque tête t des matrices W_Q^t, W_K^t et W_V^t, qui seront initialisées aléatoirement et apprises indépendamment. Une illustration de cette architecture est proposée à la Figure 2.12. En parallélisant différentes têtes d'attention, le Transformer améliore ses chances que tous les mécanismes d'attention ne captent pas les mêmes caractéristiques, permettant ainsi de produire des représentations de meilleure qualité. L'encodeur du Transformer utilise le mécanisme d'auto attention défini plus haut. Le décodeur, lui, possède un fonctionnement légèrement différent. Les clefs et les valeurs sont obtenues en sortie de l'encodeur, mais les requêtes proviennent des mots précédemment générés. Une représentation globale de l'architecture du Transformer est présentée Figure 2.13

Le modèle d'OpenAI GPT [RADFORD et al., p. d.] reprend la formulation du décodeur du transformer afin d'entraîner un modèle de langue. GPT cascade 12 décodeurs avec un

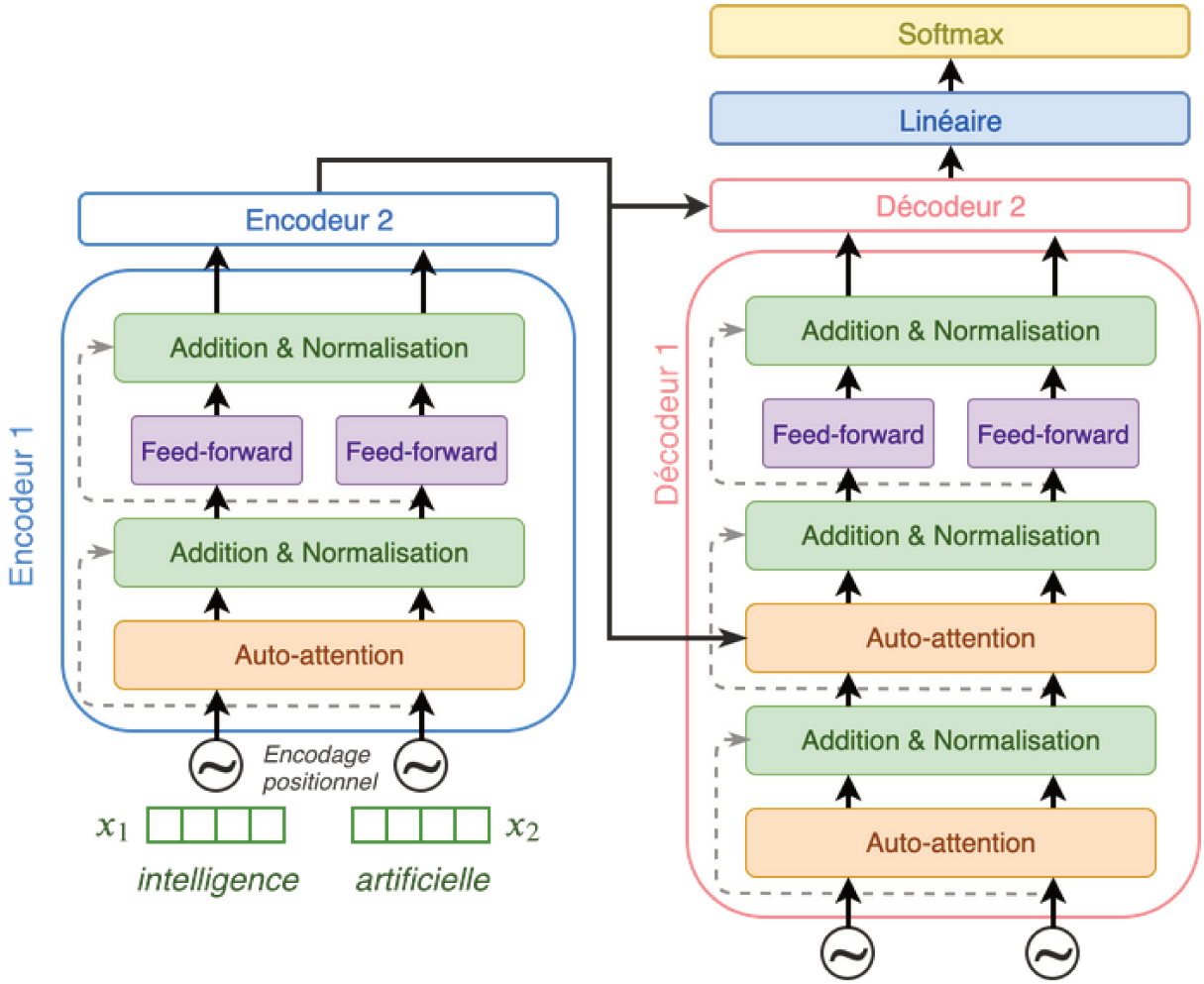


FIGURE 2.13 – Illustration de l'architecture du Transformer. En entrée une séquence de mot et leurs vecteurs de plongement x_1 et x_2. Les flèches grises représentent les connexions résiduelles.

mécanisme d'auto-attention, et cherche à prédire les mots suivants d'une séquence, cette tâche restant très générique et ne nécessitant pas d'annotations. Le modèle est pré-entraîné sur un corpus de plus de 7000 livres, et peut ensuite être raffiné de manière supervisée pour être adapté à un vaste panel de tâches spécifiques, comme l'analyse de sentiments.

BERT [DEVLIN et al., 2019] est un autre modèle de langue et repose lui sur l'encodeur du Transformer. Il propose deux implémentations : $BERT_{base}$, avec 12 couches de 12 têtes d'attention chacune, et $BERT_{large}$, avec 24 couches de 16 têtes chacune. Comme GPT, BERT est pré-entraîné sur des tâches génériques, mais elles sont au nombre de deux. La première consiste à masquer 15% des tokens de chaque phrase, en les remplaçant par un faux token `[MASK]`. Le modèle est alors optimisé afin de correctement prédire les mots masqués de la sorte. La seconde est une tâche de classification cherchant à prédire si deux phrases se suivent ou non. On forme un ensemble d'exemples composé à moité de phrases se suivant

dans le corpus d'entraînement et de phrases aléatoirement choisies. On optimise cette fois les paramètres du modèle au regard de cette tâche de classification.

Une fois pré-entraîné, BERT peut lui aussi être raffiné afin d'être adapté à des tâches de TAL plus spécifiques. Lors de sa publication, le modèle a permis d'atteindre d'excellents résultats sur une vaste gamme de tâches, tel GLUE [A. Wang et al., 2019] ou SQuAD [Rajpurkar et al., 2016] pour l'évaluation en questions-réponses. De nombreuses versions de BERT ont vu le jour, que ce soit en modifiant la procédure d'entraînement comme RoBERTa [Liu et al., 2019], en réduisant la taille du modèle par un processus de distillation comme DistilBERT [Sanh et al., 2019] ou simplement en proposant une version de BERT optimisée pour une langue particulière [Martin et al., 2020] ; [H. Le et al., 2020] ; [Nguyen et al., 2020].

Dans cette partie nous avons présenté et discuté certaines des méthodes les plus classiques permettant d'apprendre des représentations de mots et/ou de documents. Celles-ci n'ont cependant pas été cantonnées au TAL, et ont vite été adaptées à d'autres types de données, et en particulier à l'apprentissage de représentations pour les graphes.

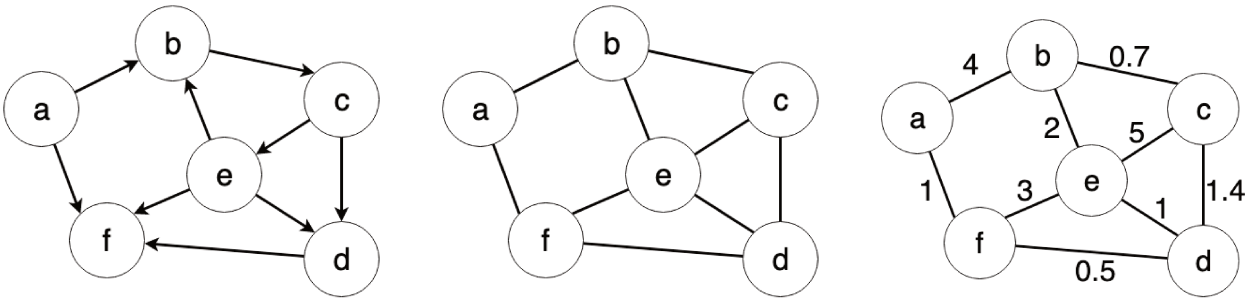
2.2 Plongement de graphes

Depuis les sept ponts de Königsberg jusqu'aux récents développements des moyens de communications, les graphes se sont révélés être de puissants outils d'analyse. La récente massification des usages du web, couplée à la nécessité grandissante de traiter informatiquement les données ont participé à de nombreuses avancées en matière d'analyse de graphes, avec une certaine attention portée à leur représentation : le plongement de graphes. Après un bref rappel sur la notion de graphe, nous discuterons dans cette section des différentes méthodes permettant l'apprentissage de plongement de sommets d'un graphe.

2.2.0.1 Rappels de théorie des graphes

On définit un graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comme la paire des deux ensembles $\mathcal{V}$, l'ensemble de sommets (indifféremment appelés noeuds par la suite) et $\mathcal{E} \subseteq (v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, l'ensemble des liens, avec $v_i \in \mathcal{V}$ un sommet, et $e_{ij} \in \mathcal{E}$ une arête entre v_i et v_j. On appelle degré du sommet v_i le nombre d'arcs reliant ce sommet, noté d_i. Un graphe sera dit orienté si les arêtes possèdent un sens, comme illustré Figure 2.14a. L'arête e_{ij} allant du noeud v_i au noeud v_j est alors différente de l'arête e_{ji} allant de v_j à v_i. Dans le cas contraire, le graphe sera dit non-orienté (Figure 2.14b). L'un et l'autre de ces types de graphe sera appelé graphe pondéré si on attribue à chaque arête une valeur. Par exemple, on peut représenter un réseau routier comme un graphe orienté, où chaque noeud est une intersection, chaque arête une

voie, pondérée par sa longueur. La Figure 2.14c présente un exemple de graphe non-orienté pondéré. Enfin, on parlera d'un graphe attribué lorsque les sommets portent également une information, que nous nommerons *attribut*. On pensera par exemple à un réseau de citations dans la littérature scientifique, où les sommets correspondent à des articles, les attributs au contenu textuel des articles, et les arcs aux citations. Bien que nous discuterons ici de quelques méthodes adaptées à l'apprentissage de représentations sur des graphes attribués, les méthodes spécialement conçues pour les réseaux de documents seront décrites dans la section suivante.



(a) Graphe orienté (b) Graphe non-orienté (c) Graphe pondéré

FIGURE 2.14 – Exemple de graphes orienté, non-orienté et pondéré.

Plusieurs concepts sont utiles à la description d'un graphe. Le premier est la matrice d'adjacence, notée $A \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$, dont chaque coefficient a_{ij} est défini comme suit :

$$a_{ij} = \begin{cases} 1 & \text{si } e_{ij} \in \mathcal{E} \\ 0 & \text{sinon} \end{cases} \tag{2.31}$$

La Figure 2.15 illustre la construction de la matrice d'adjacence d'un graphe simple.

La puissance n$^{\text{ième}}$ de la matrice d'adjacence, A^n, permet de connaître, pour chaque paire de sommets, le nombre exact de chemins distincts de longueur n entre eux. Le nombre de

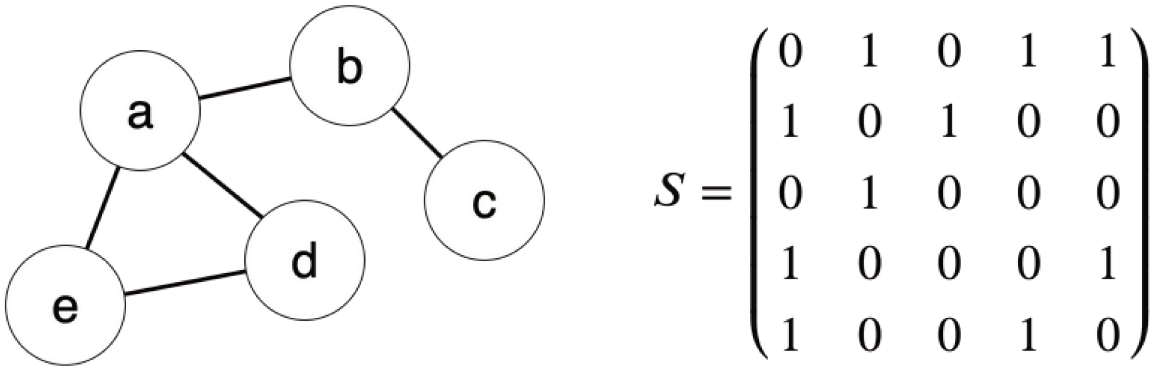


FIGURE 2.15 – Exemple de graphe non orienté avec la matrice d'adjacence correspondante.

chemins de longueur n ou moins sera alors donné par $(A + A^2 + \cdots + A^n)$. On note D la matrice des degrés, qui est la matrice diagonale de taille $|\mathcal{V}|$ des degrés des sommets de $\mathcal{G}$. On définit également la matrice Laplacienne, notée L, tel quel $L = D - A$. On appelle la matrice $L^{nor} = D^{-1}L$ le Laplacien normalisé symétrique de $\mathcal{G}$ et $L^{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. On note enfin $|A|$ le volume d'un graphe, tel que $|A| = \sum_i \sum_j a_{i,j} = \sum_i d_i$, la somme des degrés de tous les sommets.

La tâche de plongement de sommets, de façon analogue au plongement de mots ou de documents discutés à la section précédente, vise à apprendre des représentations $u_i \in \mathbb{R}^\rho$ des sommets v_i de $\mathcal{G}$, de telle façon que $\rho \ll |\mathcal{V}|$. On posera également $U \in \mathbb{R}^{|\mathcal{V}| \times \rho}$ la matrice dont chaque ligne correspond à la représentation u_i d'un sommet de $\mathcal{G}$. Par la suite, et à moins que cela ne soit explicitement spécifié, nous traiterons de graphes non orientés, aux arêtes non pondérées, et aux sommets non attribués.

2.2.1 Méthodes de réduction de dimension (manifold learning)

IsoMap [Tenenbaum et al., 2000] est une méthode de réduction de dimensions cherchant à préserver au mieux la distance géodésique[3] entre les sommets. On construit la matrice M, dont chaque coefficient m_{ij} représente la distance géodésique entre les v_i et v_j, calculée au moyen d'un algorithme de Dijkstra ou de Floyd-Warshall. On applique alors à cette matrice la méthode du positionnement multidimensionnel [Cox et al., 2001], afin d'obtenir des plongements des sommets préservant la distance géodésique des sommets du graphe construit précédemment. Il s'agit plus précisément d'estimer les valeurs de u_i et u_j, respectivement les représentations des noeuds v_i et v_j, minimisant la fonction suivante :

$$\min \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \left(m_{ij} - \|u_i - u_j\|^2\right)^2. \tag{2.32}$$

Locally Linear Embedding [Roweis et al., 2000] apporte une modification à Isomap, en proposant de s'affranchir du coûteux calcul des plus courts chemins entre chaque paire de sommets en se basant seulement sur la matrice d'adjacence A. On cherche cette fois-ci à minimiser l'erreur de reconstruction, le carré de la l-différence entre les deux plongements, comme présenté à l'Équation 2.33.

$$\min \sum_{i=1}^{|\mathcal{V}|} \|u_i - \sum_{j \in \{k | a_{ik} > 0\}} a_{ij} u_j\|^2. \tag{2.33}$$

3. En théorie des graphes, la distance géodésique entre deux sommets désigne le nombre d'arêtes qui composent le plus court chemin entre eux.

Laplacian Eigenmap [Belkin et al., 2003] est une méthode s'appuyant sur la matrice Laplacienne du graphe pour en tirer des représentations des sommets. On pose $U \in \mathbb{R}^{|\mathcal{V}|\times\rho}$ la matrice des représentations en ρ dimensions des sommets de $\mathcal{G}$, et u_i la i$^{\text{ième}}$ ligne de U, représentant le noeud v_i. On cherche alors les représentations des sommets v_i et v_j proches dans l'espace des représentations si les sommets sont liés. Plus précisément, on cherche des valeurs de u_i et u_j minimisant :

$$\sum_{i,j\in\mathcal{V}} a_{ij}(u_i - u_j)^2 \tag{2.34}$$

Cette expression peut cependant être réécrite ainsi :

$$\begin{aligned}
\frac{1}{2}\sum_{i,j\in\mathcal{V}} a_{ij}(u_i - u_j)^2 =& \frac{1}{2}\sum_{i,j\in\mathcal{V}} a_{ij}(u_i^2 + u_j^2 - 2u_iu_j) \\
=& \frac{1}{2}\left(\sum_{i\in\mathcal{V}} d_iu_i^2 + \sum_{j\in\mathcal{V}} d_ju_j^2 - 2\sum_{i,j\in\mathcal{V}} u_iu_ja_{ij}\right) \\
=& \sum_{i\in\mathcal{V}} d_iu_i^2 - \sum_{i,j\in\mathcal{V}} u_iu_ja_{ij} \\
=& U^\top DU - U^\top AU \\
=& U^\top LU
\end{aligned} \tag{2.35}$$

On cherche alors une valeur $\tilde{U}$ minimisant cette quantité, c'est-à-dire minimisant la distance entre les représentations des sommets liés, donnant l'objectif suivant :

$$\tilde{U} = \text{argmin}_U\, U^\top LU. \tag{2.36}$$

De plus, puisque cette quantité a un minimum trivial $U = 0$, les auteurs ajoutent une contrainte supplémentaire : $U^\top LU = I$. En utilisant la méthode des multiplicateurs de Lagrange, on trouve que la solution de l'Équation 2.36 est la matrice $\tilde{U}$ dont les lignes $\tilde{u}_i$ sont les solutions du problème aux valeurs propres suivant :

$$L\tilde{u}_i = \lambda_i D\tilde{u}_i, \tag{2.37}$$

avec λ_i un coefficient de Lagrange. Le vecteur de plongement de v_i peut alors être obtenu en prenant la i$^{\text{ème}}$ valeur des plus grands vecteurs propres $\tilde{u}_1, \tilde{u}_1, \cdots, \tilde{u}_\rho$.

2.2.2 Méthodes basées sur les marches aléatoires

Une marche aléatoire consiste en un chemin de longueur ℓ, partant d'un sommet $v^{(0)}$. À chaque itération t, le sommet suivant $v^{(t+1)}$ sera tiré selon une distribution uniforme entre tous les voisins de $v^{(t)}$. La Figure 2.16 propose la représentation d'une marche aléatoire sur un graphe.

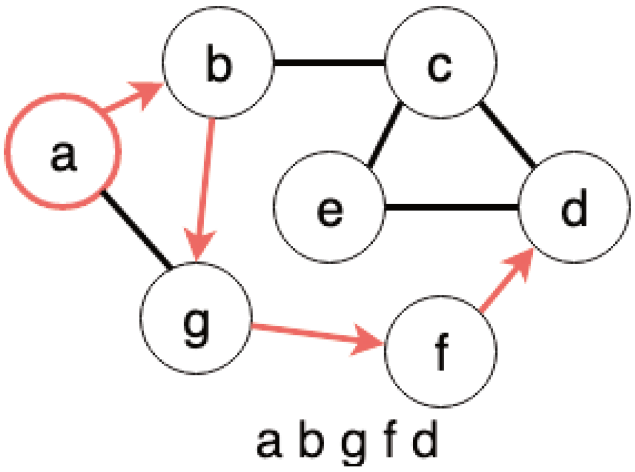


FIGURE 2.16 – Exemple de marche aléatoire de longueur 4 sur un graphe, avec l'enchaînement de sommets associés.

On définit une marche aléatoire comme une chaîne de Markov ayant pour coefficients p_{ij} de sa matrice de transition P :

$$p_{ij} = \begin{cases} \frac{1}{d_i} & \text{si } e_{ij} \in \mathcal{E} \\ 0 & \text{sinon,} \end{cases} \tag{2.38}$$

avec d_i le degré du noeud v_i. La probabilité de transition entre deux sommets liés v_i et v_j est donc une loi uniforme, fonction du degré de v_i. En effectuant n marches aléatoires de longueur ℓ sur un graphe $\mathcal{G}$, la collection ainsi obtenue peut être assimilée à un document de n phrases, chacune composée de ℓ mots, où chaque mot est un sommet de $\mathcal{G}$. Il a été noté que la distribution des fréquences d'occurrences des sommets dans les marches aléatoires sur de nombreux graphes réels tend à suivre une loi de puissance [PEROZZI et al., 2014], de la même façon que la distribution de fréquence d'occurrence des mots dans un corpus. Il devient donc pertinent d'étudier les motifs de co-occurrences, qui représentent le voisinage des sommets. On peut ainsi se rapprocher de certaines méthodes de plongements de mots dont nous avons déjà discuté à la Section 2.1.3. Nous discuterons principalement dans cette sous-section des différentes stratégies mises en place pour concevoir des marches aléatoires permettant de construire des séquences de sommets sur lesquelles seront appliqués Skip-Gram ou GloVe.

2.2.2.1 Méthodes basées sur Skip-Gram

PEROZZI et al., 2014 proposent avec DeepWalk de reprendre la formulation de Skip-Gram (avec softmax hiérarchique, *cf.* Section 2.1.3.1.1), afin de l'appliquer aux « phrases » issues des marches aléatoires. DeepWalk est constitué de deux parties. La première étape consiste donc à constituer un corpus $\mathcal{D}$ de marches aléatoires de longueur ℓ. La probabilité de transition entre

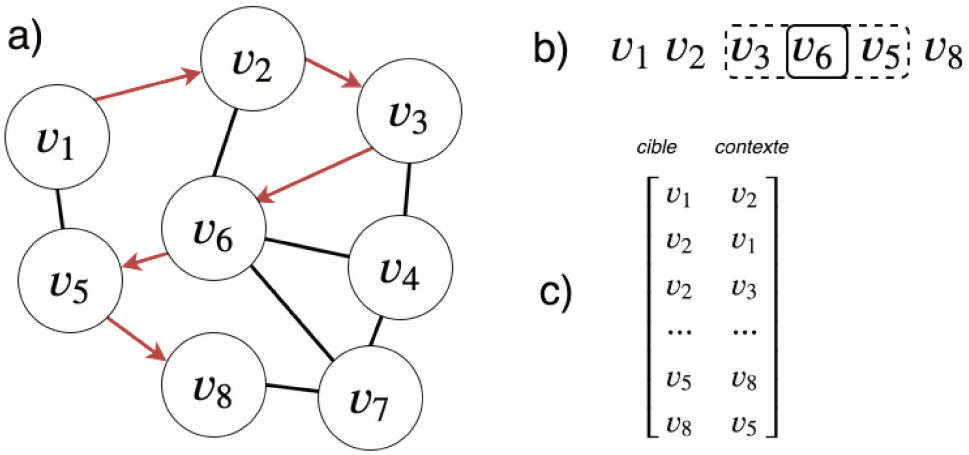


FIGURE 2.17 – Illustration de l'échantillonnage des sommets dans DeepWalk. En *(a)* on effectue une marche aléatoire de longueur $\ell = 6$ au départ du sommet v_1. La séquence de sommets obtenue est ensuite traitée comme une phrase sur laquelle on applique une fenêtre glissante (*(b)*) de taille $\tau = 1$. On obtient alors en *(c)* une série d'échantillons de co-occurrences.

deux sommets lors de ces marches aléatoires suit une loi uniforme identique à celle décrite à l'Équation 2.38. Cette étape est coûteuse en calcul, mais peut néanmoins être parallélisée.

Le second étage de DeepWalk consiste en une application de Skip-Gram sur le corpus ainsi construit afin d'apprendre les représentations des « mots », représentant ici les sommets de $\mathcal{G}$. On applique une fenêtre glissante de taille τ sur les séquences de sommets obtenues lors des marches aléatoires afin de construire des paires de sommets co-occurrents. La Figure 2.17 illustre la façon dont les paires de sommets sont échantillonnées. On apprend ensuite les représentations u_i des noeuds v_i de la même façon que celle détaillée à la Section 2.1.3.1.1.

Node2Vec [GROVER et al., 2016] apporte une modification à DeepWalk en modifiant la façon dont les marches aléatoires sont construites. Les auteurs considèrent deux stratégies de marches aléatoires :

— le parcours en profondeur, où partant d'un sommet de départ $v^{(0)}$, on choisit à chaque itération t le prochain sommet $v^{(t+1)}$ dans le voisinage du sommet courant $v^{(t)}$, en veillant à faire augmenter la distance géodésique entre le sommet de départ et le nouveau sommet. Cette méthode met en avant les caractéristiques d'homophilie[4]. Cette stratégie est illustrée par le parcours bleu à la Figure 2.18.

— le parcours en largeur, où sont d'abord privilégiés les sommets immédiatement connexes au sommet de départ, puis les sommets à une distance de 2 arêtes, et ainsi de suite.

4. Dans le contexte des graphes attribués on appelle *homophilie* le fait que les sommets aux attributs similaires tendent à être plus liés entre eux que les sommets aux attributs différents.

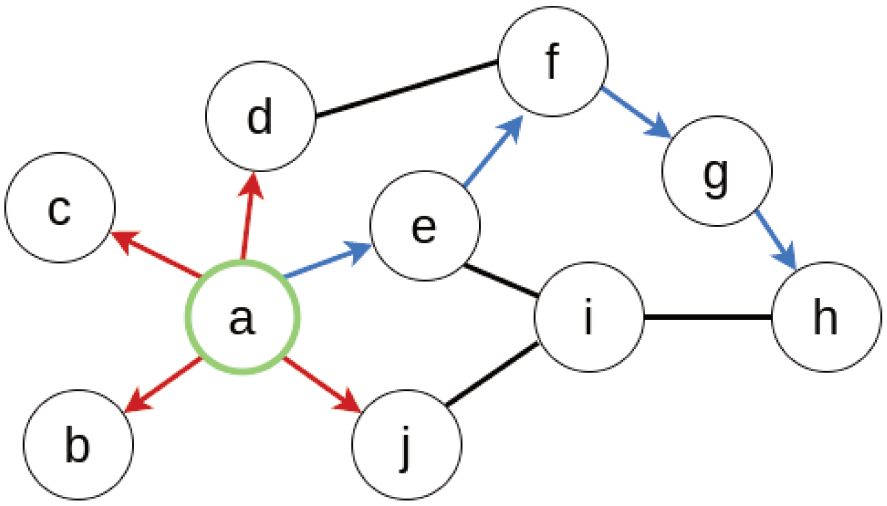


FIGURE 2.18 – Exemple de marches aléatoires de longueur 4 au départ du noeud a, en profondeur (en bleu) et en largeur (en rouge).

En mettant l'accent sur la notion de voisinage, ce type de marche aléatoire met en avant la notion d'équivalence structurelle. Cette stratégie est illustrée par le parcours rouge à la Figure 2.18.

Node2Vec adopte une stratégie pour biaiser les marches aléatoires afin de permettre un mélange de ces deux stratégies lors de la construction du corpus. En considérant une suite de sommets consécutifs v_h, v_i et v_j, comme illustré par exemple Figure 2.18, Node2Vec introduit des marches aléatoires d'ordre deux, faisant dépendre la transition entre v_i et v_j de la transition précédente entre v_h et v_i, tel que $\pi_{ij} = \alpha_{pq}(h, j)a_{ij}$ avec π_{ij} un vecteur non normalisé, représentant la transition entre v_i et v_j. On représente $\alpha_{pq}(h, j)$ tel que :

$$\alpha_{pq}(h,j) = \begin{cases} \frac{1}{p} & \text{si } m_{hj} = 0 \\ 1 & \text{si } m_{hj} = 1 \\ \frac{1}{q} & \text{si } m_{hj} = 2, \end{cases} \tag{2.39}$$

avec m_{hj} la distance géodésique entre les sommets v_h et v_j. p et q sont deux paramètres contrôlant la proportion des stratégies de parcours en longueur ou en largeur. p contrôle la probabilité que le marcheur retourne sur le noeud parcouru à l'itération précédente. Une faible valeur de p facilitera alors un parcours en largeur du graphe. Le paramètre q quant à lui détermine à quel point le marcheur favorisera le noeud v_j le plus éloigné de v_h, encourageant ainsi un parcours en profondeur du graphe lorsque $q < 1$. On définit alors une probabilité de transition entre le sommet v_i et v_j tel que :

$$p_{ij} = \begin{cases} \frac{\pi_{ij}}{Z} & \text{si } e_{ij} \in \mathcal{E} \\ 0 & \text{sinon,} \end{cases} \tag{2.40}$$

avec Z une constante de normalisation. On note que dans le cas de la stratégie employée dans DeepWalk, reposant sur une loi uniforme fonction du degré de v_i, que $\pi_{ij} = a_{ij}$.

Les séquences de sommets obtenues avec ces marches aléatoires sont ensuite passées en entrée de Skip-gram avec échantillonnage négatif. Chaque séquence issue de marches aléatoires sera traitée comme une phrase, et chaque sommet de cette séquence sera traité comme un mot. Node2Vec apprend alors des représentations des sommets en se basant sur leurs motifs de co-occurrence au sein des marches aléatoires. Les auteurs notent au travers de leurs expériences que la stratégie décrite plus haut tend à donner des représentations de meilleure qualité que celles obtenues avec DeepWalk, encodant à la fois la notion d'homophilie et celle d'équivalence structurelle.

Large-scale Information Network Embedding LINE, [J. Tang et al., 2015] est un modèle proposant également de prendre en compte les proximités de premier ordre, impliquant que les sommets similaires sont liés entre eux, ou de second ordre, qui signifie que deux sommets seront également similaires s'ils ont des voisinages semblables. En effet, l'absence de liens explicites entre deux éléments n'implique pas pour autant qu'ils ne sont pas similaires. Pour modéliser la proximité du premier ordre, LINE utilise la probabilité jointe de v_i et v_j tel que :

$$p_{1^{st}}(v_i, v_j) = \frac{1}{1 + e^{-u_i \cdot u_j}}, \tag{2.41}$$

avec u_i la représentation du sommet v_i.

La proximité du second ordre est, quant à elle, modélisée par la probabilité que le sommet v_j, représentant le contexte, soit généré en présence de v_i. Ainsi :

$$p_{2^{nd}}(v_j|v_i) = \frac{e^{c_j \cdot u_i}}{\sum_{k=1}^{|\mathcal{V}|} e^{c_k \cdot u_i}}, \tag{2.42}$$

avec c_j la représentation contextuelle de v_j. LINE pose enfin $\hat{p}_{1^{st}}(v_i, v_j) = \frac{a_{i,j}}{A}$ et $\hat{p}_{2^{nd}}(v_j|v_i) = \frac{a_{i,j}}{d_i}$, respectivement les distributions empiriques des probabilités de proximité de premier et second ordre. LINE cherche alors à minimiser la divergence de Kullback-Leibler entre $p_{1^{st}}$ et $\hat{p}_{1^{st}}$, ou $p_{2^{nd}}$ et $\hat{p}_{2^{nd}}$. L'optimisation est faite au moyen d'une descente de gradient stochastique à laquelle est couplée un échantillonnage négatif pour approcher l'Équation 2.42. Une représentation d'un sommet prenant en compte les deux premiers ordres de proximité sera alors un vecteur résultant de la concaténation des représentations apprises pour chacun des deux ordres (la dimension de cette représentation de v_i sera donc la somme de celles des premier et second ordres).

Qiu et al., 2018 proposent enfin, avec NetMF, une formulation unifiant plusieurs modèles de plongement de graphes comme DeepWalk , Node2Vec ou LINE, sous la forme d'une factorisation de matrice. Ce résultat s'appuie grandement sur les travaux de Levy et al., 2014 montrant que Skip-Gram avec échantillonnage négatif était une factorisation implicite

de la matrice SPMI, discutés à la Section 2.1.3.1.2. Les auteurs montrent que DeepWalk cherche à calculer une approximation de rang faible de la matrice suivante :

$$\log\left(|A|\left(\frac{1}{\tau}\sum_{r=1}^{\tau}(D^{-1}A)^r\right)D^{-1}\right)-\log n_k, \tag{2.43}$$

avec τ la taille de la fenêtre glissante, $|A|$ le volume de $\mathcal{G}$ et n_k le nombre d'échantillons négatifs. LINE, cas particulier de DeepWalk où $\ell = 1$, devient alors une approximation de la matrice :

$$\log\left(|A|D^{-1}AD^{-1}\right)-\log n_k. \tag{2.44}$$

Ces analyses permettent aux auteurs de proposer NetMF, une méthode de plongement de graphe reposant sur la factorisation de matrice. NetMF présente deux variantes, l'une exacte, et l'autre approchée. Le choix de l'une ou l'autre dépend principalement de la taille τ de la fenêtre choisie. Premièrement on pose la matrice M, dite *matrice DeepWalk*, tel que :

$$M=\frac{|A|}{n_k\tau}\left(\sum_{r=1}^{\tau}(D^{-1}A)^r\right)D^{-1}. \tag{2.45}$$

.

Dans le cas où τ est petit, on constate bien que les calculs successifs des puissances de $D^{-1}A$ demeurent encore facilement exécutables. Cependant l'Équation 2.43 comporte un logarithme, ce qui n'est pas sans poser certains problèmes. Premièrement, lorsque m_{ij}, un coefficient de la matrice M, est nul, on obtient $\log m_{ij} = -\infty$, rendant le calcul numériquement impossible. De plus, M est une matrice dense, ce qui rend sa décomposition en valeurs singulières coûteuse en calculs. Pour palier ces deux problèmes les auteurs reprennent l'idée de la *Shifted* PPMI de Levy et al., 2014, et définissent une matrice M' tel que $m'_{ij} = \max(m_{i,j}, 1)$. $\log(M')$ est alors une matrice creuse, et on peut facilement construire le plongement du graphe en effectuant une décomposition en valeurs singulières et en ne conservant que les ρ valeurs propres les plus importantes.

Lorsque τ devient grand, le calcul des puissances successives du Laplacien normalisé devient lourd en termes de coût de calculs, et il devient alors nécessaire de l'approximer. Le Laplacien normalisé est alors approché en ne conservant que ses g plus grandes valeurs propres :

$$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\approx U_g\Lambda_g U_g^{\top} \tag{2.46}$$

La matrice $\hat{M}$ approchant la matrice M est ensuite calculée comme suit :

$$\hat{M}=\frac{|A|}{n_k}D^{-\frac{1}{2}}U_g\left(\frac{1}{l}\sum_{r=1}^{l}\Lambda_g^r\right)U_g^{\top}D^{-\frac{1}{2}} \tag{2.47}$$

Comme pour la version exacte, on calcule ensuite $\hat{M}' = \max(\hat{M}, 1)$ puis on applique une décomposition en valeurs singulières de $\log(\hat{M}')$.

