Apprentissage de représentations d'auteurs et de documents : approches probabilistes à partir de représentations pré-entraînées.

Antoine Gourru

N° d'ordre NNT : 2021LYSE2103

THÈSE 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 23 novembre 2021, par :

Antoine GOURRU

Apprentissage de représentations d'auteurs et de documents.

Approches probabilistes à partir de représentations pré-entraînées.

Devant le jury composé de :

Christophe GRAVIER, Professeur des universités, Université Jean Monnet, Président
Marianne CLAUSEL, Professeure des universités, Université de Lorraine, Rapporteure
Benjamin PIWOWARSKI, Chercheur Première Classe, Sorbonne Université, Rapporteur
Lynda TAMINE-LECHANI, Professeure des universités, Université Toulouse 3, Examinatrice
Julien VELCIN, Professeur des universités, Université Lumière Lyon 2, Directeur de thèse
Julien JACQUES, Professeur des universités, 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 (UR 3083)
Discipline : Informatique

Apprentissage de représentations d'auteurs et de documents : approches probabilistes à partir de représentations pré-entraînées.

Antoine Gourru

Thèse soutenue publiquement le 23/11/2021 devant le jury composé de :

Marianne Clausel, Professeure des Universités, Université de Lorraine	Rapportrice
Benjamin Piwowarski, Chercheur Première classe, Sorbonne Université	Rapporteur
Christophe Gravier, Professeur des Universités, Université Jean Monnet	Examinateur
Lynda Tamine-Lechani, Professeure des Universités, Université Paul Sabatier, Toulouse	Examinatrice
Ian Davidson, Professeur Titulaire, Université de Californie - UC Davis	Invité
Julien Jacques, Professeur des Universités, Université Lumière Lyon 2	Directeur de Thèse
Julien Velcin, Professeur des Universités, Université Lumière Lyon 2	Directeur de Thèse

Résumé

La révolution numérique a entraîné une croissance exponentielle de la quantité d'informations stockées à long terme. Une part importante de cette information est textuelle (pages Web, médias sociaux, etc.). Les modèles de traitement du langage naturel (NLP), qui permettent de classer ou de regrouper cette information, ont besoin que le texte soit représenté sous forme d'objets mathématiques : on parle alors d'apprentissage de représentations. L'objectif de l'apprentissage de représentations est de construire des représentations d'objets textuels (mots, documents, auteurs) dans un espace vectoriel de faible dimension. La similarité entre les représentations vectorielles de ces objets devrait être liée à leur proximité sémantique ou à leur similarité stylistique.

En plus du texte lui-même, les documents sont souvent associés à des métadonnées. Ils peuvent être liés (par exemple, par des références hypertextes), associés à leurs auteurs, et horodatés. Il a été démontré que ces informations améliorent la qualité de la représentation d'un document. Néanmoins, l'incorporation de ces métadonnées n'est pas triviale. De plus, le langage naturel a rapidement évolué au cours des dernières décennies. Les modèles de représentation sont maintenant entraînés sur des quantités massives de données textuelles et affinés pour des tâches spécifiques. Ces modèles sont d'un grand intérêt lorsqu'on travaille avec de petits ensembles de données, permettant de transférer des connaissances à partir de sources d'information pertinentes. Il est crucial de développer des modèles d'apprentissage de représentations qui peuvent incorporer ces représentations pré-entraînées. La plupart des travaux antérieurs apprennent une représentation ponctuelle. C'est une limitation sérieuse car la langue est plus complexe que cela : les mots sont souvent polysémiques, et les documents sont, la plupart du temps, sur plusieurs sujets. Une branche de la littérature propose d'apprendre des distributions probabilistes dans un espace sémantique pour contourner ce problème.

Dans cette thèse, nous présentons tout d'abord la théorie de l'apprentissage automatique, ainsi qu'un aperçu général des travaux existants en apprentissage de représentations de mots et de documents (sans métadonnées). Nous nous concentrons ensuite sur l'apprentissage de représentations de documents liés. Nous présentons les travaux antérieurs du domaine et proposons deux contributions : le modèle RLE (Regularized Linear Embedding), et le modèle GELD (Gaussian Embedding of Linked Documents). Ensuite, nous explorons l'apprentissage des représentations d'auteurs et de documents dans le même espace vectoriel. Nous présentons les travaux les plus récents et notre contribution VADE (Variational Authors and Documents Embedding). Enfin, nous étudions la problématique de l'apprentissage de représentations dynamiques d'auteurs : leurs représentations doivent évoluer dans le temps. Nous présentons d'abord les modèles existants, puis nous proposons une contribution originale, DGEA (Dynamic Gaussian Embedding of Authors). De plus, nous proposons plusieurs axes scientifiques pour améliorer nos contributions, et quelques questions ouvertes pour de futures recherches.

Summary

"The digital revolution" led to an exponential growth of the quantity of long term stored *information*. Most of this information is textual (e.g., web pages, social media). Natural Language Processing (NLP) models, that allow to classify or to cluster textual objects, need the text to be represented as a mathematical object.

This is the goal of representation learning, which makes use of machine learning approaches to learn representations for textual objects (word, document, author) in a low dimensional vectorial space. The distance between the vectorial representations of these objects should relate with their semantic proximity or stylistic similarity.

In addition to the text itself, documents are often associated with meta data. They are often linked (e.g. hypertext references, citations in scientific production), tagged with their author, and timestamped. This information was shown to improve the quality of the documents representation and can help to guide the learning of the author representation. Nevertheless, incorporating these meta data is not trivial.

Additionally, NLP evolved rapidly during last decades. Representation models are now trained on massive amount of textual data and fine-tuned on specific tasks. These models are of high interest when working with small dataset, allowing to transfer knowledge from relevant information sources. Developing representation learning models that can incorporate these pre-trained representations is crucial. Most of prior works learn pointwise representation. This is a serious limitation as language is more complex than that : words are often polysemic, and documents are, most of the time on, several topics. A branch of the literature proposes to learn probabilistic distribution in a semantic space to circumvent this issue.

In this thesis, we first introduce theoretical of machine learning, and a general overview of existing works in representation learning for words, and documents (without meta data). We then focus on representation learning for linked documents. We present prior works of the domain and propose two contributions : the RLE (regularized Linear Embedding) model, and the GELD model (Gaussian Embedding of Linked Documents). Then, we explore representation learning for authors and documents in the same vector space. We present most recent works and our contribution VADE (Variational Authors and Documents Embedding). Finally, we study the problematic of representation learning for authors in a dynamic context : their representations should evolve over time. We first previous models, and propose an original contribution, DGEA (Dynamic Gaussian Embedding of Authors). Additionally, we propose several scientific axes to improve our contributions, and some open questions for future research.

Remerciements

Merci, d'abord, à mes deux directeurs de thèse. Vous m'avez fait assez confiance pour me laisser explorer toutes mes idées, sans jamais me fermer de portes, sans jamais douter. Merci Julien J., d'avoir toujours pris le temps de "regarder les maths" avec moi, et d'avoir fait preuve d'une exigence bienveillante, juste et constructive, tout au long de ma thèse. Votre encadrement m'a donné envie de devenir meilleur chercheur, et aussi un petit peu meilleur cycliste. Merci Julien V., je te dois beaucoup. C'est toi le premier, en cours de Java, à m'avoir fait sérieusement envisager la recherche, toi aussi le premier à m'avoir ouvert les portes du labo pour mes stages de master, puis en thèse. Merci également de m'avoir accompagné dans toutes les étapes de ma courte vie de chercheur, toujours avec énergie et sympathie, et pour tous les bons moments autour d'un verre ou d'une assiette. A vous deux, j'espère sincèrement que notre collaboration scientifique perdurera.

Merci aux membres du jury. Merci à ma rapportrice, Marianne. Merci à mon rapporteur Benjamin, merci également pour ton suivi en CSI, et pour tes questions et remarques qui m'ont donné plein de nouvelles idées à explorer. Merci à Lynda d'avoir accepté d'être examinatrice. Merci Christophe, tes relectures de certains de mes papiers ont eu un impact important, je suis ravi de te rejoindre à Saint-Étienne pour pouvoir collaborer plus concrètement sur les nombreux domaines qui nous intéressent tous les deux.

Merci aux membres permanents du laboratoire, qui ont été d'une compagnie extrêmement agréable. Merci plus particulièrement à Jairo Cugliari, Julien Ah-Pine et Stéphane Chrétien, vous m'avez été d'une aide précieuse quand je me perdais dans les maths. Merci Habiba, pour ta gentillesse et ton humour. Je voudrais également adresser un remerciement particulier à Adrien Guille. Merci de m'avoir guidé dans mes premiers pas de doctorant, et de m'avoir proposé de collaborer avec toi. Et merci aussi pour ces nombreuses heures à parler recherche, mais pas que, autour d'un Picon trop chargé, d'une terrine, d'une pastel de nata, d'un curry rouge, ou d'un tacos salade tomate oignon sauce blanche harissa gratiné chef.

Merci à mes collègues, doctorant.e.s, stagiaires, post-doctorant.e.s, et en particulier à Abderrezak, Amandine, Arwa, Camille, Carina, Clément, David, Gaël, Hugo, Javier, Loïc, Martial, Mathilde, Nicolas S., Redha, Rohit, el garca Nicolas Hernandez, Jean pour les pauses salvatrices, Robin pour nos nombreuses discussions embedding/les restos de l'espace chez Dani, et Enzo pour les sessions de bloc "veine sur le bibi". Margot, tu n'auras pas ton nom deux fois, mais un merci deux fois plus long. Merci d'avoir jeté un œil à mon manuscrit et pour tes super retours. Merci pour ces années à partager ton bureau, on s'est bien marré, on a bien papoté, on a bien travaillé, et tu as surtout été d'un très grand soutien. Je suis ravi que tu sois devenue mon amie !

Merci à mes ami.e.s, et particulièrement au noyau dur, le sang de la veine, mes gâtés : Jules, Julien, Luis et Michel, mais aussi à Pierre, qui me manque, merci aux autres, à Erwan, Oriane, Adeline,

Anthony, Ange, Kinda, Morgane, Valoche, Mélo, Bianca, Pierrot, Clémence, Aurore, Nina, Camille aka “l’enfant poulet”, Axel, Vaïana, Shaïsta, Damien, Julie, Migue, Adrian, Louise, Élise, DJ Max, Hiba, Théo, Clotilde, Max dit “de la tribu de Dana”, Aline, Ben, Andrew, Mathilde et Herbie. Et pour celles et ceux que j’ai oubliés, ne m’appelez plus.

Merci enfin à ma famille, plus particulièrement, à mes parents, pour m’avoir toujours soutenu sans condition. Merci à ma mère, toujours prête à tout pour son fils, à mon père pour ses relectures (et ses bons plats), à mes frères Benjamin et Nathan, pour ces moments ensemble qui m’ont permis de souffler un peu. Merci aussi à Sohalia, Enzo, Martine, Marie-Lou et Marina (et aussi à Manou). Un très grand merci à ma grand-mère, Mamie, la plus gentille personne sur terre, et à mon grand-père, Papy, qui m’a donné le goût de la science. Et merci à mes chats, Pépé et Bobby.

Merci enfin à Roxane. Tu m’as rencontré doctorant, pour qui la science n’attendait pas et malgré ça tu es restée. J’ai aimé chaque moment que j’ai passé avec toi, et j’attends avec impatience les prochains. Merci d’avoir été là pendant ces trois ans.

Table des matières

Chapitre 1

Introduction

1.1 Machine Learning pour le Texte : contexte et problématiques

1.1.1 Contexte

L'apprentissage automatique (*machine learning* en anglais), est un domaine scientifique dont l'objectif est d'extraire de l'information à partir de données. Les données, d'un point de vue informatique, sont des informations du monde réel qu'il est possible de représenter numériquement, de stocker et de manipuler avec un ordinateur. A partir de cette information, l'apprentissage automatique vise à construire des fonctions de décision. Ces fonctions ont pour but de prédire une valeur ou une classe à partir d'une annotation des données, on parle d'apprentissage supervisé. Elles permettent aussi d'extraire une forme d'organisation latente non observée des données, on parle alors d'apprentissage non supervisé.

Dans cette thèse, on s'intéresse à un type particulier de données : **les données textuelles numériques**. On effectue ici la distinction avec les données textuelles orales et écrites. De nombreuses méthodes permettent néanmoins de les retranscrire numériquement : on parle de reconnaissance automatique de la parole pour les premières et de reconnaissance optique de caractères pour les deuxièmes.

Index	Contenu	Classe
1	“Le machine learning entretient des liens étroits avec de nombreux domaines scientifiques”	1
2	“Démons et merveilles, vents et marées, au loin déjà la mer s’est retirée”	0
3	“Une orange sur la table, ta robe sur le tapis, et toi dans mon lit”	0

Tableau 1.1 – Exemples de documents courts, associés à des classes. Les documents 2 et 3 sont tirés de Prévert (1949)

Les données textuelles numériques sont produites massivement depuis l’avènement de l’informatique et d’un réseau mondial globalisé, le web. Elles sont nombreuses sur les médias sociaux, support de plus en plus prépondérant dans l’échange d’informations. De grandes campagnes de numérisation des archives rendent disponibles des textes anciens. Les journaux d’information ont presque tous un pendant numérique. Le code source des pages web forme un ensemble énorme de données textuelles. Les documents peuvent également être associés à différentes métadonnées telles que leurs auteurs ou leur date de publication. Dans certains cas, les documents peuvent aussi être attribués à des classes, comme le sujet général du document (par exemple, cinéma, musique, etc.), ou une mesure du sentiment ou de l’opinion portée par le texte (négatif, neutre, positif par exemple). Ces grands ensembles de documents, en raison de leur taille et de leur complexité, nécessitent d’être analysés de manière automatique. On parle de fouille automatique de texte ou de traitement automatique de la langue (TAL, en anglais NLP pour Natural Language Processing).

Le TAL est né en même temps que les ordinateurs : les travaux initiaux utilisaient principalement des systèmes basés sur des ensembles de règles prédéfinies, à l’instar du programme conversationnel ELIZA (Weizenbaum, 1966), un “robot psychologue”, ou les travaux en désambiguïsation du sens des mots de (Lesk, 1986) [1]. Plus récemment, le domaine se concentre sur le traitement statistique des données textuelles, s’inscrivant ainsi dans l’apprentissage automatique. Aujourd’hui, le paysage est largement dominé par les approches neuronales (Liu et al., 2019; Devlin et al., 2019), et plus précisément par les architectures dérivées du Transformer (Vaswani et al., 2017) que nous présenterons en sous-section 2.4.5. Les tâches principalement étudiées dans ce domaine d’analyse sont :

— la recherche d’information, ou RI. Elle consiste à retrouver une information particulière dans un corpus de documents. Par exemple, on peut chercher à déterminer le document le plus pertinent par rapport à une requête (par exemple, les recherches dans les moteurs de recherche comme Google), à une question en langage naturel formulée par un utilisateur, ou à déterminer les documents les plus proches sémantiquement ou stylistiquement d’un document donné.

— La génération automatique, utilisée dans les systèmes appelés chatbot, ou agent conversationnel, qui permettent de mener une discussion en langage naturel avec un système informatique, mais aussi dans les systèmes de question/réponse et de traduction automatique.

— La classification ou le partitionnement de documents. La classification inclut un domaine important nommé l’analyse de sentiment, qui consiste à identifier la valence émotionnelle d’un document.

La résolution de la plupart de ces tâches nécessite d’identifier une représentation mathématique des documents, et de l’information qu’ils contiennent, pour pouvoir les manipuler, les comparer ou

1. Il est intéressant de noter que ce dernier préfigure les travaux de (Mikolov et al., 2013b) qu’on présentera plus loin : il repose déjà sur l’hypothèse distributionnelle (Harris, 1954; Firth, 1957)

index	au	avec	dans	de	des	domaines	...	table	tapis	toi	une	vents	étroits
0	0	1	0	1	1	1	...	0	0	0	0	0	1
1	1	0	0	0	0	0	...	0	0	0	0	1	0
2	0	0	1	0	0	0	...	1	1	1	1	0	0

Tableau 1.2 – Matrice documents-termes du corpus du tableau 1.1

encore les indexer. On appelle ici documents des textes de longueur indifférenciée (phrase, paragraphe, article long), qui sont codés comme des séquences de caractères.

1.1.2 Représentation mathématique du texte

La plupart des algorithmes de machine learning manipulent des objets dans un espace euclidien, donc muni d'un produit scalaire. Par exemple, la méthode k-means (Lloyd, 1982), qui sépare des ensembles de vecteurs en sous-groupes homogènes, nécessite de calculer des distances euclidiennes entre les observations. Pour appliquer ces méthodes aux objets textuels, c'est-à-dire les documents ou les mots, il est donc nécessaire de leur construire une représentation vectorielle.

L'approche historique représente un document comme un vecteur sur l'ensemble du vocabulaire (Salton et al., 1975). On peut reprendre le corpus composé des trois phrases présentées en tableau 1.1 et construire la matrice documents-termes présentée en tableau 1.2. Chaque ligne correspond à un document, chaque colonne à un terme du vocabulaire. L'entrée en ligne i et colonne j est égale au nombre d'occurrences du j-ème mot du vocabulaire dans le document i.

Pour calculer la similarité entre deux documents, on peut utiliser plusieurs mesures. Les principales sont la distance euclidienne et la similarité cosinus, qu'on rappelle ici. Soit $Tf \in \mathbb{N}^{I \times J}$ la matrice documents-termes d'un corpus quelconque, composé de I documents et J mots. On note tf_i la i-ème ligne de cette matrice, qui est donc la représentation vectorielle du i-ème document, et $tf_{i,j}$ le j-ème élément de ce vecteur. La distance euclidienne entre deux documents est

$$||tf_i - tf_{i'}||_2 = \sqrt{\sum_{j=1}^{J} (tf_{i,j} - tf_{i',j})^2} \tag{1.1}$$

et la similarité cosinus est

$$cos(tf_i, tf_{i'}) = \frac{< tf_i, tf_{i'} >}{||tf_i||_2 ||tf_{i'}||_2} \tag{1.2}$$

avec $< tf_i, tf_{i'} >$ le produit scalaire entre les deux vecteurs, c'est-à-dire : $\sum_{j=1}^{J} tf_{i,j} tf_{i',j}$ et $||tf_i||_2$ la norme euclidienne de la i-ème ligne, c'est-à-dire : $\sqrt{< tf_i, tf_i >}$. Ces mesures ont été utilisées dans de nombreux cas pour résoudre des tâches diverses telles que la recherche d'information ou la classification.

Alternativement, on peut représenter les mots du vocabulaire de plusieurs façons. Les colonnes de la matrice Tf représentent les mots dans l'espace des documents. On peut ensuite comparer ces vecteurs au moyen des mesures introduites précédemment. Deuxièmement, la structure particulière en séquence des documents permet aussi de construire des matrices dites de cooccurrences. La matrice

	Distance euclidienne			Similarité cosinus		
	1.	2.	3.	1.	2.	3.
1. 'Une orange sur la table noire'	0	3.5	2.4	1	0	0.5
2. 'Le citron posé dans ma cuisine'	3.5	0	3.2	0	1	0.2
3. 'Une casquette orange sur ma tête'	2.4	3.2	0	0.5	0.2	1

Tableau 1.3 – Distance euclidienne et similarité cosinus deux à deux dans l'espace du vocabulaire pour trois documents. Le premier et le deuxième sont proches sémantiquement, mais leur distance dans cet espace est maximale, et leur similarité cosinus nulle.

de cooccurrences est de taille J par J et l'entrée d'indice (i, j) compte le nombre de fois où le mot i et le mot j sont apparus dans le même contexte : soit un document, soit une phrase, soit dans une fenêtre de k mots. On notera cette matrice C^+ dans le reste du manuscrit.

Une pondération est souvent appliquée aux matrices documents-termes de façon à donner moins d'importance aux mots très fréquents. En effet, les termes qui occurrent dans le plus de documents du corpus sont souvent peu informatifs, on parle de mots vides, ou *stop words*. Par exemple, les mots les plus courants dans cette thèse sont : "de", "la", "les", "des","et". La pondération *idf* (Jones, 1972) donne moins d'importance aux termes qui apparaissent dans beaucoup de documents :

$$idf_j = \log \frac{I}{\sum_i min(1, Tf_{i,j})}, \tag{1.3}$$

puis,

$$(Tf_{idf})_{i,j} = Tf_{i,j} \times idf_j. \tag{1.4}$$

Ces approches de représentation (documents-termes et matrice de cooccurrences) ont deux problèmes majeurs. Elles sont de taille très importante (plus précisément, c'est la dimension des variables qui pose problème) et ne construisent pas un espace réellement sémantique. Dans la prochaine sous-section, nous présenterons plus précisément ces deux problématiques avant d'étudier les méthodes permettant de les résoudre.

1.2 Apprentissage de représentation

1.2.1 Limites de l'approche historique de représentation

Le premier problème est la très grande dimension de représentation. Dans la langue française, le Larousse 2021 possède plus de 63500 mots. Début 2021, il existe 4 369 387 entrées en français dans le dictionnaire multilingue Wikitionnaire. Les matrices documents-termes et de cooccurrences sont ainsi en très grande dimension et extrêmement creuses (de nombreuses valeurs sont nulles). Plusieurs propriétés indésirables, regroupées sous le nom de fléau de la dimension, affectent particulièrement le comportement des métriques en grande dimension. Dans (Aggarwal et al., 2001), les auteurs montrent que la différence entre les distances maximales et minimales observées par rapport à un point donné augmente plus lentement que la distance minimale dans un espace de grande dimension. La notion de plus proche voisin perd donc son sens en très grande dimension. Ensuite, la complexité

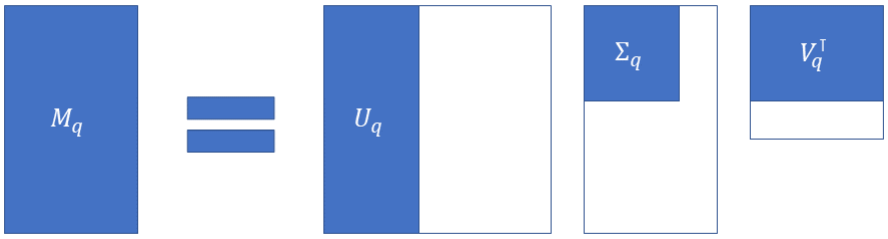


FIGURE 1.1 – Illustration de la SVD tronquée

des méthodes traditionnelles de machine learning est souvent fonction du nombre de variables. La méthode k-means (Lloyd, 1982) est en $\mathcal{O}(n^{dk+1})$, avec d la dimension des observations, k le nombre de classes et n le nombre d'individus. Le temps d'exécution augmente ainsi de façon exponentielle relativement à la dimension des données. Enfin, la taille des matrices qui représentent le texte peut nécessiter une place en mémoire importante et ainsi impliquer un fort coût de stockage.

Deuxièmement, l'espace de représentation n'est pas réellement sémantique : la proximité entre documents révèle seulement l'utilisation de termes similaires. Il ne rapproche pas deux documents portant des thématiques communes mais qui utilisent un vocabulaire différent. Par exemple, les phrases "Une orange sur la table noire" et "Le citron posé dans ma cuisine" ont une distance euclidienne maximale (racine de la somme de leurs tailles) et une similarité cosinus nulle, comme présenté en tableau 1.3. Deux mots synonymes auront une similarité presque nulle, car très rarement utilisés dans le même document.

On va donc utiliser des méthodes pour construire une nouvelle façon de représenter des objets textuels (documents, mots, auteurs) dans un espace sémantique de dimension réduite. Dans cet espace, les mots ou les documents sont représentés par des vecteurs. Deux documents, ou deux mots, qui sont proches auront un sens proche, ou seront au moins sémantiquement liés. Ces représentations servent de représentations intermédiaires, on parle parfois de *proxy*, pour résoudre de nombreuses tâches, comme la recommandation de documents, le clustering de mots ou la classification de documents.

Si historiquement, les premières approches pour contourner les limitations de la représentation documents-termes proposaient de réduire la dimension de ces matrices au moyen de méthodes classiques comme la décomposition en valeurs singulières, les approches les plus récentes s'inscrivent dans le domaine de l'apprentissage de représentation.

1.2.2 Réduction de dimension

La méthode historique propose de réduire la dimension de la matrice documents-termes en la factorisant. C'est l'Analyse Sémantique Latente (Deerwester et al., 1990). Elle repose sur la décomposition en valeurs singulières (SVD) de la matrice initiale des données. Cette décomposition factorise une matrice $M \in \mathbb{R}^{n \times r}$ en le produit de trois matrices $U\Sigma V^\top$ particulière. $U \in \mathbb{R}^{n \times n}$ et $V^\top \in \mathbb{R}^{r \times r}$ sont des matrices unitaires et Σ est diagonale positive. Les vecteurs en colonne de U sont les vecteurs singuliers gauches et les vecteurs colonne de V les vecteurs singuliers droits. La diagonale Σ contient les valeurs singulières. On peut construire une matrice de rang plus faible en

conservant seulement certains vecteurs singuliers. On parle alors de SVD tronquée, qui consiste à construire la matrice : $M_q = U_q \Sigma_q V_q^\top$, ou U_q, respectivement V, implique de conserver uniquement les q premières colonnes de U, respectivement V, et Σ_q conserve les q premières lignes et colonnes de Σ. Ce procédé est illustré en figure 1.1. Le théorème d'Eckart-Young stipule que la matrice M_q est la matrice de rang q qui minimise la norme de Frobenius $||M - M_q||_F$. Les vecteurs ligne de U_q sont utilisés comme représentations en dimension faible des documents du corpus, et les vecteurs colonne de la matrice V_q permettent de représenter les mots du vocabulaire.

Les approches les plus récentes, plutôt que de compresser la matrice documents-termes, proposent d'apprendre les représentations des objets textuels. On parle alors d'apprentissage de représentation.

1.2.3 Apprentissage de représentations : les approches modernes

L'apprentissage de représentations de mots a été largement étudié suite aux travaux de Mikolov et de ses collaborateurs (Mikolov et al., 2013b,a), même si des approches plus anciennes avaient déjà proposé de représenter les mots par des vecteurs (Bengio et al., 2003; Paccanaro and Hinton, 2000). Dans les modèles Word2Vec, l'apprentissage des représentations est guidé par la tâche de reconstruction du voisinage d'un mot ou du mot par rapport à son voisinage. La probabilité des mots de ce contexte localisé dans le document (une fenêtre de taille fixe autour d'un mot source), dépend du produit scalaire entre les représentations vectorielles des mots. Ainsi, plus les mots sont proches dans l'espace de représentation, plus ils auront de chance d'être observés dans la même fenêtre.

Ces modèles définissent des modèles de langue, c'est-à-dire des probabilités associées à chaque entrée du vocabulaire connaissant un contexte, qui dépendent des représentations des mots. Dans ces modèles, des opérations linéaires semblent corrélées avec certaines propriétés sémantiques. L'exemple célèbre énoncé dans (Mikolov et al., 2013b) montre que si on soustrait le vecteur associé au mot Allemagne au vecteur du mot Berlin, et qu'on ajoute la représentation vectorielle du mot France, on obtient un vecteur proche du vecteur représentant le mot Paris.

Dans les modèles Word2Vec, chaque mot du vocabulaire est associé à une représentation unique. Les approches les plus récentes construisent des représentations qui sont contextualisées au moyen d'architectures neuronales. Le modèle BERT (Devlin et al., 2019) propose d'utiliser les Transformers (Vaswani et al., 2017) que nous présenterons en deuxième chapitre. En sortie du réseau, un mot aura une représentation vectorielle qui dépend de son contexte d'observation (le document ou la phrase). Plus précisément, BERT manipule des unités au grain plus grossier, les *token*, qui sont des parties de mots communes dans le corpus, ce qui permet de réduire la taille du vocabulaire et donc le nombre de paramètres à apprendre. Ces approches ont rebattu les cartes du TAL : les modèles sont entraînés sur de grands corpus de données, et sont ensuite affinés (*fine-tuning*), c'est-à-dire optimisés, sur des tâches de transfert. De nombreux travaux démontrent la capacité de généralisation de ces modèles.

Concernant l'apprentissage de représentations de documents, les méthodes récentes reposent souvent sur des fonctions paramétriques d'agrégation des matrices des représentations vectorielles des mots (Conneau et al., 2017; Kiros et al., 2015; Cer et al., 2018; Reimers and Gurevych, 2019). Dans la prochaine sous-section, nous présenterons les limitations des méthodes existantes.

1.2.4 Limites de l'existant

Dans de nombreux cas, représenter les objets textuels comme des vecteurs n'est pas suffisant. Il convient d'ajouter une mesure de l'incertitude sémantique. Certains modèles proposent de représenter les mots et les documents comme des distributions de probabilités. Cela à de nombreux avantages, notamment celui d'ajouter une information qualitative utile lors de l'analyse des résultats en recommandation.

Nous avons dit précédemment que les documents étaient souvent associés à des métadonnées. De nombreuses méthodes ont montré l'intérêt d'intégrer cette information dans le processus d'apprentissage des représentations d'objets textuels. Dans cette thèse, nous nous sommes intéressés à trois types de métadonnées importants : le ou les auteurs d'un document, les liens entre documents (par exemple des liens de citations) et la date de publication des documents. Dans le cas où on incorpore l'information d'auteur, on cherchera à représenter les auteurs dans le même espace que les documents. Ensuite, l'ajout de la date permet d'apprendre des représentations dynamiques d'auteurs, c'est-à-dire des représentations qui changent au cours du temps et forment des trajectoires dans l'espace de représentation.

Nous nous sommes intéressés à l'utilisation de représentations pré-entraînées. Par exemple, une façon simple de représenter une phrase est de moyenner les représentations des mots de la phrase. L'intérêt principal est de pouvoir incorporer une connaissance externe : il est possible de récupérer des représentations de mots apprises sur de très grands corpus. Tous les modèles que nous présentons utilisent des représentations pré-entraînées, soit des mots, soit des documents.