2.2.2.2 Méthode basée sur GloVe

Brochier et al., 2019, avec Global Vectors for Node Representation (GVNR) exploitent également une méthode de plongement de mots, mais à la différence des modèles discutés plus haut reposant sur Skip-Gram, celui-ci s'appuie sur GloVe (présenté Section 2.1.3.3). GVNR apprend des représentations U et C des sommets d'un graphe en factorisant la matrice $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, la matrice de co-occurrences tronquée des noeuds du graphe. Cette matrice est construite comme suit :

- on réalise μ marches aléatoires de longueur ℓ afin de constituer un corpus de séquences de sommets ;
- on fait glisser sur chacune de ces séquences une fenêtre de taille τ afin d'incrémenter les co-occurrences entre les sommets dans la matrice S. A chaque paire de sommets (v_i, v_j) ainsi formée on incrémentera la valeur s_{ij} de $\frac{1}{q}$, avec q la distance au sein de la séquence dans la fenêtre entre v_i et v_j ;
- on met à 0 toutes les valeurs $s_{ij} < s_{min}$, où s_{min} est un hyperparamètre du modèle.

GVNR va ensuite construire les deux matrices U et C, respectivement les matrices cible et contexte dont les vecteurs lignes u_i et c_i sont les représentations cible et contexte du noeud v_i, de façon à minimiser l'erreur de reconstruction suivante :

$$\underset{U,C,b^u,b^c}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} \delta(s_{ij}) \left(u_i \cdot c_j + b_i^U + b_j^C - \log(1+s_{ij})\right)^2, \tag{2.48}$$

avec b_i^U et b_j^C des scalaires propres aux représentations cibles de v_i et contextuelles de v_j, et

$$\delta(s_{ij}) = \begin{cases} 1 & \text{si } s_{ij} > 0, \\ m_i & \text{sinon, où } m_i \sim \text{Bernoulli}(\alpha_i). \end{cases} \tag{2.49}$$

$\delta(s_{ij})$ est égal à 1 lorsque les co-occurrences sont positives, et est déterminé par une variable aléatoire suivant une loi de Bernoulli de paramètre α_i, propre à chacun des sommets du graphe. $\delta(s_{ij})$ est défini tel que :

$$\alpha_i = \begin{cases} n_k \frac{p_i}{a - p_i} & \text{si } p_i \leq (n_k+1)^{-1} \\ 1 & \text{sinon.} \end{cases} \tag{2.50}$$

$p_i = \frac{|s_i|_{\neq 0}}{|\mathcal{V}|}$ est la proportion de coefficients non nuls de la i$^{\text{ième}}$ ligne de S, et n_k est un hyperparamètre positif du modèle contrôlant la proportion de coefficients nuls intégrés dans le calcul de l'erreur. La Figure 2.19 propose un exemple d'échantillonnage des s_{ij} considérés lors de la factorisation.

Les expérimentations montrent que GVNR permet d'obtenir des représentations de qualité équivalente ou supérieure que les méthodes reposant sur SGNS sur plusieurs réseaux réels, en ayant une complexité en temps moindre.

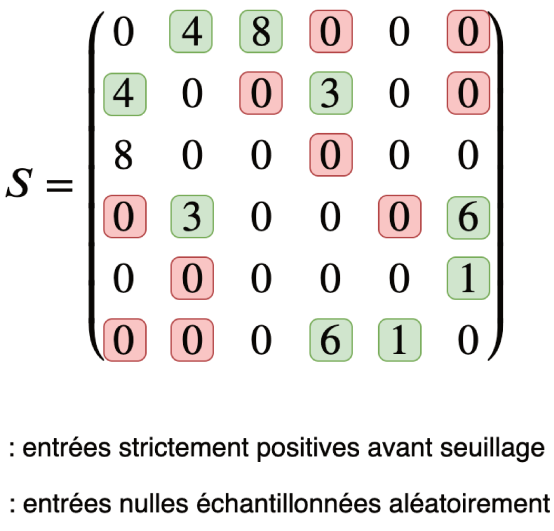


FIGURE 2.19 – Une matrice de comptage des co-occurrences de GVNR, où les entrées strictement positives et certaines entrées nulles sont échantillonnées. Ici, $n_k = 1$, signifiant que autant d'entrées positives que d'entrées nulles sont échantillonnées.

2.2.3 Méthodes à base de réseaux de neurones profonds

2.2.3.1 GraphSage

HAMILTON et al., 2017 propose avec GraphSAGE (pour *SAmple and AGgregate*) une méthode inductive d'apprentissage de représentations de sommets. Cette méthode alterne entre deux fonctions, l'une agrégeant le voisinage d'un noeud et l'autre mettant à jour sa représentation. On considère comme données d'apprentissage la matrice d'adjacence A et la matrice des attributs $X \in \mathbb{R}^{|\mathcal{V}| \times d}$, avec d la dimension des attributs, où x_i est la i^{me} ligne de X correspondant aux attributs de v_i. On note K le degré maximal de proximité considéré (pour $k = 1$ on considérera les voisins d'un noeud, pour $k = 2$ les voisins de ses voisins, etc). On note $W^{(k)}, \forall k \in \{1, \cdots, K\}$ la matrice de poids du k$^{\text{ième}}$ degré de proximité dans la fonction d'agrégation. On définit $h_{v_i}^{(k)}$ la représentation intermédiaire du sommet v_i, avec $k \in \{1, \cdots, K\}$ l'ordre de son voisinage, et, par souci de notation, on définit $u_i = h_{v_i}^{(K)}$ la représentation finale de v_i. Enfin on définit comme première représentation des données $h_{v_i}^{(0)} = x_i$, de façon à prendre en compte les attributs des sommets. Le fonctionnement de GraphSAGE est résumé à la Figure 2.20.

On apprend les représentations u_i des sommets en estimant les paramètres que sont les matrices de poids W^k de chaque couche $k \in \{1, \cdots, K\}$ et les paramètres des fonctions d'agrégation permettant de minimiser la fonction suivante :

$$\mathcal{L} = -\log(\sigma(u_i u_j) - n_k \cdot \mathbb{E}_{v_n \sim P^-(v_j)} \log(\sigma(-u_i u_n)), \tag{2.51}$$

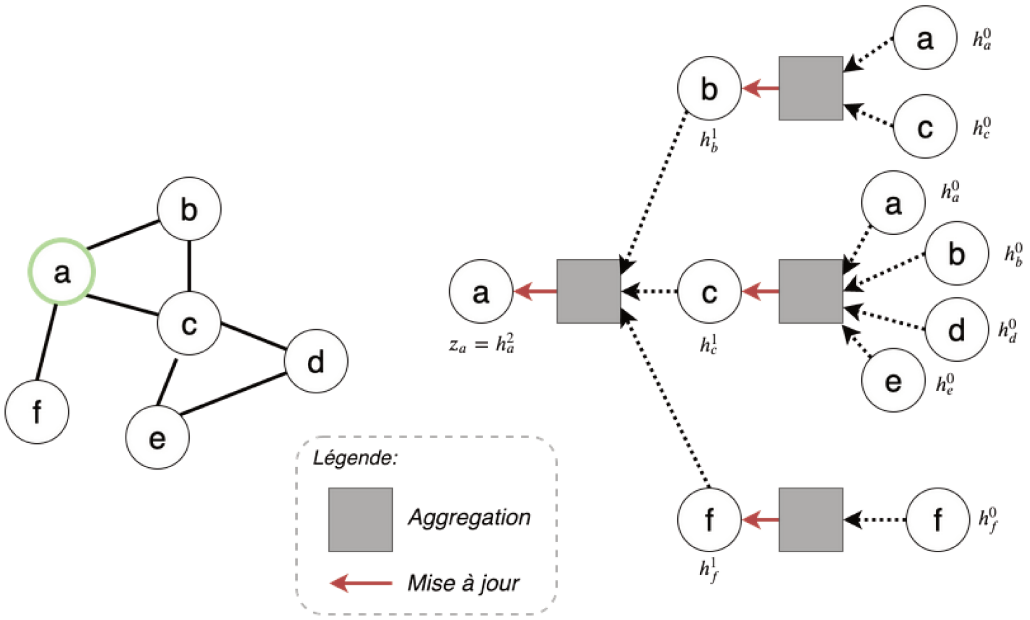


FIGURE 2.20 – Représentation du fonctionnement de GraphSAGE pour $k = 2$. A gauche un graphe de 6 sommets, et à droite le schémas d'agrégation permettant la représentation du sommet a.

avec v_j un noeud qui co-occurre avec v_i, n_k le nombre d'exemples négatifs, $P^-(v_j)$ une distribution d'échantillonnage négatif et σ la fonction sigmoïde. L'optimisation se fait au moyen d'une descente de gradient stochastique.

Les auteurs proposent plusieurs méthodes permettant d'agréger le voisinage d'un noeud. La première, et probablement la plus simple, consiste à calculer h_u^k comme la moyenne des représentations h_v^{k-1} de ses voisins. Une autre approche envisagée consiste à utiliser un LSTM afin d'agréger les représentations du voisinage d'un sommet. Enfin les auteurs proposent comme troisième approche une méthode dite de *pooling*. Cette fois, l'architecture retenue est un réseau de neurones à une couche cachée, suivie d'une couche de *max-pooling*, c'est-à-dire ne conservant pour chaque coefficient de h_u^k que la valeur la plus grande des h_v^{k-1}. Le fonctionnement du max-pooling est illustré Figure 2.21. Cela permet de capter différents aspects du voisinage. Bien que plus complexe, l'utilisation de ce type d'architecture offre une plus grande expressivité de l'apprentissage. Les expériences montrent d'ailleurs que l'utilisation de LSTM ou du *pooling* améliore sensiblement les performances du modèle, bien que les LSTM demeurent plus lents lors de l'entraînement. Les expériences montrent également que le choix de $k = 2$ permet d'obtenir les représentations de sommets les plus consistantes et offrant les meilleures performances sur des tâches de prédictions de liens, comparé à $k = 1$. Pour $k \geq 3$ le gain de performance n'est pas significatif, mais le temps de calcul augmente dramatiquement (de l'ordre d'une ou plusieurs magnitudes).

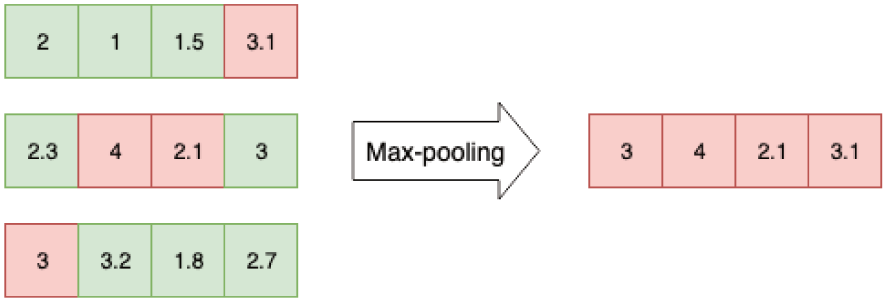


FIGURE 2.21 – Illustration du mécanisme de *max-pooling* avec trois vecteurs. Chaque coefficient x_i du vecteur construit par max-pooling correspond à la plus grande valeur parmi les i$^{\text{ième}}$ coefficient des vecteurs d'entrée.

2.2.3.2 Structural Deep Network Embedding

Structural Deep Network Embedding SDNE, [D. WANG et al., 2016] est une méthode d'apprentissage de représentation de sommets cherchant à encoder au mieux la structure du graphe. Les auteurs proposent un réseau de neurones profond constitué de plusieurs couches aux fonctions non-linéaires, permettant ainsi d'apprendre des représentations latentes des sommets dans un espace de représentation lui aussi non-linéaire. La gestion de la préservation de la structure et de l'aspect creux du réseau est, quant à lui, géré en prenant en compte non seulement le voisinage du premier ordre mais également celui du second. SDNE prend la forme d'un réseau de neurones semi-supervisé, suivant l'architecture des auto-encodeurs profonds [SALAKHUTDINOV et al., 2009].

Les auto-encodeurs constituent une catégorie d'architectures de réseaux de neurones, composées de deux parties : un encodeur et un décodeur. L'encodeur apprend une représentation en faible dimension u_i d'une donnée d'entrée x_i (par exemple une image, comme l'exemple proposé à la Figure 2.22), puis le décodeur cherche à reconstruire à partir de cette représentation une valeur $\tilde{x_i}$, supposément proche de l'original x_i. Les paramètres du réseau sont alors optimisés afin de minimiser l'erreur de reconstruction.

SDNE opère sur des graphes attribués. Les données utilisées par le modèle sont la matrice d'adjacence A, et la matrice des attributs $X \in \mathbb{R}^{|\mathcal{V}| \times d}$. On notera également $\tilde{x_i}$ la reconstruction en sortie du décodeur de l'attribut x_i du sommet v_i. Les auteurs notent que dans beaucoup de réseaux réels, l'absence d'un lien observé entre deux sommets n'indique pas nécessairement que celui-ci n'est pas légitime. On ne peut donc pas se baser uniquement sur la proximité du premier ordre comme critère de reconstruction.

L'encodeur de SDNE est un réseau à K couches, encodant les données d'entrée x_i au travers de k états cachés $h_i^{(k)}$. Par soucis de notation on désignera $h_i^{(K)}$, le dernier état caché de l'encodeur, u_i. La règle de propagation au sein de l'encodeur est la suivante :

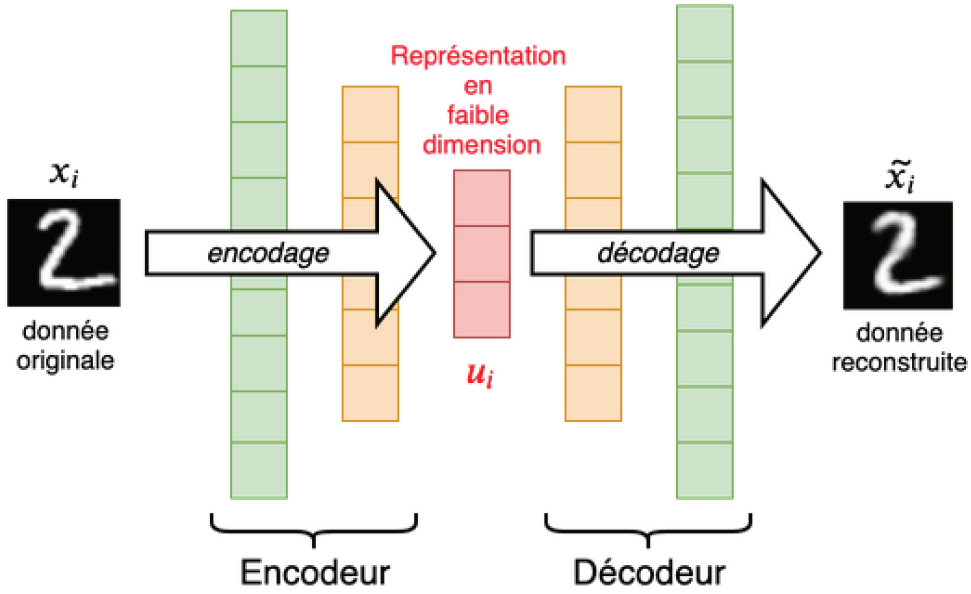


FIGURE 2.22 – Illustration d'un auto-encodeur simple, appliqué à une image issue du jeu de données MNIST. L'image originale est passée en entrée de l'encodeur, qui en produit une représentation en faible dimension (ici en rouge). Celle-ci est ensuite entrée dans le décodeur, qui cherchera à reconstruire au mieux l'image à partir de la représentation.

$$h_i^{(1)} = \sigma\left(W^{(1)} x_i + b^{(1)}\right) \tag{2.52}$$

$$h_i^{(l)} = \sigma\left(W^{(l)} h_i^{(l-1)} + b^{(l)}\right), \tag{2.53}$$

avec σ la fonction sigmoïde, $W^{(l)}$ et $b^{(l)}$ la matrice de poids et le biais de la $l^{\text{ième}}$ couche du réseau. Le décodeur permet lui de reconstruire $\tilde{x_i}$ à partir de u_i en « renversant » le processus de calcul de l'encodeur.

Les représentations u_i des sommets v_i sont alors obtenus en estimant les paramètres $W^{(l)}$ et $b^{(l)}$ en fonction des attributs x_i, minimisant une fonction de perte. On considère la fonction de perte suivante, prenant en compte la proximité au second ordre :

$$\mathcal{L}_{2^{nd}} = \sum_{i=1}^{|\mathcal{V}|} \|(\tilde{x_i} - x_i) \circ \mathbf{b_i}\|_F^2, \tag{2.54}$$

avec $\circ$ le produit de Hadamard et $\mathbf{b_i} = \{b_{i,j}\}_{j=1}^{|\mathcal{V}|}$ où :

$$b_{i,j} = \begin{cases} 1 & \text{si } a_{i,j} = 0 \\ \beta > 1 & \text{sinon.} \end{cases} \tag{2.55}$$

β est un hyperparamètre contrôlant la contribution des éléments non-nuls dans la reconstruction. Lorsque $\beta = 1$, cette expression du critère de reconstruction permet d'assurer la prise

en compte du voisinage d'ordre 2, et ainsi l'apprentissage de la structure globale du graphe. Cette partie de SDNE ne requiert pas de supervision.

Il est cependant nécessaire de considérer également la proximité de premier ordre afin de capturer la structure locale du réseau. Pour cela, on ajoute une seconde fonction de perte assurant que les représentations latentes u_i et u_j des sommets v_i et v_j soient proches l'une de l'autre. On a ainsi :

$$\mathcal{L}_{1^{st}} = \sum_{i,j=1}^{|\mathcal{V}|} a_{i,j} \|u_i - u_j\|_2^2 \tag{2.56}$$

Cette formulation n'est pas sans rappeler celle de Laplacian Eigenmap, et vise à pénaliser les plongements de sommets liés dans le graphe mais éloignés dans l'espace des représentations. L'utilisation d'une donnée connue telle que la matrice d'adjacence peut être vu comme une semi-supervision.

SDNE estime alors les représentations u_i des sommets de façon à minimiser la fonction de perte suivante, somme des deux fonctions décrites aux Équations 2.55 et 2.56, tel que :

$$\mathcal{L} = \mathcal{L}_{2^{nd}} + \alpha \mathcal{L}_{1^{st}} + \gamma \mathcal{L}_{reg}, \tag{2.57}$$

avec α un hyperparamètre contrôlant l'importance donnée à la structure locale lors de l'apprentissage (si $\alpha = 0$, seule la proximité du second ordre sera considérée), γ un hyperparamètre contrôlant le terme de régulation, et $\mathcal{L}_{reg}$ un terme de régularisation de type $L2$, prévenant le sur-apprentissage. Les plongements de sommets obtenus avec SDNE ont permis lors de sa publication d'excellents résultats sur des tâches de prédiction de liens et de classification multi-labels. La Figure 2.23 résume l'architecture de SDNE et illustre le mécanisme de semi-supervision.

2.2.3.3 Les Graph Convolutional Networks

Les *Graph Convolutional Networks* GCN, [Kipf et al., 2016a] sont une catégorie de réseaux de neurones à l couches visant à apprendre des représentations de sommets d'un graphe attribué pour des tâches de classification. Cela se fait en agrégeant successivement les représentations de leurs voisinages, au moyen d'un réseau de neurones convolutif spécialement adapté aux graphes. On considère comme données la matrice d'adjacence $A \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ et la matrice de attributs $X \in \mathbb{R}^{|\mathcal{V}|\times F}$, avec F le nombre de *features*. On définit $H^{(l)} \in \mathbb{R}^{|\mathcal{V}|\times d}$ comme la matrice d'activation de la couche l, $\hat{A} = A + I_{|\mathcal{V}|}$, $\hat{D}$ la matrice des degrés de $\hat{A}$, $W^{(l)}$ la matrice des poids de la $l^{\text{iéme}}$ couche, et σ une fonction d'activation non linéaire (type RELu). On pose également $H^{(0)} = X$. La règle de propagation est la suivante :

$$H^{(l+1)} = f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right). \tag{2.58}$$

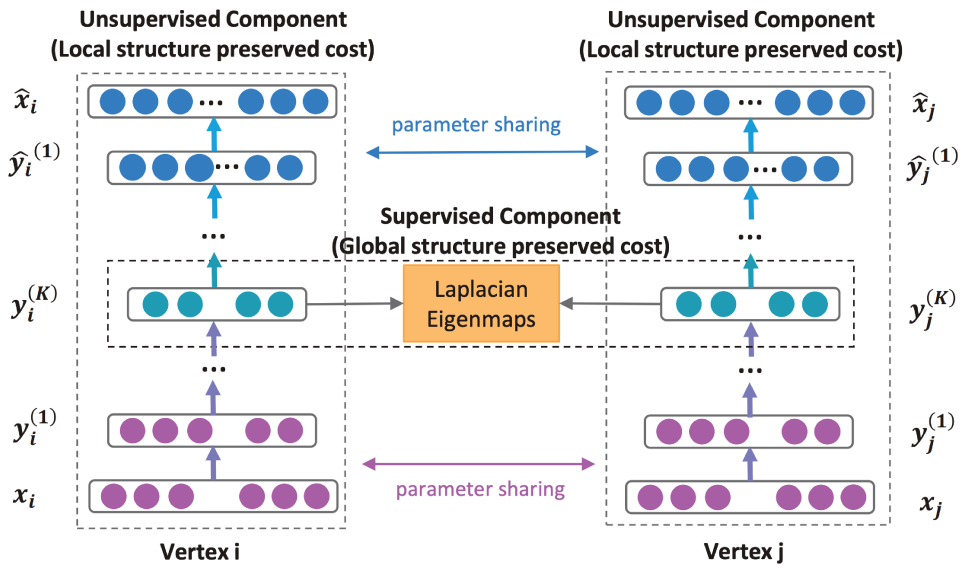


FIGURE 2.23 – Illustration de l'architecture du réseau de neurones semi-supervisé de SDNE, tirée de D. WANG et al., 2016.

Plusieurs astuces entrent en ligne de compte dans cette règle de mise à jour. Dans un premier temps, l'ajout de la matrice identité à la matrice d'adjacence permet de forcer la présence de boucles afin de prendre en compte le contenu du noeud, et non plus seulement celui de son voisinage. Ensuite, puisque A n'est pas normalisé en ligne, une multiplication avec cette matrice entraînerait un changement d'échelle des vecteurs d'attributs. Les auteurs font le choix d'une normalisation symétrique au moyen de la matrice des degrés, donnant ainsi $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$.

Les auteurs choisissent d'employer l'entropie croisée afin d'estimer les matrices de poids $W^{(0)}$ et $W^{(1)}$, optimisées par descente de gradient. En notant $\mathcal{Y}_L$ l'ensemble des sommets labellisés, la fonction à minimiser est de la forme :

$$\mathcal{L} = -\sum_{i \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{if} \ln Z_{if} \tag{2.59}$$

L'architecture proposée par les auteurs prend la forme d'un GCN à deux couches, l'augmentation du nombre de couches dégradant les performances en classification [Q. LI et al., 2018]. En effet, les GCN opèrent un lissage Laplacien (*Laplacian smoothing*) sur les noeuds, mélangeant leurs attributs et ceux de leurs voisinages. Plus le réseau sera profond, plus le lissage tendra à mélanger les points appartenant à des classes différentes, jusqu'à les rendre difficiles à distinguer.

2.2.3.4 Variational Graph Auto-Encoder

Variational Graph Auto-Encoder VGAE, [Kipf et al., 2016b] est un réseau de neurones suivant l'architecture des auto-encodeurs variationnels [Kingma et Welling, 2014] ; [Rezende et al., 2014]. L'encodeur est un GCN à deux couches, tandis que le décodeur est un simple produit scalaire. L'encodeur cherche à produire à partir de de la matrice d'adjacence $\hat{A} = A + I_{|\mathbb{V}|}$ et de la matrice des attributs $X \in \mathbb{R}^{|\mathcal{V}| \times F}$ des représentations u_i des sommets du graphe. On notera U la matrice dont les lignes sont les vecteurs u_i. Le modèle d'inférence est simplement paramétré par une GCN, tel que :

$$p(U|X, A) = \prod_{i=1}^{|\mathcal{V}|} q(u_i|X, A) \tag{2.60}$$

avec :

$$q(u_i|X, A) = \mathcal{N}(u_i|\mu_i, \text{diag}(\sigma_i^2)), \tag{2.61}$$

où $\mu = GCN_\mu(X, A)$ est la matrice des vecteurs moyenne μ_i, et de manière similaire $\log \sigma = GCN_\sigma(X, A)$. Les GCN à deux couches sont identiques à ceux décrits à la sous séction précédente 2.2.3.3, et sont de la forme $GCN(X, A) = \hat{A}\text{ReLU}(\hat{A}XW^{(0)})W^{(1)}$. On fixe la matrice de poids $W^{(0)} = X$, et ces poids sont partagés par GCN_μ et GCN_σ.

Le modèle génératif est le suivant :

$$p(A|U) = \prod_{i=1}^{|\mathcal{V}|} \prod_{j=1}^{|\mathcal{V}|} p(a_{i,j}|u_i, u_j), \tag{2.62}$$

avec :

$$p(a_{i,j}|u_i, u_j) = \sigma(u_i^\top u_j). \tag{2.63}$$

où $\sigma(\cdot)$ est la fonction sigmoïde. L'optimisation est alors faite en la borne inférieure $\mathcal{L}$ en fonction des paramètres variationnels $W^{(i)}$:

$$\mathcal{L} = \mathbb{E}_{q(U|X,A)}[\log p(A|U)] - KL[q(U|X, A)p(U)], \tag{2.64}$$

avec $p(\mathbf{Z}) = \prod_i p(\mathbf{z_i}) = \prod_i \mathcal{N}(\mathbf{z}_i \mid 0, \mathbf{I})$ un a priori gaussien.

Les auteurs proposent également une approche non variationnelle (GAE, pour Graph Auto-Encoder) qui cherchera elle à reconstruire la matrice d'adjacence $\tilde{A}$, tel que :

$$\tilde{A} = \sigma(ZZ^\top) \text{ , avec } Z = \text{GCN}(A, X) \tag{2.65}$$

2.2.3.5 Les Graphs Attentional Networks

Une autre architecture adaptée aux graphes attribués a été proposé par Veličković et al., 2018, avec les *Graph Attentional Networks* (GAT). Ces travaux étendent les concepts

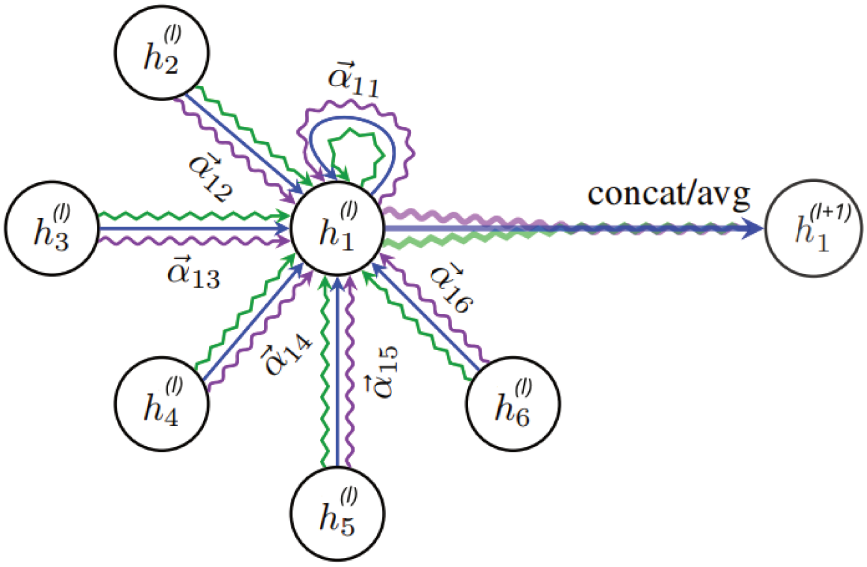


FIGURE 2.24 – Illustration, tirée de [VELIČKOVIĆ et al., 2018], de l'attention multi-têtes utilisée dans GAT (ici $k = 3$) et appliquée sur le noeud v_1 et son voisinage. Chacune des couleurs de flèches indique une tête indépendante. Les sorties de chaques têtes sont ensuite concaténées ou moyennées afin d'obtenir $h_1^{(l+1)}$.

développés avec le Transformer (*c.f.* Section 2.1.5.2) aux données structurées en graphes. Plus précisément, GAT s'appuis sur l'attention multi-têtes afin de dépasser certaines limitations des réseaux convolutionnels. X représente la matrice des attributs, et A la matrice d'adjacence du graphe. On note également K le nombre de têtes d'attention pour chaque couche, et L le nombre de couches.

Le mécanisme d'attention multi-têtes est constitué de K têtes d'attention indépendantes. Chacune des k têtes de la $l^{\text{ième}}$ couche du réseau de neurones permet d'obtenir un coefficient d'attention $c_{ij}^{(k,l)}$ entre les sommets v_i et v_j, calculé comme suit :

$$c_{ij}^{(k,l)} = \text{LeakyReLU}(\mathbf{a}[W^{(k)}h_i^{(l)} \| W^{(k)}h_j^{(l)}]), \tag{2.66}$$

avec $\|$ la fonction de concaténation, $W^{(k)} \in \mathbb{R}^{F' \times F'}$ une matrice dont les coefficients seront appris, et $\mathbf{a} \in \mathbb{R}^{2F'}$ un vecteur de poids, lui aussi appris. Ce mécanisme d'attention, nommé *attention additive*, se retrouve également dans [BAHDANAU et al., 2015] (Section 2.1.5.2). D'autres mécanismes peuvent cependant être employés. Cette fonction calcule cependant un score d'attention qui n'est pas normalisé. On introduit alors un coefficient $\alpha_{ij}^{(k,l)}$, prenant en compte les coefficients entre le noeud v_i et tous ceux de son voisinage $\mathcal{N}_i$, tel que :

$$\alpha_{ij}^{(k)} = \text{softmax}(c_{ij}^{(k,l)}) = \frac{e^{c_{ij}^{(k,l)}}}{\sum_{k \in \mathcal{N}_i} e^{c_{ij}^{(k,l)}}} \tag{2.67}$$

La représentation intermédiaire de v_i s'obtient alors en concaténant les sorties des différentes têtes.

$$h_i^{(l+1)} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} W^{(k)} h_j^{(l)}\right) \tag{2.68}$$

Cependant, lors d'une tâche de prédiction, si l'attention multi-tête est également utilisée sur la dernière couche du GAT, il est préférable d'utiliser une fonction moyennant les contributions du voisinage, comme montré à l'Équation 2.69.

$$h_i^{(l+1)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} W^{(k)} h_j^{(l)}\right) \tag{2.69}$$

Les mécanismes présentés dans GAT présentent de nombreux avantages. D'un point de vu computationnel premièrement, car les mécanismes d'attention peuvent facilement être parallélisés. Ensuite, les performances du modèle sur diverses tâches de prédiction de liens ou de clustering sont tout à fait compétitive avec l'état de l'art actuel.

2.2.3.6 Conclusion

Nous avons au cours de cette section présenté divers modèles dédiés à apprendre des représentations des sommets de graphes. Ces modèles recouvrent un large panel de méthodes, allant de la factorisation de matrices à l'apprentissage profond. Certaines approches que nous avons évoquées proposaient de prendre en compte, en plus de la structure du réseau, des attributs des sommets afin d'améliorer la qualité des plongements appris. Nous discuterons dans la prochaine section d'autres méthodes plus particulièrement adaptées à un type d'attribut : le texte.

2.3 Plongement de documents en réseaux

A la croisée de l'apprentissage de représentation de documents et de graphes se trouve le plongement de réseaux de documents, qui permet à la fois de tirer parti de l'information textuelle des documents et de la topologie du réseau. Les corpus de documents structurés de la sorte sont légion, tant les liens entre les contenus sont à la base du web. On la retrouve également fréquemment dans des domaines plus spécifiques, comme la littérature scientifique, où l'on peut construire à partir des nombreuses références d'un article un graphe de citation, ou encore les bases de connaissances comme Wikipedia pour le grand public, ou les bases industrielles comme celles développées par MeetSYS. Au cours de cette section, nous nous attacherons à décrire certaines des principales méthodes d'apprentissage de représentations

de documents en réseau, que nous avons divisé en trois catégories : les modèles thématiques génératifs, dont le principal représentant reste sans doute Relational Topic Model [Chang et al., 2009], et sur lequel s'appuient certains travaux proposés dans ce manuscrit ; les méthodes de plongement de graphes attribués s'appuyant plus spécifiquement sur l'information textuelle ; et enfin les modèles reposant sur des méthodes d'apprentissage profond.

2.3.1 Modélisation thématique génératifs de réseaux de documents

2.3.1.1 Relational Topic Model

Relational Topic Model RTM, [Chang et al., 2009] est une méthode de modélisation de thématiques qui étend LDA (discuté à la Section 2.1.2) afin de prendre en compte les liens entre les documents dans un corpus en réseau. Pour cela les auteurs ajoutent à LDA une fonction de lien permettant de rapprocher les mélanges des K thématiques latentes des documents liés, favorisant le regroupement des contenus similaires.

Plus formellement, RTM reprend le même fonctionnement que LDA. On considère un réseau de documents $\mathcal{G} = \{\mathcal{D}, \mathcal{E}\}$, avec $\mathcal{D}$ un ensemble de D documents, et $\mathcal{E}$ l'ensemble de E liens observés entres eux. On notera $e_{i,j}$ le lien observé entre les documents d_i et d_j de $\mathcal{D}$. On considère également un vocabulaire $\mathcal{V}$ de V mots. Un document d_i est assimilé à un sac de N_i mots $\{w_{i,1}, \cdots, w_{i,N_i}\}$, dont l'ordre n'importe pas. Les paramètres de RTM sont $\beta_{1:K} \in \mathbb{R}^{K \times V}$, une matrice où chaque colonne représente la distribution d'appartenance de chaque mot du vocabulaire aux K thématiques, $\alpha \in \mathbb{R}^K$ le paramètre contrôlant la distribution des thématiques des documents, ainsi que $\eta \in \mathbb{R}^K$ et $\nu \in \mathbb{R}$ les paramètres de la fonction de lien.

Le modèle génératif de RTM se décompose en deux parties : une première (en rouge), identique à LDA, vise à générer les mots qui constituent le document. Pour chaque document d_i, on tire d'abord une proportion de thématiques θ_i, puis pour chaque mot $w_{i,n}$ de ce document, on assigne une thématiques $z_{i,n}$ selon une distribution de multinomiale de paramètre θ_i avant de générer le mot $w_{i,n}$ selon une loi multinomiale de paramètre $\beta_{\cdot,z_{i,n}}$.

La seconde permettant de générer les liens entres ces documents. L'Algorithme 2 résume le fonctionnement de RTM, qui est également illustré à la Figure 2.25.

La fonction de lien, notée $\psi(\cdot)$, permet de prendre en compte l'aspect réseau du corpus. Elle modélise un indicateur binaire $y_{i,j}$ portant sur l'existence d'un lien entre deux documents d_i et d_j. Cette fonction va dépendre de deux paramètres, η et ν, ainsi que de deux ensembles de vecteurs, $\mathbf{z}_i$ et $\mathbf{z}_j$, tel que $\mathbf{z}_i = \{z_{i,1}, \cdots, z_{i,N_i}\}$, autrement dit l'ensemble des vecteurs *1 parmi K* des assignations thématiques de chaque mot de d_i. On pose également $\overline{\mathbf{z}}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} z_{i,n}$, avec $\overline{\mathbf{z}}_i \in \mathbb{R}^K$, un vecteur représentant la proportion de chacune des thématiques dans le document. Les auteurs proposent deux fonction de liens : l'une basée sur la fonction

Algorithme 2 : Modèle génératif de RTM. En rouge, la génération des thématiques et des mots de chaque documents, identique à LDA. En bleu, la partie générant les liens.

foreach $d_i \in \mathcal{D}$ **do**
  $\theta_i|\alpha \sim Dir(\alpha)$
  foreach $w_{i,n} \in d_i$ **do**
    $z_{i,n}|\theta_i \sim Multi(\theta_i)$
    $w_{i,n}|z_{i,n} \sim Multi(\beta_{\cdot,z_{i,n}})$

foreach $e_{i,j} \in \mathcal{E}$ **do**
  $y_{i,j}|\mathbf{z}_i, \mathbf{z}_j, \eta, \nu \sim \psi(\mathbf{z}_i, \mathbf{z}_j)$

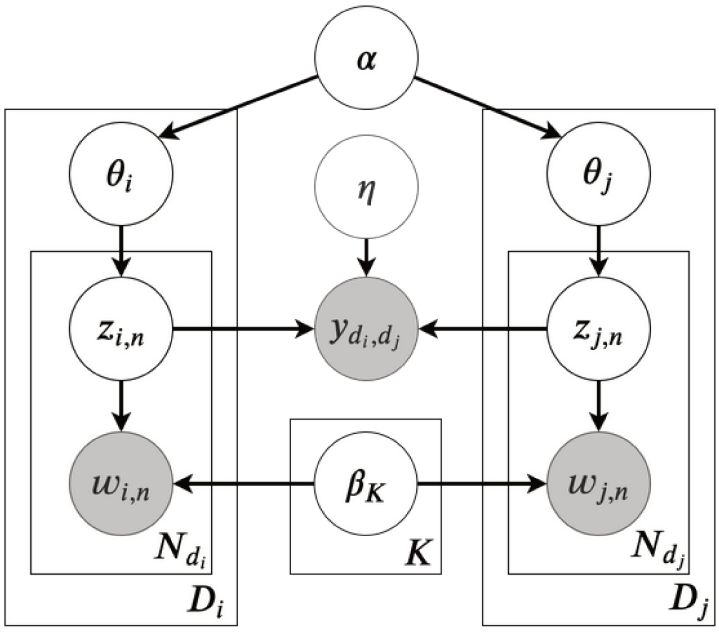


FIGURE 2.25 – Modèle graphique de RTM. En gris les observations.

exponentielle (Équation 2.70), et l'autre utilisant la fonction sigmoïde σ (Équation 2.71), tel que :

$$\psi_e(y_{i,j} = 1|\mathbf{z}_i, \mathbf{z}_j) = \exp(\eta^\top (\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j) + \nu) \tag{2.70}$$

et

$$\psi_\sigma(y_{i,j} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\eta^\top (\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j) + \nu)\,, \tag{2.71}$$

avec $\circ$ le produit de Hadamard (le produit terme à terme des deux vecteurs). $\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j$ aura ici tendance à donner plus d'importance aux thématiques similaires fortement présentes dans les deux documents, et diminuer la contribution des autres. On retrouve Figure 2.26 un graphique illustrant la forme des deux fonctions. Lors de l'estimation, les thématiques

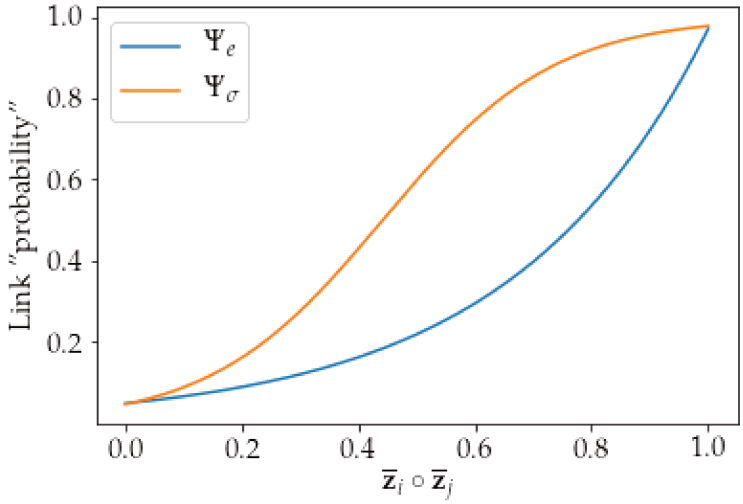


FIGURE 2.26 – Comparaison entre les fonctions exponentielle et sigmoïde.

communes des documents seront alors renforcées, dans une approche similaire à celle employée par BLEI et MCAULIFFE, 2007. C'est une hypothèse centrale du modèle qui est faite ici, en considérant qu'un lien entre deux documents est d'autant plus probable qu'ils partagent les mêmes thématiques latentes.

Pour estimer les paramètres du modèle η, ν, θ et β, les auteurs proposent une méthode reposant sur un algorithme EM variationnel. Nous détaillerons ici la phase d'*Espérance* permettant de déterminer les paramètres variationnels, et la phase de *Maximisation* afin d'estimer les paramètres du modèle.

2.3.1.1.1 Étape E : L'objectif est ici de déterminer les paramètres γ et ϕ de distributions connues, permettant d'approcher au mieux la distribution postérieure du modèle, en minimisant la divergence de Kullback-Leibler. On pose pour cela la probabilité q, tel que :

$$q(\boldsymbol{\Theta}, \mathbf{Z} \mid \boldsymbol{\gamma}, \boldsymbol{\Phi}) = \prod_d \left[q_\theta \left(\theta_d \mid \gamma_d\right) \prod_n q_z \left(z_{d,n} \mid \phi_{d,n}\right) \right]. \tag{2.72}$$

On notera par la suite que $\mathbb{E}_q \left[z_{d,n}\right] = \phi_{d,n}$. Comme pour LDA l'estimation se fait en se basant sur la ELBO, la borne inférieure obtenue par l'application de théorème de Jensen, et dont l'expression est donnée Équation 2.73.

$$\mathcal{L}(\gamma, \phi; \alpha, \beta) = \sum_{e_{ij} \in \mathcal{E}} \underbrace{\mathbb{E}_q[\log p(y_{i,j} | \mathbf{z}_i, \mathbf{z}_j, \eta, \nu)]}_{\mathcal{L}_{(d_i, d_j)}} + \sum_i^D \sum_n^{N_i} \mathbb{E}_q[\log p(z_{i,n} | \theta_i)]$$
$$+ \sum_i^D \sum_n^{N_i} \mathbb{E}_q[\log p(w_{i,n} | \beta_{1:K}, z_{d,n})] + \sum_i^D \mathbb{E}_q[\log p(\theta_i | \alpha)] + H(q). \tag{2.73}$$

La différence avec LDA se situe dans son premier terme, noté par la suite $\mathcal{L}_{(d_i,d_j)}$. En effet, l'introduction de la fonction de lien crée une dépendance entre les documents et modifie l'estimation de certains paramètres. Elle demande donc à prendre en compte les différentes fonctions de lien considérées. Afin d'alléger la notation, on pose $\overline{\pi}_{(d_i,d_j)} = \overline{\phi}_i \circ \overline{\phi}_j$, et également $\overline{\phi}_i = \mathbb{E}_q[\overline{\mathbf{z}}_i] = \frac{1}{N_i} \sum_n \phi_{i,n}$.

On note également que RTM ne prend en compte que les liens observés, et ce pour deux raisons. Premièrement, considérer l'absence de liens, $y_{ij} = 0$, n'est pas une hypothèse de modélisation pertinente dans de nombreux cas réels. Par exemple, dans le cadre d'un réseau social, deux personnes peuvent avoir des amis communs et des intérêts similaires sans pour autant être au courant de la présence de l'autre sur le réseau. Ensuite, l'avantage d'un point de vu computationnel est important, car il réduit drastiquement le nombre de paires de documents à considérer. Un corpus comme Cora [McCallum et al., 2000], qui comporte 2708 documents en réseau pour 5429 liens, requerrait 675 fois plus de calculs si l'on souhaitait prendre en compte les liens non observés. Plus généralement, les réseaux réels sont ont tendance à être très creux, et ne prendre en considération que les liens observés permet de surmonter ce problème de combinatoire.

L'espérance en q de $\mathcal{L}_{(d_i,d_j)}$ dépend de la fonction de lien choisie. Les fonctions de lien présentées ne dépendant que de $(\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j)$, il est possible, au moyen d'une approximation du premier-ordre [Braun et al., 2010], de réécrire leurs espérance tel que :

$$\mathcal{L}_{(d_i,d_j)} = \mathbb{E}_q[\psi_\bullet(\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j)] \approx \log \psi_\bullet(\mathbb{E}_q[\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j]) = \log \psi_\bullet(\overline{\pi}_{(d_i,d_j)}). \tag{2.74}$$

L'Équation 2.75 donne la forme de cette espérance pour la fonction de lien exponentielle. $\mathbb{E}_q$ pour la fonction sigmoïde est elle donnée Équation 2.76.

$$\mathbb{E}_q[\log \psi_e(\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j)] = \eta^\top \overline{\pi}_{(d_i,d_j)} + \nu. \tag{2.75}$$

$$\mathbb{E}_q[\log \psi_\sigma(\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j)] \approx \log \sigma(\eta^\top \overline{\pi}_{(d_i,d_j)} + \nu). \tag{2.76}$$

En dérivant $\mathcal{L}$, on peut alors déterminer la forme de la règle de mise à jour de $\phi_{i,n}$, donnée à l'Équation 2.77. On a toujours $\mathbb{E}_q[\log \theta_i | \gamma_i] = \Psi(\gamma_i) + \Psi(\sum_{k=1}^K \gamma_{i,k})$, puisque θ_i n'intervient pas dans la fonction de lien.

$$\phi_{i,n} \propto \exp\left(\sum_{d_i \neq d_j} \nabla_{\phi_{i,n}} \mathcal{L}_{d_i,d_j} + \mathbb{E}_q[\log \theta_i | \gamma_i] + \log \beta_{\cdot,w_{i,n}}\right) \tag{2.77}$$

L'information sur les liens, $\nabla_{\phi_{i,n}} \mathcal{L}_{d_i,d_j}$, peut être réécrite en fonction de $\overline{\pi}_{(d_i,d_j)}$, permettant de plus facilement différentier les fonctions de liens. On obtient alors :

$$\nabla_{\phi_{i,n}} \mathcal{L}_{d_i,d_j} = (\nabla_{\overline{\pi}_{(d_i,d_j)}} \mathcal{L}_{d_i,d_j}) \circ \frac{\overline{\phi}_j}{N_i} \tag{2.78}$$

On remarque dans l'équation ci-dessus que l'assignation de la thématique latente du mot $w_{i,n}$ du document d_i est « poussé » en vers les thématiques latentes des documents auxquels d_i est lié. Les gradients en $\overline{\pi}_{(d_i,d_j)}$ relatifs aux fonctions de liens sont données aux Équations 2.79 et 2.80, respectivement pour la fonction exponentielle et sigmoïde. L'estimation de $\phi_{i,n}$ se fait alors en injectant l'une ou l'autre dans l'Équation 2.78, afin de calculer la règle de mise à jour donnée Équation 2.77.

$$\nabla_{\overline{\pi}_{(d_i,d_j)}} \mathcal{L}^e_{d_i,d_j} = \eta \tag{2.79}$$

$$\nabla_{\overline{\pi}_{(d_i,d_j)}} \mathcal{L}^\sigma_{d_i,d_j} \approx (1 - \sigma(\eta^\top \overline{\pi}_{(d_i,d_j)} + \mu))\eta \tag{2.80}$$

La mise à jour du paramètre variationnel γ_i, représentant le mélange de thématiques latentes du document d_i, est elle identique à celle formulée dans LDA. En effet le paramètre θ_i n'intervient pas l'expression de la fonction de lien. Ainsi :

$$\gamma_i \leftarrow \alpha + \sum_n^{N_i} \phi_{i,n} \tag{2.81}$$

2.3.1.1.2 Étape M : Une fois les valeurs courantes des paramètres variationnels obtenues, on estime les paramètres du modèle en fonction des paramètres variationnels obtenus à l'étape précédente. La valeur de β est obtenue en utilisant la méthode des multiplicateurs de Lagrange, et ne varie pas de LDA, tel que :

$$\beta i, j = \sum_{d=1}^{D} \sum_{n=1}^{N} \phi_{d,n,i} w^j_{d,n} \tag{2.82}$$

L'estimation des paramètres η et ν va elle dépendre du choix de la fonction de lien. Comme précisé précédemment, RTM ne prend en compte que les liens observés. Il est donc nécessaire d'introduire une régularisation afin de compenser l'absence d'exemples négatifs. Si l'une des solutions envisageables consisterait à introduire une pénalisation ℓ_2, les auteurs proposent

eux d'introduire un nombre ρ d'exemples artificiels, construits à partir de l'hyperparamètre α, tel que $\overline{\pi}_\alpha = \frac{\alpha}{\mathbf{1}^\top \alpha} \circ \frac{\alpha}{\mathbf{1}^\top \alpha}$.

Les valeurs des paramètres η et ν pour la fonction sigmoïde sont approchées par descente de gradient.

$$\nabla_\eta \mathcal{L}^\sigma \approx \sum_{e_{i,j} \in \mathcal{E}} (1 - \sigma(\eta^\top \overline{\pi}_{(d_i,d_j)} + \nu))\overline{\pi}_{(d_i,d_j)} - \rho\sigma(\eta^\top \overline{\pi}_\alpha + \nu)\overline{\pi}_\alpha \tag{2.83}$$

La régularisation pour la fonction de lien exponentielle est faite différemment, afin d'obtenir une règle de mise à jour explicite. On obtient alors :

$$\eta \leftarrow \log(\overline{\mathbf{\Pi}}) - \log(\overline{\mathbf{\Pi}} + \rho\overline{\pi}_\alpha) - \mathbf{1}\nu, \tag{2.84}$$

et

$$\nu \leftarrow \log(E - \mathbf{1}^\top \overline{\mathbf{\Pi}}) - \log(\rho((1 - \mathbf{1}^\top \overline{\pi}_\alpha) + E - \mathbf{1}^\top \overline{\mathbf{\Pi}}), \tag{2.85}$$

avec $\mathbf{1}$ un vecteur de dimension K dont tous les coefficients valent 1, E le nombre de liens observés et $\overline{\mathbf{\Pi}} = \sum_{e_{i,j}} \overline{\pi}_{(d_i,d_j)}$.

2.3.1.2 Generalized RTM

N. Chen et al., 2013 proposent avec *Generalized RTM* (gRTM) d'apporter deux modifications majeures aux travaux précédents. Premièrement, et comme discuté lors de la présentations de RTM, la forme de la fonction de lien, $\eta^\top(\overline{\mathbf{z}}_i \circ \overline{\mathbf{z}}_j)$, relève d'une hypothèse forte quant à la modélisation des liens. En effet, seules sont prises en compte les interactions entre thématiques identiques. Or notre intuition nous laisse à penser que la présence de deux thématiques différentes est tout à fait capable de justifier un lien. Les auteurs reformulent alors la fonction de lien, en introduisant le paramètre $Q \in \mathbb{R}^{K \times K}$, tel que :

$$p(y_{i,j}) = 1|\mathbf{z_i}, \mathbf{z_j}, Q) = \sigma(\overline{\mathbf{z_i}}^\top Q \overline{\mathbf{z_j}}) \tag{2.86}$$

Cette forme permet de prendre en compte toutes les interactions possibles entre les thématiques latentes et permet également la prise en compte de l'orientation du graphe si tel est le cas. Les expériences menées par les auteurs montrent que la répartition des thématiques est conforme à notre intuition : les interactions entre thématiques identiques sont positives (sur la diagonale de la matrice Q), certaines thématiques interagissent également positivement avec d'autres, et la majorité des thématiques différentes interagissent négativement entre elles, autrement dit elles n'impliquent pas l'existence d'un lien.

Le second ajout est une reformulation de la procédure d'estimation des paramètres du modèle. Là où les travaux originaux se basaient sur un algorithme EM variationnel, les auteurs présentent une méthode reposant sur le *Collaped Gibbs Sampling*, méthode ayant déjà

été appliquée avec succès à LDA [Griffiths et al., 2004]. A cela se rajoute également une procédure d'augmentation des données, afin de faciliter l'intégration de la fonction de lien à la procédure d'échantillonnage. Cette augmentation est faite par l'introduction d'une variable auxiliaire de type Pólya-Gamma [Polson et al., 2013]. Bien que donnant des résultats supérieurs à la version variationnelle sur des tâches de prédiction de liens, la procédure d'estimation des paramètres demeure très coûteuse en calculs, et devient difficilement exploitable à mesure que la taille du réseau augmente.

2.3.1.3 Probabilistic Latent Document Network Embedding

Probabilistic Latent Document Network Embedding PLANE, [T. M. Le et al., 2014] est une méthode permettant à la fois d'apprendre des plongements de documents et des thématiques latentes au sein du même espace de représentation. En plus des notations introduite précédemment, on notera ρ la dimension de l'espace de représentation, et K le nombre de thématiques. L'objectif de PLANE est d'apprendre conjointement une représentation de rang faible $x_i \in \mathbb{R}^\rho$ des documents d_i, une représentation $\phi_k \in \mathbb{R}^\rho$ des thématiques et une distribution des mots sur les K thématiques considérées.

Algorithme 3 : Modèle génératif de PLANE

foreach $k \in \{1, \cdots, K\}$ **do**
  $\beta_k \sim \text{Dirichlet}(\lambda)$
  $\phi_k \sim \mathcal{N}(0, \varphi^{-1} I)$

foreach $d_i \in \mathcal{D}$ **do**
  $x_i \sim \mathcal{N}(0, \gamma^{-1} I)$
  foreach $w_{i,n} \in \{1, \cdots, N_i\}$ **do**
    $z_{i,n} \sim \text{Multi}(\{p(k|x_i, \mathbf{\Phi}\}_{k=1}^K)$
    $w_{i,n} \sim \text{Multi}(\beta_{z_{i,n}})$

foreach $e_{ij} \in \mathcal{E}$ **do**
  $y_{ij} \sim \text{Bernoulli}(p(y_{ij} = 1|x_i, x_j, \eta))$

L'Algorithme 3 décrit le modèle génératif de PLANE, et peut être découper en trois parties. La première, surlignée en rouge, correspond à la la génération des paramètres du modèles. Pour chaque thématique k, on tire la probabilité β_k de chaque mot d'y appartenir selon une distribution de Dirichlet d'hyperparamètre λ, et les coordonnées ϕ_k de cette thématique selon une loi normale, centrée en 0 et dont la variance est contrôlée par l'hyperparamètre φ.

La seconde, en bleu, permet de générer d'abord le plongement x_i de chaque document d_i selon une loi normale, puis pour chaque mot du document, de tirer $z_{i,n}$ la thématique

correspondante. La probabilité que le n$^{\text{ième}}$ mot de d_i soit associé à la thématique k est calculée comme suit :

$$p(k|x_i, \mathbf{\Phi}) = \frac{exp(-\frac{1}{2}\|x_i - \phi_k\|^2)}{\sum_{k'=1}^{K} exp(-\frac{1}{2}\|x_i - \phi_{k'}\|^2)}, \tag{2.87}$$

avec $\mathbf{\Phi} = \{\phi_k\}_{k=1}^{K}$ l'ensemble des coordonnées des K thématiques. Concrètement, plus la représentation x_i d'un document sera proche de la représentation ϕ_k d'une thématique, plus les mots qui le composent auront de chance d'être affectés à cette thématiques. Ainsi, deux documents proches dans l'espace des représentations se verront partager les mêmes thématiques. On génère enfin les mots $w_{i,n}$ selon une loi multinomiale de paramètre $\beta_{z_{i,n}}$.

La dernière partie enfin, en vert, permet de modéliser les liens entre les documents, en fonction des plongements générés à l'étape précédente. Comme pour RTM, on tire un indicateur binaire y_{ij} symbolisant l'existence ou non d'un lien. Un lien entre deux documents sera d'autant plus probable que leurs plongements dans l'espace de représentation seront proches l'un de l'autre, ce qui se traduit par l'équation suivante :

$$p(y_{ij} = 1|x_i, x_j, \eta) = exp(-\eta\|x_i - x_j\|^2), \tag{2.88}$$

où η est un paramètre du modèle à apprendre. Cette fonction permet d'intégrer l'information sur la structure du réseau au modèle, et ainsi de rapprocher les représentations des documents liés, tout comme les thématiques qui leurs sont associées. L'estimation des paramètres β, x_i, ϕ_z et η est faite au moyen d'un algorithme EM variationnel, d'une manière similaire à l'estimation des paramètres de RTM.

2.3.1.4 Latent topic models for hypertext

Gruber, Rosen-Zvi et al., 2008 propose avec *Latent Topic Models for Hypertext* (LTMH) une méthode visant à modéliser les liens hypertextes allant d'un mot (ou un groupe de mots) vers un document, afin d'en générer des nouveaux. Le mécanisme employé diffère cependant des méthodes précédemment citées.

Les variables observées sont ici les mots $w_{i,n}$, ainsi que les liens $L^{dj}_{w_{i,n}}$, partant de $w_{i,n}$ et poitant vers d_j. Le modèle génératif de LTMH, illustré à l'Algorithme 4, est composé de deux parties : une partie permettant la génération des documents (en rouge), strictement équivalente à LDA, et une partie générant les liens (en bleu). Le fonctionnement de LDA ayant été traité dans la première section de ce chapitre, nous nous attarderons ici sur le processus de génération des liens.

LTMH cherche à produire des liens hypertexte provenant des mots $w_{i,n}$ d'un document d_i pointant vers un document d_j. On notera ce lien L_n, et celui-ci peut prendre valeurs dans $\{1, \cdots, D, \emptyset\}$, avec $\emptyset$ représentant un lien vers aucun document. Dans un premier temps, on considère le paramètre $\lambda \in \mathbb{R}^{D+1}$, représentant la probabilité de considérer un lien vers

Algorithme 4 : Modèle génératif de LTMH

foreach $d_i \in \mathcal{D}$ **do**
  $\theta_i|\alpha \sim Dir(\alpha)$
  foreach $w_{i,n} \in d_i$ **do**
    $z_{i,n}|\theta_i \sim Multi(\theta_i)$
    $w_{i,n}|z_{i,n} \sim Multi(\beta_{z_{i,n}})$

$\lambda \sim Dir(\gamma)$
foreach $d_i \in \mathcal{D}$ **do**
  foreach $w_{i,n} \in d_i$ **do**
    $\tau_n \sim Multi(\lambda)$
    if $\tau_n \neq \emptyset$
      $z^L \sim Multi(\theta_{\tau_n})$
    if $z^L = z_{i,n}$
      $L_i = \tau_i$

l'un des D documents, ou bien aucun lien. Ce paramètre est tiré selon une loi de Dirichlet d'hyperparamètre $\gamma \in \mathbb{R}^{D+1}$. Dans la mesure où l'on sait que la majorité des mots d'un document ne portent pas d'hyperliens, l'a priori γ doit être fortement asymétrique de façon à privilégier les liens vers aucun document, tel que $\gamma_i \ll \gamma_\emptyset$ pour tout document d_i.

Ensuite, pour chaque document d_i de $\mathcal{D}$ puis pour chacun des mots $w_{i,n}$ qui le composent, on tire au moyen d'un loi multinomiale de paramètre λ la variable aléatoire τ_n, représentant la cible potentielle du lien hypertexte partant du mot. Si $\tau_n = \emptyset$, alors on considère que $w_{i,n}$ n'est lié à aucun document. Dans le cas contraire, où $\tau_n = d_j$, on tire une seconde affectation de thématiques à $w_{i,n}$, notée $z^L_{i,n}$, selon une loi multinomiale de paramètre θ_j. Dans le cas où $z_{i,n} = z^L_{i,n}$, on considérera alors un hyperlien entre le mot $w_{i,n}$ et le document d_j. Pour qu'un hyperlien existe entre un mot $w_{i,n}$ et d_j, il faut donc que la thématiques affectée à ce mot soit la même, qu'elle soit générée selon le mélange de thématique de son document d'origine θ_i ou de celle du document lié ; θ_j. Ce processus peut être répété plusieurs fois dans l'optique d'améliorer les performances du modèle [Gruber, Rosen-Zvi et al., p. d.] L'estimation des paramètres est faite par maximum a posteriori au moyen d'un algorithme EM.

2.3.2 Méthodes de plongement de graphes attribués adaptées au texte

2.3.2.1 Text Attributed DeepWalk

C. Yang et al., 2015 proposent avec Text Attributed DeepWalk (TADW) une extension de DeepWalk (*cf.* Section 2.2.2.1) afin de prendre également en compte les attributs textuels

des sommets. TADW, à l'instar de NetMF, interprète DeepWalk comme une factorisation d'une matrice M, mais propose d'incorporer également le contenu textuel des sommets dans l'apprentissage de leurs représentations. Cette matrice est construite à partir de la matrice d'adjacence A telle que $M = \frac{A+A^2}{2}$. TADW prend donc en compte les voisinages d'ordre un et deux de chaque documents du réseau. Le contenu textuel prend lui la forme d'une matrice $X \in \mathbb{R}^{\rho_t \times D}$, avec D le nombre de documents du corpus en réseau. Chaque colonne $x_{.,i}$ de X correspond à la représentation du document d_i obtenue par LSA (Section 2.1.1), en dimension ρ_t. Plus précisément, TADW correspond à une décomposition de rang faible de M en deux matrices U et H. La matrice U représente l'aspect « réseau » et HX l'aspect « texte ».

En employant la méthode de complétion inductive de matrice [Natarajan et al., 2014] permettant une tri-factorisation de M, TADW estime U et H, compte tenu de X, de façon à résoudre un problème des moindre carrés, tout en imposant une pénalisation sur la norme de Frobenius aux deux premières matrices. Le problème d'optimisation est alors le suivant :

$$\min_{U,H} \|M - U^\top HX\|_F^2 + \frac{\lambda}{2}\left(\|U\|_F^2 + \|H\|_F^2\right), \tag{2.89}$$

avec $\lambda \in \mathbb{R}$ un hyperparamètre. U et H sont estimées alternativement afin de minimiser l'Équation 2.89. La matrice finale des plongements de documents est obtenue en concaténant les matrices U et HX.

2.3.2.2 Gaussian Embedding of Linked Documents

Gourru et al., 2020 proposent avec Gaussian Embedding of Linked Documents (GELD) une méthode de plongement de documents en réseaux, se basant sur des plongements de mots pré-appris, qui ne représente plus un document d_i comme un point dans un espace de représentation, mais comme une distribution gaussienne de moyenne μ_i et de variance σ_i^2. Cette variance permet de modéliser l'incertitude liée à la représentation d'un document. Celle-ci aura tendance à augmenter si le document est lié à des documents qui lui sont éloignés et qu'ils différent de part leur contenu textuel. De plus GELD permet de représenter dans le même espace, de dimension $\mathbb{R}^\rho$, les mots et les documents. Il est ainsi possible de calculer des similarités entre des termes et des documents, afin par exemple d'expliquer un ensemble de documents par un groupe de mots qui leur sont les plus proches.

GELD cherche à apprendre une représentation du document d_i sous la forme d'une distribution gaussienne de moyenne $\mu_i \in \mathbb{R}^\rho$ et de variance diagonale $\sigma_i^2 I$ avec $\sigma^2 \in \mathbb{R}^\rho$. On note $A \in \mathbb{N}^{D\times D}$ la matrice d'adjacence, dont le coefficient a_{ij} représente le nombre de fois que le document d_i cite le document d_j. On considère également un vocabulaire fixe comptant V mots. La matrice $X \in \mathbb{N}^{D\times V}$ correspond à la matrice de comptage de $\mathcal{D}$, où chaque ligne c_i est la représentation sac de mots de d_i, et dont le coefficient c_{ik} est le nombre d'occurrences du $k^{\text{ième}}$ mot du vocabulaire dans le contenu textuel de d_i. On note enfin trois ensembles :

— $\mathcal{S}_i^t = \{u_1^i, \cdots, u_N^i\}$, l'ensemble des N plongements de mots de d_i, avec $u_i^n \in \mathbb{R}^\rho$ le plongement du $n^{\text{ième}}$ mot de d_i.
— $\mathcal{S}_i^s = \{\mu_1^i, \cdots, \mu_M^i\}$, l'ensemble des M moyennes des documents liés à d_i.
— $\mathcal{S}_i = \mathcal{S}_i^t \cup \mathcal{S}_i^s$, l'union des deux ensembles précédents, tel que $\mathcal{S}_i = \{u_1^i, \cdots, u_N^i, \mu_1^i, \cdots, \mu_M^i\}$.

On considère les vecteurs $\mathcal{S}_{i,j}$ composant $\mathcal{S}_i$ comme indépendants, et ceux-ci sont tirés selon la même distribution gaussienne, de la forme :

$$\mathcal{S}_{i,j} \sim \mathcal{N}(\mu_i, \sigma_i^2 I). \tag{2.90}$$

En notant $\mathcal{S} = \{\mathcal{S}_i\}_{i=1}^D$, $\mu = \{\mu_i\}_{i=1}^D$ et $\sigma^2 = \{\sigma_i^2\}_{i=1}^D$ on peut écrire log-vraisemblance des données telle que :

$$\mathcal{L}(\mathcal{D}; \mu, \sigma^2) = \underbrace{\sum_{i=1}^{D}\sum_{j=1}^{D} a_{ij}\mathcal{N}(\mu_j; \mu_i, \sigma_i^2)}_{\mathcal{L}_s} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{V} c_{ik}\mathcal{N}(u_k; \mu_i, \sigma_i^2)}_{\mathcal{L}_t}. \tag{2.91}$$

$\mathcal{L}_s$ et $\mathcal{L}_t$ représentent respectivement la partie de la log-vraisemblances relatives à la structure du réseau et aux contenus textuels. Cependant le nombre de citations partant d'un document et le nombre de mots qui le composent sont rarement équivalents. Dans de nombreux réseaux réels le nombre de mots dépasse nettement le nombre de citations. Afin de palier ce déséquilibre, les auteurs introduisent donc $\eta \in [0, 1]$, un paramètre permettant de définir l'importance accordée à chacun des deux aspects du réseau de documents, de façon à proposer une version modifiée de l'Équation 2.91 :

$$\tilde{\mathcal{L}} = (1 - \eta)\mathcal{L}_t + \eta\mathcal{L}_s. \tag{2.92}$$

$\tilde{\mathcal{L}}$ est maximisée alternativement en chaque étape k les paramètres $\mu_i^{(k)}$ et $\sigma^{2(k)}$, jusqu'à convergence. Les valeurs initiales des paramètres sont calculées comme les moyennes et variances empiriques des plongements de mots.

2.3.2.3 GVNR-t

GVNR-t est une extension de GVNR permettant d'apprendre des représentations de (*cf.* Section 2.2.2.2) de documents en réseau. L'information textuelle est représentée sous la forme d'une matrice de fréquence des mots $X \in \mathbb{N}^{D \times V}$, avec D le nombre de documents du corpus en réseau, et V le nombre de mots du vocabulaire, où x_i, la i^{me} ligne de X, correspond à la représentation sac de mots du document d_i. À cela s'ajoute la matrice $W \in \mathbb{R}^{V \times \rho}$, donc chaque ligne est la représentation d'un mot, en dimension ρ.

De la même façon que dans les travaux d'Arora et al., 2017, un document est considéré dans GVNR-t comme la moyenne des vecteurs des mots qui le composent. Cette moyenne est calculée, pour le document d_i, comme $x_i W$.

Le fonctionnement de GVNR-t est identique à celui de GVNR, avec pour seule différence le fait que la matrice des vecteurs contexte C précédemment apprise est maintenant remplacée par la matrice des moyennes des représentations des mots, tel que la i^{me} ligne de C devient $c_i = x_i W$. En remplacent ce terme dans l'Équation 2.48 on cherche les valeurs des paramètres U, W, b_i^U et b_i^W minimisant la fonction suivante :

$$\underset{U,C,b^u,b^c}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \delta(s_{ij}) \left(u_i \cdot x_j W + b_i^U + b_j^W - \log(1 + s_{ij})\right)^2. \tag{2.93}$$

On peut ainsi considérer que GVNR-t encode d'un côté l'aspect structurel du réseau de document, sous la forme de la matrice U, et de l'autre l'aspect textuel des documents, représenté par la matrice XW. Le plongement complet du document d_i est la concaténation des vecteurs encodant ces deux aspects, autrement dit $u_i \oplus x_i W$.

2.3.2.4 Inductive Document Network Embedding

Inductive Document Network Embedding IDNE, [Brochier et al., 2020] est une méthode visant à apprendre de façon inductive au sein d'un même espace de représentation, de dimension ρ, les mots, les documents et les thématiques issus d'un corpus en réseau. Il s'appuiera pour cela sur un mécanisme d'attention entre les mots et les thématiques.

On considère toujours un réseau de D documents $\mathcal{G} = \{\mathcal{D}, \mathcal{E}\}$, avec $A \in \{0,1\}^{D \times D}$ sa matrice d'adjacence, et $X \in \mathbb{N}^{D \times V}$ la matrice dont chaque ligne x_i est la représentation sac-de-mots (TF) du document d_i, avec V la taille du vocabulaire. On définit enfin l'hyperparamètre $K \in \mathbb{N}$, le nombre de thématiques. IDNE cherche à apprendre les matrices $W \in \mathbb{R}^{V \times \rho}$ et $T \in \mathbb{R}^{K \times \rho}$, respectivement la matrice des plongements des mots du vocabulaire et la matrice des vecteurs thématiques, afin de construire $U \in \mathbb{R}^{D \times \rho}$, la matrice des plongements des documents.

Dans un premier temps, pour chaque document d_i, on calcule les scores d'attention $C_i \in \mathbb{R}^{K \times V}$ entre les mots qui le composent et les thématiques, tel que :

$$C_i = ReLU(TW^\top \mathrm{diag}(X_i)). \tag{2.94}$$

La fonction ReLU ($ReLU(x) = max(0, x)$) [Nair et al., 2010] permet à la fois d'assurer la non négativité des éléments. On normalise ensuite en colonnes, afin que chacune des colonnes de C_i somme à 1. Il est alors possible de calculer une représentation $u_i^{(k)}$ de chaque document pour chacune des thématiques k, comme suit :

$$u_i^{(k)} = \frac{C_{i,k} \mathrm{diag}(x_i) W}{|x_i|_1}. \tag{2.95}$$

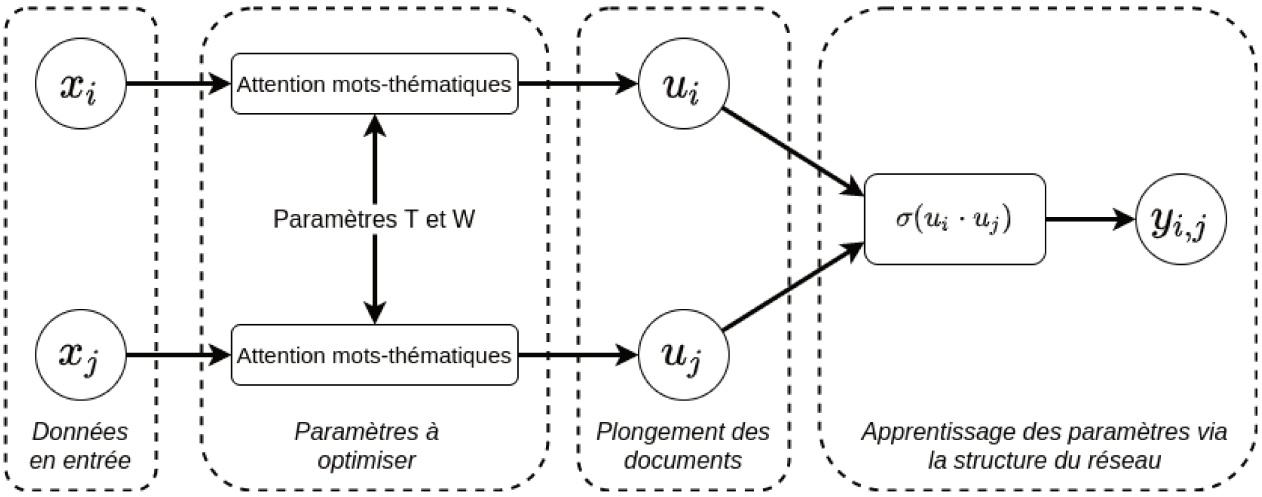


FIGURE 2.27 – Illustration de l'architecture de IDNE. Le premier étage correspond aux données TF des documents d_i et d_j. Le second illustre le calcul des coefficients d'attention, où les paramètres T et W sont partagés pour tous les documents. On construit ensuite les plongements des documents. Le dernier étage correspond à l'apprentissage des paramètres, en cherchant à prédire l'existence d'un lien entre les deux documents.