Enfin, on parle de méthode transductive quand les représentations sont construites pour un ensemble d'objets fixé a priori, c'est-à-dire qu'on ne peut pas représenter un document ou un mot qui n'a pas été vu pendant l'entraînement. A l'inverse, les méthodes inductives construisent une fonction qui permet de représenter n'importe quel objet textuel, même s'il n'a pas été vu pendant la phase d'entraînement. Plusieurs méthodes proposées dans ce manuscrit ont la particularité d'être transductives notamment grâce à l'incorporation de représentations pré-entraînées.

1.3 Organisation du manuscrit et contributions

Dans le chapitre 2, nous présenterons les approches existantes d'apprentissage de représentations de mots et les documents. Au-delà de l'intérêt historique, cette partie permet d'introduire des concepts qui ont en partie inspiré les contributions présentées en chapitres 3, 4 et 5.

Dans le chapitre 3, nous détaillerons l'apprentissage de représentations de documents organisés en graphe, c'est-à-dire qu'il existe une information additionnelle permettant de construire des liens entre les documents. Après un état de l'art de ce domaine précis, nous présenterons deux contributions. Le modèle RLE [1] a été publié dans la conférence ECIR 2020. Le modèle GELD [2] a été publié dans la conférence IJCAI 2020.

Dans le chapitre 4, nous étudierons l'apprentissage de représentations d'auteurs. Après un état de l'art sur le sujet, nous présenterons le modèle VADE [4], dont une première version plus simple a été publiée dans la conférence EGC [3].

Dans le chapitre 5, nous introduirons l'apprentissage de représentations dynamiques d'auteurs. Nous présenterons les méthodes existantes, puis notre contribution, le cadre générique DGEA [5], et deux modèles qui découlent de ce cadre : le modèle K-DGEA et le modèle R-DGEA.

[1] Document network projection in pretrained word embedding space. Gourru, A., Guille, A., Velcin, J., Jacques, J., ECIR 2020

[2] Gaussian Embedding of Linked Documents from a Pretrained Semantic Space. Gourru, A., Velcin, J., Jacques, J., IJCAI 2020

[3] Apprentissage Conjoint de Représentations d'Auteurs et de Documents. Gourru, A., Yadav, R., Velcin, J., Extraction et Gestion des Connaissances : Actes EGC 2021.

[4] Variational Embedding of Authors and Documents. Gourru, A., Yadav, R., Velcin, J.,Jacques, J., en cours de soumission.

[5] A General Framework for Dynamic Gaussian Embedding of Authors. Gourru, A., Velcin, J.,Jacques, J., en cours de soumission

1.4 Acronymes

- AR : Apprentissage de représentation
- SVD : Singular Value Decomposition (décomposition en valeurs singulières)
- VAE : Variational Auto-Encoder (auto-encoder variationnel)
- MLP : MultiLayer Perceptron (perceptron multicouche)
- DAN : Deep Averaging Network
- EM : Expectation Maximisation
- VEM : Variational Expectation Maximisation
- VIB : Variational Information Bottleneck
- Noms de modèles :
  - BERT : Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)
  - TADW : Text Attributed DeepWalk (Yang et al., 2015)
  - SGNS : Skip Gram with Negative Sampling (Mikolov et al., 2013b)
  - CBOW : (Mikolov et al., 2013a)
  - AANE : Accelerated Attributed Network Embedding Huang et al. (2017)
  - STNE : Self-Translation Network Embedding (Liu et al., 2018)
  - IDNE : Inductive Document Network Embedding (Brochier et al., 2020)
  - GVNR : Global Vector for Node Representation (Brochier et al., 2019)
  - VGAE : Variational Graph Auto-Encoder (Kipf and Welling, 2016)
  - RLE : Regularized Linear Embedding (Gourru et al., 2020a)
  - GELD : Gaussian Embedding of Linked Document (Gourru et al., 2020b)
  - PAD : Projection d'Auteurs et de Documents
  - VADE : Variationnal Author and Document Embedding (Gourru et al., 2021)
  - DGEA : Dynamic Gaussian Embedding of Authors

1.5 Notations

— $x_{1:n} : \{x_1, x_2, ..., x_n\}$

— $f(x; \theta)$: fonction d'une variable x et de paramètres θ.

— $\nabla_x f(x)$: gradient de la fonction f par rapport à x, c'est-à-dire le vecteur des dérivées partielles en chaque coordonnée de x.

— x_i la i-ème ligne d'une matrice X quelconque

— $x_{i,j}$ ou $X_{i,j}$: j-ème élément de la i-ème ligne de X, une matrice quelconque.

— $X^\top$: transposée de la matrice X.

— $tr(X)$: traces de la matrice X.

— Soit X une matrice de taille $I \times J$, on note X_{l1} la normalisation telle que la $i$ème ligne vaut : $(x_{l1})_i = x_i / ||x_i||_1 = x_i / \sum_{j=1}^{J} |x_{i,j}|$ et X_{l2} la normalisation avec : $(x_{l2})_i = x_i / ||x_i||_2$.

— $\sigma(x)$: fonction sigmoïde, définie par $\sigma(x) = \frac{1}{1+\exp(-x)}$.

— T : nombre total de tranches temporelles indexées par t.

— $x^{(t)}$: x est observé dans la t-ème tranche.

— Probabilités :

  — $p(x; \theta)$: notation générale de la probabilité de x par rapport à une loi de probabilité de paramètres θ. Nous utilisons également cette notation pour la vraisemblance d'une observation, c'est-à-dire la valeur de la loi pour une réalisation de x, et donc une fonction de θ. Nous noterons également la vraisemblance $\mathcal{L}(\theta; x)$ pour expliciter certaines démonstrations.

  — $\mathcal{N}(\mu, \Sigma)$: loi normale de paramètres μ la moyenne et Σ la matrice de variance-covariance. On note $\mathcal{N}(x; \mu, \Sigma)$ la vraisemblance/probabilité de l'observation x.

  — $\mathcal{B}(\lambda)$: loi de Bernoulli de paramètre λ

  — $\mathcal{P}(\lambda)$: loi de Poisson de paramètre λ

  — $\mathcal{M}(\lambda)$: loi multinomiale de paramètre λ

  — $\mathcal{W}^{-1}(\Psi, \nu)$: loi de Wishart inverse de paramètres Ψ et ν

  — $\mathcal{D}ir(\pi)$: loi de Dirichlet de paramètre π

  — $\mathbb{E}[x]$ ou $\mathbb{E}_{p(x)}[x]$: espérance de x par rapport à la loi $p(x)$.

— Données textuelles :

  — Tf : matrice documents-termes, de dimension $I \times J$

  — $tf_{i,j}$: nombre d'occurrences du j-ème mot dans le i-ème document.

  — Tf_{idf} : matrice T pondérée par idf (*inverse document frequency*)

  — C^+ : matrice de cooccurrences, avec $C^+_{i,j}$: nombre de cooccurrences observées entre les mots d'indices i et j et $C^+_i = \sum_j C^+_{ij}$.

  — C^- : matrice de cooccurrences négatives, donc C^-_{ij} le nombre de fois ou j a été tiré comme exemple négatif pour une paire contenant i.

  - $\mathcal{C} : \{C^+, C^-\}$
  - l_i : taille du i-ème document du corpus
  - $d_i = f_{map}(\mathcal{D}_i, cont; \theta)$: représentation vectorielle d'un document, avec $cont$ le contexte d'observation (un ensemble de mots, de documents, etc.).
  - $\mathcal{D}_i^w$: séquence des représentations vectorielles des mots du document
  - $\mathcal{D}_{i,j}^w$ le j-ème élément de cette séquence.
  - $\mathcal{D}_i$: i-ème document du corpus, donc $(w^{i,1}, w^{i,1}, ..., w^{i,l_i})$, ou $w^{i,j}$
- Graphes :
  - V : l'ensemble des noeuds
  - E : l'ensemble des arêtes ou arcs
  - $vol(G)$: le volume du graphe
  - $deg(i)$: le degré du noeud i
  - A : la matrice d'adjacence
  - A_{l1} : la matrice d'adjacence normalisée en ligne
  - S : matrice de similarité quelconque
  - $\mathcal{E}_i$: voisinage du noeud i
- Fonctions :
  - $f_{map}(w, cont; \theta)$: fonction de paramètres θ qui associe une entité w (mot, document ou auteur) à un vecteur en fonction d'un contexte $cont$. Quand $f_{map}(w, cont; U) = f_{map}(w; U)$ et U est une matrice contenant les représentations de chaque élément, on notera simplement $f_{map}(w, cont; U) = u_i$, ou i est l'index de w dans le *vocabulaire* des entités.
  - $f_{mlp}(x; \theta)$: réseaux de neurones multi-couches
  - $f_{rnn}(x; \theta)$: réseau de neurones récurrents
  - $f_{att}(x, y; \theta)$: fonction d'attention
  - $f_{sdp}(x, y; \theta)$: fonction d'attention de type *scaled dot product*
  - $f_{asdp}(x; \theta)$: fonction de *self-attention* de type *scaled dot product*
  - $f_{rnn}(x; \theta)$: réseau de neurones récurrents
  - relu(x) : fonction Unité Linéaire Rectifiée
  - tanh(x) : fonction tangente hyperbolique
  - $[a, b]$: concaténation du vecteur a et du vecteur b

Chapitre 2

État de l'art

2.1 Introduction

Dans cette section, nous allons tout d'abord présenter un ensemble de notions nécessaires à la compréhension des contributions présentées dans ce manuscrit. Nous présenterons d'abord l'apprentissage automatique, incluant la différence entre approches supervisées et non supervisées, puis quelques détails techniques primordiaux en apprentissage automatique (fonction de perte,

optimisation, régularisation). Ensuite, nous présenterons les approches qui utilisent des modèles probabilistes. Nous détaillerons plusieurs algorithmes d'apprentissage des paramètres de ces modèles : le maximum de vraisemblance, l'algorithme EM et les approches variationnelles, qui sont utilisées dans le modèle GELD (Chapitre 3), VADE (Chapitre 4) et DGEA (Chapitre 5). Nous présenterons ensuite un état de l'art en apprentissage de représentations d'objets textuels. Nous présenterons d'abord les approches de représentations de mots, incluant l'apprentissage de densités gaussiennes, dynamiques et contextualisées. Ces travaux sont utilisés dans toutes nos contributions. Enfin, nous présenterons les travaux en apprentissage de représentations de documents, incluant les approches historiques, les méthodes récentes par agrégation, puis les méthodes qui représentent les documents par des densités. Ces travaux sont intégrés aux approches VADE et DGEA.

2.2 Apprentissage automatique

Nos travaux s'intègrent dans le domaine de l'apprentissage de représentation, qui fait lui-même partie de l'apprentissage automatique. L'apprentissage automatique, ou *machine learning* entretient des liens étroits avec de nombreux domaines scientifiques, principalement l'optimisation, les statistiques, les probabilités et l'informatique. Son objectif général est d'apprendre des fonctions de décision par rapport à un ensemble d'observations, ou données. Une donnée, ou observation, notée x, peut prendre des formes multiples : vecteur dans $\mathbb{R}^p$, image, texte, son. Les données prennent autant de formes qu'il existe de problématiques. Pour simplifier, on dira qu'on cherche à déterminer un ensemble de paramètres $\theta = \{\theta_e\}_{e=1}^{E}$ tels que ce que retourne un ensemble de fonctions générales $\{f(x;\theta_e)\}_{f=1}^{F}$ soit :

— *informatif*, c'est-à-dire qu'il extrait une information qui n'est pas directement observable, comme une forme de régularité des données (on parle parfois de pattern),

— *exact*, dans le sens où l'on veut qu'elles modélisent le plus fidèlement les données, ou leurs étiquettes. L'apprentissage des paramètres de ces fonctions nécessite donc de définir une fonction objectif qui mesure l'adéquation entre les sorties de ces fonctions et ce que l'on sait des données. On manipule des fonctions de perte et de fonctions de vraisemblance qu'on regroupe sous le nom de fonctions objectif.

On sépare historiquement l'apprentissage automatique en deux sous-ensembles de méthodes : l'apprentissage supervisé et non-supervisé, qu'on va détailler dans la prochaine sous-section.

2.2.1 Apprentissage supervisé et non-supervisé

Si chaque observation est associée à une étiquette, ou *label*, l'objectif est que $f(x;\theta)$ (ici la fonction d'intérêt est unique) prédise l'étiquette d'une observation. On peut calculer une mesure d'erreur, qui est d'autant plus grande que la sortie de $f(x;\theta)$ s'éloigne de l'étiquette réelle d'une observation. On cherchera ainsi à déterminer des paramètres θ qui minimisent l'erreur sur un ensemble de données. On parle ici d'apprentissage supervisé. Par exemple on peut étudier le cas de la classification de texte. Chaque donnée textuelle est associée à un label $y = 1$ s'il parle de machine learning, $y = 0$ sinon, comme illustré en tableau 1.1. On peut par exemple calculer l'erreur pour un

couple (x_i, y_i) associé au i-ème individu statistique du jeu de donnée :

$$\text{Erreur}(\theta; (x_i, y_i)) = |f(x_i; \theta) - y_i|. \tag{2.1}$$

Dans la phase d'apprentissage, on cherchera à minimiser une agrégation de cette quantité sur un ensemble d'observations, appelé ensemble d'entraînement. On peut utiliser par exemple la somme de l'erreur calculée pour chaque couple. L'agrégation sur un échantillon est nommée généralement la perte, ou *loss* en anglais.

La seconde famille d'apprentissage est dite non-supervisée. Elle s'intéresse aux cas où les données ne sont pas associées à des étiquettes. On définit alors généralement un modèle des données et on cherche les paramètres θ, ici du modèle, qui rendent les observations vraisemblables par rapport au modèle choisi. La plupart des algorithmes d'apprentissage non supervisé, dont le clustering, identifient deux fonctions : $f_{enc}(x, \theta)$ qui compresse les données initiales, une seconde $f_{dec}(z, \theta)$ qui les décompresse. On peut calculer une erreur entre la donnée telle que retournée par la succession de l'action de compression et de décompression à la donnée originale. Par exemple, on parle de *clustering* quand on identifie un nombre réduit de groupes, et qu'on assigne chaque observation à un ou plusieurs groupes, tel que les données associées au même groupe soient proches par rapport à une mesure définie a priori. Par exemple, dans l'algorithme K-Means (Lloyd, 1982), la fonction de compression associe une observation dans $\mathbb{R}^p$ à un groupe représenté par son centroïde, parmi un nombre E fini de groupes. Les centroïdes $\{\mu_e\}_{e=1}^{E}$ vivent dans le même espace que les observations. La décompression est la fonction identité : on va donc simplement mesurer l'erreur comme la distance euclidienne entre l'observation initiale et le centroïde. Similairement au cas supervisé, on minimise la perte calculée sur un ensemble d'entraînement par rapport aux paramètres, i.e. les centroïdes. Pour faire le lien avec ce qui a été dit précédemment, la composition des fonctions d'encodage et de décodage doit être exact, c'est-à-dire reconstruire x sans erreur, et l'encodage doit être informatif : révéler une notion d'organisation latente des données. Nous présenterons dans la prochaine sous-section quelques détails techniques concernant l'optimisation de ces différentes fonctions.