$C_{i,k}$ désigne la $k^{\text{ième}}$ ligne de la matrice C_i. Ce vecteur va alors pondérer la représentation des mots contenu dans d_i, afin de produire une représentation spécifique du document pour la thématique k. Le dénominateur permet de contenir les représentations de documents dans le même ordre de grandeur, afin de stabiliser l'apprentissage. On remarque une similitude avec l'Equation 2.30, du Transformer, où ici $C_{i,k}$ joue le rôle de requête (Q), et $\text{diag}(x_i)W$ celui à la fois de clef (K) et valeurs (V). La représentation finale de d_i, prenant en compte les contributions de toutes les thématiques, est alors $u_i = \sum_{k=1}^{K} u_i^{(k)}$, autrement dit la somme des représentations thématiques de d_i.

Enfin, l'apprentissage des paramètres va exploiter la structure du réseau, de façon à rapprocher dans l'espace des représentations les documents proches dans le graphe, et éloigner les plus distants. On construit alors une matrice binaire $S \in \{0,1\}^{D \times D}$ encodant les voisinages d'ordre 1 et 2. Un coefficient $s_{ij} = 1$ indiquera ainsi que les documents d_i et d_j sont soit directement liés dans le graphe, soit ont au moins un voisin en commun. Formellement, les coefficients de S sont construits tel que :

$$s_{i,j} = \begin{cases} 1 & \text{si } (A + A^2)_{i,j} > 0 \\ 0 & \text{sinon.} \end{cases} \tag{2.96}$$

On définit la probabilité que deux documents soient liés (représenté par un indicateur binaire y_{ij}) comme $p(y_{ij} = 1 | u_i, u_j; T, W) = \sigma(u_i \cdot u_j)$. L'apprentissage des paramètres est fait en maximisant la vraisemblance de S selon T et W, montrée à l'Équation 2.97. L'optimisation est faite au moyen de l'algorithme de descente de gradient stochastique ADAM [KINGMA et

Ba, 2015]. Lorsque W et T sont appris, il est alors possible de représenter de nouveaux documents (tant que les mots les constituants appartiennent au vocabulaire considéré lors de l'entraînement) et prédire de façon inductive leurs liens avec le reste du corpus.

$$\mathcal{L}(T, W) = \sum_{i=1}^{D} \sum_{j=1}^{D} s_{i,j} \log \sigma(u_i \cdot u_j) + (1 - s_{i,j}) \log \sigma(-u_i \cdot u_j) \tag{2.97}$$

2.3.3 Méthodes basée sur l'apprentissage profond

2.3.3.1 Context-Aware Network Embedding

Context-Aware Network Embedding CANE, [Tu et al., 2017] est un modèle de plongement de réseaux de documents visant à apprendre des représentations qui encodent à la fois la structure du graphe et le contenu textuel des sommets. Une des particularités de CANE est d'apprendre autant de représentations d'un document qu'il a de voisins, afin de mieux capturer les propriétés des contenus textuels susceptibles d'expliquer la présence d'un lien. Dans la description suivante nous emploierons s en exposant afin d'indiquer qu'un objet est lié à la dimension structurelle du réseau de document, et t en exposant s'il est lié à l'aspect textuel.

CANE cherche à apprendre les valeurs des paramètres suivants :

— W, la matrice de plongement des mots,

— Q, la matrice d'attention mutuelle,

— U^s et H^s, les matrices des représentations des sommets.

Ces paramètres sont appris de façon à minimiser simultanément la distance entre les représentations structurelles u_i^s et h_j^s (respectivement des lignes de U^s et H^s) des sommets liés d_i et d_j, la distance entre les représentations textuelles u_i^t et h_j^t des deux documents liés et la distance entre les représentations textuelles (calculée à partir des paramètres W) et structurelles des documents proches dans le réseaux de documents. L'objectif à minimiser est alors de la forme :

$$\mathcal{L} = \sum_{e_{ij} \in \mathcal{E}} \mathcal{L}^s(e_{ij}) + \mathcal{L}^t(e_{ij}). \tag{2.98}$$

$\mathcal{L}^s(e_{ij})$, l'objectif relatif à la structure du réseau de documents pour le lien e_{ij}, vise à mesurer la log-vraisemblance d'un lien en se basant sur les représentations structurelles des sommets d_i et d_j, tel que :

$$\mathcal{L}^s(e_{ij}) = a_{ij} p^{ss}(d_j | d_i), \tag{2.99}$$

avec

$$p^{ss}(v_j | v_i) = \frac{e^{u_i^s h_j^s}}{\sum_{v_k \in \mathcal{V}} e^{u_i^s h_k^s}} \tag{2.100}$$

L'objectif $\mathcal{L}^t(e_{ij})$, cherchant à rapprocher les représentations textuelles des autres représentations est, elle, de la forme :

$$\mathcal{L}^t(e_{ij}) = a_{ij}(\alpha \cdot p^{tt}(d_j|d_i) + \beta \cdot p^{ts}(d_j|d_i) + \gamma \cdot p^{st}(d_j|d_i)). \tag{2.101}$$

Les trois probabilités p^{tt}, p^{ts} et p^{st} sont de la même forme que celle décrite à l'Équation 2.100, en modifiant les représentations u_i^s et h_j^s pour les représentations adéquates.

On considère pour chacun des deux documents liés d_i et d_j leur séquence de mots respective $x_i = (w_1, \cdots, w_N)$ et $x_j = (w_1, \cdots, w_M)$. On obtient alors les séquences de plongements de mots $\mathbf{x_i} = (\mathbf{w_1}, \cdots, \mathbf{w_N})$ et $\mathbf{x_j} = (\mathbf{w_1}, \cdots, \mathbf{w_M})$, au moyen de la matrice de plongement W, qui sera apprise. Ces deux séquences sont alors passées dans une couche de convolution afin d'obtenir deux matrices $S^i \in \mathbb{R}^{\rho \times N}$ et $S^j \in \mathbb{R}^{\rho \times M}$, dont les lignes sont autant de représentations intermédiaires. On construit ensuite une matrice de corrélation $F = \tanh\left(S^{i\top} Q S^j\right)$, donc chaque coefficient f_{kl} est un score de corrélation entre entre les vecteurs s_k^i et s_l^j. Ce score est paramétré par la matrice d'attention mutuelle Q, qui est un paramètre du modèle à estimer. On calcule enfin un vecteur d'attention pour chacune des deux séquences au moyen d'un max-pooling sur F, en ligne (pour v_i) ou en colonne (pour v_j). Le plongement d'un document sera alors donné par le produit scalaire de son vecteur d'attention et de sa matrice S^i. L'optimisation des paramètres est effectuée en employant l'algorithme de descente de gradient stochastique ADAM.

2.3.3.2 Neural Relational Topic Model

Bai et al., 2018 proposent avec Neural Relational Topic Model (NRTM) d'adapter l'architecture de RTM [Chang et al., 2009] en se basant sur des méthodes neurales. RTM est composé de deux parties, l'une modélisant les documents et l'autre modélisant les liens, et il en va de même pour NRTM. La modélisation des documents est faite au moyen d'une architecture nommée par les auteurs *Stacked Variational Auto-Encoder* (SVAE), proposant une adaptation au cadre variationnel des *Stacked Auto-Encoders* [Bengio, Lamblin et al., 2007]. La prédiction des liens s'effectue, elle, en utilisant un perceptron multi-couches (MLP) cherchant à classifier si le vecteur résultant de la concaténation des mélanges de thématiques de deux documents suggère un lien entre eux. La Figure 2.29 illustre l'architecture du modèle.

L'ensemble des N_i mots du document $d_i \in \mathcal{D}$ est noté $w_i \in [w_{i,n}, \cdots, w_{i,N_d}]$, avec $w_{i,n} \in \{0,1\}^V$. Puisque l'ordre des mots n'a pas d'importance dans NRTM, et par soucis de performance afin de ne pas répéter certains calculs, les mots sont agrégés pour chaque document d_i tel que $x_i = \sum_{n=1}^{N_i} w_{i,n}$, de façon à obtenir un vecteur de comptage des mots. L'auto-encodeur prend la forme d'un réseau de neurones à L couches, l'encodeur et le décodeur en comptant chacun $\frac{L}{2}$. Le SVAE sera d'abord entraîné seul, afin d'apprendre des

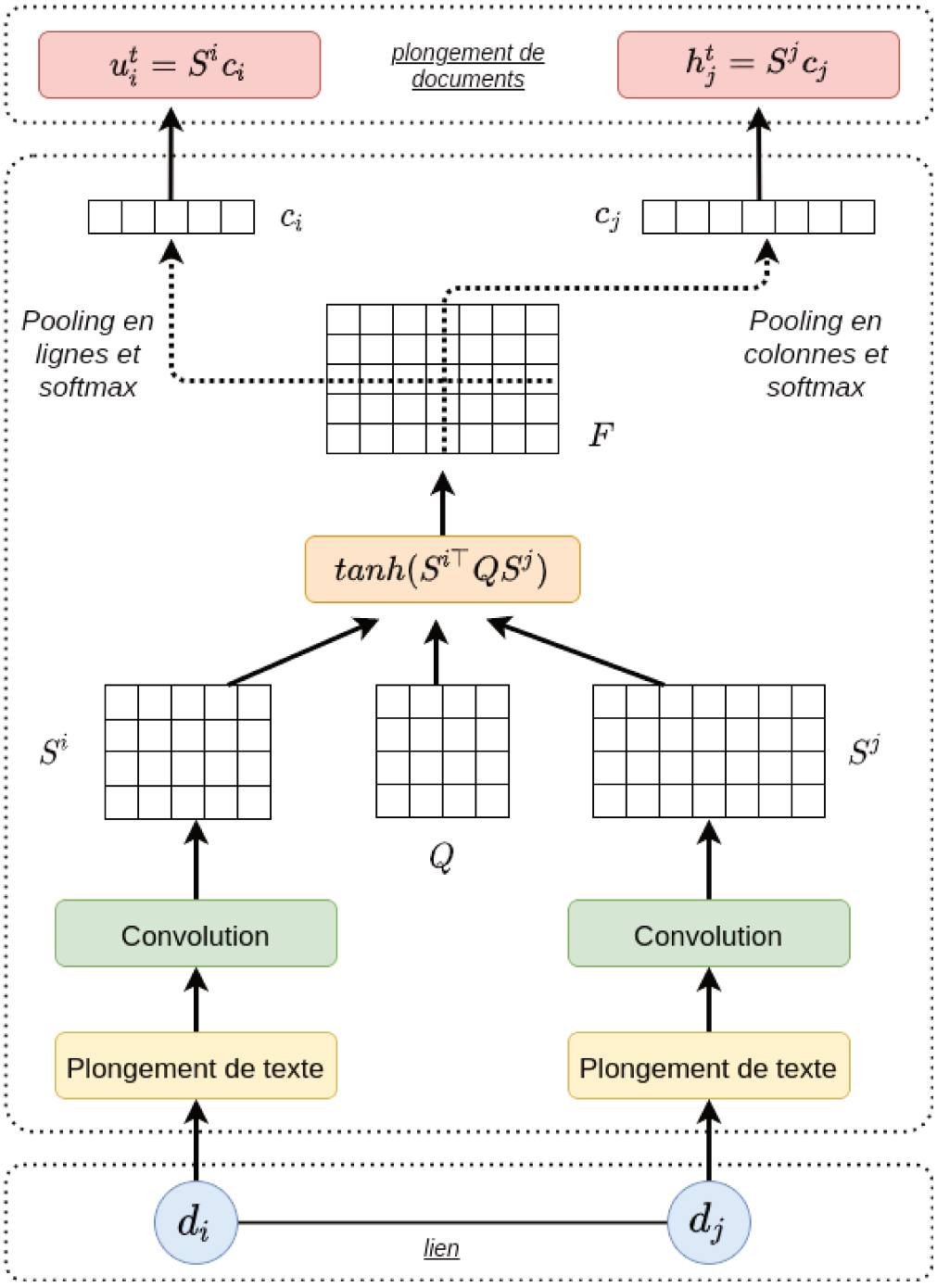


FIGURE 2.28 – Vue d'ensemble de CANE. Pour chaque lien les mots des deux documents sont plongés avant d'être passés dans une couche de convolution afin d'en extraire les caractéristiques locales. La matrice de corrélation F est alors construite à l'aide de la matrice d'attention Q. Les poids d'attentions sont obtenus par un pooling en ligne ou en colonne. La produit de la matrice issue de la convolution et du vecteur d'attention permet enfin d'obtenir une représentation du contenu textuel.

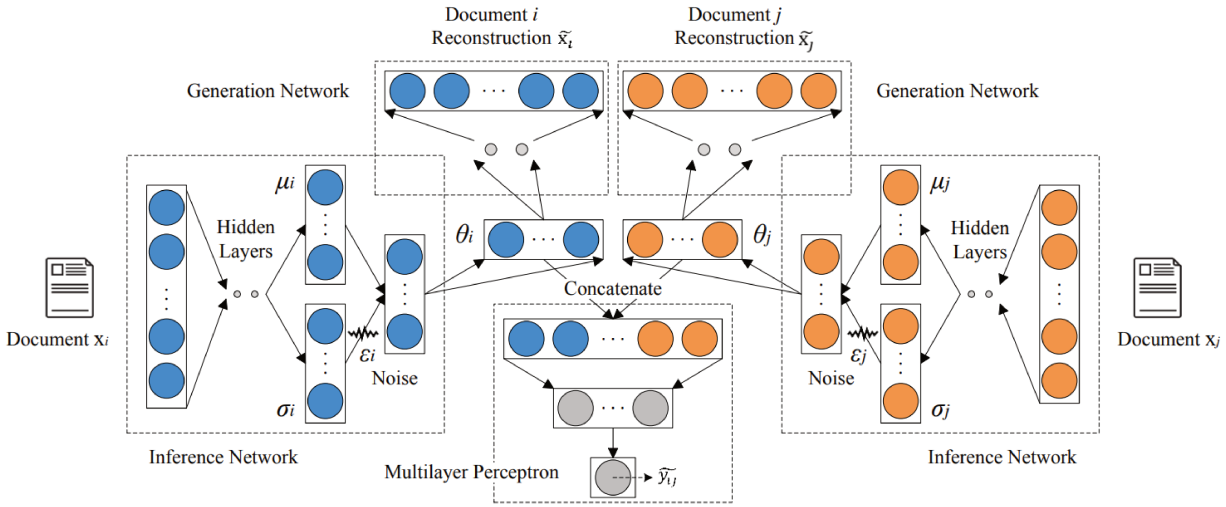


FIGURE 2.29 – Illustration de l'architecture de NRTM, tirée de BAI et al., 2018. Le documents x_i et x_j sont passée en entrée du SVAE afin de générer les mélanges de thématiques θ_i et θ_j. On passe ensuite le résultats de la concaténation de θ_i et θ_j dans un perceptron multi-couches afin d'obtenir y_{ij}, variable indiquant un lien entre les deux documents.

représentations des documents, puis sera affiné lors de l'entraînement du MLP, en profitant de la rétropropagation de son erreur.

L'encodeur va générer pour chaque document d_i, à partir du vecteur de comptage x_i, un vecteur moyenne $\mu_i \in \mathbb{R}^K$ et un vecteur $\sigma_i \in \mathbb{R}^K$, avec K le nombre de thématiques fixé. Pour permettre la rétropropagation du gradient au travers des variables aléatoires, on utilise l'astuce de reparamétrisation [KINGMA et WELLING, 2014], en posant $\theta_i = \mu_i + \epsilon\sigma$, avec $\theta_i \in \mathbb{R}^K$ un vecteur représentant la proportion de thématiques du document d_i, et $\epsilon = \mathcal{N}(0, 1)$ un bruit généré aléatoirement selon une loi normale.

Le décodeur va lui chercher à reconstruire un vecteur $\tilde{x}_i$ à partir du θ_i généré par l'encodeur. On pose pour cela $\beta \in \mathbb{R}^{K \times V}$, une matrice dont les colonnes décrivent la probabilité de chaque mot du vocabulaire d'appartenir à la thématique correspondante, de la même manière que dans LDA ou RTM, de telle façon que $\beta = \prod_{l=\frac{L}{2}}^{L} W^{(l)}$, avec $W^{(l)}$ la matrice des poids de la $l^{\text{ième}}$ couche du réseau de neurones. Contrairement à d'autres méthodes de modélisation de thématiques utilisant des auto-encodeurs [SRIVASTAVA et al., 2017] ; [MIAO et al., 2016], le décodeur de NRTM possède plusieurs couches, permettant une plus grande expressivité dans la génération des thématiques. Les expériences montrent d'ailleurs une cohérence des thématiques [RÖDER et al., 2015] de NRTM, comparativement à ses concurrents. L'apprentissage des paramètres est fait en minimisant l'erreur de reconstruction entre le x_i original et sa version reconstruite $\tilde{x}_i$.

Les liens entre les documents d_i et d_j sont eux modélisés au moyen d'un perceptron multi-couches (à 4 couches entièrement connectées dans les expériences menées par les auteurs).

Celui-ci reçoit en entrée un vecteur $\theta_{ij} \in \mathbb{R}^{2K}$, tel que θ_{ij} soit le résultat de la concaténation des vecteurs θ_i et θ_j. Ce classifieur cherche à prédire $\tilde{y}_{ij} \in \{0, 1\}$, indiquant si les deux documents sont liés (1) et non (0). Afin d'assurer de meilleurs résultats, on ajoute aux liens existants $\mathcal{E}$ un ensemble $\mathcal{E}^-$ d'exemple négatifs. On cherchera alors à maximiser la log-vraisemblance suivante, par descente de gradient :

$$\mathcal{L}_{MLP} = \sum_{e_{ij} \in \mathcal{E}} \log \tilde{y}_{ij} + \sum_{e_{ik} \in \mathcal{E}^-} \log(1 - \tilde{y}_{ik}) \tag{2.102}$$

2.3.3.3 Relational Deep Learning

H. Wang, Shi et al., 2017 proposent un modèle d'apprentissage profond, à L couches, adapté à la modélisation de thématiques de réseaux de documents : Relational Deep Learning (RDL). Ce modèle permet d'apprendre des représentations de K thématiques des documents d'un corpus adaptées à des tâches de prédiction de liens. On notera $X_C \in \mathbb{R}^{D \times V}$ la matrice dont la i$^{\text{ème}}$ ligne contient le vecteur sac-de-mots correspondant au document d_i. X_C et $\mathcal{E}$ sont tout deux les variables observées du modèle. Par la suite, on dénotera par $\lambda_\cdot$ les différents hyperparamètres. Enfin, pour une couche l, on notera $W^{(l)}$ la matrice de poids, $b^{(l)}$ son vecteur de biais, et $X_i^{(l)}$ la représentation intermédiaire du document d_i.

Algorithme 5 : Modèle génératif de RDL

foreach *couche l de RDL* **do**
  foreach *colonne n de $W^{(l)}$* **do**
    $W^{(l)}_{:,n} \sim \mathcal{N}(0, \lambda_w^{-1} I_{(l-1)})$
  $b^{(l)} \sim \mathcal{N}(0, \lambda_w^{-1} I_l)$
  foreach *ligne i de $X^{(l)}$* **do**
    $X_i^{(l)} \sim \mathcal{N}(\sigma(X_i^{(l-1)} W^{(l)} + b^{(l)}), \lambda_s^{-1} I_l)$

foreach *document d_i* **do**
  $\tilde{X}_{C,i} \sim \mathcal{N}(X_i^{(L)}, \lambda_n^{-1} I_V)$

foreach *document d_i* **do**
  $\phi_i \sim \mathcal{N}(X_i^{(\frac{L}{2})^\top}, \lambda_p^{-1} I_K)$

$\eta_i \sim \mathcal{N}(0, \lambda_e^{-1} I_K)$
foreach $e_{ij} \in \mathcal{E}$ **do**
  $y_{ij} | \phi_i, \phi_j \sim \psi(\cdot | \phi_i, \phi_j, \eta)$

RDL repose sur une architecture *Probabilist Stacked Denoising Auto-Encoders* [H. Wang, N. Wang et al., 2015]. A partir d'un vecteur sac de mots $X_{C,i} \in \mathbb{R}^V$, un encodeur produira

à la couche $\frac{L}{2}$ une représentation intermédiaire $X_i^{(\frac{L}{2})} \in \mathbb{R}^K$, puis un décodeur cherchera à reconstruire à la couche L le vecteur sac de mot d'origine, noté $\tilde{X}_{C,i}$. L'Algorithme 5 détaille le modèle génératif de la méthode. Bien que la formulation de RDL puisse sembler dense, on peut la décomposer en quatre parties afin d'en faciliter la lecture :

— Le premier bloc, en rouge, correspond à la génération des paramètres $W^{(l)}$ et $b^{(l)}$, respectivement la matrice de poids et le vecteur de biais de la couche l, selon des distributions normales. La variable latente correspondant à la représentation intermédiaire du document d_i à la couche l, $X_i^{(l)}$, est alors tirée selon une loi normale contrôlée par les paramètres $W^{(l)}$ et $b^{(l)}$.
— La seconde partie, en bleu dans le modèle génératif, va chercher, pour chaque document d_i, à reconstruire le vecteur sac de mots $\tilde{X}_{C,i}$ à partir de la représentation intermédiaire issue de la dernière couche L.
— Le troisième bloc, en jaune, génère la représentation thématique du document d_i, à partir d'une loi normale centrée sur la représentation intermédiaire générée à la couche $\frac{L}{2}$.
— La dernière partie, en vert, va permettre la modélisation des liens. RDL adopte une fonction de lien similaire à la fonction sigmoide de RTM (Équation 2.71), de la forme :

$$\psi(y_{ij} = 1 | \phi_i, \phi_j, \eta) = \sigma(\eta^\top (\phi_i \circ \phi_j)). \tag{2.103}$$

Comme dans les travaux de [Chang et al., 2009], on cherche avec Relational Deep Learning à générer un indicateur binaire y_{ij} indiquant l'existence ou non d'un lien entre les documents d_i et d_j. Cette probabilité sera d'autant plus grande que les thématiques des deux documents sont plus proches. On note également que là encore, seul les liens véritablement observés sont pris en compte.

L'apprentissage des paramètres W, b et η est faite au moyen d'un algorithme EM variationnel adaptée pour pouvoir gérer les multiples transformations non linéaires propres aux réseaux de neurones profonds. La procédure étant complexe, nous n'en décrirons ici que les grandes lignes. Il s'agit de maximiser la log-vraisemblance de $\{X^{(l)}\}$, X_C, $\{W^{(l)}\}$, $\{b^{(l)}\}$, $\{\phi_i\}$, η et $\{e_{ij}\}$ en fonction des paramètres. On y retrouve les différentes erreurs de reconstruction de l'auto-encodeur et la génération des liens, ainsi les termes de régularisation des

paramètres :

$$\begin{aligned}\mathcal{L} = &-\frac{\lambda_p}{2}\sum_{i=1}^{D}\|\phi_i - X_i^{\frac{L}{2}\top}\|_2^2 - \frac{\lambda_n}{2}\sum_{i=1}^{D}\|X_i^{(L)} - X_{C,i}\|_2^2 \\ &-\frac{\lambda_s}{2}\sum_{l=1}^{L}\sum_{i=1}^{D}\|\sigma(X_i^{(l-1)}W^{(l)} + b^{(l)}) - X_i^{(l)}\|_2^2 \\ &+\sum_{e_{ij}\in\mathcal{E}}\log\sigma(\eta^\top(\phi_i \circ \phi_j)) \\ &-\frac{\lambda_w}{2}\sum_{l=1}^{L}(\|W^{(l)}\|_F^2 + \|b^{(l)}\|_F^2) - \frac{\lambda_e}{2}\|\eta\|_2^2 \end{aligned} \tag{2.104}$$

La première étape (*Espérance*) va consister à trouver les paramètres variationnels permettant d'approcher au mieux la variable latente ϕ_i. Puis, lors de l'étape de Maximisation, les paramètres W, b et η sont mis à jour de façon à maximiser l'Equation 2.104, compte tenu de la valeur courante de ϕ_i.

Les expériences menées par les auteurs montrent que RDL permets d'obtenir des résultats supérieurs sur des tâches de prédiction de liens entre documents, comparé à d'autres variantes de RTM ne reposant pas sur un réseau de neurones profond.

2.3.3.4 Adjacent-Encoder-X

C. Zhang et al., 2020 proposent une architecture d'auto-encodeurs, *Adjacent-Encoder-X* (AdjE), visant à apprendre des représentations thématiques de documents en réseau, en s'appuyant de deux façons différentes sur la topologie du réseau. La première consiste en un mécanisme d'attention permettant de prendre en compte les thématiques du voisinage d'un document. La seconde consiste à reconstruire, en parallèle du contenu textuel du voisinage du document, leurs vecteurs d'adjacence. *Adjacent-Encoder-X* prend son nom de la forme de son architecture, montrée Figure 2.30. Ces représentations peuvent être utilisées pour des tâches de classification ou de prédictions de liens.

On considère comme données d'entrée la matrice $X \in \mathbb{N}^{D\times V}$ dont chaque ligne $x_i \in \mathbb{N}^V$ est le vecteur de comptage du contenu textuel du document d_i, et A la matrice d'adjacence du réseau. Dans un premier temps, les auteurs log-normalisent ces vecteurs, tel que $\mathbf{x}_i = \frac{\log(1+x_{i,n})}{max_{m\in\mathcal{V})}\log(1+x_{i,m})}$. Pour un document d_i, on apprend alors en sortie de l'encodeur une représentation latente, de K thématiques, $h_i \in \mathbb{R}^K$ tel que $h_i = tanh(W_1\mathbf{x}_i + W_2 a_i + b)$, avec $W_1 \in \mathbb{R}^{K\times V}$ la matrice de poids du contenu textuel, $W_2 \in \mathbb{R}^{K\times D}$ la matrice de poids liés aux vecteurs d'adjacence, et $b \in \mathbb{R}^K$ un vecteur de biais. À partir de ces représentations intermédiaires, on peut alors calculer pour d_i les coefficients d'attention pour chacun des documents d_j de son voisinage (voisinage noté $\mathcal{N}_i$) de la façon présentée ci-dessous :

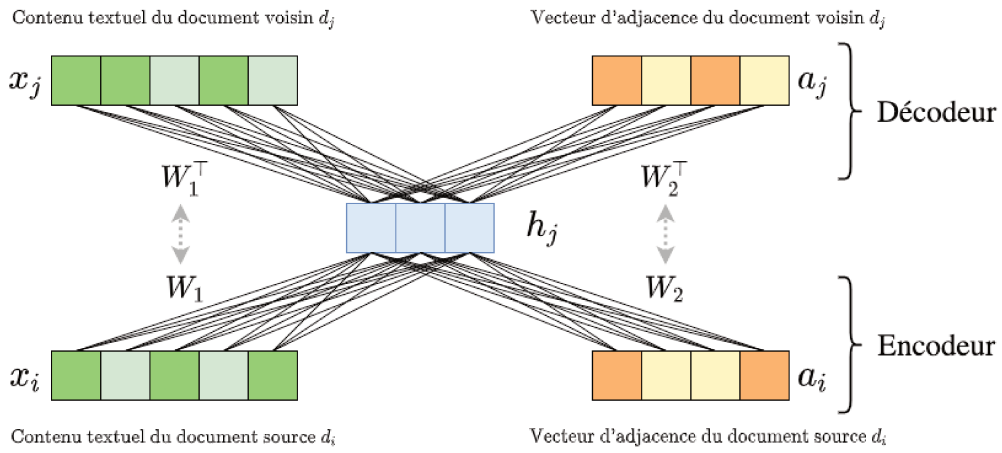


FIGURE 2.30 – Illustration de Adjacent-Encoder-X comme décrit par C. ZHANG et al., 2020. À partir des informations textuels x_i (en vert) et du réseau a_i (en orange) du document d_i, l'encodeur produit une représentation intermédiaire h_i de ce document (en bleu). Le décodeur reconstruit à partir de cette représentation intermédiaire le contenu textuel et le vecteur d'adjacence du document d_j, voisin de d_j. La double flèche grise indique la liaison des paramètres W_1 et W_2.

$$c_{ij} = \frac{exp(h_i^\top h_j)}{\sum_{d_k \in \mathcal{N}(d_i)} exp(h_i^\top h_k)}. \tag{2.105}$$

Ce coefficient sera d'autant plus grand que les représentations latentes des documents sont plus proches. Puisque ces documents sont liés, on propage ces représentations thématiques au document cible d_i, sa représentation thématique finale devenant $\mathbf{h}_i = \sum_{d_j \in \mathcal{N}_i} c_{ij} h_j$.

Le décodeur va, à partir du vecteur $\mathbf{h}_i$ représentant d_i, reconstruire son voisinage, et plus précisément, pour chaque $d_j \in \mathcal{N}_i$, la représentation log-normalisée $\tilde{\mathbf{x}}_j$ et le vecteur d'adjacence $\tilde{a}_j$. Ces reconstructions sont calculées comme suit :

$$\tilde{\mathbf{h}}_i = \sigma(W_1^\top \mathbf{h}_i + b_1), \quad \tilde{a_i} = \sigma(W_2^\top \mathbf{h}_i + b_2) \tag{2.106}$$

On remarque que l'encodeur et le décodeurs partagent tout deux les matrices de poids W_1 et W_2 (à une transposée près). Cette liaison des poids (en anglais *weights tying*) [PRESS et al., 2017] ; [HAKAN et al., 2016] joue un double rôle. Le premier est de réduire le nombre de paramètres du modèle, favorisant la vitesse d'entraînement du modèle. Le second est d'appliquer une régularisation sur les paramètres pour limiter les risques de sur-apprentissage.

L'apprentissage des paramètres W_1, W_2, b, b_1 et b_2 se fait en minimisant, pour chaque document d_i et chaque document d_j de son voisinage, l'entropie croisée binaire décrite Équation 2.107. L'optimisation est faite au moyen d'une descente de gradient stochastique ADAM.

$$\begin{aligned}\mathcal{L}(x_j, \tilde{x}_i, a_j, \tilde{a}_i) =& \mathcal{L}(x_j, \tilde{x}_i) + \mathcal{L}(a_j, \tilde{a}_i) \\ =& -\sum_{n \in \mathcal{V}} [x_{jn} \log(\tilde{x}_{in}) + (1 - x_{jn}) \log(1 - \tilde{x}_{in})] \\ & -\sum_{m \in \mathcal{D}} [a_{jm} \log(\tilde{a}_{im}) + (1 - a_{jm}) \log(1 - \tilde{a}_{im})] \end{aligned} \tag{2.107}$$

Cette méthode permet d'étudier les thématiques en s'intéressant aux valeurs de la matrice W_1, dont chaque ligne représente l'appartenance de chaque mot à la thématique correspondante. Les expériences montrent qu'elle propose une meilleure cohérence des thématiques que les autres modèles thématiques relationnels présentés dans cette section.

2.4 Conclusion

Au cours de ce chapitre nous avons exploré et détaillé certaines des principales méthodes visant à apprendre des représentations de mots, de documents, de graphes et de réseaux de documents. Certaines des méthodes présentées sont à la base des travaux proposés dans la suite de ce manuscrit, et d'autres seront le point de départ de perspectives de recherches s'inscrivant dans leur continuité.

Chapitre 3

Gestion de contenus ou de liens manquants

Au cours de ce chapitre nous discuterons de l'application d'une méthode de traduction, originellement utilisée pour la traduction de mots d'une langue vers une autre, pour le passage d'un espace de représentations de documents textuels à celui des représentations des sommets du réseau de documents associés.

3.1 Problématique

Lors de tâches de recommandation de contenus, par exemple au sein d'une base de connaissances, plusieurs méthodes peuvent être envisagées afin déterminer quels documents sont les plus susceptibles d'être similaires. L'une des plus simples est probablement l'apprentissage de représentations : une fois lesdites représentations apprises, on cherchera les contenus les plus proches de l'élément considéré, par exemple au moyen d'une similarité cosinus. Dans le cas particulier d'une base de connaissances, dont les éléments sont organisés en réseau et dont les contenus peuvent être de natures différentes, on pourra s'appuyer sur la structure du graphe afin d'apprendre des plongements des sommets, préservant ainsi un degré d'abstraction suffisant pour englober indifféremment des documents textuels, des fichiers ou des notices d'auteur. Se pose alors la question de comment traiter un nouveau contenu n'ayant pas encore été intégré au graphe. Beaucoup de méthodes se basent pour l'apprentissage sur des marches aléatoires, comme DeepWalk [Perozzi et al., 2014] ou Node2Vec [Grover et al., 2016]. Puisque les représentations des sommets sont appris de la même façon que les mots dans Word2Vec, elles ne peuvent fournir la représentation d'un nouveau document non présent dans les données d'entraînement, puisque celui-ci ne peut être rencontré lors des marches aléatoires.

En parallèle, beaucoup de contenus d'une base de connaissance sont des textes, et des méthodes comme Doc2Vec [Mikolov, Sutskever et al., 2013] offrent la possibilité de représenter des documents textuels sous forme de vecteur denses. Le cas de Doc2Vec est intéressant à plusieurs égards. Premièrement, comme pour DeepWalk ou Node2Vec, il prend comme base Word2Vec, une méthode d'apprentissage de représentations de mots reposant sur l'hypothèse distributionnelle. De plus, Doc2Vec permet, une fois les représentations des mots appris lors de la phase d'entraînement, d'estimer le plongement d'un nouveau document en effectuant une nouvelle étape d'estimation, en conservant fixes les autres paramètres.

La travail présenté dans ce chapitre propose une tâche de traduction permettant d'inférer la position d'un contenu inconnu, dans l'espace de représentation des documents à partir de son plongement dans l'espace des sommets du graphe, ou inversement. L'hypothèse faite ici est que la topologie d'un réseau de documents est corrélée au contenu des documents. Pour cela, nous nous appuyons sur les travaux de Smith et al., 2017, qui proposent d'appliquer le problèmes des procrustes orthogonaux [Schönemann, 1966] à deux ensembles de plongements de mots de deux langues différentes, afin d'apprendre une transformation linéaire entre ces deux espaces. Nous commencerons par présenter brièvement la preuve concernant le calcul de la matrice de passage, avant de présenter la contribution. Nous terminerons ce chapitre par une discussion sur l'utilisation de la méthode dans le cadre industriel dans lequel se place cette thèse.

3.2 Procrustes orthogonaux

Le problème des procrustes est un problème d'approximation de matrice. Considérant deux matrices, $A \in \mathbb{R}^{n\times d}$ et $B \in \mathbb{R}^{n\times d}$, on cherche une matrice $W \in \mathbb{R}^{d\times d}$ tel que W minimise $\|AW - B\|_F^2$. Schönemann, 1966 propose une solution à ce problème, lorsque l'on contraint W à être une matrice orthogonale, afin de préserver la norme des vecteurs dans A. On commence par développer l'expression $\|AW - B\|_F^2$:

$$\begin{aligned}\|AW - B\|_F^2 =& \sum_i \sum_j (AW - B)_{ij}^2 \\ =& \sum_i \sum_j (AW)_{ij}^2 + (B)_{ij}^2 - 2(AW)_{ij}(B)_{ij} \\ =& \|AW\|_F^2 + \|B\|_F^2 - 2tr(W^\top A^\top B)\end{aligned}$$

Or, puisque W est une matrice orthogonale, on a $\|AW\|_F^2 = tr(AWW^\top A^\top)) = \|A\|_F^2$. Ainsi,

$$\|AW - B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2tr(W^\top A^\top B) \quad . \tag{3.1}$$

Déterminer la valeur de W minimisant $\|AW - B\|_F^2$ revient alors à déterminer celle maximisant $tr(W^\top A^\top B)$. On utilise la décomposition en valeurs singulières de $A^\top B$, telle que $A^\top B = U^\top \Sigma^\top V$, où U et V sont deux matrices orthogonales, et Σ la matrice diagonales des valeurs singulières (non négatives) de $A^\top B$. On a alors :

$$\begin{aligned}tr(W^\top A^\top B) =& tr(WU^\top \Sigma^\top V) \\ =& tr(V^\top W^\top U\Sigma)\end{aligned}$$

Premièrement, on pose $Z = V^\top W^\top U$. Puisque Z est le produit de matrices orthogonales, elle est elle même orthogonale. De plus, puisque Σ est une matrice diagonale, on a $tr(Z\Sigma) = \sum_i Z_{ii}\Sigma_{ii}$. Le maximum est donc atteint en choisissant W de façon à ce que $Z = I$. $\|AW - B\|_F^2$ est alors à son minimum lorsque $W = UV^\top$.

3.3 Contribution

Student Research Abstract: Coping for Missing Content and Missing Links in Document Network Embedding

Jean Dupuy
Université de Lyon, Lyon 2, ERIC EA3083 – MeetSYS
Lyon, France
jean.dupuy@univ-lyon2.fr

ABSTRACT

Searching through networks of documents is an important task. A promising path to improve the performance of information retrieval systems in this context is to leverage dense node and content representations learned with embedding techniques. However, these techniques cannot learn representations for documents that are either isolated or whose content is missing. To tackle this issue, assuming that the topology of the network and the content of the documents correlate, we propose to estimate the missing node representations from the available content representations, and conversely. Inspired by recent advances in machine translation, we detail in this paper how to learn a linear transformation from a set of aligned content and node representations. The projection matrix is efficiently calculated in terms of the singular value decomposition. The usefulness of the proposed method is highlighted by the improved ability to predict the neighborhood of nodes whose links are unobserved based on the projected content representations, and to retrieve similar documents when content is missing, based on the projected node representations.

CCS CONCEPTS

• **Computing methodologies** → **Machine translation**; *Learning latent representations*; • **Information systems** → *Information retrieval*;

KEYWORDS

Document network, Document representation, Node representation, Translation

ACM Reference Format:
Jean Dupuy. 2020. Student Research Abstract: Coping for Missing Content and Missing Links in Document Network Embedding. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC'20), March 30-April 3, 2020, Brno, CzechRepublic.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3341105.3374222

1 INTRODUCTION

Searching through a network of documents, such as scientific articles [2], Web pages or social media posts [5], is a common task and a long-standing research topic. Recent works propose to improve the performance of information retrieval systems by leveraging dense vector space representations of the content and the nodes [16, 19]. Quite often, for various reasons, such as technical issues or legal limitations (*e.g.* paywall), not all links can be observed nor can all the content be retrieved.

These representations are learned with algorithms similar to those devised for word embedding, *e.g.* DeepWalk [14], Node2Vec [4] or GVNR [3] for node representations, or Doc2Vec [7] for content. Obviously, missing content prevents applying any document embedding techniques. Also, since network embedding algorithms perform random walks to measure the co-occurrence frequency between nodes, they cannot learn representations of isolated nodes.

In this paper, we address this issue by formulating a translation task, from available content representations to node representations, and from available node representations to content representations. Assuming that the topology of a network of documents correlates with the content of the documents, we propose to estimate the missing node (resp. document) representations as a linear transformation, *i.e.* projection, of the content (resp. node) representations. More specifically, we show how to construct a dictionary of pairs of content and node representations that allows learning a projection matrix, inspired by recent advances in machine translation with pre-trained word embeddings [6, 12, 18]. Note that a self-translation network embedding algorithm, STNE [9], has been recently proposed, however it addresses a different task. It learns to translate a sequence of content to a sequence of nodes to learn richer node representations and it cannot operate on isolated nodes nor deal with missing content.

Our approach proves relevant, as evidenced by the quality of the alignment between the estimated representations and the true representations. Practically speaking, our method has two main applications. On the one hand, estimating the node representation helps in recovering the missing links. On the other hand, estimating the content representation from the node representation allows measuring the similarity with the content of other documents and thus provides helpful clues about the missing content.

The rest of this paper is organized as follows. In section 2, we survey related work on document and network embedding, as well as embedding-based bilingual translation. We then describe our proposal to estimate the missing representations in section 3. Next, in section 4, we evaluate the quality of the projection and assess the improvement brought by our method in predicting the neighborhood of nodes whose links are missing and retrieve documents similar to those without content. Finally, we conclude and provide future directions in section 5.

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03.
https://doi.org/10.1145/3341105.3374222

2 RELATED WORK

Word embedding, *i.e.*, the task of learning dense vector space representations of words, is tightly connected to the tasks of learning document and node representations, *i.e.* document embedding and network embedding. In this section, we first briefly survey works related to word embedding and then present models adapted for document and network embedding. Lastly, we cover recent advances in offline bilingual translation via word embeddings, on which our work is based.

2.1 Word Embedding

Word embedding builds upon the distributional hypothesis. It states that distributional similarity and meaning similarity are correlated, which allows learning representations of words based on the contexts in which they occur, the context of a word being co-occurring words in a short window.

Mikolov *et al.* [11] propose two models, namely Skip-Gram and Continuous Bag-of-words (CBOW). The first models the conditional probability of occurrence of a word, given a word in the context. The second models the conditional probability of occurrence of a word, given the set of words in the context. In both cases, the probability is defined as a softmax function parameterized by the dot products of the vector representations of the words.

Because the softmax makes the estimation of the word representations computationally expensive, Mikolov *et al.* [13] propose approximate solutions based either on the hierarchical softmax or negative sampling, an approximation akin to noise-contrastive estimation.

2.2 Content Representation Learning

Le and Mikolov [7] extends the Skip-Gram and CBOW models to learn representations of documents, under the names, respectively, Distributed Bag-of-words (DV-DBOW) and Distributed Memory (DV-DM). They rely on a simple trick, that consists in considering each document as a distinct new vocabulary entry, that occurs in every window in the document. Thus, these models learn document representations that are good at predicting the words they contain.

2.3 Node Representation Learning

Even though the distributional hypothesis originated in linguistics, Perozzi *et al.* [14] show that sequences of nodes generated by random walks and sentences share some statistical properties. In particular, they show that the frequency at which nodes appear in short random walks follows a power-law distribution, like the frequency of words in language. Consequently, they propose to learn node representations under the distributional hypothesis. The proposed model, DeepWalk, consists in learning the representations based on the Skip-Gram model with the hierarchical softmax approximation, from a corpus of node sequences – deemed equivalent to sentences – generated by truncated random walks.

LINE [17] refines the objective optimized by DeepWalk, trying to preserve both the first and second order proximities in the embedding space. The representations are learned based on the Skip-Gram model with negative sampling rather than the hierarchical softmax approximation, using a specific stochastic gradient descent algorithm for weighted networks, that samples the edges according to their weights.

Node2vec [4] performs biased random walks, in order to better balance the exploration-exploitation trade-off, arguing that the added flexibility in exploring neighborhoods helps learning richer representations. The representations are also learned based on the Skip-Gram model with negative sampling.

2.4 Bilingual Translation

Mikolov *et al.* [12] first propose to learn a linear transformation from the embedding space of a source language into the embedding space of a target language. They suggest to estimate a projection matrix from pre-trained vectors and a bilingual dictionary. To this end they formulate a least square problem based on the Euclidean distance between aligned vectors, which they solve with the regular stochastic gradient descent algorithm.

Even though this approach yields encouraging results, Xing *et al.* [18] argue that the problem is ill-posed, since the similarity between representations is usually measured in terms of the cosine similarity, rather than the Euclidean distance. They show how to learn unitary word embeddings and formulate a maximization problem in terms of the dot product between some source vectors and projected target vectors. They also require the projection matrix to be orthogonal, so that the projected vectors remain unitary in order to preserve the equivalence between the dot product and the cosine similarity. Eventually, they devise a gradient descent based algorithm to learn the projection matrix. Yet, their approach is computationally expensive, because it requires to orthogonalize the projection matrix after each update, by computing its singular value decomposition.

Smith *et al.* [6] show that finding the projection matrix that solves the problem stated by Xing *et al.* is actually equivalent to solving a least square problem based on the Euclidean distance, when the vectors are unitary and the projection matrix is orthogonal. Furthermore, they link this problem with the orthogonal Procrustes problem, which admits an analytical solution. The projection matrix is obtained from the singular value decomposition of the similarity matrix between the aligned source and target vectors.

3 PROPOSED METHOD

In this section, we describe our proposal, which consists in adapting techniques devised for bilingual translation to learn a projection from the content representations (pre-trained with, *e.g.* Doc2Vec) to the node representations (pre-trained with, *e.g.* DeepWalk), and vice versa.

Consider a corpus of documents organized in a network, so that the content of some documents is missing and the links of some other documents are unobserved. We assume a set of pre-trained node representations $A = \{a_i \in \mathbb{R}^k\}$ and a set of document representations $B = \{b_i \in \mathbb{R}^k\}$. Our goal is to estimate the missing node representations in A, from the content representations in B ; conversely, estimate the missing content representations in B, from the node representations in A.

We posit that the topology of the network and the content of the documents correlate. Thus, we aim at finding a projection matrix

$W \in \mathbb{R}^{k \times k}$ that maps content representations to node representations, and node representations to content representations. More particularly, given a document i, we want W to maximize the cosine of the angle $\theta_{a_i, b_i W}$ between a_i and $b_i W$. That is to say, we want to find a linear transformation so that the vectors a_i and $b_i W$ point in the same direction. More formally, based on a set of pairs of node representations and document representations, S, we want to find a matrix W so that:

$$\max_W \sum_{i \in S} \cos(\theta_{a_i, b_i W}), \tag{1}$$

In the rest of this section, we describe how to construct S and then detail how to learn the projection matrix W.

3.1 Dictionary construction

Rather than learning the projection from all the available pairs of node representations and content representations, we suggest to restrain to a specific subset. The aim in doing so is (i) favor the learning of a good projection and (ii) limit the computational cost.

Bojanowski *et al.* [1] note that Skip-Gram, the model behind most network embedding algorithms, struggles to learn good representations for rare words. This also holds for network embedding algorithms, since Perrozi *et al.* [14] show that the frequency at which nodes occur in short random walks and the frequency of words in language follow a similar law. Hence, we propose to avoid rare nodes and to construct the dictionary only from the m nodes that occur the most in short random walks on the network. Formally, with f_i the frequency of node i, we construct the set S of cardinality m that maximizes $\sum_{i \in S} f_i$.

3.2 Projection Learning

Following [18], we rewrite formula 1 in terms of the dot product, and thus require the node and content representation to be unitary vectors and constrain W to be orthogonal:

$$\max_W \sum_{i \in S} a_i \cdot b_i W, \text{ subject to } W^\top W = I. \tag{2}$$

Yet, rather than learning the representations again as Xing *et al.* suggest, we simply row-normalize A and B. Then, thanks to the proportional relationship between the opposite of the dot product and the squared Euclidean distance highlighted in [6], because $||a_i - b_i W||^2 = ||a_i||^2 + ||b_i||^2 - 2(a_i \cdot b_i W)$, we transform the problem into a minimization one:

$$\min_W \sum_{i \in S} ||a_i - b_i W||^2, \text{ subject to } W^\top W = I. \tag{3}$$

This is a classic linear algebra problem, which solution is given in [15]. It consists in calculating the singular value decomposition of the matrix $M = A_S^\top B_S$, with $A_S \in \mathbb{R}^{m \times k}$ and $B_S \in \mathbb{R}^{m \times k}$ the aligned node and content representations according to the dictionary S. With $M = U \Sigma V^\top$, the matrix W is defined as:

$$W = U V^\top. \tag{4}$$

We estimate the node representation of an isolated node by calculating $\tilde{a} = bW$. Similarly, we estimate the content representation of a document whose content is missing by calculating $\tilde{b} = aW^\top$.

Table 1: Network properties.

Name	# of documents	# of links
Cora	2,211	5,001
HepTh	1,038	1,974

4 EXPERIMENTS

In this section, we evaluate our proposal on two networks of documents. First, we focus on missing links and show that estimating the node representations from the content representations allows to better reconstruct the neighborhood of nodes whose links are unobserved. Second, we focus on missing content and show that estimating the content representations from node representations allows to retrieve a significant fraction of the similar documents.

4.1 Experimental setup

4.1.1 Data. We consider two well known networks of scientific articles, Cora [10] and HepTh [8], whose properties are summed up in Table 1. No content nor citation links are missing. These datasets available online[1].

4.1.2 Node and Content Representations. We experiment with DeepWalk [14] to learn the node representations, with the following configuration: 80 random walks length 80 from each node, and a co-occurrence window of size 10. Regarding content representation, we experiment with the Distributed Memory (DV-DM) variant of Doc2Vec [7] with a window of size 10. Here after, we report the results for representations of size 500.

4.1.3 Projection Learning. We learn the projection matrix via the exact singular value decomposition, based on a dictionary of size 1400 for Cora, and a dictionary of size 550 for HepTh.

4.1.4 Similarity Metric. We measure the similarity between vector representations in terms of cosine similarity [11].

4.2 Missing Links: Neighborhood Reconstruction

We begin by assessing the quality of the estimated node representations. The evaluation task consists in reconstructing the neighborhood of a document whose links are unobserved. We adopt a leave-one-out strategy and proceed in the following manner. First, we learn all the content representations from the entire corpus. Then, for each document i, we do the following:

(1) Hide all the incoming and outgoing links of document i;
(2) Learn the node representations on this sub-network;
(3) Learn the projection matrix on this sub-network;
(4) Measure the similarity between the estimated node representation $\tilde{a}_i = b_i W$ and all the other node representations $\{a_j\}$;
(5) Order the documents accordingly and measure the precision at n, $P@n$ w.r.t. the true links.

[1]Get the data: https://github.com/thunlp/CANE/tree/master/datasets/cora, https://github.com/thunlp/CANE/tree/master/datasets/HepTh

Table 2: Precision gain in link prediction.

		P@5	P@10	P@20	P@50
2*Cora	$\cos(\theta_{b_i,b_j})$	+4.4%	+23.4%	+20.1%	+20.0%
	Random_{200}	+681.4%	+475.9%	+166.1%	+51.9%
2*Hepth	$\cos(\theta_{b_i,b_j})$	+69.2%	+42.9%	+5.9%	+2.6%
	Random_{200}	+516.6%	+375.1%	+350.4%	+175.6%

We report the relative gain in average precision in Table 2, w.r.t. two baselines:

- $\cos(\theta_{b_i,b_j})$: Documents ordered according to the similarity between the content representations;
- Random_{200}: Documents ordered randomly, picked among the 200 documents with the highest degrees.

It reveals that the estimated node representations, obtained via the linear transformation of the content representations always lead to a better precision in predicting the neighborhood of a document.

4.3 Missing Content: Similar Document Retrieval

We now switch our attention to missing content. We assess the quality of the estimated content representations following a similar leave-one-out methodology. We learn all the the content representations and define the true sets of similar document $S_i = \{j | \cos(\theta_{b_i,b_j}) > 0.2\}$. The evaluation task consists in reconstructing the set of similar documents for a document whose content is unknown. We proceed as follows. First, we learn all the node representations from the whole network. Then, for each document i, we do the following:

(1) Hide document i from the corpus;
(2) Retrain the content representations on this sub-corpus;
(3) Learn the projection matrix on this sub-corpus;
(4) Measure the similarity between the estimated content representation $\tilde{b}_i = a_i W^\top$ and all the content representations $\{b_j\}$;
(5) Measure the accuracy by comparing the true set of similar documents S_i and the estimated set of similar documents $\tilde{S}_i = \{j | \cos(\theta_{\tilde{b}_i,b_j}) > 0.2\}$.

We report the average precision following this procedure in the first row of Table 3. The second row shows the average precision based on the node representations, where $\tilde{S}_i = \{j | \cos(\theta_{a_i,a_j}) > 0.2\}$. It reveals that estimating the missing content representation from the corresponding node representation doubles the accuracy in document retrieval, in comparison with the accuracy obtained by measuring the similarity based on the node representations. In particular, we are able to recover, on average, two-thirds of the similar documents in the Hepth corpus.

Table 3: Average accuracy in similar document retrieval.

	Cora	Hepth
$\tilde{S}_i = \{j \mid \cos(\theta_{\tilde{b}_i,b_j}) > 0.2\}$	**0.36**	**0.65**
$\tilde{S}_i = \{j \mid \cos(\theta_{a_i,a_j}) > 0.2\}$	0.13	0.30

5 CONCLUSION

In this paper, we described a way to efficiently learn a linear function to map pre-trained node and content representations learned from a network of documents. In presence of missing links and missing content, this method helps in reconstructing the neighborhood of isolated documents and also helps in recovering documents that are similar to a document whose content is unknown. In future work, we would like to investigate how to learn a non-linear transformation to refine the mapping between the two spaces.

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)* 5 (2017).
[2] Robin Brochier. 2019. Representation Learning for Recommender Systems with Application to the Scientific Literature. In *Companion Proceedings of Web Conference (WWW)*.
[3] Robin Brochier, Adrien Guille, and Julien Velcin. 2019. Global Vectors for Node Representations. In *Proceedings of The Web Conference (WWW)*.
[4] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*.
[5] Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A. Zighed. 2013. Information Diffusion in Online Social Networks: A Survey. *SIGMOD Rec.* 42, 2 (2013).
[6] Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
[7] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the International Conference on International Conference on Machine Learning (ICML)*.
[8] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*.
[9] Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to Node: Self-Translation Network Embedding. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*.
[10] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval* 3, 2 (2000).
[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* (2013).
[12] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. *CoRR* abs/1309.4168 (2013).
[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*.
[15] Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (1966).
[16] Dominic Seyler, Praveen Chandar, and Matthew Davis. 2018. An Information Retrieval Framework for Contextual Suggestion Based on Heterogeneous Information Network Embeddings. In *Proceedings of the International Conference on Research Development in Information Retrieval (SIGIR)*.
[17] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web (WWW)*.
[18] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.
[19] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *Proceedings of the The Web Conference (WWW)*.

3.4 Perspective d'implémentation pour l'entreprise

Au cours de ce chapitre nous avons proposé l'application d'une méthode de traduction à une tâche de mise en relation de plongements de documents vers des plongements de sommets de réseaux de documents associés. Un cas d'usage peut être identifié afin d'implémenter cette méthode dans le cadre d'$i2Kn$.

Celui-ci vise à proposer de meilleures recommandations pour de nouveaux documents. Les expériences menées montrent en effet un gain en terme de prédiction de liens lors de l'utilisation d'une matrice de passage optimale entre les représentations des contenus et celle des sommets du réseau. En estimant la représentation d'un nouveau document à partir de son contenu au moyen de Doc2Vec, on utilisera la projection de cette représentation dans l'espace des représentations des sommets pour déterminer les documents les plus proches. De plus, puisque les méthodes d'apprentissage de sommets considérés sont agnostiques au type de contenus (*eg.* DeepWalk ou Node2Vec), il est envisageable de rapprocher un nouveau contenu textuel d'un éventuel contenu de nature différente, comme un contributeur, ou un fichier.

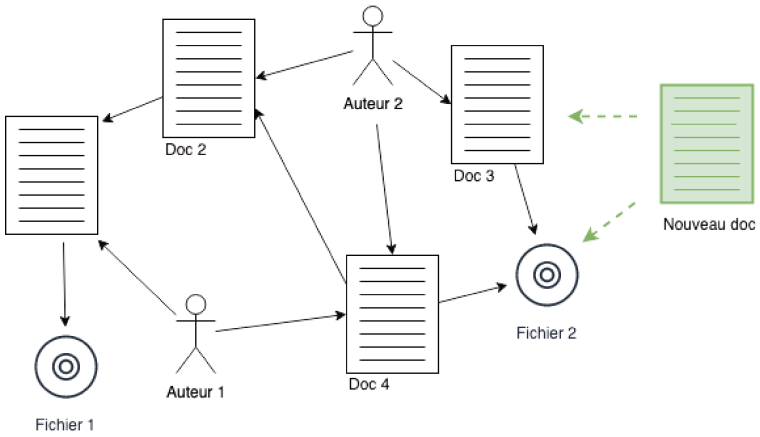


FIGURE 3.1 – Illustration d'une base de connaissance

La mise en place d'un tel système peut alors être utilisé lors de recommandations intra-base de connaissances. Il est nécessaire d'apprendre une matrice W permettant le passage entre les deux espaces de représentation. Dans le cadre d'une base déjà existante, on cherchera l'ensemble des documents appartenant à la plus grande composante connexe du réseau afin de construire le corpus servant à déterminer la matrice W.

3.5 Conclusion

Durant ce chapitre nous avons détaillé une méthode proposant l'application d'un mécanisme simple de traduction en TAL à la gestion de contenus manquant dans les réseaux

de documents. L'hypothèse est que la structure des réseaux de documents est corrélée aux contenus textuels de ceux-ci. Les expériences menées tendent à montrer que l'emploi d'une matrice de passage entre les espaces de représentation des contenus et du réseau permet à la fois de mieux reconstruire le voisinage d'un nouveau document dont seul le contenu est accessible, ainsi que de mieux déterminer les documents aux contenus similaires lorsque la seule information disponible se trouve être les documents liés (par exemple, un article derrière un péage de lecture numérique, dont on ne peut accéder qu'aux citations). Cette méthode pourra trouver sa place dans l'architecture logicielle d'une base de connaissances pour améliorer la recommandation de documents liés lors de la consultation de nouveaux contenus.

Chapitre 4

Modélisation contextuelle de liens dans des réseaux de documents

Comme exposé en introduction, les hyperliens dans les bases de connaissances sont presque toujours générés soit à la main, comme c'est le cas dans Wikipedia par exemple, soit au moyen d'heuristiques simples, comme l'utilisation d'algorithmes de recherche de sous-chaîne entre le contenu d'un document et les titres des tous les autres. Cette dernière méthode, bien que semblant relativement automatique de prime abord cache très souvent un travail des contributeurs afin de faire explicitement apparaître les termes exactes du titre, ou créer un alias aux titres afin que ceux-ci puissent se retrouver lors de paraphrases.

4.1 Problématique

Fort de ce constat, il serait alors intéressant de proposer une méthode permettant de positionner correctement un hyperlien (une ancre et un lien vers un autre document) au sein d'un document. Localiser automatiquement ces ancres peut alors être utilisé de plusieurs façons. Les liens peuvent être soit générés à la volée lors de l'affichage d'une page par un lecteur, soit être proposés lors de la phase d'édition d'une page. La première forme ne nécessite pas de supervision de la part du contributeur, mais ouvre la voie à l'incursion d'erreurs au niveaux des hyperliens entre les pages. La seconde option demande elle une certaine supervision venant du contributeur, lui demandant de valider manuellement les suggestions du modèle.

On propose donc une méthode permettant de déterminer les mots, ou groupes de mots, pouvant être à l'origine d'un hyperlien entre un document source et un document cible : *Contextual Relational Topic Model*. Ces travaux reposent sur le cadre de la modélisation thématiques, cadre ayant émergé suite à des travaux comme pLSA et LDA, tous deux décrits à la Section 2.1.1. Plus précisément, afin de prendre en compte l'organisation en réseaux des documents, ces travaux reposent sur *Relational Topic Model* [Chang et al., 2009], détaillé à la Section 2.3.1.

Les deux articles suivant développent ce modèle. La première, Section 4.2, propose une première version de *Contextual Relational Topic Model*, qui s'attarde sur les différentes façons envisagées pour agréger le contexte d'apparition d'une ancre. La seconde, Section 4.3, propose une formulation alternative de CRTM, y appliquant certaines améliorations.

4.2 Contribution 1

Contextual-RTM : un cadre général pour la modélisation de thématiques dans les réseaux de documents

Jean Dupuy[***], Adrien Guille[**], Julien Jacques[**]

[*]MeetSYS
[**]Université de Lyon, Lyon 2, ERIC EA3083

Résumé. Les longs parcours de bases de connaissance ou de grands corpus de documents en réseau peuvent parfois perdre le lecteur et lui faire manquer certains liens entre les documents qu'il consulte. Nous proposons ici un cadre de modélisation thématique pour les réseaux de documents, Contextual-RTM, qui généralise *Relational Topic Model* (RTM). Alors que RTM est agnostique à la position des liens dans les documents, Contextual-RTM les prend en compte, permettant une meilleure contextualisation. Nous définissons le contexte comme une fenêtre de taille ajustable centrée autour du lien et proposons 3 méthodes d'agrégation du contexte : uniforme, positionnelle et sémantique. Contextual-RTM se montre compétitif sur des tâches d'identification de mots à l'origine de liens entre documents. Nous intégrons ces méthodes dans un système d'aide à la lecture capable d'inférer localement des liens latents entre documents. Ainsi le lecteur garde une trace de ses précédentes lectures, et s'en voit recommandé de nouvelles.

1 Introduction

Nous nous intéressons à la prédiction de liens entre documents, avec comme objectif la mise au point d'un système de recommandation et d'aide à la lecture. Nous nous attachons plus particulièrement aux réseaux de documents où les liens entre les pages sont localisés dans le texte. C'est le cas par exemple des liens hypertextes dans les pages Wikipédia. C'est également le cas des citations dans les articles scientifiques.

Nous portons une attention particulière à la localisation des liens dans le texte, et aux mots qui les entourent (que nous appelons leur contexte) car nous pensons que cette information est porteuse de sens et permet de mieux expliquer l'existence d'un lien entre deux documents. Par exemple les liens qui se trouvent dans l'introduction d'un article scientifique sont généralement vers d'autres articles relativement généraux, ceux dans le corps de l'article plus spécialisés sur une tâche similaire à celle qui est traitée dans l'article, et ceux en conclusion vers des tâches spécifiques mais différentes.

Tandis que les système existants recommandent un ensemble de documents à lire, étant donné le dernier document consulté, nous souhaitons mettre au point un système

guidant encore plus la lecture. Notre objectif est de positionner directement les recommandations à l'intérieur du document en cours de lecture. Un tel système ne peut-être mis en œuvre sur la base des méthodes de prédiction de liens existantes, qui modélisent un lien entre un document A et un document B.

Par conséquent, nous proposons dans cet article une nouvelle méthode de prédiction de lien, modélisant les liens entre une portion du document A (un mot ou groupe de quelques mots) et un document B. Celle-ci repose sur une généralisation d'un modèle thématique probabiliste : Relational Topic Model Chang et Blei (2009).

RTM modélise les liens par une fonction de score, dont le résultat est d'autant plus grand que les proportions de thématiques des deux documents liés sont plus similaires. Cependant RTM ne prend pas en compte le contexte d'apparition des liens car le score de lien est fonction de tous les mots de chacun des deux documents. Nous proposons avec CRTM trois variantes de cette fonction score de lien, qui diffèrent selon l'importance qui est donnée aux mots du contexte. Ces trois variantes (uniforme, positionnelle et sémantique) nous semblent couvrir les approches les plus générales permettant de prendre en compte le contexte d'apparition des liens. D'autres méthodes plus spécifiques peuvent bien entendu être envisagées.

Nous montrons d'abord que CRTM surpasse RTM sur une tâche de prédiction des mots à l'origine des liens, tant en terme de précision @k qu'en terme d'aire sous la courbe ROC. Pour ces évaluations RTM est entraîné avec la fonction de lien originale. Nous substituons celle-ci à une fonction de lien de CRTM lors de la prédiction, car la fonction de lien originale ne permet pas de répondre à la tâche évaluée. Nous montrons ainsi que la prise en compte du contexte lors de l'entraînement des modèles donne de meilleurs résultats que son exploitation uniquement lors de la prédiction. Nous illustrons ensuite sur des exemples comment elle permet de mieux positionner les documents recommandés, en les associant à des termes pertinents dans les documents en cours de lecture.

2 État de l'art

Les corpus de documents sont étudiés depuis longtemps. En particulier, beaucoup d'attention a été donnée à la représentations de ces documents, soit par des méthodes probabilistes, par exemple pour inférer une structure thématique latente comme dans *Latent Dirichlet Allocation* (LDA) de Blei et al. (2003), soit par des méthodes visant à apprendre des représentations denses en faible dimension comme Doc2Vec de Le et Mikolov (2014).

Très souvent ces corpus ne sont pas une simple collection de textes et s'avèrent être structurés en réseaux. On pense notamment à la littérature scientifique, les contenus encyclopédiques comme Wikipedia, les bases de connaissances ou plus généralement le Web. De nombreuses méthodes s'appuient sur cette information supplémentaire pour déterminer de meilleures représentations. Certaines méthodes se basent sur des modèles probabilistes pour représenter les thématiques latentes, comme NetPLSA (Zhang et al., 2016) ou RTM (Chang et Blei, 2009) et d'autres sur des méthodes d'apprentissage de plongements de document comme TADW (Yang et al., 2015), CANE (Tu et al., 2017),

GVNR (Brochier et al., 2019), RLE (Gourru et al., 2020) ou IDNE (Brochier et al., 2020).

Nous retenons l'approche fondée sur la modélisation de thématiques car elle fournit un cadre formel pour apprendre les représentations des documents et autorise une prise en compte simple du contexte. Pour toute ces raisons nous nous basons sur *Relational Topic Model*. Puisque RTM se base sur LDA, nous commençons par détailler ce dernier avant de présenter RTM.

2.1 LDA

Latent Dirichlet Allocation (Blei et al., 2003) est un modèle génératif probabiliste. En supposant qu'un document répond d'un mélange d'un petit nombre K de thématiques, il vise à en attribuer une à chaque mots observés.

Modèle : Soit un corpus de documents, dont chaque document $d \in [1, D]$ comporte N_d mots appartenant à un vocabulaire fixe V. Soit $\mathbf{w}$ tous les mots observés de touts les documents, on note $w_{d,n} \in V$ la valeur prise par le n-*ième* mot du document d. On considère plusieurs variable latentes pour expliquer $\mathbf{w}$: $\mathbf{z}$ et $\boldsymbol{\theta}$, avec $z_{d,n} \in \{1, \cdots, K\}$ l'affectation thématique de $w_{d,n}$ et θ_d le mélange de thématiques du document d. On introduit enfin $\alpha \in \mathbb{R}^K$ et $\beta \in \mathbb{R}^{K \times V}$, respectivement le paramètre de la distribution des thématiques des documents et l'a priori sur la distribution des mots par thématiques.

Processus génératif et estimation : Pour chaque document d on tire le mélange de K thématiques θ_d selon une distribution de Dirichlet de paramètre α. Ensuite, pour chaque mot $w_{d,n}$ de d on tire sa thématique latente $z_{d,n}$ à partir d'une distribution multinomiale de paramètre θ_d. Enfin on tire $w_{d,n}$ selon une distribution multinomiale de paramètre $\beta_{z_{d,n}}$. L'estimation des paramètres se fait en trouvant le maximum de vraisemblance. La vraisemblance étant impossible à calculer en pratique on utilise un algorithme Variational EM (Jordan et al., 1999) afin d'optimiser la ELBO (Bishop, 2006) alternativement vis à vis des paramètres variationnels et des paramètres du modèle.

2.2 RTM

Comme dit précédemment, *Relational Topic Model* (Chang et Blei, 2009) étend LDA en y incorporant une information sur les liens entre les documents.

Soit $\mathbf{y}$ la variable aléatoire binaire, indiquant l'observation ou non d'un lien entre deux document. On définit la fonction ψ comme étant une fonction de score de lien :

$$\psi_e(y = 1) = \exp\left(\boldsymbol{\eta}^{\mathrm{T}} \left(\overline{\mathbf{z}}_d \circ \overline{\mathbf{z}}_{d'}\right) + \nu\right) \tag{1}$$

où $\overline{\mathbf{z}}_d = \frac{1}{N_d} \sum_n z_{d,n}$ représente la répartition des thématiques latentes du document d, et ν et η des paramètres.

On remarque que le produit de Hadamard $\overline{\mathbf{z}}_d \circ \overline{\mathbf{z}}_{d'}$ amplifie les thématiques communes et tend à gommer les thématiques distinctes entre les deux documents. De plus, de part la nature linéaire de ces fonctions de score, on constate que le paramètre η,

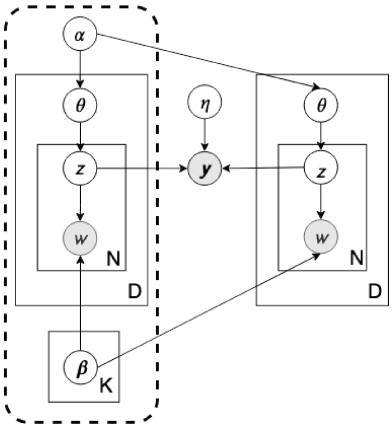


FIG. 1 : Modèle graphique de *Relational Topic Model* (LDA entouré en pointillés)

qui est un vecteur de taille K, ne permet pas de modéliser un éventuel lien entre deux thématiques différentes. On peut donc raisonnablement penser qu'il permet principalement d'amoindrir l'importance de certaines thématiques ne présentant pas d'intérêts pour la modélisation des liens. L'estimation des paramètres β, η et ν s'effectue de la même façon que pour LDA. A noter également que les auteurs ne prennent en compte que les cas où $y = 1$, considérant qu'un lien non observé n'induit pas qu'il n'existe pas. Le processus génératif de RTM est résumé dans l'Algorithme 1 et la figure 1.

pour tous *document d de D* **faire**
  tirer la proportion de thématiques de d : $\theta_d|\alpha \sim Dirichlet\,(\alpha)$
  pour tous *mot* $w_{d,n}$ *de d* **faire**
    tirer l'assignation thématique : $z_{d,n}|\theta_d \sim Multinomiale\,(\theta_d)$
    tirer le mot : $w_{d,n}|z_{d,n} \sim Multinomiale\,\left(\beta_{z_{d,n}}\right)$
pour tous *paire de documents (d,d′)* **faire**
  tirer l'indicateur de lien : $y_{d,n}\,|\boldsymbol{z}_d, \boldsymbol{z}_{d'} \sim \psi\,(\cdot|\boldsymbol{z}_d, \boldsymbol{z}_{d'})$

Algorithme 1 : Processus génératif de RTM

3 Généralisation de RTM : Contextual-RTM

Contextual-RTM généralise RTM, en proposant maintenant de prendre en compte la localisation et le contexte d'apparition d'un lien au sein du texte, permettant ainsi des usages différents et pouvant être adapté à des corpus de natures variées.