2.2.2 Optimisation

Nous avons évoqué précédemment la minimisation, respectivement maximisation, d'une fonction d'erreur, respectivement de vraisemblance, par rapport à un ensemble de variables θ pour un ensemble d'observations qu'on a appelé données d'entraînement. On parle plus généralement d'optimisation, qui est un processus complexe. Nous présenterons ici seulement le cas de la minimisation, mais la maximisation d'une fonction est équivalente, sachant qu'elle revient à minimiser l'opposé de la fonction. Pour un ensemble d'observations $X = x_{1:n} = \{x_1, x_2, ..., x_n\}$, on note l'erreur totale $J(\theta; X)$, qui est à valeur dans $\mathbb{R}$. On la calcule alors comme :

$$J(\theta; X) = \sum_{i=1}^{N} \text{Erreur}(\theta; x_i) \tag{2.2}$$

Le minimum global de $J(\theta; X)$ est la valeur θ^* pour laquelle :

$$\forall \theta, J(\theta; X) \geq J(\theta^*; X) \tag{2.3}$$

On cherche en machine learning à identifier cette valeur, ou, au moins une valeur de minimum local dans le cas de fonctions non convexes, c'est-à-dire qu'on observe la propriété précédente sur une partie du domaine de la fonction (au voisinage du minimum). Ces minimums sont des points critiques, ou stationnaires, c'est-à-dire que, si la fonction est dérivable, la dérivée vaut 0 en ces points. C'est la condition nécessaire de premier ordre. Dans le cas d'un minimiseur, global ou local, on a aussi la propriété que la matrice hessienne est définie positive sur l'intervalle considéré. On peut donc minimiser la fonction de perte si on arrive à identifier analytiquement la valeur θ^* telle que le gradient soit nul, après avoir vérifié que la hessienne est définie positive sur l'intervalle. Dans certains cas, on a recours à des approches numériques, comme l'approche de la descente du gradient.

L'algorithme de la descente de gradient consiste à identifier une valeur initiale de θ, puis la mettre à jour de façon itérative en suivant la règle suivante, avec η une valeur réelle appelée taux d'apprentissage :

$$\theta^{new} = \theta^{old} - \eta \nabla_\theta J(\theta^{old}; X) \tag{2.4}$$

jusqu'à ce que $J(\theta; X)$ converge, c'est-à-dire que la différence entre la valeur de $J(.; X)$ d'une mise à jour à l'autre soit inférieure à un seuil choisi a priori. ∇_θ indique qu'on calcule le gradient de la fonction par rapport à θ. On est assuré de minimiser la fonction si on prend un taux d'apprentissage suffisamment petit. L'algorithme de la descente de gradient stochastique permet d'optimiser les fonctions de pertes qui s'écrivent sous la forme $J(\theta; X) = \sum_{i=1}^{N} Erreur(\theta; x_i)$. On met alors à jour les paramètres en calculant la dérivée de la perte pour chaque observation, ou pour chaque sous-ensemble de taille fixe des données totales qu'on appelle *batch*.

On introduit en plus régulièrement des contraintes additionnelles à la fonction de perte. Ces contraintes peuvent prendre des formes multiples. La plus commune consiste à limiter la norme des paramètres. Si les paramètres sont tous des vecteurs dans $\mathbb{R}^p$, et que J est une mesure d'erreur à valeur dans $\mathbb{R}$:

$$\arg\min_\theta J(\theta; X) + \sum_{e=1}^{E} ||\theta_e||_q. \tag{2.5}$$

On parle alors de régularisation. La norme L1 entraînera les valeurs de θ à être creuses (certaines valeurs seront nulles), la norme L2 limitera simplement l'amplitude des valeurs de θ.

Les différentes approches du machine learning se différencient par la forme des fonctions d'intérêt et la méthode d'optimisation. Une famille particulière de fonctions d'intérêt est devenue centrale ces dernières années : les réseaux de neurones artificiels, qu'on va présenter dans la prochaine sous-section.

2.2.3 Les réseaux de neurones

L'architecture la plus simple est le perceptron multicouche, ou *MLP* (Multi Layer Perceptron). On définit d'abord la fonction d'une *couche* de m neurones, dont les paramètres θ sont une matrice W_e dans $\mathbb{R}^{m\times(p+1)}$. Les entrées sont des vecteurs x dans $\mathbb{R}^p$. Les vecteurs sont considérés en colonnes. On note x_b le vecteur x auquel on a ajouté une dimension qui vaut toujours 1. En sortie on obtient un vecteur dans $\mathbb{R}^m$:

$$f_e(x; W_e) = h(W_e \cdot x_b). \tag{2.6}$$

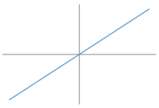
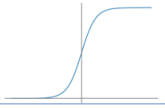
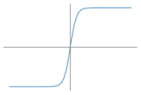
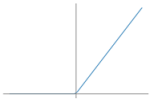
Nom	Fonction	Courbe
Linéaire	$h(x) = x$	
Sigmoïde	$h(x) = \frac{1}{1+e^{x}}$	
Tanh	$h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$h(x) = \max(0, x)$	

FIGURE 2.1 – Quelques fonctions d'activation usuelles.

h est une fonction qu'on appelle fonction d'activation. Elle prend plusieurs formes, les plus communes étant la fonction linéaire, la fonction sigmoïde, la fonction tanh et la fonction ReLU. Ces fonctions sont décrites dans la figure 2.1. Un perceptron multicouche à E couches, dans sa forme fonctionnelle, calcule une composition de ces fonctions, ou la fonction finale f_E dépend de l'espace de sortie désiré :

$$f_{mlp}(x;\theta) = (f_1 \circ f_2 \circ ... \circ f_E)(x). \tag{2.7}$$

Comme précédemment, on mesure l'adéquation entre la sortie de cette fonction et ce qu'on sait des données. Dans le cas supervisé avec deux classes, la dernière couche comporte un seul neurone et l'activation est la fonction sigmoïde. On calcule ensuite l'erreur pour un jeu de données d'entraînement en utilisant l'entropie croisée binaire par exemple :

$$J(\theta = \{W_e\}_{e=1}^{E}; X, Y) = \sum_{i=1}^{N} -\left(y_i \log\left(f_{mlp}(x_i;\theta)\right) - (1-y_i)\log\left(1 - f_{mlp}(x_i;\theta)\right).\right) \tag{2.8}$$

La fonction de perte est la plupart du temps minimisée au moyen de la descente de gradient stochastique par batch. Les fonctions de pertes les plus courantes sont l'erreur quadratique moyenne $\frac{1}{n}\sum_i ||f_{mlp}(x_i) - y_i||_2^2$ et l'entropie croisée $\sum_i y_i \log f_{mlp}(x_i)$. Dans ce dernier cas, la fonction de la dernière couche de f_{mlp} est une softmax, ou n'importe quelle fonction qui permet de retourner un vecteur de probabilité sommant à 1, et y_i est lui aussi un vecteur de valeurs entières sommant à 1. La forme particulière de la fonction d'un réseau de neurones multicouches permet de dériver facilement l'erreur au moyen de la règle de la dérivation en chaîne et d'utiliser la méthode de la rétropropagation du gradient, qu'on ne détaillera pas ici.

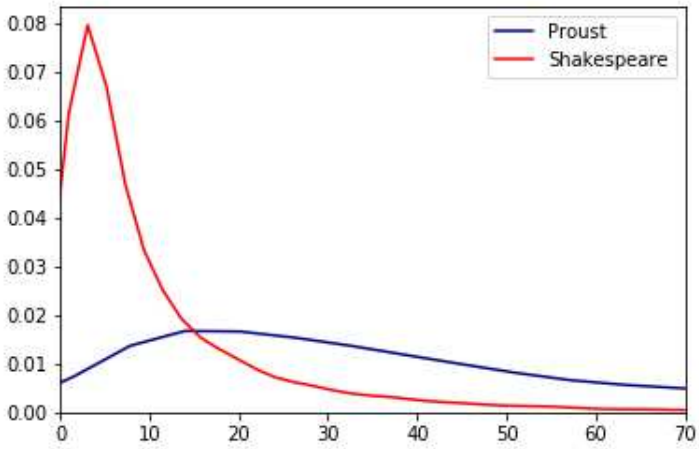


FIGURE 2.2 – Distribution des tailles de phrases pour trois textes de Shakespeare (Hamlet, le roi Lear, et Roméo et Juliette) vs Proust (à la recherche du temps perdu tome 1 et 2)

Les réseaux de neurones forment une grande famille de fonctions, ou architectures, qui possède de nombreuses propriétés pertinentes. Notamment, leur entraînement, c'est-à-dire la détermination des paramètres qui minimise la fonction de perte, passe mieux à l'échelle que d'autres approches d'apprentissage, en raison de l'utilisation d'une optimisation par batch. Ensuite, l'amélioration des architectures de calcul de type GPU/TPU accélère nettement la phase d'apprentissage. Les produits matriciels sont largement parallélisés grâce à de nombreuses librairies de calcul bas niveau. Les modèles deviennent ainsi de plus en plus complexes, et donc flexibles, permettant de généraliser à partir de très grands ensembles d'entraînement.

Dans cette section nous avons présenté les principes fondamentaux de l'apprentissage statistique, ou machine learning, ainsi qu'un type particulier de fonction de décision, les réseaux de neurones. Dans la prochaine partie, nous présentons les modèles probabilistes d'apprentissage. Ils permettent de modéliser les observations comme des variables aléatoires, issues d'un ensemble de paramètres, de variables latentes et de lois de probabilité en interaction.

2.3 Modélisation probabiliste

Dans le cadre de la modélisation probabiliste, les données sont issues d'un processus stochastique plus ou moins complexe. Ce processus met en jeu des variables qui ne sont pas directement observées. On parle de variables latentes si elles sont elles-mêmes issues de processus stochastique, de paramètres sinon. Dans nos travaux, les représentations vectorielles d'objets textuels sont considérées comme des réalisations de variables aléatoires, qui dépendent de paramètres non observés : les représentations des objets textuels dans l'espace sémantique latent.

Pour définir les meilleurs paramètres, on a recours généralement à la méthode du maximum de vraisemblance.

2.3.1 Le maximum de vraisemblance

On note l'ensemble des observations : $X = x_{1:n} = \{x_1, x_2, ..., x_n\}$, appelées aussi évidences. On note généralement les variables latentes $Z = z_{1:m}$. Les lois d'intérêts sont $p(X|Z)$ et $p(Z)$ qui sont en général des choix de modélisation. La plupart du temps, les observations sont considérées indépendantes et identiquement distribuées, c'est-à-dire que la probabilité d'un échantillon est définie par :

$$p(X;\theta) = \prod_{i}^{n} p(x_i;\theta). \tag{2.9}$$

Alternativement, la fonction de θ, à échantillon X fixé, est appelée vraisemblance. On la note $\mathcal{L}(\theta; X)$ pour un échantillon. Elle mesure l'adéquation des paramètres aux données observées, qui est d'autant plus élevée que les données sont probables au regard des paramètres. C'est la fonction qu'on va maximiser lors de la phase d'apprentissage. Elle a une forme fonctionnelle équivalente à la loi d'un échantillon, on les utilisera donc sans distinction dans certaines démonstrations. On note la vraisemblance pour une donnée $\mathcal{L}(\theta; x)$ si une loi est définie pour chaque observation. On manipule souvent le logarithme de la vraisemblance, plus facile à dériver. Dans la phase d'apprentissage, on va donc maximiser :

$$\arg\max_{\theta} \log \mathcal{L}(\theta; X) = \sum_{i}^{n} \log \mathcal{L}(\theta; x_i). \tag{2.10}$$

Dans le cas où on introduit l'ensemble des variables latentes (ici continues) :

$$p(X;\theta) = \int \prod_{i}^{n} p(x_i|Z;\theta)p(Z)d_Z. \tag{2.11}$$

En Figure 2.2, on affiche la distribution des tailles des phrases d'un échantillon de la production littéraire de deux auteurs. Les distributions sont bien différentes. A partir de ces mesures de taille, on peut construire un modèle "jouet" à titre d'illustration : si on ne connaît pas l'auteur d'une phrase et qu'on observe la distribution de toutes les données, on peut modéliser la taille des phrases au moyen d'un processus stochastique. La taille d'une phrase quelconque est x. Chaque phrase est associée à une variable latente $z \in \{0, 1\}$, qui n'est pas observée. On a ensuite $\theta = \{\theta_1, \theta_2, \theta_3\}$ l'ensemble des paramètres qui sont des scalaires. On note $\mathcal{B}$ la loi de Bernoulli et $\mathcal{P}$ la loi de Poisson. On peut définir le processus génératif suivant pour la taille d'une phrase :

$$\begin{aligned} z_i &\sim \mathcal{B}(\theta_1) \\ \lambda &= z_i\theta_2 + (1 - z_i)\theta_3 \\ x_i &\sim \mathcal{P}(\lambda). \end{aligned}$$

On peut ensuite exprimer la vraisemblance et la maximiser pour un ensemble d'entraînement, et donc identifier les phrases écrites par les mêmes auteurs (Shakespeare ou Proust).

La présence de variables latentes rend souvent la vraisemblance difficile à calculer, et donc à maximiser en θ. De nombreux modèles nécessitent d'avoir recours à l'algorithme Expectation Maximisation (EM) (Dempster et al., 1977), qui est itératif.

2.3.2 L'algorithme EM

On peut décomposer la vraisemblance (Bishop, 2006c), avec $q(Z)$ une loi sur les variables latentes. On commence par développer la divergence de Kullbach-Liebler entre $q(Z)$ et $p(Z|X)$

$$\begin{aligned} KL(q(Z)||p(Z|X)) &= \int q(Z) \log \frac{q(Z)}{p(Z|X)} d_Z \\ &= \int q(Z) \log \frac{q(Z)p(X)}{p(X,Z)} d_Z \\ &= \int q(Z) \log \frac{q(Z)}{p(X,Z)} d_Z + \int q(Z) \log p(X) d_Z \\ &= \int q(Z) \log \frac{q(Z)}{p(X,Z)} d_Z + \log p(X) \int q(Z) d_Z \\ &= \int q(Z) \log \frac{q(Z)}{p(X,Z)} d_Z + \log p(X) \\ &= -\mathcal{E}(q) + \log p(X) \end{aligned} \tag{2.12}$$

et donc :

$$\begin{aligned} \log p(X) &= KL(q(Z)||p(Z|X)) + \mathcal{E}(q) \\ &= KL(q(Z)||p(Z|X)) + \mathbb{E}_{q(Z)}[\log(\frac{p(X,Z;\theta)}{q(Z)})]. \end{aligned} \tag{2.13}$$

$\mathcal{E}(q)$ est une borne inférieure de $\log p(X)$, appelée ELBO, et est maximale à θ fixé si $KL(q(Z)||p(Z|X)) = 0$, c'est-à-dire que $q(Z) = p(Z|X)$. On appelle la vraisemblance complétée la fonction des paramètres et des variables latentes à X fixés qu'on note $\mathcal{L}(\theta, Z; X)$ et qui est équivalente à la probabilité jointe $p(X, Z; \theta)$. L'algorithme EM consiste à alterner les deux étapes suivantes, avec θ^{old} la valeur de θ à l'itération précédente :

— E step : calculer $p(Z|X; \theta^{old})$

— M step : calculer $\theta^{new} = \arg\max_\theta \mathbb{E}_{p(Z|X;\theta^{old})}[\log(\frac{\mathcal{L}(\theta,Z;X)}{q(Z)})]$

Or $\mathbb{E}_{p(Z|X;\theta^{old})}[\log q(Z)]$ dépend seulement de θ^{old}, on doit donc calculer uniquement :

$$\theta^{new} = \arg\max_\theta \mathbb{E}_{p(Z|X;\theta^{old})}[\mathcal{L}(\theta, Z; X)]. \tag{2.14}$$

Malheureusement, $p(Z|X; \theta)$ n'est parfois pas calculable en forme close. Plusieurs approches permettent néanmoins d'optimiser la vraisemblance, au moyen de tirages (les méthodes de Monte-Carlo par chaînes de Markov (Bishop, 2006a)), ou en identifiant une borne inférieure de la vraisemblance. Dans ce deuxième cas, on résout un problème alternatif, qui assure néanmoins de faire croitre la valeur de la vraisemblance. C'est l'inférence variationnelle (Jordan et al., 1999), que nous présenterons dans la prochaine sous-section.

2.3.3 L'inférence variationnelle

On choisit d'abord une famille de distributions $q(Z;\lambda)$, où λ sont les paramètres variationnels. A θ fixés, on maximise $\mathcal{E}(q)$ par rapport à λ. Cela a pour effet de minimiser $KL(q(Z)||p(Z|X))$, $\log p(X)$ étant fixe, et donc de rapprocher $q(Z)$ de $p(Z|X)$. On alternera la maximisation en λ et en θ de façon à faire croître la vraisemblance des données (algorithme VEM).

L'utilisation de familles de distribution simples permet de faciliter l'étape d'apprentissage. Par exemple, la famille est souvent choisie pour se factoriser sur toutes les variables Z : $q(Z;\lambda) = \prod_i^m q(z_i;\lambda)$. On appelle ça l'approximation en champs moyens.

Enfin, des approches numériques permettent d'obtenir des approximations de la ELBO, qui peut être, elle aussi, difficile à calculer en raison de l'intégrale introduite par le calcul de l'espérance. L'approche consiste à réaliser L tirages aléatoires de z suivant q

$$\mathbb{E}_q[\log(\frac{p(X,Z;\theta)}{q(Z)})] \approx \frac{1}{L}\sum_{l=1}^{L}\log(\frac{p(X,Z^{(l)};\theta)}{q(Z^{(l)})}),\ Z^{(l)} \sim q(Z;\lambda). \tag{2.15}$$

Kingma and Welling (2014) proposent une manière particulière de tirage qui permet dans des cas simples, de réaliser la maximisation de la vraisemblance au moyen d'un réseau de neurones de type auto-encoder : l'astuce de la reparamétrisation. $q(Z;\lambda)$ peut être "reparamétrée" par $g(X,\epsilon;\lambda)$, différentiable en X avec ϵ une variable aléatoire auxiliaire (Kingma and Welling, 2014). On va remplacer

$$Z \sim q(Z;\lambda) \tag{2.16}$$

par

$$Z = g(X,\epsilon;\lambda),\ \epsilon \sim p(\epsilon). \tag{2.17}$$

Dans le cas d'une loi normale avec variance diagonale par exemple, où chaque observation x_i est associée à une variable latente z_i, on a :

$$z \sim q(z;\lambda) \iff z = \mu_\lambda(x;\lambda) + \sigma^2(x;\lambda)\odot\epsilon,\ \epsilon \sim \mathcal{N}(0,1) \tag{2.18}$$

où μ et σ^2 sont des fonctions de x paramétrées par λ et $\odot$ le produit de Hadamard (produit terme à terme). En réécrivant la ELBO, on fait apparaître le parallèle avec un auto-encoder :

$$\begin{aligned}\mathcal{E}(q) =& \frac{1}{L}\sum_{l=1}^{L}\log p(X|Z^{(l)}) - KL(q_\lambda(Z)||p(Z))\\ & Z^{(l)} \sim q_\lambda(Z).\end{aligned} \tag{2.19}$$

On génère un "code" à partir d'une observation, et on essaye de maximiser la probabilité de l'observation reconstruite à partir de ce code. L'astuce de la reparamétrisation permet d'encoder le modèle sous la forme d'un réseau de neurones, et d'optimiser les paramètres par descente de gradient stochastique et retro-propagation.

Le *Variational Information Bottleneck* ou VIB utilise le principe de l'inférence variationnel pour construire des représentations compressées d'observations étiquetées. C'est cette approche que nous avons utilisée dans le Chapitre 3 pour apprendre des représentations de documents et d'auteurs.

2.3.4 Approche VIB

Les données sont un ensemble de couples (x, y) où y est une étiquette associée à x. Chaque couple est associé à une variable latente z qui n'est pas observée. On note $X = x_{1:n}$, $Z = z_{1:n}$, et $Y = y_{1:n}$. La méthode de l'*Information Bottleneck* proposée par Tishby et al. (1999) consiste à apprendre une représentation z qui maximise la compression de x, tout en étant informative par rapport aux labels y. Cette notion de compression s'intègre bien dans le cadre de l'apprentissage de représentation, comme le montrent Oh et al. (2019). La fonction objectif est donnée par :

$$\arg\max_{Z} I(Z, Y) - \beta I(Z, X) \tag{2.20}$$

où I est l'information mutuelle définie par :

$$I(x, y) = \int\int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} d_x d_y. \tag{2.21}$$

$\beta \geq 0$ contrôle l'équilibre entre les deux sous objectifs : le premier terme de l'équation 2.20 encourage z à bien prédire y ; le deuxième terme encourage z à «oublier» x de façon à le compresser. L'information mutuelle n'est souvent pas calculable en forme close. Alemi et al. (2017) proposent d'utiliser une approche variationnelle (le VIB). Ici le principe est un peu différent de ce qui a été décrit précédemment : p(z|x) est un choix de modélisation, et on va approximer p(z), et p(y|z). On construit une borne inférieure qui est :

$$\mathcal{E}_{VIB}(q, r) = \mathbb{E}_{Z\sim p(Z|X)}[\log q(Y|Z)] - \beta KL(p(Z|X)||r(Z)). \tag{2.22}$$

$p(Z|X)$ est appelé encodeur, $q(Y|Z)$ approxime $p(Y|Z)$ et $r(Z)$ approxime $p(z)$. Maximiser en les paramètres de q et z permet de faire croître l'équation (2.21).

On a présenté dans cette section la modélisation probabiliste des données ainsi que quelques algorithmes d'apprentissage couramment utilisés pour déterminer les paramètres optimaux des modèles. Dans la prochaine partie, nous présenterons les approches d'apprentissage de représentations de mots, et de documents, qui sont au coeur de la plupart de nos contributions.

2.4 Apprentissage de représentations de mots

Nous rappelons que $Tf \in \mathbb{N}^{I\times J}$ est la matrice documents-termes de comptage d'occurrences des mots dans les documents : $Tf_{j,k}$ est le nombre de fois ou le k-ème mot apparaît dans le j-ème document, cf tableau 1.2. Le vocabulaire est composé de K mots différents, où w_k est le k-ème mot du vocabulaire. On rappelle qu'on note σ la fonction sigmoïde.

Dans cette partie, nous présenterons d'abord les travaux fondateurs de Mikolov et al. (2013b), et quelques autres approches. Nous présenterons ensuite les modèles qui permettent d'apprendre une mesure d'incertitude sur la représentation du mot. Dans un troisième temps, nous présenterons les approches de représentations contextuelles, à l'image des travaux de Devlin et al. (2019) (le modèle BERT et ses dérivés). Ces approches intègrent des architectures neuronales complexes, les CNN, les RNN ou les Transformers, que nous présenterons également.

Nous résumons les travaux présentés en tableau 2.1.

Nom	article	Statique	Distribution	Dynamique	Contextualisées
SGNS	(Mikolov et al., 2013b)	✓			
CBOW	(Mikolov et al., 2013a)	✓			
Glove	(Pennington et al., 2014)	✓			
Word2Gauss	(Vilnis and McCallum, 2015)	✓	✓		
VBSG	(Barkan, 2017)	✓	✓		
BSG	(Bražinskas et al., 2018)	✓	✓		
DWE	(Bamler and Mandt, 2017)		✓	✓	
DSG	(Rudolph and Blei, 2018)		✓	✓	
DW2V	(Yao et al., 2018)			✓	
CoVE	(McCann et al., 2017)	✓			✓
BERT	(Devlin et al., 2019)	✓			✓

Tableau 2.1 – Tableau récapitulatif des approches de représentations de mots présentées dans ce chapitre

2.4.1 Word2Vec et associés

L'apprentissage de représentations de mots a été popularisé par les travaux de Mikolov et al. (2013b,a). Ces travaux ne sont néanmoins pas précurseurs de cette approche. Des travaux, moins cités, avaient proposé d'apprendre des représentations vectorielles de mots en faible dimension. Certains utilisaient déjà des réseaux de neurones (Bengio et al., 2003), d'autres étendent l'approche de (Deerwester et al., 1990), comme (Schütze, 1993).

Le recours croissant aux réseaux de neurones pourrait expliquer la popularité des modèles Word2Vec : ils nécessitent de manipuler une représentation continue des données. Les nombreuses architectures initialement développées pour d'autres types de données (e.g. les RNN, que nous présenterons en sous-section 2.4.4) sont ainsi plus facilement applicables à des matrices de réels en dimension raisonnable, ou en tout cas plus faibles que la matrice document-termes.

Les modèles Word2Vec reposent sur une hypothèse formulée par Harris (1954) et Firth (1957). Le contexte d'un mot lui donne son sens, c'est-à-dire que deux mots qui cooccurrent avec les mêmes mots auront un sens proche. Par exemple, les synonymes ne cooccurrent généralement pas, mais seront observés dans des contextes similaires. Les colonnes qui leur correspondent dans la matrice documents-termes T sont orthogonales. En revanche, les colonnes de deux mots synonymes dans la matrice $Tf^\top.Tf$ (la matrice qui compte les cooccurrences au sein des documents) seront proches.

Word2Vec fait référence à deux modèles : CBOW et Skip Gram. Chaque mot est associé à deux représentations : la représentation source et contexte. On introduit la fonction de *mapping*, $f_{map}(w;\theta)$, qui associe à un mot d'un vocabulaire de taille K, sa représentation stockée dans une matrice de paramètres $\theta \in \mathbb{R}^{K\times R}$, ou R est nettement inférieur à K. Les représentations sources forment la matrice U, les représentations contextes une matrice V. Dans le cas où il n'y a pas d'ambiguïté, on notera plus simplement $u_i = f_{map}(w_i;U)$, $v_j = f_{map}(w_j;V)$. Les matrices U et V sont les paramètres à optimiser dans la phase d'apprentissage.

On détaillera d'abord le modèle Skip-Gram avec tirage négatif (*negative sampling*), noté SGNS. On considère une fenêtre de taille c. Pour chaque paire de mots, on calcule le nombre de fois où ils

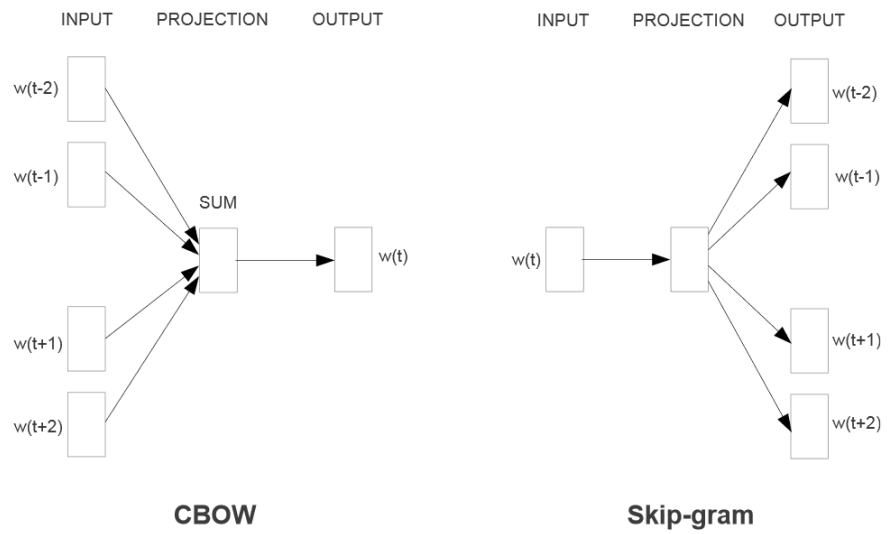


FIGURE 2.3 – Différence Skip Gram et CBOW, illustration issue de (Mikolov et al., 2013a)

ont cooccurré dans cette fenêtre. Par exemple, dans cette dernière phrase, le mot "nombre" et "fois" ont cooccurré une fois dans une fenêtre de taille c=2 mais aucune dans une fenêtre de taille c=1. On note C^+ la matrice de cooccurrences, avec $C^+_{i,j}$ le nombre de cooccurrences observées entre les mots d'indices i et j et $C^+_i = \sum_j C^+_{ij}$. On peut ainsi définir la probabilité empirique de chaque mot qu'on écrit :

$$p(w_i) = \frac{C^+_i}{\sum\limits_j C^+_j}. \tag{2.23}$$

Les auteurs modélisent la probabilité d'observer w_j dans une fenêtre de taille c autour d'une occurrence de w_i par :

$$p(w_j|w_i) = \frac{\exp(u_i^\top . v_j)}{\sum\limits_{j'=1}^{K} \exp(u_i^\top . v_{j'})}. \tag{2.24}$$

La fonction softmax permet d'obtenir une distribution sur le vocabulaire et donc d'interpréter la sortie comme un vecteur de probabilités. On obtient pour un corpus la log-vraisemblance suivante :

$$\log \mathcal{L}(C^+; U, V) = \sum_{i=1}^{K} \sum_{j=1}^{K} C^+_{ij} \log p(w_i|w_j). \tag{2.25}$$

La vraisemblance est difficile à maximiser : son calcul nécessite de dériver une somme de fonctions exponentielles. Les auteurs proposent d'utiliser la méthode de la *noise contrastive estimation* (Gutmann and Hyvärinen, 2010) : plutôt que de prédire le mot source, l'objectif devient une classification

binaire entre un exemple négatif tiré aléatoirement et une observation, dite exemple positif. Cette approche revient à sous-échantillonner les observations négatives, qui sont largement majoritaires dans une matrice de cooccurrence. Pour chaque observation de cooccurrence, les auteurs proposent de tirer n_e échantillons négatifs au moyen d'une loi catégorielle dont les paramètres sont fonction de la fréquence des mots du vocabulaire : $Multi\left((p(w_j)^{3/4})_{j=1}^K\right)$. Plus un mot est courant dans le corpus, et plus sa probabilité d'être tiré sera grande. La vraisemblance devient :

$$\begin{gathered}\mathcal{L}(C^+;U,V)=\sum_{i=1}^{K}\sum_{j=1}^{K}C_{ij}^{+}\left(\log\sigma(u_i^\top.v_j)+\sum_{s=1}^{n_e}\log\sigma(-u_i^\top.v_{l^{(s)}})\right)\\ l^{(s)}\sim Multi\left((p(w_j)^{3/4})_{j=1}^K\right).\end{gathered}\tag{2.26}$$

En notant C_{ij}^- le nombre de fois où j a été tiré comme exemple négatif pour une paire contenant i, et $\mathcal{C}=\{C^+,C^-\}$, on peut récrire plus simplement :

$$\mathcal{L}(\mathcal{C};U,V)=\sum_{i=1}^{K}\sum_{j=1}^{K}C_{ij}^{+}\log\sigma(u_i^\top.v_j)+C_{ij}^{-}\log\sigma(-u_i^\top.v_j).\tag{2.27}$$

SGNS rapproche les représentations vectorielles des mots qui cooccurrent, et éloigne la représentation d'un mot des mots tirés comme exemples négatifs.