Là où RTM suppose qu'un lien entre deux documents d et d' est marqué par la similitude de $\overline{\mathbf{z}}_d$ et $\overline{\mathbf{z}}_{d'}$, nous modifions la manière de calculer $\overline{\mathbf{z}}_d$ afin de prendre en compte les thématiques des mots du contexte d'apparition des liens. Nous proposons

ici 3 variations de la fonction de lien présentée à l'Equation (1), permettant de prendre en compte ce contexte, de façon uniforme, positionnelle ou sémantique. Nous choisissons d'utiliser la version exponentielle de la fonction de lien car selon Chang et Blei (2009) c'est celle qui donne de meilleurs résultats lors de leurs évaluations.

3.1 Approche Uniforme

On peut considérer qu'un lien est déterminé à égale importance par tous les mots de son contexte immédiat, et les thématiques qui s'y rapportent. Ainsi dans l'Equation 2, nous remplaçons la moyenne $\overline{\mathbf{z}}_d$ des assignations des thématiques dans l'ensemble du document de l'Equation 1 par celle des mots contenus dans une fenêtre de taille l autour du lien :

$$\psi_{uni}(y=1) = \exp\left(\boldsymbol{\eta}^{\mathrm{T}}\left(\overline{\mathbf{z}}_{d,n}^{(u)} \circ \overline{\mathbf{z}}_{d'}\right) + \nu\right), \tag{2}$$

avec

$$\overline{\mathbf{z}}_{d,n}^{(u)} = \frac{1}{2l+1}\sum_{i=n-l}^{n+l} z_{d,i}.$$

Une fenêtre couvrant tout le document permettra de retrouver l'expression de RTM, et une fenêtre de taille nulle ne prendra en compte que le mot à l'origine du lien. On peut également ajuster la taille de cette fenêtre pour la faire coïncider avec une phrase ou un paragraphe.

3.2 Approche positionnelle

On peut également supposer que les mots influencent d'autant plus l'apparition d'un lien qu'ils en sont plus proches dans le texte. Pour modéliser cela nous proposons de lisser les assignations thématiques des mots présents dans une fenêtre de taille l au moyen d'une pondération de type gaussienne, centrée sur l'origine du lien, comme présenté dans l'Equation 3 :

$$\psi_{pos}(y=1) = \exp\left(\boldsymbol{\eta}^{\mathrm{T}}\left(\overline{\mathbf{z}}_{d,n}^{(p)} \circ \overline{\mathbf{z}}_{d'}\right) + \nu\right), \tag{3}$$

avec :

$$\overline{\mathbf{z}}_{d,n}^{(p)} = \frac{1}{2l+1}\sum_{i=n-l}^{n+l} e^{-\frac{1}{2}\left(\frac{i-n}{\sigma}\right)^2} z_{d,i},$$

où σ est l'écart-type choisi.

3.3 Approche sémantique

La dernière approche suppose que tous les mots ne sont pas pertinents pour induire un lien, et que seuls les plus proches sémantiquement ont une véritable influence. Cette approche requiert d'incorporer dans notre modèle de la connaissance externe sur cette proximité sémantique, ce que nous proposons de faire en utilisant des plongements de

mots pré-appris. Nous utilisons dans l'équation 4 le *Scaled-Dot Product* décrit dans Vaswani et al. (2017), variante normalisée de *Dot-Product Attention* des mêmes auteurs permettant que le softmax ne prenne des valeurs trop extrêmes, pour donner plus de poids aux thématiques latentes des mots qui sont les plus proches sémantiquement de ceux à l'origine du lien. On gommera ainsi facilement les mots vides ou non lexicaux présents dans la phrase tout en renforçant l'importance des mots similaires :

$$\psi_{sem}(y=1) = \exp\left(\boldsymbol{\eta}^{\mathrm{T}}\left(\overline{\mathbf{z}}_{d,n}^{(s)} \circ \overline{\mathbf{z}}_{d'}\right) + \nu\right), \tag{4}$$

avec :

$$\overline{\mathbf{z}}_{d,n}^{(s)} = \text{softmax}\left(\frac{e_n.E_l^T}{\sqrt{dim}}\right) z_{d,l}.$$

où :

- e_n est la représentation du mot à la source du lien,
- E_l est la matrice des représentations de tous les mots de la fenêtre,
- dim est la dimension de l'espace de représentation des mots,
- z_l est la matrice des assignations de thématique de tous les mots de la fenêtre.

Cette méthode permet tout autant d'utiliser des plongement de mots entraînés sur les données étudiées que des versions pré-entraînées sur un autre corpus.

4 Expériences

Nous évaluons l'intérêt de la contextualisation des liens en comparant Contextual-RTM à RTM sur leur capacité à retrouver les mots à l'origine d'un lien entre deux documents. Dans un premier temps nous présentons les corpus utilisés et comment ils ont été construits, puis les tâches et les métriques utilisées. Enfin nous commenterons les résultats quantitatifs et qualitatifs de CRTM par rapport à RTM.

4.1 Corpus

Les modèles sont évalués sur des jeux de données constitués des paragraphes introductifs d'articles Wikipedia appartenant à diverses catégories. En effet les corpus habituellement utilisés en prédiction de liens ne donnent pas la localisation des liens dans le texte. Ces corpus sont soit vectorisés (i.e. sac de mots) soit pré-traités (sans marquage des liens dans le texte).

Ils sont construits en parcourant récursivement la chaîne de citations en partant de la page principale d'une catégorie Wikipedia. A chaque itération on ajoute les documents non encore vus, jusqu'à ce que la profondeur désirée soit atteinte.

Deux raisons nous ont poussé à utiliser les paragraphes introductifs de Wikipedia. Premièrement ils sont relativement complets et comportent déjà un nombre important de liens, les rendant comparable à une fiche de base de connaissance en entreprise. Ensuite le format Wikitext, dérivé de Markdown, permet un pré-traitement simple des documents, en particulier la représentation des liens. Un lien dans une page Wikipedia prendra la forme suivante : `[[United_Kingdom|UK]]`. On retrouve à gauche de la barre verticale le titre de la page vers lequel pointe le lien, et à droite le mots qui portera l'hyperlien.

Le tableau 1 résume les propriétés de jeux de données utilisés dans nos évaluations. Celles-ci se feront en deux langues : Français et Anglais.

Catégories	*Physics*	*Arts*	*Society*	*Physique*	*Société*
Langue	Anglais	Anglais	Anglais	Français	Français
Nb de pages	12769	15007	16716	8281	14770
Nb de liens	119671	97774	108275	53738	88643
Nb moyen de mots	149	153	146	149	151
Nb moyen de phrases	7.84	7.74	7.70	4.71	4.65
Connectivité moyenne	9.37	6.52	6.47	6.49	6.00

TAB. 1 : Description des jeux de données.

4.2 Tâches et métriques d'évaluation

Nous évaluons RTM et CRTM sur deux tâches : leur capacité à retrouver le mot à l'origine d'un lien donné entre deux documents en le plaçant parmi les mots les plus probables, et une évaluation qualitative pour juger de la cohérence des mots suggérés. Pour chaque corpus, nous cachons certains liens avant l'entraînement des modèles.

Précision à K et courbe ROC : Pour tous les liens retirés avant l'entraînement des modèles on cherche à prédire quels mots en sont à l'origine et dans quelle proportion ils apparaissent dans le top-K des mots les plus probables. Nous présentons pour chaque corpus et chaque modèle la moyenne des précisions à K pour tous les liens cachés ainsi que l'aire sous la courbe ROC.

Évaluation qualitative : Nous cherchons ici à montrer la capacité de Contextual-RTM à proposer un ensemble de mots cohérent liants des pages et mettant en lumière des liens non triviaux pouvant échapper au lecteur. Nous proposons pour cela de comparer pour deux documents les mots suggérés par RTM et Contextual-RTM les liants à un troisième.

4.3 Méthodes

Pour la première évaluation nous retirons aléatoirement 2000 liens de chaque corpus. Nous comparons RTM aux quatre variantes de Contextual-RTM que nous proposons :

- **RTM** : entraîné comme décrit par Chang et Blei (2009), c'est à dire avec pour fonction de lien Ψ_e (cf. Eq 1). Il est impossible avec cette fonction de lien de mesurer la probabilité que les mots soient à l'origine des liens (tous les mots d'un document ont la même probabilité d'être à l'origine du lien). En conséquence, lors de l'évaluation, nous substituons ψ_e par les fonctions de lien des variantes de CRTM que nous proposons.
- **Variantes de CRTM** :
  - CRTM-Uni-0 : une variante uniforme où le contexte se limite aux mots à l'origine du lien (taille de fenêtre nulle).

— CRTM-Uni-S : une variante uniforme où le contexte correspond à tous les mots de la phrase.
— CRTM-Pos-S : une variante positionnelle où le contexte correspond à tous les mots de la phrase.
— CRTM-Sem-S : une variante sémantique où le contexte corresponds à tous les mots de la phrase. Les représentations des mots utilisés ont été obtenues à partir de *Skip-gram* avec échantillonage négatif (Mikolov et al., 2013), entraîné sur les corpus étudiés, avec une fenêtre de 10 mots.

Tous les modèles sont entraînés avec 50 thématiques, qui conduit aux meilleurs résultats pour RTM et Contextuel-RTM.

Nous fixons l'hyperparamètre α à 5.0 pour tous les modèles (identique à la valeur choisie par Chang et Blei (2009) dans leurs évaluations), celui-ci ayant donné les meilleurs résultats individuellement pour chacun d'entre eux.

4.4 Résultats

4.4.1 Quantitatif

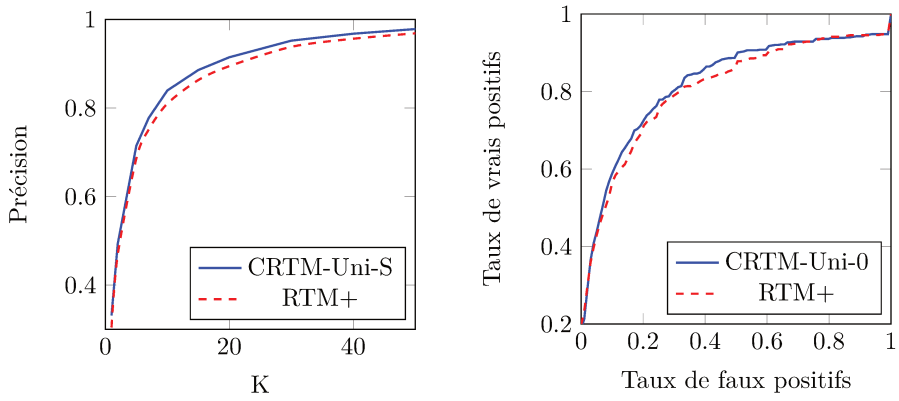
Nous étudions ici la précision moyenne à K pour différents corpus, reportée dans la Table 2. Nous présentons pour RTM le meilleur (RTM+) et le pire (RTM-) résultat obtenus avec les différentes fonctions de score utilisées. On constate que la prise en compte du contexte lors de l'entraînement des modèles offre de meilleures performances que RTM lorsqu'il s'agit de retrouver le mot à l'origine d'un lien dans les 10 mots jugés les plus probables. On note également que la version de Contextual-RTM uniforme au niveau de la phrase donne de meilleurs résultats en précision à 10 sur trois des cinq réseaux étudiés et la variante sémantique au niveau de la phrase (CRTM-Sem-S) sur deux, tout en étant la seule à se placer toujours devant RTM+. À noter que quelque soit le corpus, RTM obtient de meilleurs résultats en utilisant la fonction de lien ψ_{uni} avec une fenêtre de 0.

Corpus	**RTM**		**CRTM**			
	-	+	**Uni-0**	**Uni-S**	**Pos-S**	**Sem-S**
Physics	0.738	0.813	0.840	**0.842**	0.724	0.829
Arts	0.782	0.819	0.815	**0.833**	0.830	**0.833**
Society	0.787	0.818	0.858	**0.862**	0.844	0.822
Physique	0.796	0.820	0.833	0.826	0.837	**0.844**
Société	0.781	0.820	**0.833**	0.817	0.808	0.830

Tab. 2 : Précision moyenne sur les 10 mots les plus probables inférés par les modèles (en gras le meilleur résultat par corpus et souligné les résultats supérieurs à RTM).

Ne pouvant pas reporter les résultats pour toutes les valeurs de K dans cet article nous présentons à la Figure 2 (a) l'évolution de la précision moyenne de CRTM-Uni-S et RTM+, qui utilise pour l'évaluation la fonction de lien ψ_{uni} avec une fenêtre de 0. On remarque que les modèles arrivent tous les deux à retrouver la très grande majorité

de ces mots bien avant que le cinquantième mots soit atteint. Contextual-RTM semble néanmoins retrouver ces liens plus rapidement.



(a) Précision à K (b) Courbe ROC

FIG. 2 : (a) : Évolution des précisions moyennes à K en fonction de K entre CRTM-Uni-S et RTM sur le corpus Société. (b) : Courbes ROC moyennées entre CRTM-Uni-0 et RTM sur le corpus Society.

On présente dans le tableau 3 les aires sous la courbe ROC moyennée de chaque modèles pour les jeux de données étudiés, et à la Figure 2 (b), à titre d'exemple, les courbes ROC moyennes de RTM+ et CRTM-Uni-S.

Corpus	**RTM+**	**CRTM**			
		Uni-0	**Uni-S**	**Pos-S**	**Sem-S**
Physics	0.865	**0.874**	0.868	0.862	0.873
Arts	0.829	0.829	0.826	0.830	**0.832**
Society	0.802	**0.820**	0.811	0.812	0.817
Physique	0.832	**0.852**	0.844	0.842	0.850
Société	0.811	0.814	0.812	0.812	**0.822**

TAB. 3 : Aires sous les courbe ROC (en gras le meilleur résultat et souligné les résultats supérieurs à RTM).

On remarque que la prise en compte du contexte d'apparition des liens améliore systématiquement les performances sur la tâche. La version ne prenant en compte que les mots à l'origine des liens donne les meilleurs résultats sur trois des cinq corpus étudiés, et la version sémantique utilisant le produit scalaire normalisé sur les deux autres, tout en se plaçant seconde le reste du temps.

4.4.2 Influence de la taille de la fenêtre

Puisque Contextual-RTM comprend le contexte comme étant une fenêtre de l mots de part et d'autre du mot à l'origine d'un lien, nous étudions l'évolution de la précision à 10 pour les trois variantes de CRTM suivant neuf tailles de fenêtre, allant de zéro à l'intégralité du contenu d'un document. La Figure 3 présente l'évolution de la précision à 10 moyennée pour toutes les variantes de CRTM pour tous les corpus anglophones en fonction de la taille de la fenêtre. On remarque que les petites tailles de fenêtres tendent à donner de meilleurs résultats contrairement à un contexte de plus de 100 mots ou équivalent à un document entier. De plus on note que les meilleurs précision moyenne proviennent d'un contexte inférieur ou égale à une vingtaine de mots (une fenêtre de 10), ce qui correspond environ au nombre moyen de mots dans une phrase des corpus.

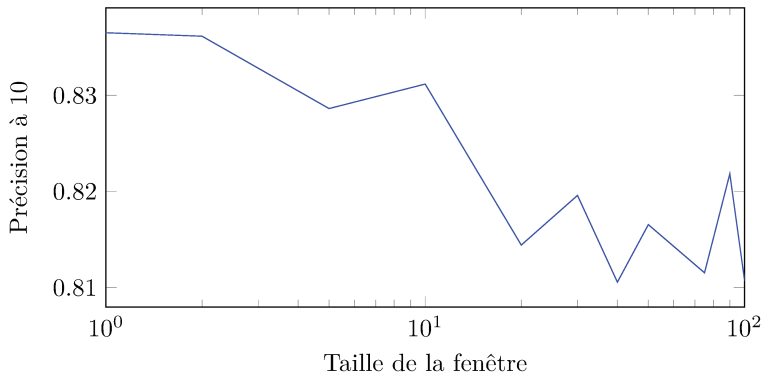


FIG. 3 : Précision à 10 moyennée pour tous les corpus anglophones pour les trois variantes de CRTM en fonction de la taille de la fenêtre (de 0, à 100 mots)

4.4.3 Qualitatif

Au delà de l'évaluation permettant de juger de la performance de ces modèles à retrouver l'origine d'un lien, nous désirons également juger de la cohérence des autres mots suggérés par les modèles.

Pour illustrer cela nous présentons à la Table 4 un exemple de parcours de pages Wikipedia. Le premier document est le paragraphe d'introduction du philosophe et mathématicien britannique Bertrand Russel. Le deuxième est celui du philosophe français Jean-Paul Sartre et le troisième celui de la page décrivant le concept de rationalité. Pour chacun des deux derniers document nous inférons avec RTM (qui utilise la fonction de lien ψ_{uni} avec une fenêtre de 0, qui donne le meilleurs résultats) et CRTM-Sem-S, la variante sémantique de CRTM au niveau des phrases, quels mots contribuent le plus aux liens entre ces documents et le premier document.

Concernant l'introduction de la page de Sartre on remarque que les deux modèles les identifient comme des *philosophes* ayant marqué la vie *intellectuelle*. Cependant

notre modèle met également en avant des liens plus ténus, comme des points de vues politiques similaires (*humanisme*) ou encore le fait qu'ils se soient vus tout deux proposer le prix Nobel de littérature, quand RTM propose des liens moins pertinents comme *prolifique* ou *idées*.

Dans le dernier document, parlant du concept de rationalité, on note une fois de plus que les deux modèles mettent en valeur la *philosophie* et le *rationalisme* comme lien entre les deux documents. RTM propose cependant d'autres termes moins pertinents comme *fonder* quand notre modèle retrouve des termes liés à Russel comme *science*, *logique*, ou *Descartes*.

Bertrand Arthur William Russell, né le 18 mai 1872 à Trellech (Monmouthshire, pays de Galles), et mort le 2 février 1970 près de Penrhyndeudraeth, au pays de Galles, est un mathématicien, logicien, philosophe, épistémologue, homme politique et moraliste britannique. […]

Jean-Paul Sartre		**Rationalité**	
CRTM-pos- S	**RTM**	**CRTM-pos-S**	**RTM**
philosophe	philosophe	philosophie	philosophie
intellectuelle	Sartre	rationalisme	sociologie
existentialisme	intellectuelle	Descartes	humain
humanisme	philosophique	science	rationalisme
littéraire	prolifique	logique	fonder
Nobel	idées	connaissance	économique

Tab. 4 : Mots proposés par CRTM-pos-S et RTM pour les pages wikipedia relatives à J-P Sartre et à la Rationalité, après lecture de celle sur Bertrand Russell.

5 Conclusion et perspectives

Nous avons présenté dans cet article Contextual-RTM, un cadre de modélisation probabiliste de thématiques latentes pour les réseaux de documents, généralisant *Relational Topic Model*, et permettant de prendre en compte le contexte d'apparition des liens de différentes façons. Nous avons montré que les trois méthodes introduites étaient compétitives dans une tache d'identification de mots à l'origine de liens. Nous avons également montré que les mots suggérés avaient tendance à être plus plausibles et cohérents. Nous prévoyons dans nos travaux futurs d'étudier l'intérêt d'adopter une forme linéaire pour la fonction ψ afin de capturer des motifs thématiques plus complexes susceptibles de mieux expliquer les liens entre les documents.

Références

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

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

Brochier, R., A. Guille, et J. Velcin (2019). Global vectors for node representations. In *The World Wide Web Conference*, pp. 2587–2593.

Brochier, R., A. Guille, et J. Velcin (2020). Inductive document network embedding with topic-word attention. In *European Conference on Information Retrieval*, pp. 326–340. Springer.

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

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

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, et L. K. Saul (1999). An introduction to variational methods for graphical models. *Machine learning 37*(2), 183–233.

Le, Q. et T. Mikolov (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pp. 1188–1196.

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

Tu, C., H. Liu, Z. Liu, et M. Sun (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)*, pp. 1722–1731.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, et I. Polosukhin (2017). Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008.

Yang, C., Z. Liu, D. Zhao, M. Sun, et E. Y. Chang (2015). Network representation learning with rich text information. In *IJCAI*, Volume 2015, pp. 2111–2117.

Zhang, F., N. J. Yuan, D. Lian, X. Xie, et W.-Y. Ma (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 353–362.

Summary

In the context of documents networks, we propose Contextual-RTM for documents contents and links between documents, by generalizing the Relation Topic Model. The originality of our approach is to model the location of a link into a document, i.e. taking in account the context where a link to another document occurs. Contextual-RTM is competitive with state of the art in retrieving link location on large networks of web documents. Consequently Contextual-RTM could assist readers making connections between documents.

4.3 Contribution 2

Anchor Prediction: A Topic Modeling Approach

Jean Dupuy
Université de Lyon, Lyon 2, ERIC
EA3083 – MeetSYS
jean.dupuy@meetsys.com

Adrien Guille
Université de Lyon, Lyon 2, ERIC
EA3083
adrien.guille@univ-lyon2.fr

Julien Jacques
Université de Lyon, Lyon 2, ERIC
EA3083
julien.jacques@univ-lyon2.fr

ABSTRACT

Networks of documents connected by hyperlinks, such as Wikipedia, are ubiquitous. Hyperlinks are inserted by the authors to enrich the text and facilitate the navigation through the network. However, authors tend to insert only a fraction of the relevant hyperlinks, mainly because this is a time consuming task. In this paper we address an annotation, which we refer to as *anchor prediction*. Even though it is conceptually close to link prediction or entity linking, it is a different task that require developing a specific method to solve it. Given a source document and a target document, this task consists in automatically identifying anchors in the source document, i.e words or terms that should carry a hyperlink pointing towards the target document. We propose a contextualized relational topic model, CRTM, that models directed links between documents as a function of the local context of the anchor in the source document and the whole content of the target document. The model can be used to predict anchors in a source document, given the target document, without relying on a dictionary of previously seen mention or title, nor any external knowledge graph. Authors can benefit from CRTM, by letting it automatically suggest hyperlinks, given a new document and the set of target document to connect to. It can also benefit to readers, by dynamically inserting hyperlinks between the documents they're reading. Experiments conducted on several Wikipedia corpora (in English, Italian and German) highlight the practical usefulness of anchor prediction and demonstrate the relevancy of our approach.

CCS CONCEPTS

• **Information systems** → **Document topic models**.

KEYWORDS

Anchor prediction, Topic modeling, Annotation, Document network

ACM Reference Format:
Jean Dupuy, Adrien Guille, and Julien Jacques. 2018. Anchor Prediction: A Topic Modeling Approach. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Wikipedia is an online and collaborative encyclopedia which is one the most visited website of the world. This attractiveness leads to the creation of nearly 600 new pages everyday in the English version of the encyclopedia[1], and other languages tend to follow this trend. In Wikipedia, articles are linked together through a network of hyperlinks. Hyperlinks allow to enrich text and facilitate horizontal reading, by giving access to additional information during the reading. Formally, a hyperlink is a directed link, from a word or term – which we refer to as *anchor* – in a source document towards a target document. Because hyperlinks are manually inserted by contributors, a time consuming task, they tend to insert few of them, leaving relevant anchors unconnected. Thus, finding ways to improve the connectivity between pages by automatically inserting hyperlinks in the pages would facilitate the access to information and knowledge. Finally, recommending anchors from a document to another specified one allow contextual hyperlink generation, which can be used by the reader to track terms relative to a specific concept during its navigation though Wikipedia.

Problem definition. Given a source document and a target document, we address the issue of automatically identifying potential anchors in the source document, given a target document. We refer to this task as *Anchor Prediction*. It is different from link prediction, where the target document isn't given, and where the locations of the anchors don't matter. It also differs from entity-linking where targets aren't documents but entries of an ontology. Finally anchor prediction is close to an annotation task, but unlike the latter it aims to identify words carrying hyperlinks between a pair of specifics document, rather than looking for probables mentions of an entire set of entities.

Use cases. Predicting anchors can be beneficial for both contributors and readers. Having wrote a new document (i.e. the source), rather than directly inserting hyperlinks, a contributor could simply specify a set of target documents, and let the system automatically locate the anchors and thus insert the hyperlinks. This set of document could also be build with any recommendation method. During a reading session, contextual hyperlinks could be automatically inserted in the document being read (i.e. the source document), pointing towards previously read documents (i.e. targets), thus helping the users contextualizing their reading.

Related work. A lot of attention has been devoted to developing means to infer latent links between documents [6, 8, 27, 42]. Prior work on this issue views a link as a connection between two entire documents, following the more general definition of link prediction in simple networks [28]. In other words, most of existing methods

[1] https://en.wikipedia.org/wiki/Wikipedia:Statistics

– including the most recent ones [9, 21, 46], with the notable exception of [7] – ignore the locations of the links, i.e. anchors, in the documents and cannot trivially be adapted to solve the anchor prediction task. Yet, they provide grounds to develop new methods suited to anchor prediction.

Other settings like automatic annotation of documents have been also studied in the last decades. An annotation method aims to find in a text the most relevant mentions of documents that belong to a predefined corpus (Wikipedia for the large majority), and returns pairs of anchors and linked documents. Those systems are build upon a wide range of machine learning methods, including statistical analysis of existing anchors [14, 34], topic modeling [20] or deep learning [15, 26]. However those methods are mostly designed to identify anchors pointing to another documents by relying on dictionaries of previously seen mentions and documents title, or knowledge graphs or onthologies.

Proposal. In this paper, we propose a new method based on a contextualized relational topic model, which we refer to as *CRTM*. We generalize the Relational Topic Model, RTM [10], itself based upon the well-known Latent Dirichlet Allocation (LDA) topic model [5]. To incorporate the network structure in LDA, RTM models undirected links between documents as a function of the topics assigned to all the words in each document. In contrast, *CRTM* models directed links between documents as a function of the topics assigned to the words surrounding the anchor (*i.e.* its context) in the source document and the topics assigned to all the words in the target document. We further improve the link function by (i) incorporating an attention mechanism to better estimate the importance of each topic for anchor prediction and (ii) incorporating additional parameters to learn new hidden representations more fitted to anchor prediction. CRTM is computationally light, language agnostic and is solely trained on texts and existing anchors, without any use of external knowledge bases nor documents titles.

Results. We assess the ability of CRTM to correctly predict anchors on several corpora made of English, Italian and German Wikipedia articles. We compare its performance with those of other variants of RTM to highlight its relevancy, and a widely used annotation tool. We also conduct an ablation study, which reveals that both our improvements to the link function are crucial to the performance of CRTM. To connect our work with prior work on link prediction, we provide additional results on this task, incorporating recent baselines. We show that, even though not initially designed to solve this task, CRTM still manages to beat some baselines, being beaten only by the most recent and specialized methods for link prediction. To conclude this paper, we show how to use CRTM in practice to predict anchors. We provide examples that show that CRTM can help in automatically detecting words or terms that should carry hyperlinks.

2 RELATED WORK

In this section we discuss the different parts of the literature we lean on. Firstly, because CRTM is a topic model, we start by reviewing prior work on topic modeling. Next, we briefly survey work on link prediction, a close task to anchor prediction. Lastly, we discuss work that leverage anchor links in other settings.

2.1 Topic Modeling

We can classify topic models in two categories: LDA-based models and neural topic models.

2.1.1 LDA-based Topic Models. These methods model documents as topic mixtures, each word in a document being assigned to one topic. LDA [5] is a generalization of pLSA [23]. It is a generative model to infer latent topics to describe a set of documents D. For each document $d \in D$, LDA infers a vector θ_d of proportions over a small number K of latent topics. The topics are characterized by a distribution β over a fixed vocabulary V. Each word occurrence $w_{d,i}$ in a document d is independently assigned a topic, encoded in a one-hot manner as $z_{d,i} \in \{0;1\}^K$, according to the distribution over words for this topic, $\beta_{\bullet,z_{d,i}}$. The Hidden Topic Markov Model (HTMM) [19] models the topics of all the words in a same document as a Markov Chain, effectively lifting the independence assumption between neighboring word occurrences. More precisely HTMM expects that all words in a same sentence share the same topic, and that consecutive sentences are more likely to have the same topic. Biterm Topic Model (BTM) [41] proposes to address a shortcoming of LDA, dealing with short texts, by modeling the generation of word co-occurrence patterns. By taking advantage of co-occurrence at the corpus-level, BTM solves the sparsity problem at the document level, which makes it suitable for short texts.

2.1.2 Neural Topic Models. With the advent of deep generative models and variational auto-encoders [24], topic modeling has been cast into the deep learning framework. NTM-R [13] relies on neural variational inference and pre-trained word embeddings, and is trained with an objective towards topic coherence. W-LDA [33] proposes to use Wasserstein auto-encoders [37] to address the problem of topic disentanglement described in [3], and enforces the Dirichlet prior. GraphBTM [48], based on the same assumption than BTM, represents biterms as a graph and uses a Graph Neural Network (GCN) [25] to extract topic mixtures from them. [47] proposes the Graph Topic Model (GTM), which extracts topic mixtures by processing a word-document graph with a GNN, to further improve topic coherence. Note that the word-document graph is distinct from an explicit document-document graph, as it solely computed from word occurrences in the documents.

It should be noted that neural topic models, such as NTM-R, W-LDA, GraphBTM or GTM, only compute topic mixtures and do not explicitly assign a topic to each word occurrence, as opposed to LDA-based models.

2.2 Link Prediction

Link prediction is a related task to the one we propose. We classify existing approaches in two categories: approaches based on topic modeling, and approaches based on node embedding.

2.2.1 Methods based on Topic Modeling. The Relation Topic Model (RTM) [10] is an extension of LDA that incorporates explicit links between documents, so that it is suitable for link prediction. Here, we quickly develop the internal functioning of this model, since CRTM generalizes it. Given a set L of explicit links between the documents in a corpus D, RTM extends LDA with the addition of a link function that models the likelihood of the links between two documents in terms of their topic proportions. This function,

$\psi(y_{d,d'} = 1)$, with $y_{d,d'} \in \{0;1\}$ a binary variable that indicates whether d and d' are linked, parametrized by $\eta, \nu \in \mathbb{R}^K$, is defined as:

$$\psi_{(d,d')}(y = 1) = \exp\left(\eta^\top (\overline{\mathbf{z}}_d \circ \overline{\mathbf{z}}_{d'}) + \nu\right), \quad (1)$$

where $\circ$ denotes the Hadamard product and $\overline{\mathbf{z}}_d = \frac{1}{N_d}\sum_i z_{d,i}$ and $\overline{\mathbf{z}}_{d'} = \frac{1}{N_d}\sum_i z_{d',i}$ represents the proportions of latent topics in document d and d' respectively. Overall, the generative process of RTM goes as follows:

- For each document d in D:
  - draw topic proportion for $d : \theta_d|\alpha \sim Dirichlet(\alpha)$
  - for each word occurrence $w_{d,i}$ in d:
    - draw topic assignment : $z_{d,i}|\theta_d \sim Multi(\theta_d)$
    - draw word : $w_{d,i}|z_{d,i} \sim Multi(\beta_{z_{d,i}})$
- For each pair of documents $(d, d') \in L$:
  - draw link indicator : $y_{d,n} \sim \psi(\cdot|\eta, \nu, z_d, z_{d'})$

$Multi(.)$ is the multinomial distribution, α the Dirichlet parameter, z_d a matrix whose rows are the $z_{d,n}$ vectors.

RTM has been modified or extended in different ways. gRTM [12] proposes to capture all pairwise topic relationships between documents and relies on a different estimation procedure based on collapsed Gibbs sampling with data augmentation. [45] proposes a non-probabilistic formulation of RTM to control the sparsity of document's topics. [44] incorporates in RTM the weighted stochastic block model [1] to identify blocks of strongly connected documents. [43] proposes a joint model that uses link structure to define clusters of documents.RTM has also triggered works in the field of Neural Topic Modeling. For instance, Relational Deep Learning (RDL) [39] is a deep hierarchical Bayesian model designed for link prediction. [2] proposes Neural Relational Topic Model (NRTM) to mimic the RTM architecture with components from deep learning. For the LDA part NRTM uses a stacked variational auto-encoder and a multi-layer perceptron to model the link function.

It should be noted that, as previously stated, neural approaches don't explicitly assign a topic to each word occurrence in a document. Thus they aren't naturally suited to anchor prediction, since they don't allow to straightforwardly extract local topical contexts.

2.2.2 Node embedding with textual information. These methods learn document embeddings in a Euclidean space and then straightforwardly predict links according to the dot-product between these embeddings. Several neural architectures have been explored to learn the embeddings from the documents' content and the network structure, such as Graph2Gauss [6], a deep energy-based encoder embedding each nodes as a Gaussian distribution, STNE, a self-translating recurrent network [27] or IDNE, an attention-based network [9]. Methods based on matrix-factorization have also been investigated, like TADW [42], an extension of the matrix formulation of DeepWalk that leverages both the network structure and the content of the documents. Similarly, GVNR-t [8], adapts the GloVe algorithm to embed networks of documents. Some methods, like RLE [16], aims at projecting linked documents in a pre-trained word embedding space.

It should be noted that all these approaches model links at the document-level, ignoring anchors (i.e. the position of the hyperlinks in the source document).

2.3 Leveraging anchors

In this last section, we discuss works that incorporate anchors to solve annotation tasks.

While some topic models take advantage of the anchor information [17, 18, 35], those methods are not suitable for a anchor prediction task. However many methods are designed to annotate documents by identifying portions of text related to another document. The first method to address this task WIKIFY [31]. TAGME [14] is an efficient method designed to annotate shorts texts with Wikipedia entries, combining three steps. For a given text, TAGME first looks for all mentions of entries in a dictionary containing titles of Wikipedia pages and anchor texts. Then an entity disambiguation step is performed using a voting scheme. The score of each mention-entity pair is computed as the sum of votes given by candidate entities of all other mentions in the text, taking into account the semantic association between two entities[30] and the probability of an entity being the link target of a given mention[29]. This step return on anchor per mention. Finally all the candidates below a certain threshold of coherence are pruned. WAT [34] is another method that extends TAGME by redefining the disambiguation step using graph-based methodologies [22]. [36] is one of the first deep-learning based method designed to deal with this task, by leveraging the semantics of mention, context and entity as well as their compositionality in a unified way. BLINK [26] is another deep learning based method using a two stages approach for entity linking and annotation, based on fine-tuned BERT architectures.