Le modèle CBOW prédit un mot central à partir de ses voisins. Avec C_o l'ensemble des indices des mots observés dans une fenêtre de taille c, le modèle de langue définit la probabilité que le mot w_i soit au centre de la fenêtre par :

$$p(w_i|C_o)=\frac{\exp(u_i^\top.\sum\limits_{j\in C_o}v_j)}{\sum\limits_{i'=1}^{K}\exp(u_{i'}^\top.\sum\limits_{j\in C_o}v_j)}.\tag{2.28}$$

L'optimisation est faite au moyen de l'algorithme de rétropropagation avec descente de gradient stochastique.

Glove Pennington et al. (2014) est un deuxième modèle important d'apprentissage de représentations de mots. Les auteurs proposent d'utiliser une approche par factorisation de matrice plutôt qu'un réseau de neurone. Ainsi, les représentations vectorielles sont apprises de façon à ce que le produit scalaire entre les représentations de deux mots soit égal au logarithme du nombre de fois où ils cooccurrent (à un biais près). Une légère correction est appliquée pour donner plus d'importance aux valeurs élevées de cooccurrences, plus informatives. On note $u_i=f_{map}(w_i;U)$ et $v_i=f_{map}(w_i;V)$ les représentations vectorielles des mots apprises par le modèle. En introduisant des vecteurs de biais a et b dans $\mathbb{R}^K$, qui permettent de compenser les déséquilibres de nombre de cooccurrence, et un hyperparamètre $\alpha\in\mathbb{R}$, l'objectif à minimiser est :

$$\begin{gathered}J=\sum_{i=1}^{K}\sum_{j=1}^{K}f\left(C_{ij}^{+}\right)\left(u_i^\top.v_j+a_i+b_j-\log C_{ij}^{+}\right)^2\\ f(x)=\begin{cases}(x/x_{\max})^{\alpha} & \text{si } x<x_{\max}\\ 1 & \text{autrement .}\end{cases}\end{gathered}\tag{2.29}$$

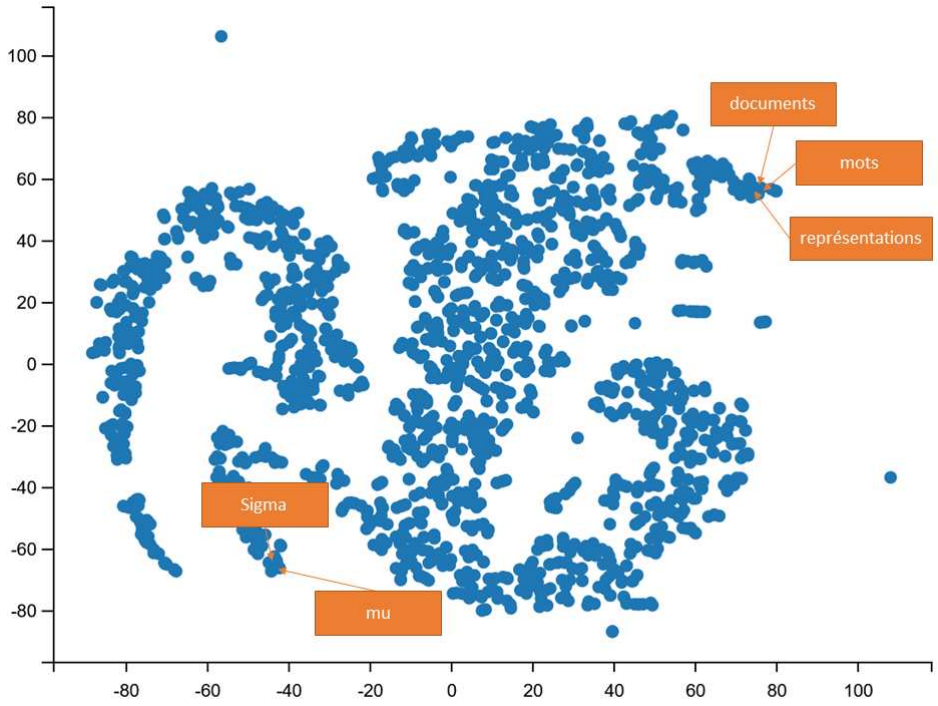


FIGURE 2.4 – SGNS appliqué au contenu textuel de ce manuscrit (10 exemples négatifs, fenêtre de taille 3, les mots vides ont été retirés). La dimension est réduite à 2 au moyen de la méthode T-SNE (Maaten and Hinton, 2008), avec une perplexité de 5.

Les auteurs montrent que la complexité est légèrement moins importante que le modèle SGNS. Les expérimentations des auteurs démontrent la supériorité de Glove par rapport à Word2Vec sur des tâches de résolution d'analogie, de similarité de mots et de reconnaissance d'entités nommées. Il est également plus rapide à entraîner. Cette approche est encore largement utilisée : des représentations pré-entraînées sur de très grands corpus ont été mises à disposition par les auteurs. Elles permettent de transférer les connaissances de ressources textuelles conséquentes.

Les travaux que nous avons présentés jusque-là représentent les mots comme un vecteur dans un espace sémantique de faible dimension. Dans la prochaine sous-section, nous présentons les approches qui proposent d'apprendre des distributions de probabilités.

2.4.2 Modélisation de l'incertitude dans les plongements de mots

Les mots peuvent être associés à des concepts difficilement représentables par un seul point. Plusieurs travaux proposent donc de représenter les mots comme des densités de probabilités, de façon à encoder une notion qu'on appellera *l'incertitude sémantique*. Intuitivement, plus un mot est utilisé dans des contextes sémantiques différents, plus son sens est incertain. Les synonymes d'un mot peuvent être très éloignés sémantiquement, comme le mot "souris", périphérique informatique et rongeur. La majorité des travaux utilise des lois gaussiennes pour représenter les mots.

On optimise ainsi les paramètres des fonctions qui associent un mot du vocabulaire à un des

paramètres de la loi considérée. Pour une loi gaussienne, on apprend les paramètres $\mu_i = f_{map}(w_i; \mu)$ qui associe chaque mot à sa moyenne, stockée dans une matrice $\mu \in \mathbb{R}^{K \times R}$, et $\Sigma_i = f_{map}(w_i; \mathbf{\Sigma})$ qui associe un mot à sa variance-covariance dans l'espace des représentations de mots, stockée dans un ensemble $\mathbf{\Sigma}$ de K matrices $\Sigma_i \in \mathbb{R}^{R \times R}$, symétriques et semi-définies positives.

Vilnis and McCallum (2015) proposent d'apprendre des représentations gaussiennes de mots en utilisant l'apprentissage à base d'énergie. Chaque mot du vocabulaire étant représenté par une densité gaussienne, la distance entre deux mots peut se calculer au moyen de la divergence de Kullback-Leibler. Soit $tr()$ la fonction qui retourne la trace d'une matrice (la somme de ses valeurs diagonales) et $det()$ son déterminant, on obtient :

$$\begin{aligned} KL\left(w_i, w_j\right) &= \int_{x \in \mathbb{R}^n} \mathcal{N}\left(x; \mu_i, \Sigma_i\right) \log \frac{\mathcal{N}\left(x; \mu_j, \Sigma_j\right)}{\mathcal{N}\left(x; \mu_i, \Sigma_i\right)} dx \\ &= \frac{1}{2}\left(\operatorname{tr}\left(\Sigma_i^{-1}\Sigma_j\right) + \left(\mu_i - \mu_j\right)^\top \Sigma_i^{-1}\left(\mu_i - \mu_j\right) - d - \log \frac{\det\left(\Sigma_j\right)}{\det\left(\Sigma_i\right)}\right). \end{aligned} \tag{2.30}$$

L'objectif est de minimiser cette valeur pour des mots observés dans le même contexte, la maximiser pour deux mots qui n'apparaissent pas ensemble. Plus concrètement, les auteurs maximisent une mesure de rang à marge maximum (*max-margin*) pour une paire de mots (w_i, w_j) qui cooccurrent dans le corpus et un exemple négatif w_l :

$$L_m\left(w_i, w_j, w_l\right) = \max\left(0, m + KL\left(w_i, w_j\right) - KL\left(w_i, w_l\right)\right). \tag{2.31}$$

La somme de cet objectif pour toutes les paires de mots observées dans les documents de l'ensemble d'entraînement est minimisée. Les exemples négatifs sont tirés similairement à Mikolov et al. (2013b). Ce modèle semble capable de capter des concepts d'inclusion sémantique, grâce à l'asymétrie de la divergence de Kullback-Leibler. Enfin, cette approche permet de guider l'apprentissage des représentations par des mesures directes de distance entre les densités des mots, contrairement aux approches suivantes, qui ajoutent seulement des a priori bayésiens sur les représentations vectorielles des mots.

Barkan (2017) reprennent la vraisemblance de SGNS décrite en équation (2.27), c'est-à-dire que la probabilité que w_i soit dans le contexte de w_j est donnée par :

$$p(w_i|w_j) = \sigma(u_i^\top . v_j) \prod_{s=1}^{n_e} \sigma(-u_i^\top . v_{l^{(s)}})) \tag{2.32}$$

avec $u_i = f_{map}(w_i; U)$ et $v_j = f_{map}(w_j; V)$. Ils proposent d'ajouter les a priori suivant :

$$\begin{aligned} u_i &\sim \mathcal{N}(0, \Sigma_0) \\ v_i &\sim \mathcal{N}(0, \Sigma_0). \end{aligned} \tag{2.33}$$

Pour maximiser la vraisemblance des données (les matrices de cooccurrences positives et négatives C^- et C^+), les auteurs utilisent une approche variationnelle qui implique de définir une distribution q. Elle est définie par :

$$q_\lambda(U, V) = \prod_{i=1}^{K} \mathcal{N}(u_i; \mu_{u_i}, \Sigma_{u_i}) \mathcal{N}(v_i; \mu_{v_i}, \Sigma_{v_i}). \tag{2.34}$$

On cherche à maximiser la ELBO, définie dans ce contexte par

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(\mathcal{C}|U,V)] + \mathbb{E}_q[\log p(U,V)] - \mathbb{E}_q[\log q_\lambda(U,V)]. \tag{2.35}$$

Ils introduisent ainsi une moyenne source μ_{u_i} et une moyenne cible μ_{v_i} pour chaque mot du vocabulaire, mais aussi une variance source Σ_{u_i} et cible Σ_{v_i}. Les distributions variationnelles rendent compte, selon les auteurs, de l'incertitude sémantique des mots du vocabulaire.

Bražinskas et al. (2018) propose un modèle très proche de celui de Barkan (2017). La différence principale est que les variances et les moyennes des a priori des représentations ne sont pas partagées par tous les mots du vocabulaire :

$$u_i \sim \mathcal{N}(\mu_i, \Sigma_i) \tag{2.36}$$

et la probabilité est définie par :

$$p(w_j|w_i) = \frac{\mathcal{N}(u_i; \mu_j, \Sigma_j) \times p(w_j)}{\sum_{j'=1}^{K} \mathcal{N}(u_i; \mu_{j'}, \Sigma_{j'}) \times p(w_{j'})}. \tag{2.37}$$

Ils implémentent l'optimisation en utilisant l'approche de l'auto-encodeur variationnel de (Kingma and Welling, 2014).

Dans les sections précédentes, nous avons présenté des approches apprenant soit des vecteurs soit des densités pour représenter les mots. Néanmoins, dans de nombreux jeux de données, les données textuelles ont été observées sur un temps long. Le sens des mots change, d'autres apparaissent (par exemple, "ordinateur"), et les contextes d'observation, dans le sens de collocations, évoluent au cours des années (par exemple, "souris"). Il apparaît donc nécessaire de prendre en compte cette information de glissement sémantique dans l'apprentissage des représentations. On parle d'apprentissage de représentations dynamiques.

2.4.3 Apprentissage de représentations dynamiques de mots

La plupart des approches discrétisent le temps en construisant un nombre fini de périodes temporelles. Elles apprennent ensuite une représentation par période pour chacun des mots du vocabulaire. Les matrices des représentations vectorielles sont donc indexées par rapport à la tranche temporelle à laquelle elles correspondent. On considère un ensemble de T tranches temporelles qu'on indice par t. Les premières observations commencent à $t = 1$. On calcule les cooccurrences pour chaque tranche, et les données deviennent T couples de matrices $\mathcal{C}^{(t)} = \{C^{(t)^+}, C^{(t)^-}\} : C^{(t)^+}$ est la matrice des cooccurrences positives et $C^{(t)^+}$ des cooccurrences négatives observées dans la t-ème tranche temporelle .

Dans le cas transductif, c'est-à-dire qu'on ne peut ni construire de représentations des mots qui ne sont pas dans le jeu de donnée d'entraînement, ni de tranche temporelle non observée, on apprendra T, parfois $T + 1$, matrices $U^{(t)} \in \mathbb{R}^{K \times R}$. On notera la t-ème représentation du i-ème mot du vocabulaire : $u_{i,t}$. Dans certains cas, on sépare, similairement à (Mikolov et al., 2013b), les représentations sources et contextes. On notera par simplicité l'ensemble des données $\mathcal{C}$ et l'ensemble des représentations U (et V lorsque deux représentations sont apprises).