3 CONTEXTUALIZED RELATIONAL TOPIC MODEL

In this section, we describe our proposal: the Contextualized Relational Topic Model, which we refer to as CRTM. It generalizes RTM so that links are modeled as a function of the topics assigned to the words in the context of the anchor in the source document and the topics assigned to all the words in the target document. The context is a set of words that surround the anchor in the text. It can be rather narrow, *e.g.* only the anchor word or the sentence in which it appears, or wider, up to the all the words in the document. In the latter case, CRTM is equivalent to RTM. We further refine the link function by (i) incorporating an attention mechanism and (ii) introducing additional parameters.

3.1 The CRTM Model

Let d and d' be two documents, and $c = \{w_i\}_{i=1}^C$ be the context of the link, *i.e.* a set of $C \geq 1$ words, in the source document d. The link function of *CRTM* is defined as:

$$\psi_{d,d'}(y = 1; c) = \exp\left(\eta^\top \left(Q\overline{\mathbf{z}}_{d,c} \circ Q\overline{\mathbf{z}}_{d'}\right) + \nu\right), \quad (2)$$

where $\overline{\mathbf{z}}_{d,c}$ is an attention-based weighted average of the per-word topic assignments in c and $Q \in \mathbb{R}^{K\times K}$ is a set of additional parameters. We describe these two components in the following subsections.

3.1.1 Attention-based Weighted Average. Rather than calculating $\overline{\mathbf{z}}_{d,c}$ as a simple average, we rely on pre-trained word embeddings to calculate a weighted average of the topic assignments in c using an attention mechanism. Assuming a word embedding $u_i \in \mathbb{R}^p$ for

each word $w_i \in V$, we compute the attention scores $s \in \mathbb{R}^k$, by measuring the scaled dot-product [38] between the embedding of the word carrying the link, u_{link} and the embedding u_j of each word w_j in c:

$$s_j = \frac{u_{\text{link}} \cdot u_j}{\sqrt{p}}. \tag{3}$$

With the attention weights $a = \text{softmax}(s)$, we calculate the weighted average:

$$\overline{z}_{d,c} = \sum_{w_j \in c} a_j z_{d,j}, \tag{4}$$

where $z_{d,j}$ is the one-hot encoding of the topic assignment for the word w_j in context c. This way, the topics assigned to words that are semantically closer to the word that carries the link are given more importance.

3.1.2 Additional Parameters. The form of the link function in RTM implies that linked documents should have similar topic proportions. We argue that this implicit assumption is too restricting as complementary documents that exhibit different topic proportions could be linked. Hence, we incorporate additional parameters $Q \in \mathbb{R}^{K \times K}$ so that CRTM learns new representations of the documents d and d' as linear transforms of $\overline{z}_{d,c}$ and $\overline{z}_{d'}$: *i.e.* $Q\overline{z}_{d,c}$ and $Q\overline{z}_{d'}$. This way, RTM can compare d and d' beyond simple same-topic interactions. Note that the size of Q is dictated by the estimation procedure, because it requires η to be a K-dimensional vector, which in turns enforces $Q\overline{z}_{d,c}$ and $Q\overline{z}_{d'}$ to be K-dimensional vectors.

3.2 Parameter Estimation

We estimate the parameters adapting the procedure in RTM, by maximizing the likelihood using the variational Expectation-Maximization algorithm [4]. The addition of Q in ψ slightly modifies the expression of the ELBO and therefore the update rules of the variational parameters of θ_d, and parameters for ψ: η and ν. The expected value of the link function becomes:

$$\mathbb{E}_q\left[\log p\left(y_{d,d'} = 1 \mid \overline{z}_d, \overline{z}_{d'}, \boldsymbol{\eta}\right)\right] = \boldsymbol{\eta}^T\left(Q\overline{\phi}_{d,c} \circ Q\overline{\phi}_{d'}\right) + \nu,$$

where $\phi_{d,n}$ is the variational parameter of $z_{d,n}$ and $\overline{\phi}_d = \frac{1}{N_d}\sum_n \phi_{d,n}$.

Parameters γ_d and $\phi_{d,i}$, respectively the variational parameters of θ_d and $z_{d,i}$, are updated at the E-step as follow:

$$\gamma_d = \alpha + \sum_i \phi_{d,i}, \tag{5}$$

and

$$\phi_{d,i} \propto \exp\left(\log \beta_{\cdot,w_{d,i}} + \Psi(\gamma_d) - \mathbf{1}\Psi(\mathbf{1}^T\gamma_d)) + \sum_{d' \neq d} \eta \circ \frac{Q^2\overline{\phi}_{d,i}}{N_d}\right), \tag{6}$$

with $\Psi(.)$ the Digamma function, 1 a K-dimensional vector of ones, and N_d the number of words in document d.

During the M-step we update the model's parameters β, η, ν and Q, according to the new values of γ_d and $\phi_{d,i}$. The update rule for β didn't change from LDA:

$$\beta_{k,w} \propto \sum_d \sum_n \mathbf{1}(w_{d,n} = w)\phi^k_{d,n}. \tag{7}$$

The link function's parameters η and ν are updated as follow:

$$\eta \leftarrow \log\left(\overline{\Pi}\right) - \log\left(\overline{\Pi} + \rho\overline{\pi}_\alpha\right) - \mathbf{1}\nu, \tag{8}$$

and

$$\nu \leftarrow \log\left(L - \mathbf{1}^T\overline{\Pi}\right) - \log\left(\rho(1 - \mathbf{1}^T\overline{\pi}_\alpha) + L - \mathbf{1}^T\overline{\Pi}\right), \tag{9}$$

with $\overline{\Pi} = \sum_{(d,d')} Q\overline{\phi_d} \circ Q\overline{\phi_{d'}}$, $\overline{\pi}_\alpha = Q\frac{\alpha}{\mathbf{1}^T\alpha} \circ Q\frac{\alpha}{\mathbf{1}^T\alpha}$ and L the total number of observed links. ρ denote the number of negative examples needed, following the regularization procedure provided by [11]. Instead of using a $L2$ regularizer, we introduce negative observations, where $y = 0$. The observations are associated with a similarity $\overline{\pi}_\alpha$, the expected Hadamard product of any two documents given the Dirichlet prior of the model.

Finally the coefficients of Q are updated with

$$Q_{i,j} \leftarrow Q_{i,j} + l \times \sum_{(d,d' \in L)} \frac{\eta_i}{K}\left(\overline{\phi}_{d,j}\sum_{n=1}^{K} Q_{i,n}\overline{\phi}_{d',n} + \overline{\phi}_{d',j}\sum_{n=1}^{K} Q_{i,n}\overline{\phi}_{d,n}\right) \tag{10}$$

, where l is a learning rate. Note that updates to Q are strictly positive, which could cause numerical instability. While we could constrain Q to prevent its norm from monotonically increasing and rely on the Lagrange multiplier to estimate it, we choose to address this issue differently and simply normalize each row of Q with its $L2$-norm after each update. This tricks has been shown to work well in [40] for instance.

3.3 Time Complexity

Estimating the parameters of the LDA component of CRTM has a time complexity of $O(|D| \times K \times N)$, with $|D|$ the number of documents, K the number of topics and N the average number of words per document. Estimating the parameters of the relational component of CRTM has a complexity of $O(L \times K^2)$, where L is the number of observed links in the network, and K^2 is engendered by the addition of Q in the link function. The proposed model thus have an overall $O(|D| \times K \times N + L \times K^2)$ time complexity.

3.4 Relationship to gRTM

The link function of gRTM [12] incorporates a matrix $Q' \in \mathbb{R}^{K \times K}$, as follow:

$$\psi_{d,d'}(y = 1) = \sigma\left(\overline{z}_d Q' \overline{z}_{d'}\right), \tag{11}$$

where σ can be either the sigmoid function or the exponential function, as in RTM. This formulation arises from the fact that $\eta^\top(\overline{z}_d \circ \overline{z}_{d'}) = \overline{z}_d \text{diag}(\eta)\overline{z}_{d'}$. Rather than enforcing Q' to be a diagonal matrix to stick to RTM, gRTM considers Q' as a full matrix that allows capturing inter-topics interactions. This is quite different from what *CRTM* does, as the purpose of Q isn't to replace η but rather to project both $\overline{z}_{d,c}$ and $\overline{z}_{d'}$ to a new comparison space to improve link prediction.

4 EXPERIMENTS

First, we quantitatively evaluate CRTM against RTM, TAGME and variants of RTM on the task of anchor prediction. Then we highlight the importance of contextualizing the link function, and show that both refinements we bring to the function are crucial to the performance of CRTM. Then we evaluate CRTM on a link prediction task against RTM and recent baselines. This experiment is less a way to evaluate if our model can reach state of the art results, since CRTM is designed for anchor prediction, than assess if our modifications

leads to a weaker generalization. Lastly, we illustrate how to use our model for anchor prediction in practice.

4.1 Datasets

To demonstrate the relevancy of our proposal we construct six datasets using Wikipedia. Each dataset is made of introductory sections of articles belonging to a specific category. Articles are sampled in a snow ball fashion following the links, starting from the main page of the category. Furthermore, since we have to construct our own datasets, we can easily evaluate our work on different languages, while commonly used datasets tend to be in only in English. We consider three languages: English, Italian and German. For each language we extract a dataset related to the main category about physics (*Physics, Fisica* and *Physik*), and society (*Society, Società* and *Gesellschaft*). We choose these categories to take into account different editorial policies, writing styles and vocabularies. Table 1 summarizes the general properties of these datasets.

4.2 Tasks and metrics

We consider two tasks. The first is an anchor prediction task, and consists in locating the words carrying hyperlinks, hidden during training, given a link between a source and a target documents. Based on the scores given by $\psi_{d,d'}$, we:

- Calculate the precision at n, *i.e.* the fraction of hidden links for which the word carrying the hyperlink is ranked among the top n words.
- Construct the ROC curve, to visually assess the overall capacity of the models to give high ranks to the word carrying the hidden hyperlinks.

The second is a link prediction task. We hide a percent of edges and compare the cosine similarity between hidden pairs and negative examples of unconnected documents. We report Area Under the ROC Curve.

4.3 Methodology

4.3.1 Anchor Prediction. We hide one link per document, which represents between 21% to 33% of the links depending on the connectivity of the datasets. The set of hidden links is split into a a validation set, used to control the convergence during the training phase, and a test set, used to calculate the precision and construct the ROC curves. We compare CRTM, with the context matching the sentence in which the link occurs, against *ad hoc* variants of RTM. gRTM was considered then discarded because of its prohibitive runtime on large corpora.

- **CRTM**: We define the context as the sentence in which the hyperlink occurs. After gridsearching we set the learning rate l to 0.01 in the update rule for Q, which yield the best results. Word embeddings are trained independently on each dataset using Skip-Gram with negative sampling (window size of 10 and 20 negative samples per true sample) [32]. By training one word-embedding set per corpora, we did not introduce external knowledge to our model and remain fair to the other baselines.
- **RTM**: We train the model following [10], with the original link function in Eq. 1. However this link function doesn't allow ranking the words because it is independent to the link's location. Hence, during the test phase, we change RTM's link function to $\psi_{(d,d')}(y = 1; w_i) = \exp(\eta^\top (z_{d,i} \circ \overline{\mathbf{z}}_{d'}) + \nu)$, *i.e.* we replace $\overline{\mathbf{z}}_{d'}$ with $z_{d,i}$, the topic assignment for word i. This trick allows us to measure the probability of each word w_i to carry the link. Note that we are limited to $z_{d,i}$, as considering larger context at test time would require modifications akin those proposed in CRTM.
- **TAGME**: In order to evaluate TAGME we use the annotation service provided by the D4Science's API[2]. For each hidden hyperlink we report if TAGME identify the right linked document and the corresponding anchor. As TAGME only output one result per linked document because of its disambiguation process, only P@1 is reported. We set the confidence score threshold at 0.1, discarding all results below this value. This value is the smaller threshold recommended by the documentation.

We also consider simpler variants of CRTM for the purpose of the ablation study:

- **CRTM**$_1$ (no context): We consider singleton contexts that consist only contain the word carrying the link.
- **CRTM**$_U$ (uniform weighting): We match the contexts with sentences and remove the attention mechanism by directly calculating $\overline{\mathbf{z}}_{d,c}$ as a simple average. Because this link function is doesn't take the exact location of the word into account, we again use the trick that consists in restricting the context to a single word at test time.
- **CRTM**$_P$ (positional weighting): We match the contexts with sentences and replace the attention weights with weights calculated according to a Gaussian smoothing centered on the position of the word carrying the link. The weight for a word w_i in the context is calculated as $g_i = e^{-\frac{1}{2}\left(\frac{\text{dist}(i)}{\sigma}\right)^2}$, where $\text{dist}(i)$ is the distance between w_i and w_{link}, the word carrying the link. σ is the standard deviation. We set it to $\sigma = 3$ by gridsearch.
- **CRTM**$_I$ (no representation learning): We match the contexts with sentences and remove the additional parameters by setting $Q = I_K$, with I_K the identity matrix.

All models are trained with 50 topics, which we've found to give the best performance for [10] and CRTM. We set the hyperparameter α to 5.0 for all models which leads to consistently good performances for *CRTM* and RTM (identical to the value chosen by [10] in their evaluations), and set $\rho = 2000$ negative examples for the regularization in η and ν estimation.

4.3.2 Link Prediction. We hide 10% of edges before training models. For the four baselines learning documents embeddings, we compare the cosine similarity between hidden pairs and 10 negative examples of unconnected documents. For both RTM and CRTM we use the link function defined at Equation 1 and compare the probability of the true link and 10 negative examples. Because CRTM link function doesn't take into account all the content of the source document, Equation 2 is not suitable for link prediction. Switching

[2]https://sobigdata.d4science.org/web/tagme/tagme-help

Table 1: Dataset properties.

Category	*Physics*	*Society*	*Fisica*	*Società*	*Physik*	*Gesellschaft*
Language	English	English	Italian	Italian	German	German
Nb of pages	6327	14486	2254	3106	3438	12262
Nb of links	18821	30749	7145	7093	10727	35710
Average number of words per doc	183	194	121	130	120	111
Average number of sentences per doc	9.77	10.05	5.79	5.94	8.69	8.12
Average degree	5.9	4.2	6.3	4.5	6.2	5.8

from Equation 2 to 1 doesn't use the parameter Q learned by CRTM and this modification will be discussed in Section 4.5.

We compare CRTM to RTM and 4 recent document network embedding methods specifically designed for link prediction:

- **CRTM**: The context is still defined as the sentence in which the anchor link occurs and we use the same parameters and word-embeddings used for the previous evaluation. We use Equation 1 for computing the probability of two documents being linked.
- **RTM**: We use the parameters previously described for anchor prediction.
- **RLE**: The parameter λ is set to 0.7, and RLE outputs document embeddings in dimension $d = 160$. The word embeddings are obtained with Skip-Gram with negative sampling, using a window of 15 words and 5 negative examples per true sample, as described in [16].
- **GELD**: We fix $\alpha = 0.85$ after gridsearching, and run the model for 40 epochs. GELD outputs vectors in dimension $d = 160$. The word embeddings are obtained with Skip-Gram with negative sampling, using a window of 15 words and 5 negative examples per true sample.
- **TADW**: Following [42], we reduce the dimension of word vectors to 200 via SVD decomposition of the TFIDF matrix, select $k = 80$ and $\lambda = 0.2$. We run TADW for 20 epochs.
- **GVNRt**: We run 80 random walks per node of length 40. We fix the window size at 5. GVNRt run for 20 epochs and document embeddings are obtained by concatenating matrix I and J.

4.4 Anchor Prediction

Table 2 reports the average precision at 1 and the precision at 5 for all models over 10 runs, for all five datasets. Figure 1 shows averaged ROC curves for CRTM and RTM on each datasets.

4.4.1 Comparison of CRTM with RTM. *CRTM* outperforms RTM by a large margin on all datasets. In particular, CRTM exceeds 0.5 in precision at 1 on all English and German datasets, when RTM only score 0.41 in the most favorable case. This superiority of CRTM is also illustrated by the ROC curves on Figure 1. *CRTM* also outperforms RTM in the precision at 1 on the two Italian corpora, but performs equally as good as RTM in the precision at 5. This can be explained in part by the fact that, even though Italian and English documents have the same number of average number words per sentence (about 21.5 in Italian and 19.7 in English), the Italian corpora are smaller, and Italian documents are shorter but with a greater average degree. This may degrade CRTM's ability to generalize over anchor modelization. A similar explication could be given for the results in the German corpora. German and English datasets tends to have a similar average number of sentences per document, but German sentences are much shorter on our datasets. Furthermore the German language uses declensions and much more surcomposition than English in its vocabulary which could lead to more unique words, and so a weaker topic assignation of those words.

Overall, CRTM manages to rank the words carrying the hidden links higher than RTM, as evidenced by the ROC curves in Figure 1.

4.4.2 Comparison of CRTM and TAGME. . TAGME outperforms CRTM in terms of P@1 on the two German corpora, and on the *Physics* one. However CRTM exceeds its competitor on the *Society* by 0.02 in precision at 1. For the two Italian datasets TAGME still achieves better performance on P@1, but close to CRTM's one on the *Società* dataset. Yet those results may be tempered by the fundamental differences between CTRM and TAGME. As TAGME relies on a predefined dictionary made of previously observed anchors and documents titles, the annotation process is deterministic and TAGME can't infer unseen anchors. On the other hand CRTM aims to infer anchors depending on their topic and context. This means that top words predicted by CTRM are not necessarily inappropriate, even though they didn't match the exact anchor of the removed hyperlink. By saying this we could mitigate those results and advance that CRTM's P@5 performance, exceeding by a large margin TAGME's P@1 on all datasets, can lead to a better diversity of links. We further illustrate this point in Section 4.6.

4.4.3 Execution time. We report in Table 3 the average wall-clock execution time for all models, across all datasets. All models are coded in Python 3 and trained using a single core on a computer with an *Intel i7-6700K* CPU and 64GiB of RAM. We notice that all the average runtimes are in $O(10^2)$ seconds, including the time required to train the word embeddings in *CRTM*. Interestingly, while an iteration takes longer to compute in *CRTM*, more than twice the time needed for RTM, CRTM needs fewer iterations to converge and thus ends up running faster than RTM.

4.4.4 Ablation Study. In this section, we study the performance of CRTM and its variants to show the importance of each of its components.

Impact of the attention mechanism. To judge the impact that the attention mechanism has on the performance of CRTM, we must compare it with these of CRTM_1 (no context), CRTM_U (simple

Table 2: Average P@1 and P@5 per corpus for CRTM and baseline (standard deviation in parentheses).

Corpus	*Physics*		*Society*		*Fisica*		*Società*		*Physik*		*Gesellschaft*	
	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5
CRTM	.65 (.02)	**.86** (.02)	**.65** (.03)	**.88** (.01)	.45 (.04)	**.76** (.03)	.38 (.02)	**.69** (.01)	.50 (.02)	**.81** (.001)	.53 (.02)	**.81** (.01)
RTM	.34 (.03)	.71 (.03)	.32 (.02)	.69 (.02)	.33 (.02)	.72 (.02)	.31 (.02)	.68 (.02)	.39 (.03)	.75 (.02)	.41 (.02)	.75 (.01)
TagMe	**.71** (.00)	-	.63 (.00)	-	**.51** (.01)	-	**.40** (.01)	-	**.65** (.01)	-	**.60** (.01)	-
CRTM$_1$	.37 (.02)	.78 (.02)	.36 (.02)	.76 (.01)	.33 (.03)	.73 (.02)	.33 (.02)	.67 (.03)	.39 (.01)	.74 (.01)	.41 (.02)	.76 (.01)
CRTM$_U$	.37 (.02)	.76 (.02)	.36 (.02)	.74 (.01)	.32 (.02)	.70 (.02)	.33 (.02)	**.69** (.01)	.39 (.02)	.76 (.02)	.42 (.01)	.76 (.01)
CRTM$_P$	.37 (.01)	.78 (.01)	.36 (.01)	.75 (.02)	.34 (.02)	.72 (.01)	.33 (.02)	.67 (.03)	.39 (.03)	.75 (.02)	.41 (.01)	76 (.01)
CRTM$_I$	.37 (.02)	.76 (.02)	.35 (.01)	.74 (.02)	.33 (.03)	.72 (.02)	.33 (.02)	.67 (.03)	.38 (.005)	.74 (.003)	.42 (.01)	.77 (.01)

Table 3: Average wall-clock runtime in seconds (time to train the embeddings in parentheses when applicable).

	Average runtime (s)	Average runtime/iter (s)
CRTM	$2.2 \times 10^2 (+10^2)$	28.4
RTM	4.6×10^2	11.4
CRTM$_1$	14.3×10^2	12.2
CRTM$_U$	6.1×10^2	18.8
CRTM$_P$	6.4×10^2	18.2
CRTM$_I$	$5.1 \times 10^2 (+10^2)$	12.73

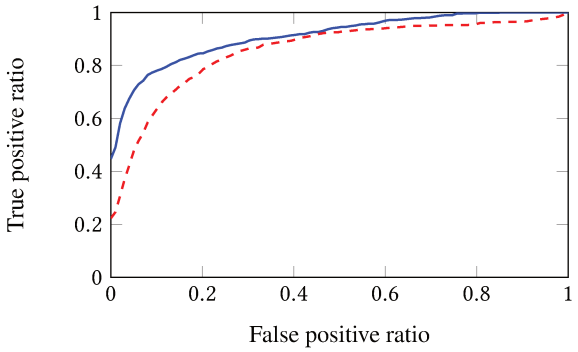
average) and CRTM$_P$ (Gaussian smoothing around the word's position). CRTM$_1$ has slightly better performance than RTM and is largely outperformed by CRTM. This tends to show that restricting the context to the sole word carrying the link is often too drastic, which can also explain why RTM struggle to match the performance of CRTM. CRTM$_U$ manages to equal CRTM on the *Società* dataset in precision at 5. However it is consistently outperformed by CRTM in all the other settings, and in particular in precision at 1. This suggests that simply averaging the per-word topic assignments is sub-optimal. On the other hand, CRTM$_P$ manages to improve over RTM, with a relative gain of up to 7% in the most favorable cases. This seems to indicate that the topic assignments of the words closest to the one carrying the hyperlink are more important. Yet, CRTM$_P$ is still outperformed by CRTM in the precision at 1 and 5 on all datasets, which shows that the attention mechanism is an important component of CRTM.

Impact of Q. Here we compare the performance of CRTM with the performance of CRTM$_I$ (where $Q = I_K$). CRTM consistently outperformed CRTM$_I$ in precision at 1 and 5 on the six considered corpora. We can interpret that by saying that learning new representations is beneficial.

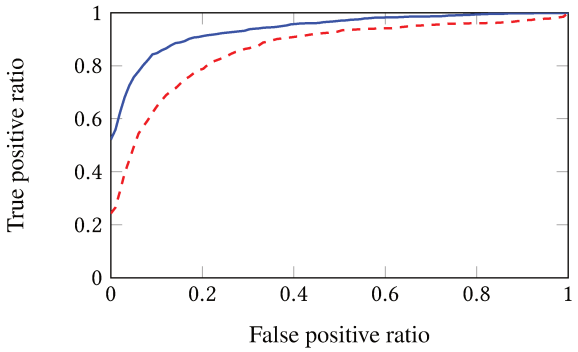
4.5 Link Prediction

Table 4 reports the AUC scores for all datasets. We run each model five times, on each dataset, and report mean AUC with standard deviation.

Results analysis. CRTM beats TADW by a clear margin on five of six datasets. CRTM's performance matches RTM or slightly improves upon it, while coming close to RLE and GELD on English datasets. GVNR-t clearly performs better than all methods. Still, we



(a) *Physics*



(b) *Society*

Figure 1: Averaged ROC curves of CRTM (blue solid line) and RTM (red dashed line) on the two English corpora.

observe that CRTM achieves consistent performances, with at least 0.80 AUC in all cases. It is worth noting that GVNR-t has a training time of several hours on our datasets (due to the computational costs of random walks), while CRTM is trained in less than 20 minutes on the same computer. This experiment shows that taking into account the context of anchor links at least doesn't degrade CRTM results in link prediction w.r.t RTM, and leads in some cases to a minor performance boost.

Impact of Q. As mention in section 4.3.2, CRTM's link function is not suitable for link prediction as it requires knowing the anchor.

Table 4: AUC for link prediction (standard deviation in parentheses)

	Physics	*Society*	*Fisica*	*Società*	*Physik*	*Gesellschaft*
CRTM	.87 (.004)	.87 (.003)	.81 (.01)	.78 (.02)	.82 (.002)	.80 (.01)
RTM	.85 (.02)	.86 (.01)	.80 (.01)	.78 (.03)	.82 (.005)	.78 (.01)
RLE	.89 (.002)	.91 (.003)	.90 (.002)	.89 (.001)	.86 (.002)	.88 (.001)
GELD	.89 (.001)	.91 (.001)	.89 (.001)	.88 (.002)	.87 (.001)	.85 (.001)
TADW	.83 (.003)	.82 (.003)	.80 (.005)	.73 (.001)	.67 (.003)	.65 (.004)
GVNR-t	.97 (.03)	.96 (.04)	.95 (.001)	.95 (.04)	.96 (.03)	.96 (.04)

Therefore once we've trained CRTM we substitute its link function with the simpler RTM's link function to do link prediction, thus omitting the transformation Q. We further investigate the impact of this parameter by introducing the transformation Q in the link function of RTM. The overall average AUC in link prediction on the six datasets falls to 0.63, while it is about 0.80 without using the parameter Q. This result, coupled with Section 4.4.4's conclusion, suggest that (i) CRTM learns topics that are useful for link prediction and that (ii) Q confers it the ability to efficiently recombine topics so that they are suited to anchor prediction. This means that a single CRTM model could solve both the link prediction (even though it doesn't match the performance of the most up-to-date techniques) and anchor prediction tasks, by simply changing the link function at prediction time.

4.6 Case Study

Here we show the anchors that CRTM is able to automatically detect, given a pair of source and target documents. To this end, CRTM was trained with all hyperlinks removed in the source documents, to prevent it from simply listing anchors seen during training. These anchors could be used by writers, to automatically insert hyperlinks towards related pages. They could also be useful for readers, as they could serve as contextual hyperlinks, linking the page they are reading with those they've previously read. We also report the anchors found by RTM, to highlight how those found by CRTM are more relevant.

As an example, Table 5 shows the five most likely anchors, given the page about "Semiconductor" as the source, and the page about "Transistor" as the target, and Table 6 shows the five most likely anchors, given the page about "Computer" as the source, and the page about "Transistor" as the target. We note that CRTM always rank the word transistor first, the most natural word to be an anchor. RTM actually manages to identify it as an important word too, but ranks it lower. CRTM also ranks MOSFET at the second or third place, which is a type of transistor. In addition, we observe that RTM predicts some less interesting words, such as *loom*, *fabricated* or *enable*, while CRTM highlights relevant but less obvious terms, like *Moore* (from the Moore's law).

Table 5: Five most likely anchors in the "Semiconductor" page, connected to the "Transistor" page.

Transistor, A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. [. . .]

↑

Semiconductor

CRTM	RTM
transistor	loom
MOSFET	MOSFET
semiconductor	transistor
circuit	enable
Moore	circuit

Table 6: Five most likely anchors in the "Computer" page, connected to the "Transistor" page.

Transistor, A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. [. . .]

↑

Computer

CRTM	RTM
transistor	staircase
electron	atom
MOSFET	dopant
dopant	fabricated
electrical	ability

5 CONCLUSION AND FUTURE WORK

We have presented the Contextualized Relational Topic Model, CRTM, a probabilistic modeling framework to infer latent topics in networks of documents, that explicitly accounts for the locations of the links in the text. We've experimentally shown the relevancy of our approach through a quantitative evaluation based on several Wikipedia datasets, in English, Italian and German. We've also shown that CRTM has a competitive runtime, which makes it usable in practice to solve tasks akin to anchor prediction without relying on external information like a knowledge graph. We also demonstrated that taking anchor links into account doesn't degrade the model's performance in link prediction. From a qualitative point of view, we've illustrated how CRTM can assist knowledge bases contributors while they specify anchor links after writing. In future work, we'd like to investigate more complex link functions based upon a more sophisticated modeling of topics and documents, and extend our work to more recent works, like Graph Neural Networks and Neural Topic Models.

REFERENCES

[1] Christopher Aicher, Abigail Z Jacobs, and Aaron Clauset. 2013. Adapting the stochastic block model to edge-weighted networks. *arXiv preprint arXiv:1305.5782* (2013).

[2] Haoli Bai, Zhuangbin Chen, Michael R Lyu, Irwin King, and Zenglin Xu. 2018. Neural relational topic models for scientific article analysis. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 27–36.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[6] Aleksandar Bojchevski and Stephan Günnemann. [n.d.]. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *Proceeding of ICLR*.

[7] Robin Brochier and Frédéric Béchet. 2021. Predicting Links on Wikipedia with Anchor Text Information. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[8] Robin Brochier, Adrien Guille, and Julien Velcin. 2019. Global vectors for node representations. In *The World Wide Web Conference*. 2587–2593.

[9] Robin Brochier, Adrien Guille, and Julien Velcin. 2020. Inductive Document Network Embedding with Topic-Word Attention. In *Proceedings of ECIR*. Springer, 326–340.

[10] Jonathan Chang and David Blei. 2009. Relational topic models for document networks. In *Artificial Intelligence and Statistics*. 81–88.

[11] Jonathan Chang and David M Blei. 2010. Hierarchical relational models for document networks. *The Annals of Applied Statistics* (2010), 124–150.

[12] Ning Chen, Jun Zhu, Fei Xia, and Bo Zhang. 2013. Generalized relational topic models with data augmentation. In *Proceedings of IJCAI*. 1273–1279.

[13] Ran Ding, Ramesh Nallapati, and Bing Xiang. 2018. Coherence-Aware Neural Topic Modeling. In *EMNLP*.

[14] Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*. 1625–1628.

[15] Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2619–2629.

[16] Antoine Gourru, Adrien Guille, Julien Velcin, and Julien Jacques. 2020. Document Network Projection in Pretrained Word Embedding Space. In *ECIR*. Springer, 150–157.

[17] Amit Gruber, Michal Rosen-Zvi, and Yair Weiss. 2008. Latent topic models for hypertext. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. 230–239.

[18] Amit Gruber, Michal Rosen-Zvi, and Yair Weiss. 2008. Topic Models for Hypertext: How Many Words is a Single Link Worth? (2008).

[19] Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *Artificial intelligence and statistics*. PMLR, 163–170.

[20] Xianpei Han and Le Sun. 2012. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 105–115.

[21] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. 2020. Inductive Link Prediction for Nodes Having Only Attribute Information. In *Proceedings of IJCAI*. 1209–1215.

[22] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 782–792.

[23] Thomas Hofmann. 1999. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization. In *Proceedings of NeurIPS*. 914–920.

[24] Diederik P Kingma, Max Welling, et al. 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[26] Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. 2020. Zero-shot Entity Linking with Dense Entity Retrieval. In *EMNLP*.

[27] Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to node: Self-translation network embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1794–1802.

[28] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A Survey of Link Prediction in Complex Networks. 49, 4 (2016).

[29] Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with Wikipedia. In *Proceedings of the AAAI WikiAI workshop*, Vol. 1. 19–24.

[30] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*. 101–110.

[31] Rada Mihalcea and Andras Csomai. 2007. Wikify! Linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 233–242.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[33] Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic Modeling with Wasserstein Autoencoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6345–6381.

[34] Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation*. 55–62.

[35] Congkai Sun, Bin Gao, Zhenfu Cao, and Hang Li. 2008. HTM: A topic model for hypertexts. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 514–522.

[36] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-fourth international joint conference on artificial intelligence*.

[37] I Tolstikhin, O Bousquet, S Gelly, and B Schölkopf. 2018. Wasserstein Auto-Encoders. In *6th International Conference on Learning Representations (ICLR 2018)*. OpenReview. net.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[39] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. 2017. Relational deep learning: A deep latent variable model for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[40] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1006–1011.

[41] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*. 1445–1456.

[42] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of IJCAI*. 2111–2117.

[43] Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. 2015. Birds of a feather linked together: A discriminative topic model using link-based priors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 261–266.

[44] Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. 2016. A discriminative topic model using document network structure. In *Proceedings of the 54th ACL Annual Meeting (Volume 1: Long Papers)*. 686–696.

[45] Aonan Zhang, Jun Zhu, and Bo Zhang. 2013. Sparse relational topic models for document networks. In *Joint ECML and KDD*. Springer, 670–685.

[46] Jian Zhang, Jun Zheng, Jinyin Chen, and Qi Xuan. 2020. Hyper-Substructure Enhanced Link Predictor. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 2305–2308. https://doi.org/10.1145/3340531.3412096

[47] Deyu Zhou, Xuemeng Hu, and Rui Wang. 2020. Neural Topic Modeling by Incorporating Document Relationship Graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3790–3796.

[48] Qile Zhu, Zheng Feng, and Xiaolin Li. 2018. GraphBTM: Graph enhanced autoencoded variational inference for biterm topic model. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 4663–4672.

4.4 Perspectives d'implémentation pour l'entreprise

Ces deux contributions proposent donc une méthode pour prédire les ancres d'éventuels hyperliens entre deux documents. Deux scénarios peuvent être envisagés pour exploiter la prédiction d'ancres : le premier prend la forme d'une aide à la lecture lors d'une phase de recherche d'information ; le second comme un outil de recommandation d'hyperliens pour les contributeurs.

4.4.1 La prédiction d'ancres pour assister la lecture

Dans le premier scénario, la prédiction des ancres va être employée afin d'assister un lecteur durant un parcours de base. Les bases de connaissances construites par MeetSYS et ses clients sont principalement constituées de pages décrivant des phénomènes physiques ou industriels, ou de descriptions de problèmes rencontrés. L'idée est donc le sélectionner une page, décrivant par exemple un problème industriel documenté, et déterminer lors de la consultation d'autres contenus quels termes sont susceptibles de porter un hyperlien vers le document cible, comme illustré à la Figure 4.1.

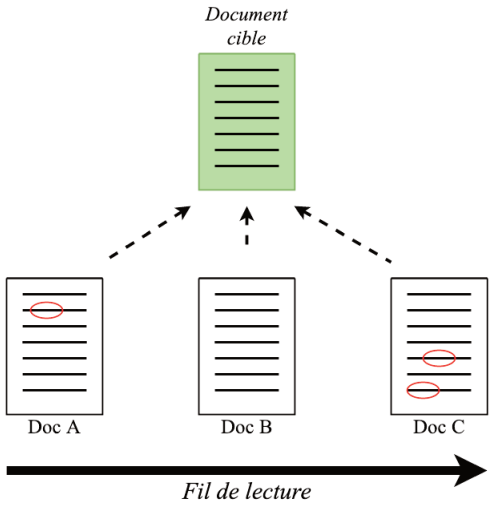


FIGURE 4.1 – Représentation schématique de l'utilisation de CRTM en temps d'accompagnement à la lecture pour la résolution de problèmes.

Ce système s'avère relativement simple à mettre en place, car outre l'apprentissage des paramètres du modèle, il est simplement nécessaire de fournir au serveur l'identifiant de la page sélectionnée (lié à la session) et celui de la page en cours de consultation. Pour permettre à plus ou moins d'ancres d'être proposées, il est nécessaire de définir une valeur seuil en dessous de laquelle la probabilité qu'un mot porte un hyperlien n'est pas présenté.