La première, proposée par Bamler and Mandt (2017), utilise le même objectif que SGNS. La vraisemblance des observations positives et négatives pour une tranche temporelle dépend des représentations vectorielles au temps t, et est donnée par :

$$p(\mathcal{C}^{(t)}|U^{(t)}, V^{(t)}) = \sum_{i=1}^{K}\sum_{j=1}^{K} {C_{ij}^{(t)}}^{+} \log \sigma(u_{i,t}^{\top}.v_{j,t}) + {C_{ij}^{(t)}}^{-} \log \sigma(-u_{i,t}^{\top}.v_{j,t}). \tag{2.38}$$

De façon à prendre en compte le temps, les auteurs considèrent ensuite un processus de diffusion des vecteurs de plongement en ajoutant l'a priori suivant :

$$p(u_{i,t+1}|u_{i,t}) \sim \mathcal{N}(u_{i,t}, \sigma_t^2)\mathcal{N}(0, \sigma_0^2)$$
$$\sigma_t^2 = D\nabla_{t,t+1}$$

où D est une constante globale de diffusion et $\nabla_{t,t+1}$ la différence de temps entre les deux tranches temporelles. L'inférence est réalisée au moyen d'une approche variationnelle. On cherche donc à approximer $p(U, V|\mathcal{C})$ par une distribution $q_\lambda(U, V)$ qui maximise

$$\arg\max_q \mathcal{L}(q) = \mathbb{E}_q[\log p(\mathcal{C}, U, V)] - \mathbb{E}_q[\log q_\lambda(U, V)]. \tag{2.39}$$

Les auteurs proposent d'utiliser une distribution qui se factorise par tranches de temps $q_\lambda(U, V) = \prod_{t=1}^{T} q_\lambda(U^{(t)}, V^{(t)})$. En introduisant les paramètres variationnels, les auteurs proposent :

$$q_\lambda(U^{(t)}, V^{(t)}) = \prod_{i=1}^{K} \mathcal{N}(u_{i,t}; \mu_{u_{i,t}}, \Sigma_{u_{i,t}})\mathcal{N}(v_{i,t}; \mu_{v_{i,t}}, \Sigma_{v_{i,t}}). \tag{2.40}$$

Dans (Rudolph and Blei, 2018), les auteurs étendent le modèle statique de (Rudolph et al., 2016) appliqué à l'apprentissage de représentations de mots : l'apprentissage de représentations de Bernoulli. Il est assez proche du modèle CBOW. Soit $x_i \in \{0, 1\}$, qui prend la valeur 1 si le i-ème mot est observé au centre de la fenêtre, 0 sinon. La probabilité de x_i, avec C_o l'ensemble des indices des mots observés dans le contexte :

$$p(x_i = 1|C_o) = \sigma(u_i^{\top} \sum_{j \in C_o} v_j). \tag{2.41}$$

Les observations négatives sont sous-échantillonnées, de sorte à ce qu'on retrouve le modèle CBOW avec échantillonnage négatif. Les auteurs proposent ensuite dans (Rudolph and Blei, 2018) d'ajouter les a priori suivants :

$$u_{i,t} \sim \mathcal{N}(u_{t-1}, \lambda^{-1}I)$$
$$u_{i,0} \sim \mathcal{N}(0, \lambda_0^{-1}I)$$
$$v_{i,t} \sim \mathcal{N}(0, \lambda_0^{-1}I).$$

Ils maximisent ensuite la log-vraisemblance complétée. On note $\mathcal{R}_i$ l'ensemble des n_e ensembles de contextes tirés négativement pour le i-ème mot. La quantité suivante est maximisée :

$$\begin{aligned}\log p(\mathcal{C}, u, v) = &\sum_{t=1}^{T}\sum_{i=1}^{K}\sum_{j=1}^{K} {C_{ij}^{(t)}}^{+} \log\sigma(u_{i,t}^{\top}\sum_{j\in C_o} v_{i,t}) + \sum_{C'_o\in\mathcal{R}_i} \log\sigma(-u_{i,t}^{\top}\sum_{j\in C'_o} v_{i,t}) \\ &- \frac{\lambda_0}{2}\left(\sum_{i=1}^{K}||v_i||^2 + ||u_{i,1}||^2\right) - \frac{\lambda}{2}\sum_{i=1}^{K}||u_{v,t} - u_{v,t-1}||^2.\end{aligned} \tag{2.42}$$

Dans (Yao et al., 2018), les auteurs étendent la modélisation de (Levy and Goldberg, 2014) au cas dynamique. Dans (Levy and Goldberg, 2014), les représentations vectorielles sont construites en factorisant la matrice PPMI (Positive Pointwise Mutual Information) des mots et de leur contexte. Cette matrice contient les estimations ponctuelles de l'information mutuelle entre mots du vocabulaire, où les valeurs négatives sont fixées à 0. Sous certaines hypothèses, cette formulation est équivalente à SGNS. Soit Y_t la matrice PPMI calculée sur l'intervalle temporel t, et $U^{(t)}$ les représentations vectorielles des mots au temps t. Avec λ et τ des hyperparamètres réels, la fonction à optimiser est :

$$\min_{U_t}\sum_t ||Y^{(t)} - U^{(t)}{U^{(t)}}^{\top}||_F^2 + \lambda\sum_t ||U^{(t)}||_F^2 + \tau\sum_t ||U^{(t)} - U^{(t-1)}||_F^2. \tag{2.43}$$

On a présenté jusque-là des modèles qui apprennent une représentation unique pour chaque mot du vocabulaire, ignorant les problèmes de synonymies. Les modèles plus récents apprennent des représentations de mots en contexte, c'est-à-dire que le vecteur représentant le mot dépendra de sa position dans la phrase et des mots qui l'entourent. Cette contextualisation des représentations se fait généralement au moyen de mécanismes d'attention ou de réseaux de neurones récurrents.

2.4.4 Représentations contextualisées - Réseaux récurrents et convolutifs

De façon à prendre en compte le contexte d'observation des mots, plusieurs méthodes ont été proposées. Elles reposent principalement sur trois architectures de réseaux de neurones : les réseaux de neurones convolutifs, les réseaux de neurones récurrents (Hochreiter et al., 1997; Cho et al., 2014) et les Transformers (Vaswani et al., 2017). On présentera dans cette sous-section les deux premières approches.

Réseaux convolutifs

Les réseaux de neurones convolutifs (LeCun et al., 2015) ont été principalement utilisés pour le traitement d'image. Ils consistent à faire passer des filtres de convolution sur une matrice, ou un tenseur, puis de réduire la taille de la matrice en sortie en utilisant une opération de *pooling*, c'est-à-dire une opération d'agrégation d'un ensemble de valeurs. En général, on agrège en calculant la moyenne ou le maximum, qu'on nomme *mean pooling* et *max pooling*). Dans le cas du texte, il est possible d'utiliser les CNN sur les matrices des représentations vectorielles des mots (Kim, 2014). Les filtres sont de dimension $l_{cnn} \times r$, où l_{cnn} donne la taille des séquences de mots à prendre en compte et r la dimension de l'espace de représentation des mots. On parle de convolution 1D car la

convolution n'est pas effectuée sur l'axe des variables de la représentation des mots mais seulement sur l'axe temporel, c'est-à-dire l'axe de la séquence des mots. Pour les images, la convolution est réalisée sur des zones spatiales en deux dimensions (hauteur et largeur), donc une convolution 2D (ou 3D pour les images en couleur). Après une étape de pooling sur la dimension temporelle, on obtient donc une représentation vectorielle contextualisée de la phrase dans un nouvel espace dont la dimension dépend des paramètres de la convolution.

Réseaux récurrents

Les réseaux de neurones récurrents permettent de modéliser des séquences de vecteurs réels. Un vecteur interne, appelé état caché, implémente une forme de mémoire de l'historique des vecteurs passés en entrée. Soit une séquence de vecteurs réels $(x^{(t)})_{t=0}^{T}$ et $a^{(t)}$ la valeur de l'état caché au temps t, qui est un vecteur généralement initialisé à 0. Dans sa forme fonctionnelle, un RNN simple calculera pour $t > 0$, avec W_1, W_2 et b les paramètres du réseau :

$$a^{(t)} = h(W_1 a^{(t-1)} + W_2 x^{(t)} + b) \tag{2.44}$$

où h est une fonction d'activation, souvent tanh ou ReLU, W_1 et W_2 et b sont les paramètres du réseau. Par la suite, on note $f_{rnn}(x^{(t)}, a^{(t-1)})$ la fonction qui fournit en sortie le vecteur $a^{(t)}$. Plusieurs améliorations ont été proposées, telles que l'architecture GRU, pour Gated Rectified Unit. On rappelle qu'on note $[a, b]$ la concaténation des vecteurs a et b. Avec W_c, b_c, W_u, b_u les paramètres du réseau, le GRU calcule :

$$\begin{aligned} \tilde{a}^{(t)} &= \tanh(W_c[a^{(t-1)}, x^{(t)}] + b_c) \\ \Gamma_u &= \sigma(W_u[a^{(t-1)}, x^{(t)}] + b_u) \\ a^{(t)} &= \Gamma_u \times \tilde{a}^{(t)} + (1 - \Gamma_u) \times a^{(t-1)}. \end{aligned} \tag{2.45}$$

Dans les réseaux LSTM, les auteurs proposent d'introduire un deuxième vecteur de mémoire, qu'on note c. Avec $W_c, b_c, W_u, b_u, W_f, b_f, W_o, b_o$ les paramètres du réseau, le LSTM calcule :

$$\begin{aligned} \tilde{c}^{(t)} &= \tanh(W_c[a^{(t-1)}, x^{(t)}] + b_c) \\ \Gamma_u &= \sigma(W_u[a^{(t-1)}, x^{(t)}] + b_u) \\ \Gamma_f &= \sigma(W_f[a^{(t-1)}, x^{(t)}] + b_f) \\ \Gamma_o &= \sigma(W_o[a^{(t-1)}, x^{(t)}] + b_o) \\ c^{(t)} &= \Gamma_u \times \tilde{c}^{(t)} + \Gamma_f \times c^{(t-1)} \\ a^{(t)} &= \Gamma_o \times \tanh(c^{(t)}). \end{aligned} \tag{2.46}$$

Le dernier état caché est ensuite souvent donné à un MLP spécifique pour résoudre la tâche étudiée. Dans le cas de données textuelles, x est une séquence de représentations vectorielles des mots de la phrase. Les représentations sont soit pré-entrainées, soit apprises simultanément aux paramètres du réseau. La séquence d'états cachés du RNN produit ainsi des représentations contextualisées des mots. Dans (McCann et al., 2017; Melamud et al., 2016), les auteurs utilisent des RNN bidirectionnels,

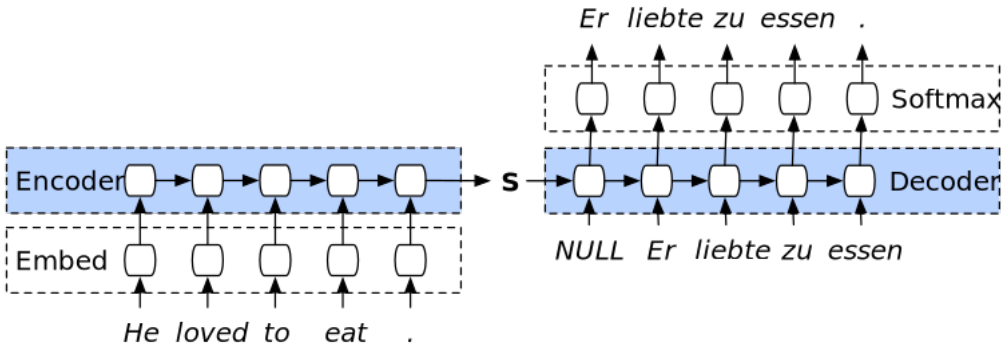


FIGURE 2.5 – Schéma du fonctionnement général de l'architecture Seq2Seq, source : `https://smerity.com/articles/2016/google_nmt_arch.html`

c'est-à-dire que la séquence de mots est passée dans le RNN, puis est renversée (on place le premier mot en dernier le deuxième en avant-dernier, etc) et est donnée à un deuxième RNN. La concaténation des états cachés finaux *forward* (issu de la passe en avant) et *backward* (issu de la passe en arrière) donne ainsi une représentation contextualisée qui prend en compte tous les mots de la séquence, et pas seulement ceux venant avant le mot concerné.

On va maintenant présenter l'architecture seq2seq, qui a été largement utilisée dans des tâches de traduction automatique. Les mécanismes d'attention, nés du développement de cette architecture, sont à la base d'une nouvelle famille d'apprentissage de représentations contextualisées que nous présenterons ensuite.

Architecture Seq2Seq

L'architecture Seq2Seq repose sur deux réseaux : un encodeur et un décodeur. En figure 2.5, nous présentons un schéma général du fonctionnement de cette architecture. Dans la phase d'entraînement, chaque phrase dans une langue source est donnée au réseau avec sa traduction dans la langue cible. Chaque mot est remplacé par une représentation vectorielle (apprise en même temps que les paramètres de l'architecture, ou pré-entraînée). On obtient donc deux séquences de vecteurs. La phrase source est passée dans un RNN encodeur de paramètres θ, qui retourne un état caché final, qui contient une forme de mémoire de la séquence de vecteurs. Soit une phrase de longueur T, on conserve donc l'état $a_{enc}^{(T)}$ résultat de la fonction :

$$a_{enc}^{(T)} = f_{rnn}(x^{(T)}, f_{rnn}(x^{(T-1)}, f_{rnn}(x^{(T-2)}, ...))). \tag{2.47}$$

Cet état caché sert à initialiser le décodeur. Pour la traduction dans la langue cible, on génère une séquence de vecteurs au moyen d'un RNN décodeur de paramètres θ'. Un token spécial "START" est ajouté au début de la phrase à décoder dont la représentation vectorielle est $x_{dec}^{(0)}$. Le premier mot est

$$\begin{aligned} a_{dec}^{(1)} &= f_{rnn}(x_{dec}^{(0)}, a_{enc}^{(T)}; \theta') \\ y_{dec}^{(1)} &= f_{mlp}(a_{dec}^{(1)}) \end{aligned} \tag{2.48}$$

où $y_{dec}^{(1)}$ est une distribution sur le vocabulaire. La stratégie la plus simple pour la génération est de sélectionner le mot de probabilité maximum. Pour générer le mot suivant on calcule, avec $x_{dec}^{(1)}$ la représentation vectorielle du mot tiré :

$$\begin{aligned} a_{dec}^{(2)} &= f_{rnn}(x_{dec}^{(1)}, a_{dec}^{(1)}; \theta') \\ y_{dec}^{(2)} &= f_{mlp}(a_{dec}^{(2)}). \end{aligned} \tag{2.49}$$

On continue ensuite à générer séquentiellement un nombre fixé de mots, ou jusqu'à ce que le modèle génère un token spécial indiquant la fin de la phrase, ou du document. D'autres approches de génération que nous ne détaillerons pas, telles que la recherche par faisceau, ou *beam search* sont parfois utilisées.

Dans la phase d'apprentissage, il est courant d'utiliser la méthode du "teacher forcing". On donne au décodeur la vraie séquence de mots à traduire, c'est-à-dire que $x_{dec}^{(l)}$ est la représentation du l-ème mot réellement observé, de la séquence cible. Chaque sortie est comparée avec la sortie attendue, généralement avec une perte d'entropie croisée.