4.4.2 La recommandation d'hyperliens pour le contributeurs

La prédiction d'ancres peut également être employée dans un autre contexte : la recommandation d'hyperliens aux contributeurs lors de la publication d'une page. Cette recommandation devra cependant se faire en deux temps, comme montré Figure 4.2. En effet, puisque de CRTM ne permet pas de prédire le document en même temps que l'ancre associée, il est nécessaire de trouver d'abord les n documents les plus proches. On utilisera ainsi pour cela les représentations de documents déjà apprises et une mesure de similarité, comme par exemple une similarité cosinus. Sans cette étape de filtrage, le nombre de documents à examiner serait trop important, et engendrerait un coup en terme de calcule pouvant devenir prohibitif pour une base importante.

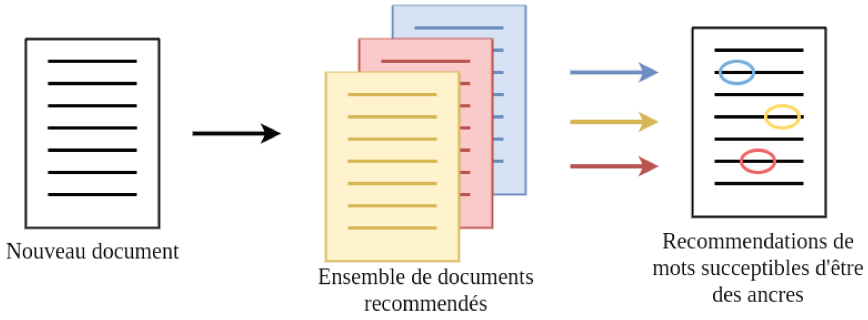


FIGURE 4.2 – Processus de recommandation d'hyperliens pour un nouveaux document.

Une fois construit le sous ensemble des documents les plus proches du document considéré, on peut alors appliquer CRTM sur les n paires de documents formées avec le document source. Les paires mot-document cible proposées sont alors filtrées afin de ne garder que celles proposant les plus grandes valeurs de $\psi_{(d_n,d')}$. Libre alors au contributeur de valider, ou non, les hyperliens proposés.

4.5 Conclusion

Au long de ce chapitre nous avons présenté une méthode de modélisation de thématiques adaptée aux réseaux de documents, destinée à prédire les ancres des hyperliens. Au cours des différentes expériences menées nous avons montré que les améliorations proposées permettent d'obtenir les meilleurs résultats, tant quantitativement qu'au niveau de la cohérence des termes. De toutes les formes d'agrégation du contexte envisagées, l'approche sémantique, privilégiant les mots sémantiquement proches de l'ancre, donne les meilleurs résultats. De plus, nous avons montré que l'utilisation de la matrice Q dans la fonction de lien, permettant l'interaction de toutes les paires de thématiques, améliorait sensiblement les performances du modèle.

Chapitre 5

Conclusion

En conclusion de ce manuscrit, nous commencerons par résumer brièvement les travaux proposés, avant de détailler différentes perspectives de recherches qui ont pu naître suite à ces contributions.

5.1 Résumé des travaux présentés

Au cours de ce manuscrit de thèse, nous nous sommes intéressés à l'utilisation de l'information portée par la structure en réseau des éléments des bases de connaissances, dans le but de pouvoir les enrichir avec des contenus non encore intégrés, ou encore, en cherchant à peupler les pages avec des hyperliens.

En premier lieu, nous sommes revenus sur les principales méthodes existantes permettant d'apprendre des représentations de mots, de documents, de réseaux et de réseaux de documents. Pour chacun de ces cas d'application, nous nous sommes employés à en retracer l'origine et les évolutions jusqu'aux méthodes les plus récentes.

Ensuite, nous avons présenté une méthode permettant de déterminer une matrice de passage entre deux espaces de représentations, l'un propre aux sommets du graphe, et l'autre aux documents textuels. Les expériences réalisées montrent que lors d'une tâche de reconstruction de voisinage, utiliser l'approximation de la représentation du contenu textuel dans l'espace de représentation des sommets donne de meilleurs résultats, comparé à la simple utilisation des représentations des contenus textuels. De plus, cette méthode offre également la possibilité de lier de nouveaux documents textuels à d'autres contenus de natures différentes, comme des fichiers, ou des pages personnelles de contributeurs.

Enfin, nous avons proposé une méthode permettant de prédire les mots ou groupes de mots d'une page susceptibles de porter un hyperlien vers une autre. Ce modèle, nommé *Contextual Relational Topic Model* (CRTM), repose sur la modélisation de thématiques appliquée aux réseaux de documents, et cherche à apprendre ces thématiques de façon à déterminer si un mot porte un lien vers une page, en fonction des mots qui l'entourent. Nous avons proposé plusieurs façons permettant de prendre en compte le voisinage d'un mot portant un lien, celui-ci étant agrégé de façon (1) uniforme, (2) positionnelle ou (3) sémantique. Les expériences menées sur différents sous-ensembles de Wikipedia montrent une certaine habilité du modèle à retrouver les mots susceptibles d'être une ancre. De plus, en regardant de plus près les mots suggérés par CRTM, on remarque que ceux-ci semblent plus cohérents que ceux proposés par Relational Topic Model.

5.2 Perspectives et continuité des travaux

Les travaux menés durant ce doctorat ont permis d'explorer plusieurs axes de recherches, faisant naître de nouvelles interrogations et perspectives d'améliorations des méthodes proposées. Nous aborderons cette partie en trois temps, en discutant des limites des méthodes proposées, et des pistes à explorer.

5.2.1 La prédiction d'ancres

Plusieurs constats sont à faire quant à la méthode proposée au Chapitre 3. Premièrement, CRTM s'appuie sur un cadre de modélisation qui, s'il est toujours largement utilisé dans l'industrie [Kang et al., 2019] ; [Sutherland et al., 2020], est depuis longtemps concurrencé par des méthodes plus récentes. Comme présenté au cours de l'état de l'art, les méthodes reposant sur l'architecture *Transformer*, par exemple, offrent des capacités de modélisation de la langue d'excellente qualité. En parallèle, les architectures de type *Graph Convolutional Network* ou *Graph Attention Network* sont aujourd'hui prédominantes sur les questions d'apprentissage de représentation de graphes, attribués ou non.

Une des limitations principales de CRTM se trouve dans son incapacité à prédire la page cible d'un hyperlien. En l'état actuel des choses, afin de positionner efficacement un hyperlien, CRTM nécessite en effet que l'on lui fournisse un lien partant d'une page, et pointant vers une autre. Ce mode de fonctionnement, s'il est compatible avec certaines utilisations, réduit fondamentalement les cas d'usages, de part une combinatoire importante à mesure que le corpus devient important. Il est donc nécessaire d'adjoindre au modèle un moteur de recommandations afin de limiter le nombre de documents cibles potentiels. On imagine alors l'intérêt que pourrait trouver un modèle capable de prédire à la fois les liens entre les documents, et les ancres qui leurs sont associées.

5.2.2 Exploitation d'autres types données

Durant ces travaux de thèse nous nous sommes limité à travailler sur les données directement présentes dans les bases de connaissances, à savoir le texte, et les liens entre eux. D'autres données sont cependant rendues disponibles par l'exploitation de ces bases, et parmi elles, les parcours utilisateurs. Tout comme les liens entre les pages, les données relatives aux chemins empruntés par les utilisateurs dans leur recherche d'information peuvent également révéler d'autres informations sur les liens entre les différentes pages, comme par exemple des expériences métiers plus ténues, que la proximité sémantique ne permettrait pas de identifier. Ce type de données a été très largement étudié dans la littérature [F. Zhang et al., 2016] ; [Huang et al., 2019], et notamment dans les modèles de recommandations.

5.2.3 Archipel de bases de connaissances et apprentissage fédéré

La question de la recommandation multi-bases est également un enjeu important pour l'écosystème de MeetSYS. On se pose ici la question de comment pouvoir représenter les documents au sein de chaque base de façon à pouvoir, de façon transparente, recommander des documents d'une base B, en partant d'un document d'une base A. Plusieurs méthodes

peuvent être envisagées. La plus simple consisterait peut-être sa baser sur la méthode décrite au Chapitre 2, afin d'apprendre des matrices de passages entre les différentes bases. Cela nécessiterait cependant que l'on inclue un sous ensemble commun à toutes les bases (dans notre cas, par exemple, la base ouverte de faits scientifiques proposée par l'entreprise), afin de pouvoir apprendre correctement la matrice.

Une seconde approche peut cependant être envisagée, dès lors que l'on reformule le problème. Au lieu de considérer les espaces de représentations des documents des bases comme autant d'espaces indépendants dont il faut apprendre comment projeter un élément de l'un vers l'autre, il pourrait également être intéressant d'entraîner conjointement ces modèles. Cela ne va cependant pas sans soulever des problèmes de confidentialité des données. Il serait en effet impossible de regrouper toutes les bases afin d'entraîner une méthode d'apprentissage de représentations sur l'ensemble de ce corpus, pour d'évidentes contraintes de propriétés industrielles.

Cette question n'est toutefois pas neuve, et représente tout un pan de l'apprentissage automatique : l'apprentissage fédéré [B. McMahan et al., 2017]. L'apprentissage fédéré constitue un cadre où, selon la définition donnée par Kairouz et al., 2019, *[...] plusieurs entités (des clients) collaborent à résoudre une tâche d'apprentissage, coordonnées par un serveur central. Les données brutes de chaque client sont stockées localement et ne sont ni échangées ni transférées ; au lieu de cela, des mises à jour ciblées, destinées à être immédiatement agrégées sont utilisées pour atteindre l'objectif d'apprentissage.* Les données brutes (dans notre cas les informations contenues dans les bases de connaissances) ne transitent alors jamais sur le réseau, et seuls les gradients des modèles locaux sont envoyés au serveur central pour y être agrégés. Les deux contraintes émises plus haut seraient alors respectées : d'une part, la confidentialité des données serait assurée, en ceci qu'elles ne quitteraient pas le réseau interne des entreprises, et d'autre part toutes les bases de connaissances partageraient le même espace de représentation, offrant la possibilité d'une complète interopérabilité entre elles. Le cadre de l'apprentissage fédéré est déjà employé dans de nombreux travaux en traitement du langage naturel [Zhu et al., 2020] ; [Lin et al., 2021] ; [C. Wang et al., 2021], et représente une direction prometteuse pour l'apprentissage de représentations de documents.

Bibliographie

Ackoff, Russell L (1989). « From data to wisdom ». In : *Journal of applied systems analysis* 16.1, p. 3-9.

Arora, Sanjeev, Yingyu Liang et Tengyu Ma (2017). « A simple but tough-to-beat baseline for sentence embeddings ». In : *5th International Conference on Learning Representations, ICLR 2017*.

Bahdanau, Dzmitry, Kyunghyun Cho et Yoshua Bengio (2015). « Neural Machine Translation by Jointly Learning to Align and Translate ». In : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Sous la dir. d'Yoshua Bengio et Yann LeCun. url : http://arxiv.org/abs/1409.0473.

Bai, Haoli, Zhuangbin Chen, Michael R Lyu, Irwin King et Zenglin Xu (2018). « Neural relational topic models for scientific article analysis ». In : *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, p. 27-36.

Balasubramanyan, Ramnath, William Cohen, Douglas Pierce et David Redlawsk (2012). « Modeling polarizing topics: When do different political communities respond differently to the same news? » In : *Proceedings of the International AAAI Conference on Web and Social Media*. T. 6. 1.

Belkin, Mikhail et Partha Niyogi (2003). « Laplacian eigenmaps for dimensionality reduction and data representation ». In : *Neural computation* 15.6, p. 1373-1396.

Bengio, Yoshua, Aaron Courville et Pascal Vincent (2013). « Representation learning: A review and new perspectives ». In : *IEEE transactions on pattern analysis and machine intelligence* 35.8, p. 1798-1828.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent et Christian Janvin (2003). « A neural probabilistic language model ». In : *The journal of machine learning research* 3, p. 1137-1155.

Bengio, Yoshua, Pascal Lamblin, Dan Popovici et Hugo Larochelle (2007). « Greedy layer-wise training of deep networks ». In : *Advances in neural information processing systems*, p. 153-160.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.

Blei, David M et John D Lafferty (2006). « Dynamic topic models ». In : *Proceedings of the 23rd international conference on Machine learning*, p. 113-120.

Blei, David M et Jon D McAuliffe (2007). « Supervised Topic Models ». In : *NIPS*.

Blei, David M, Andrew Y Ng et Michael I Jordan (2003). « Latent dirichlet allocation ». In : *the Journal of machine Learning research* 3, p. 993-1022.

Bojanowski, Piotr, Edouard Grave, Armand Joulin et Tomas Mikolov (2017). « Enriching word vectors with subword information ». In : *Transactions of the Association for Computational Linguistics* 5, p. 135-146.

Bolukbasi, Tolga, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama et Adam Tauman Kalai (2016). « Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings ». In : *NIPS*.

Braun, Michael et Jon McAuliffe (2010). « Variational inference for large-scale models of discrete choice ». In : *Journal of the American Statistical Association* 105.489, p. 324-335.

Britz, Denny, Anna Goldie, Minh-Thang Luong et Quoc Le (2017). « Massive Exploration of Neural Machine Translation Architectures ». In : *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 1442-1451.

Brochier, Robin, Adrien Guille et Julien Velcin (2019). « Global Vectors for Node Representations ». In : *Proceedings of the 2019 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

— (2020). « Inductive document network embedding with topic-word attention ». In : *Advances in Information Retrieval* 12035, p. 326.

Caliskan, Aylin, Joanna J Bryson et Arvind Narayanan (2017). « Semantics Derived Automatically From Language Corpora Contain Human-Like Biases ». In : *Science (New York, NY)* 356.6334, p. 183-186.

Chang, Jonathan et David Blei (2009). « Relational topic models for document networks ». In : *Artificial intelligence and statistics*. PMLR, p. 81-88.

Chen, Ning, Jun Zhu, Fei Xia et Bo Zhang (2013). « Generalized relational topic models with data augmentation ». In : *Twenty-Third International Joint Conference on Artificial Intelligence*.

Cho, Kyunghyun, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk et Yoshua Bengio (2014). « Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation ». In :

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau et Yoshua Bengio (2014). « On the Properties of Neural Machine Translation: Encoder–Decoder Approaches ». In : *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, p. 103-111.

Collobert, Ronan et Jason Weston (2008). « A unified architecture for natural language processing: Deep neural networks with multitask learning ». In : *Proceedings of the 25th international conference on Machine learning*, p. 160-167.

Cox, Michael AA et Trevor F Cox (2001). « Multidimensional scaling ». In : *Handbook of data visualization*. Springer, p. 315-347.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer et Richard Harshman (1990). « Indexing by latent semantic analysis ». In : *Journal of the American society for information science* 41.6, p. 391-407.

Dempster, Arthur P, Nan M Laird et Donald B Rubin (1977). « Maximum likelihood from incomplete data via the EM algorithm ». In : *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, p. 1-22.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee et Kristina Toutanova (juin 2019). « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short*

Papers). Minneapolis, Minnesota : Association for Computational Linguistics, p. 4171-4186. URL : https://www.aclweb.org/anthology/N19-1423.

Dosovitskiy, Alexey et al. (2021). « An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale ». In : *International Conference on Learning Representations*. URL : https://openreview.net/forum?id=YicbFdNTTy.

Dupuy, Jean (2019). « Recherche d'information entre des bases de connaissances. » In : *CORIA*.

— (2020). « Coping for missing content and missing links in document network embedding: student research abstract ». In : *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, p. 477-480.

Dupuy, Jean, Adrien Guille et Julien Jacques (2021). « Contextual-RTM: un cadre général pour la modélisation de thématiques dans les réseaux de documents ». In : *Extraction et Gestion des Connaissances: Actes EGC'2021*.

Eckart, Carl et Gale Young (1936). « The approximation of one matrix by another of lower rank ». In : *Psychometrika* 1.3, p. 211-218.

Ermine, Jean-Louis (2003). *La gestion des connaissances*. Hermes Sciences Publications, p. 166. URL : https://hal.archives-ouvertes.fr/hal-00997696.

Firth, John R (1957). « A synopsis of linguistic theory, 1930-1955 ». In : *Studies in linguistic analysis*.

Gershman, Samuel et Joshua B Tenenbaum (2015). « Phrase similarity in humans and machines. » In : Citeseer.

Goodman, Joshua (2001). « Classes for fast maximum entropy training ». In : *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*. T. 1. IEEE, p. 561-564.

Gourru, Antoine, Julien Velcin et Julien Jacques (2020). « Gaussian Embedding of Linked Documents from a Pretrained Semantic Space. » In : *IJCAI*, p. 3912-3918.

Griffiths, Thomas L et Mark Steyvers (2004). « Finding scientific topics ». In : *Proceedings of the National academy of Sciences* 101.suppl 1, p. 5228-5235.

GROVER, Aditya et Jure LESKOVEC (2016). « node2vec: Scalable feature learning for networks ». In : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 855-864.

GRUBER, Amit, Michal ROSEN-ZVI et Yair WEISS (2008). « Latent topic models for hypertext ». In : *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, p. 230-239.

— (p. d.). « Topic Models for Hypertext: How Many Words is a Single Link Worth? » In : ().

GRUBER, Amit, Yair WEISS et Michal ROSEN-ZVI (2007). « Hidden topic markov models ». In : *Artificial intelligence and statistics*. PMLR, p. 163-170.

GUO, Jiafeng, Gu XU, Xueqi CHENG et Hang LI (2009). « Named entity recognition in query ». In : *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, p. 267-274.

GUTMANN, Michael et Aapo HYVÄRINEN (2010). « Noise-contrastive estimation: A new estimation principle for unnormalized statistical models ». In : *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop et Conference Proceedings, p. 297-304.

HAKAN, Inan, Khosravi KHASHAYAR et Socher RICHARD (2016). « Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling ». In : *CoRR* abs/1611.01462. arXiv : 1611.01462. URL : http://arxiv.org/abs/1611.01462.

HAMILTON, William L, Rex YING et Jure LESKOVEC (2017). « Inductive representation learning on large graphs ». In : *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 1025-1035.

HARRIS, Zellig S (1954). « Distributional structure ». In : *Word* 10.2-3, p. 146-162.

HEINTZ, Ilana et al. (2013). « Automatic extraction of linguistic metaphors with lda topic modeling ». In : *Proceedings of the First Workshop on Metaphor in NLP*, p. 58-66.

HOCHREITER, Sepp et Jürgen SCHMIDHUBER (1997). « Long short-term memory ». In : *Neural computation* 9.8, p. 1735-1780.

Hoffman, Matthew, Francis R Bach et David M Blei (2010). « Online learning for latent dirichlet allocation ». In : *advances in neural information processing systems*. Citeseer, p. 856-864.

Hofmann, Thomas (1999). « Probabilistic Latent Semantic Analysis ». In : p. 289-296.

Hotelling, Harold (1933). « Analysis of a complex of statistical variables into principal components. » In : *Journal of educational psychology* 24.6, p. 417.

Hu, Yuheng, Ajita John, Fei Wang et Subbarao Kambhampati (2012). « Et-lda: Joint topic modeling for aligning events and their twitter feedback ». In : *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 26. 1.

Huang, Xiaowen, Quan Fang, Shengsheng Qian, Jitao Sang, Yan Li et Changsheng Xu (2019). « Explainable interaction-driven user modeling over knowledge graph for sequential recommendation ». In : *Proceedings of the 27th ACM International Conference on Multimedia*, p. 548-556.

Jagarlamudi, Jagadeesh et Hal Daumé (2010). « Extracting multilingual topics from unaligned comparable corpora ». In : *European Conference on Information Retrieval*. Springer, p. 444-456.

Jones, Karen Sparck (1972). « A statistical interpretation of term specificity and its application in retrieval ». In : *Journal of documentation*.

Kairouz, Peter et al. (2019). « Advances and open problems in federated learning ». In : *arXiv preprint arXiv:1912.04977*.

Kang, Jiho, Junseok Lee, Dongsik Jang et Sangsung Park (2019). « A methodology of partner selection for sustainable industry-university cooperation based on LDA topic model ». In : *Sustainability* 11.12, p. 3478.

Kingma, Diederik P et Jimmy Ba (2015). « Adam: A Method for Stochastic Optimization ». In : *ICLR (Poster)*.

Kingma, Diederik P et Max Welling (2014). « Auto-encoding variational Bayes ». In : *International Conference on Learning Representations (ICLR)*.

Kipf, Thomas N et Max Welling (2016a). « Semi-Supervised Classification with Graph Convolutional Networks ». In : *arXiv preprint arXiv:1609.02907*.

— (2016b). « Variational Graph Auto-Encoders ». In : *NIPS Workshop on Bayesian Deep Learning*.

Kiros, Ryan et al. (2015). « Skip-Thought Vectors ». In : *NIPS*.

Kusner, Matt, Yu Sun, Nicholas Kolkin et Kilian Weinberger (2015). « From word embeddings to document distances ». In : *International conference on machine learning*. PMLR, p. 957-966.

Kuyumcu, Birol, Cuneyt Aksakalli et Selman Delil (2019). « An automated new approach in fast text classification (fastText) A case study for Turkish text classification without pre-processing ». In : *Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval*, p. 1-4.

Le, Hang et al. (mai 2020). « FlauBERT: Unsupervised Language Model Pre-training for French ». In : *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France : European Language Resources Association, p. 2479-2490. url : https://www.aclweb.org/anthology/2020.lrec-1.302.

Le, Quoc et Tomas Mikolov (2014). « Distributed representations of sentences and documents ». In : *International conference on machine learning*. PMLR, p. 1188-1196.

Le, Tuan MV et Hady W Lauw (2014). « Probabilistic latent document network embedding ». In : *2014 IEEE International Conference on Data Mining*. IEEE, p. 270-279.

Lecun, Y., L. Bottou, Y. Bengio et P. Haffner (1998). « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11, p. 2278-2324.

Levy, Omer et Yoav Goldberg (2014). « Neural word embedding as implicit matrix factorization ». In : *Advances in neural information processing systems* 27, p. 2177-2185.

Li, Qimai, Zhichao Han et Xiao-Ming Wu (2018). « Deeper insights into graph convolutional networks for semi-supervised learning ». In : *Thirty-Second AAAI conference on artificial intelligence*.

Li, Yitan, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong et Enhong Chen (2015). « Word embedding revisited: A new representation learning and explicit matrix factorization perspective ». In : *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Lin, Bill Yuchen et al. (2021). « FedNLP: Benchmarking Federated Learning Methods for Natural Language Processing Tasks ». In : *arXiv preprint arXiv:2104.08815*.

Liu, Yinhan et al. (2019). « RoBERTa: A Robustly Optimized BERT Pretraining Approach ». In :

Luhn, H. P. (1957). « A Statistical Approach to Mechanized Encoding and Searching of Literary Information ». In : *IBM Journal of Research and Development* 1.4, p. 309-317.

Lukins, Stacy K, Nicholas A Kraft et Letha H Etzkorn (2010). « Bug localization using latent dirichlet allocation ». In : *Information and Software Technology* 52.9, p. 972-990.

Luong, Minh-Thang, Hieu Pham et Christopher D Manning (2015). « Effective Approaches to Attention-based Neural Machine Translation ». In : *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412-1421.

Martin, Louis et al. (juil. 2020). « CamemBERT: a Tasty French Language Model ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online : Association for Computational Linguistics, p. 7203-7219. url : https://www.aclweb.org/anthology/2020.acl-main.645.

McCallum, Andrew Kachites, Kamal Nigam, Jason Rennie et Kristie Seymore (2000). « Automating the construction of internet portals with machine learning ». In : *Information Retrieval* 3.2, p. 127-163.

McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson et Blaise Aguera y Arcas (2017). « Communication-efficient learning of deep networks from decentralized data ». In : *Artificial intelligence and statistics*. PMLR, p. 1273-1282.

Miao, Yishu, Lei Yu et Phil Blunsom (2016). « Neural variational inference for text processing ». In : *International conference on machine learning*. PMLR, p. 1727-1736.

Mikolov, Tomas, Kai Chen, Greg Corrado et Jeffrey Dean (2013). « Efficient estimation of word representations in vector space ». In : *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Quoc Le et Ilya Sutskever (2013). « Exploiting similarities among languages for machine translation ». In : *arXiv preprint arXiv:1309.4168*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado et Jeffrey Dean (2013). « Distributed representations of words and phrases and their compositionality ». In : *Pro-*

ceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2, p. 3111-3119.

Millar, Jeremy R, Gilbert L Peterson et Michael J Mendenhall (2009). « Document Clustering and Visualization with Latent Dirichlet Allocation and Self-Organizing Maps. » In : *FLAIRS Conference*. T. 21, p. 69-74.

Mnih, Andriy et Geoffrey E Hinton (2008). « A scalable hierarchical distributed language model ». In : *Advances in neural information processing systems* 21, p. 1081-1088.

Morin, Frederic et Yoshua Bengio (2005). « Hierarchical probabilistic neural network language model. » In : *Aistats*. T. 5. Citeseer, p. 246-252.

Nair, Vinod et Geoffrey E Hinton (2010). « Rectified linear units improve restricted boltzmann machines ». In : *Icml*.

Natarajan, Nagarajan et Inderjit S Dhillon (2014). « Inductive matrix completion for predicting gene–disease associations ». In : *Bioinformatics* 30.12, p. i60-i68.

Nguyen, Dat Quoc et Anh Tuan Nguyen (2020). « PhoBERT: Pre-trained language models for Vietnamese ». In : *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, p. 1037-1042.

Nonaka, Ikujirō, Nonaka Ikujiro, Hirotaka Takeuchi et al. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. T. 105. OUP USA.

Parisi, Giorgio (1988). *Statistical field theory*. Addison-Wesley.

Pennington, Jeffrey, Richard Socher et Christopher D Manning (2014). « Glove: Global vectors for word representation ». In : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, p. 1532-1543.

Perina, Alessandro, Pietro Lovato, Vittorio Murino et Manuele Bicego (2010). « Biologically-aware latent dirichlet allocation (balda) for the classification of expression microarray ». In : *IAPR International Conference on Pattern Recognition in Bioinformatics*. Springer, p. 230-241.

Perozzi, Bryan, Rami Al-Rfou et Steven Skiena (2014). « Deepwalk: Online learning of social representations ». In : *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 701-710.

Peters, Matthew E. et al. (juin 2018). « Deep Contextualized Word Representations ». In : *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana : Association for Computational Linguistics, p. 2227-2237. URL : https://www.aclweb.org/anthology/N18-1202.

Polson, Nicholas G, James G Scott et Jesse Windle (2013). « Bayesian inference for logistic models using Pólya–Gamma latent variables ». In : *Journal of the American statistical Association* 108.504, p. 1339-1349.

Ponzi, Leonard J (2004). « Knowledge management: Birth of a discipline ». In : *Knowledge management lessons learned: What works and what doesn't*, p. 9-26.

Press, Ofir et Lior Wolf (2017). « Using the Output Embedding to Improve Language Models ». In : *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, p. 157-163.

Qiu, Jiezhong, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang et Jie Tang (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*, p. 459-467.

Radford, Alec, Karthik Narasimhan, Tim Salimans et Ilya Sutskever (p. d.). « Improving Language Understanding by Generative Pre-Training ». In : ().

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev et Percy Liang (nov. 2016). « SQuAD: 100,000+ Questions for Machine Comprehension of Text ». In : *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas : Association for Computational Linguistics, p. 2383-2392. URL : https://www.aclweb.org/anthology/D16-1264.

Rezende, Danilo Jimenez, Shakir Mohamed et Daan Wierstra (2014). « Stochastic backpropagation and approximate inference in deep generative models ». In : *International conference on machine learning*. PMLR, p. 1278-1286.

Robert, Christian et George Casella (2011). « A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data ». In : *Statistical Science*, p. 102-115.

RÖDER, Michael, Andreas BOTH et Alexander HINNEBURG (2015). « Exploring the space of topic coherence measures ». In : *Proceedings of the eighth ACM international conference on Web search and data mining*, p. 399-408.

ROSEN-ZVI, Michal, Thomas GRIFFITHS, Mark STEYVERS et Padhraic SMYTH (2004). « The author-topic model for authors and documents ». In : *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, p. 487-494.

ROWEIS, Sam T et Lawrence K SAUL (2000). « Nonlinear dimensionality reduction by locally linear embedding ». In : *science* 290.5500, p. 2323-2326.

SALAKHUTDINOV, Ruslan et Geoffrey HINTON (2009). « Semantic hashing ». In : *International Journal of Approximate Reasoning* 50.7, p. 969-978.

SALTON, Gerard, Chung-Shu YANG et CLEMENT T YU (1975). « A theory of term importance in automatic text analysis ». In : *Journal of the American society for Information Science* 26.1, p. 33-44.

SANH, Victor, Lysandre DEBUT, Julien CHAUMOND et Thomas WOLF (2019). « DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter ». In : *arXiv preprint arXiv:1910.01108*.

SCHÖNEMANN, Peter H (1966). « A generalized solution of the orthogonal procrustes problem ». In : *Psychometrika* 31.1, p. 1-10.

SMITH, Samuel L., David H. P. TURBAN, Steven HAMBLIN et Nils Y. HAMMERLA (2017). « Offline bilingual word vectors, orthogonal transformations and the inverted softmax ». In : *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL : https://openreview.net/forum?id=r1Aab85gg.

SONG, Min, Meen Chul KIM et Yoo Kyung JEONG (2014). « Analyzing the political landscape of 2012 Korean presidential election in Twitter ». In : *IEEE Intelligent Systems* 29.2, p. 18-26.

SRIVASTAVA, Akash et Charles SUTTON (2017). « Autoencoding Variational Inference for Topic Models ». In : *5th International Conference on Learning Representations*.

SUTHERLAND, Ian et Kiattipoom KIATKAWSIN (2020). « Determinants of guest experience in Airbnb: a topic modeling approach using LDA ». In : *Sustainability* 12.8, p. 3402.

Tan, Shulong et al. (2013). « Interpreting the public sentiment variations on twitter ». In : *IEEE transactions on knowledge and data engineering* 26.5, p. 1158-1170.

Tang, Hong et al. (2012). « A multiscale latent Dirichlet allocation model for object-oriented clustering of VHR panchromatic satellite images ». In : *IEEE Transactions on Geoscience and Remote Sensing* 51.3, p. 1680-1692.

Tang, Jian, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan et Qiaozhu Mei (2015). « Line: Large-scale information network embedding ». In : *Proceedings of the 24th international conference on world wide web*, p. 1067-1077.

Tenenbaum, Joshua B, Vin De Silva et John C Langford (2000). « A global geometric framework for nonlinear dimensionality reduction ». In : *science* 290.5500, p. 2319-2323.

Thongtan, Tan et Tanasanee Phienthrakul (juil. 2019). « Sentiment Classification Using Document Embeddings Trained with Cosine Similarity ». In : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy : Association for Computational Linguistics, p. 407-414. URL : https://www.aclweb.org/anthology/P19-2057.

Tu, Cunchao, Han Liu, Zhiyuan Liu et Maosong Sun (juil. 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)*. Vancouver, Canada : Association for Computational Linguistics, p. 1722-1731. URL : https://aclanthology.org/P17-1158.

Vaswani, Ashish et al. (2017). « Attention is All you Need ». In : *NIPS*.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò et Yoshua Bengio (2018). « Graph Attention Networks ». In : *International Conference on Learning Representations*. URL : https://openreview.net/forum?id=rJXMpikCZ.

Vulic, Ivan, Wim De Smet et Marie-Francine Moens (2011). « Identifying word translations from comparable corpora using latent topic models ». In : *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*. T. 2. ACL; East Stroudsburg, PA, p. 479-484.

Wallach, Hanna M, David M Mimno et Andrew McCallum (2009). « Rethinking LDA: Why priors matter ». In : *Advances in neural information processing systems*, p. 1973-1981.

WANG, Alex, Amanpreet SINGH, Julian MICHAEL, Felix HILL, Omer LEVY et Samuel R BOWMAN (2019). « Glue: A multi-task benchmark and analysis platform for natural language understanding ». In : *7th International Conference on Learning Representations, ICLR 2019*.

WANG, Chenghong et al. (2021). « A Secure and Efficient Federated Learning Framework for NLP ». In : *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 7676-7682.

WANG, Daixin, Peng CUI et Wenwu ZHU (2016). « Structural Deep Network Embedding ». In : *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA : ACM, p. 1225-1234. URL : http://doi.acm.org/10.1145/2939672.2939753.

WANG, Hao, Xingjian SHI et Dit-Yan YEUNG (2017). « Relational deep learning: A deep latent variable model for link prediction ». In : *Thirty-first AAAI conference on artificial intelligence*.

WANG, Hao, Naiyan WANG et Dit-Yan YEUNG (2015). « Collaborative deep learning for recommender systems ». In : *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, p. 1235-1244.

WANG, Tianlu, Xi Victoria LIN, Nazneen Fatema RAJANI, Bryan MCCANN, Vicente ORDONEZ et Caiming XIONG (2020). « Double-Hard Debias: Tailoring Word Embeddings for Gender Bias Mitigation ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 5443-5453.

WEI, Xing et W Bruce CROFT (2006). « LDA-based document models for ad-hoc retrieval ». In : *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 178-185.

WU, Lingfei, Ian EH YEN, Fangli XU, Pradeep RAVIKUMAR et Michael WITBROCK (2018). « D2KE: From Distance to Kernel and Embedding ». In : *stat* 1050, p. 16.

WU, Lingfei, Ian En-Hsu YEN et al. (oct. 2018). « Word Mover's Embedding: From Word2Vec to Document Embedding ». In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium : Association for Computational Linguistics, p. 4524-4534. URL : https://www.aclweb.org/anthology/D18-1482.

Xu, Kelvin et al. (2015). « Show, attend and tell: Neural image caption generation with visual attention ». In : *International conference on machine learning*. PMLR, p. 2048-2057.

Yan, Xiaohui, Jiafeng Guo, Yanyan Lan et Xueqi Cheng (2013). « A biterm topic model for short texts ». In : *Proceedings of the 22nd international conference on World Wide Web*, p. 1445-1456.

Yang, Cheng, Zhiyuan Liu, Deli Zhao, Maosong Sun et Edward Chang (2015). « Network representation learning with rich text information ». In : *Twenty-fourth international joint conference on artificial intelligence*.

Yang, Xinli, David Lo, Li Li, Xin Xia, Tegawendé F Bissyandé et Jacques Klein (2017). « Characterizing malicious android apps by mining topic-specific data flow signatures ». In : *Information and Software Technology* 90, p. 27-39.

Zhang, Ce et Hady W Lauw (2020). « Topic modeling on document networks with adjacent-encoder ». In : *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 34. 04, p. 6737-6745.

Zhang, Fuzheng, Nicholas Jing Yuan, Defu Lian, Xing Xie et Wei-Ying Ma (2016). « Collaborative knowledge base embedding for recommender systems ». In : *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, p. 353-362.

Zhang, Li, Xiaoping Sun et Hai Zhuge (2015). « Topic discovery of clusters from documents with geographical location ». In : *Concurrency and Computation: Practice and Experience* 27.15, p. 4015-4038.

Zhu, Xinghua, Jianzong Wang, Zhenhou Hong et Jing Xiao (2020). « Empirical studies of institutional federated learning for natural language processing ». In : *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, p. 625-634.