L'inconvénient majeur de cette architecture Seq2Seq est que le document source est représenté par un vecteur unique, c'est-à-dire qu'on génère uniquement en fonction d'une information sémantique globale du document. Pour contourner ce problème, les mécanismes d'attention ont été proposés par Bahdanau et al. (2015) et (Luong et al., 2015).

2.4.5 Représentations contextualisées - Attention et Transformers

Dans la sous-section précédente, on a présenté l'architecture seq2seq. On présente ici le mécanisme d'attention qui permet de contrevenir à la représentation par un vecteur unique de la phrase lors de la traduction. Ce mécanisme est à la base d'une architecture récente : le Transformer que nous présenterons également.

Mécanismes d'attention

On conserve A_{enc} la matrice contenant sur chaque ligne les états cachés de l'encodeur, qu'on utilise au moment de décoder la séquence cible. Dans sa forme la plus simple, l'attention de type "dot product" (Luong et al., 2015) calcule :

$$a_{co}^{(t)} = softmax\left((a_{dec}^{(t)})^\top A_{enc}^\top\right) A_{enc}. \tag{2.50}$$

Le vecteur obtenu est concaténé à l'entrée $x_{dec}^{(t+1)}$ mais aussi à $a_{dec}^{(t+1)}$ avant d'être donné en entrée à f_{mlp}. Dans leur forme générale, les mécanismes d'attention calculent :

$$softmax(f_{att}(y, X)) \cdot XW_1. \tag{2.51}$$

La fonction $f_{att}(y, X)$ définit la manière dont sont calculés les poids d'attention. Soit y un vecteur de $\mathbb{R}^r$ et X une matrice dans $\mathbb{R}^{n\times r}$, l'attention consiste à construire une moyenne pondérée de la

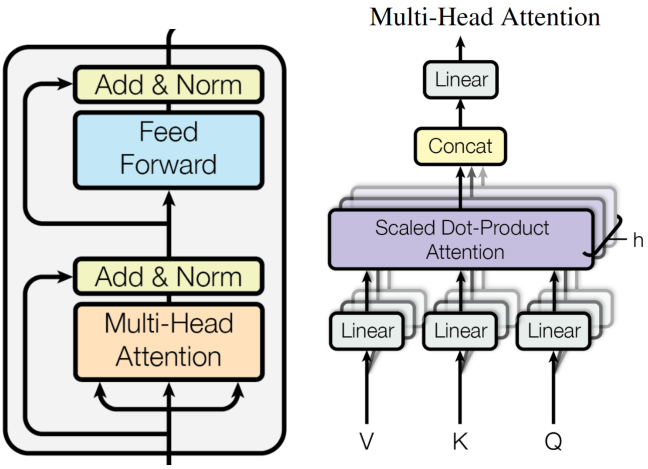


FIGURE 2.6 – Bloc Transformer et l'attention Multi-tête, schéma présenté dans (Vaswani et al., 2017)

matrice X. Les poids des vecteurs lignes qui sont les plus proches de y seront plus élevée. L'attention de type *scaled dot product*, est définie par :

$$f_{sdp}(y, X; W_2, W_3) = \frac{(y^\top W_2)(XW_3)^\top}{\sqrt{h}} \tag{2.52}$$

avec h la dimension de $a^\top W_2$. On parle de *self-attention* quand y est un des vecteurs ligne de X. On obtient la matrice X contextualisée quand on calcule

$$f_{asdp}(X; W_1, W_2, W_2) = softmax(\frac{(XW_2)(XW_3)^\top}{\sqrt{h}})XW_1 = X_{cont}. \tag{2.53}$$

Dans la prochaine partie, nous présenterons une architecture qui repose sur ces mécanismes d'attention et qui a permis l'émergence d'une nouvelle famille de modèles dont BERT (Devlin et al., 2019) est le représentant le plus connu.

Transformer et BERT

Le Transformer (Vaswani et al., 2017) est une architecture complexe de type Seq2Seq originalement développée pour résoudre une tâche de traduction. Elle repose sur une contextualisation des représentations des mots d'un document au moyen de l'attention *scaled dot product*. Plus précisément, les auteurs proposent d'utiliser un mécanisme d'attention multi-tête. Le Transformer est composé d'un encodeur et d'un décodeur. La majorité des travaux récents n'utilisent que l'encodeur du Transformer. Il est lui-même composé d'une succession de ce qu'on appellera bloc transformer.

Le bloc Transformer est une architecture qui permet de contextualiser la représentation d'un mot en fonction des autres mots de la séquence dans laquelle il a été observé. Plus formellement, soit X, la matrice de la séquence des représentations vectorielles des mots. Elle est découpée sur l'axe des colonnes (donc les axes de représentation) en h sous matrices de dimension $n \times r_h$, où $r_h = r/h$, avec r la dimension de représentation initiale. Les auteurs proposent d'appliquer l'attention *scaled dot product*, que nous avons présentée précédemment, à chaque sous matrice, puis concatènent les

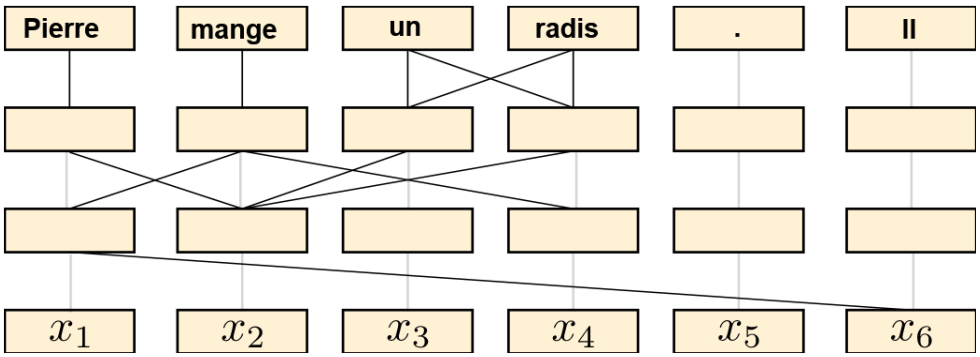


FIGURE 2.7 – Visualisation des poids d'attention permettant de contextualiser une phrase passée dans un Transformer. Chaque ligne correspond au passage dans un bloc Transformer. Dans la dernière passe, le mot "Il" porte principalement son attention sur le mot "Pierre" pour construire la représentation contextualisée finale. Source : Benjamin Piwowarski, Séminaire IXXI, 11-06-21 `http://www.ixxi.fr/agenda/seminaires/cycles-de-seminaires/intelligence-artificielle-et-langage/representations-continues-pour-lacces-a-linformation`

sorties (cf figure 2.6), c'est-à-dire les h sous-matrices, contextualisées. Cette nouvelle matrice est ajoutée à X (on parle de connexion résiduelle). La matrice obtenue est normalisée avant d'être passée à un MLP, puis on réitère la connexion résiduelle et la normalisation. Un schéma du bloc Transformer est présenté en figure 2.6.

Plusieurs blocs Transformer sont superposés pour créer l'encodeur et le décodeur du Transformer (à quelques détails près). De façon à rendre compte de la nature séquentielle des données textuelles, la matrice initiale des représentations vectorielles est additionnée à une matrice de positions. La matrice de position $M^p \in \mathbb{R}^{n \times r}$, est définie par :

$$\begin{aligned} M^p_{i,2j} &= \sin\left(\frac{i}{10000^{\frac{2j}{r}}}\right) \\ M^p_{i,2j+1} &= \cos\left(\frac{i}{10000^{\frac{2j}{r}}}\right) \end{aligned} \tag{2.54}$$

Dans la figure 2.7, nous présentons un exemple de contextualisation des représentations vectorielles des mots d'une phrase simple.

Le modèle BERT (Devlin et al., 2019) utilise l'encodeur du Transformer (plusieurs blocs transformer empilés), et manipule donc des représentations contextualisées des mots. Dans sa forme originale, le réseau est entraîné sur deux tâches.

La première consiste à prédire des tokens (des mots ou parties de mots qui forment le vocabulaire), masqués dans le jeu de donnée d'entraînement. Concrètement, un pourcentage prédéfini des tokens du document est remplacé par un token spécial (<MASK>). La matrice des représentations vectorielles des tokens contenus dans le document est passée dans le Transformer, qui retourne donc une matrice de même dimension, où chaque vecteur a été contextualisé par les couches d'attention successives. Les tokens masqués sont passés dans un MLP qui ressort une probabilité sur l'ensemble du vocabulaire.

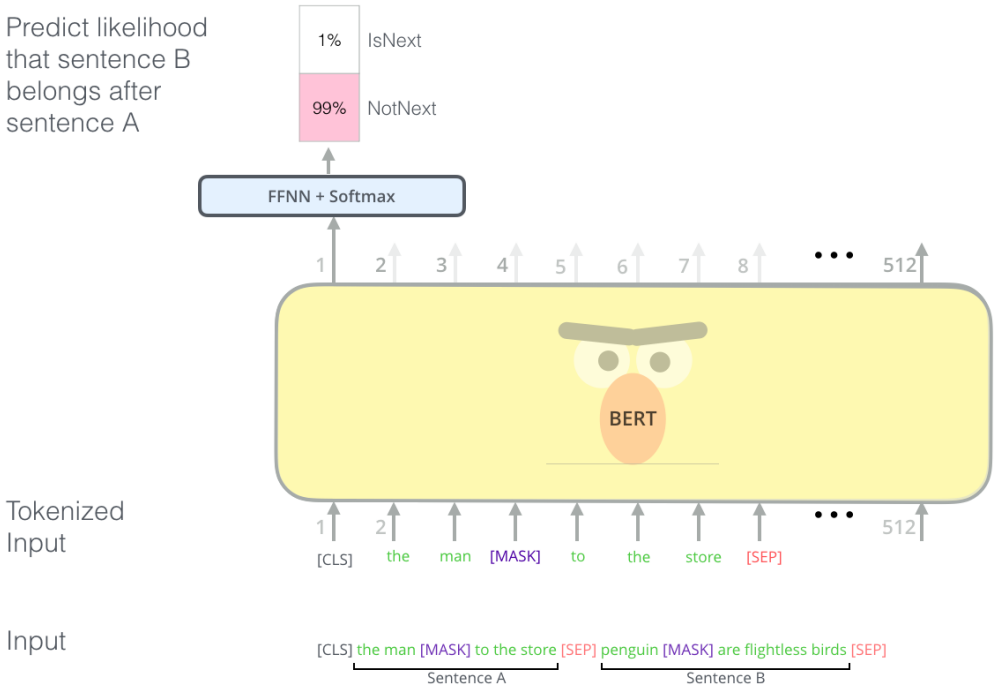


FIGURE 2.8 – Illustration de la tâche de Next Sentence Prediction du modèle BERT (Devlin et al., 2019), tirée de `http://jalammar.github.io/illustrated-bert/`

La rétropropagation est réalisée après calcul d'une perte d'entropie croisée comparant le mot prédit par la dernière couche au mot original avant masquage. Cette première tâche est appelé le modèle de langue masqué (*masked language model*).

La deuxième tâche est appelée prédiction de la phrase suivante (*next sentence prediction*). Un ensemble de paires de phrases est construit de façon à ce que 50% des phrases soient consécutives dans le jeu de données d'entraînement et 50% soient des exemples négatifs, c'est-à-dire que la deuxième phrase est choisie aléatoirement dans le corpus. Les deux phrases de chaque paire sont concaténées, séparées par un token spécial (le token SEP), puis données en entrée au Transformer. Un token spécial est ajouté en début de séquence (le token CLS). En sortie du Transformer, le vecteur associé au token CLS est passé dans un MLP qui ressort une probabilité binaire. La perte est calculée par rapport au label, qui vaut 1 si la deuxième phrase suivait la première, 0 sinon. De cette façon, le modèle apprend à concentrer l'information sémantique générale de la phrase au niveau du token CLS.

Les deux tâches sont illustrées en figures 2.9 et 2.8. Plusieurs variantes du modèle BERT ont été proposées, comme le modèle RoBERTa (Liu et al., 2019), qui modifie le procédé d'apprentissage et améliore les performances de BERT ou les modèles CamemBERT (Martin et al., 2020) et FlauBERT (Le et al., 2020) spécialisés sur des jeux de données en français.

On a vu jusqu'ici des méthodes qui représentent les mots, comme des vecteurs (contextualisés ou non), comme des distributions ou comme des séquences de vecteurs. Nous présenterons dans la section suivante les méthodes d'apprentissage de représentations de documents.

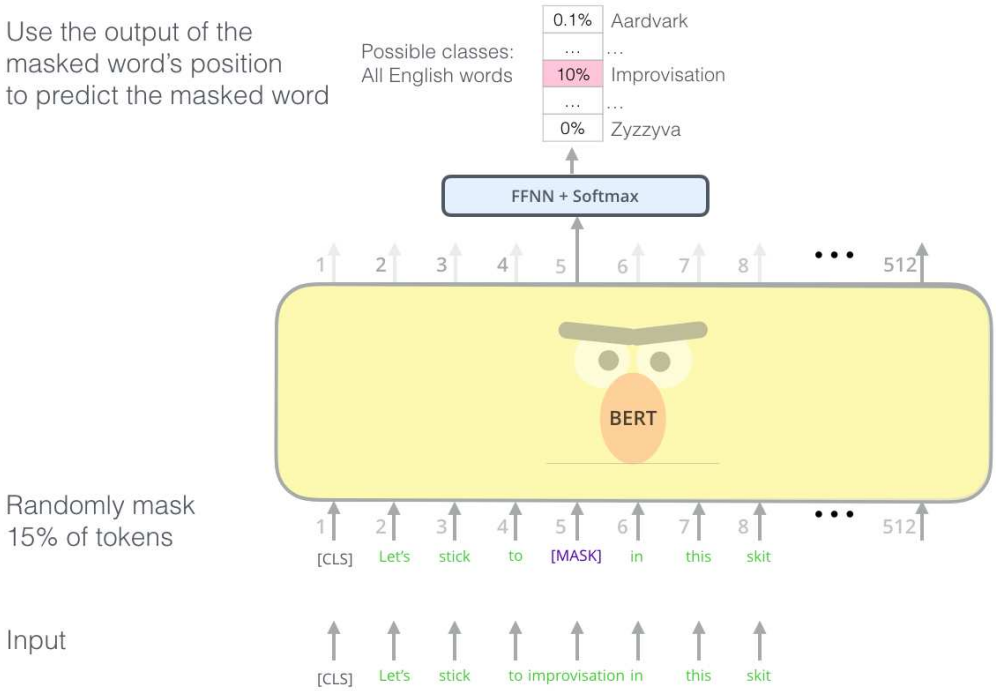


FIGURE 2.9 – Illustration de la tâche de Masked Language Modeling du modèle BERT (Devlin et al., 2019), tirée de `http://jalammar.github.io/illustrated-bert/`

2.5 Apprentissage de représentations de documents

Nous avons décrit précédemment les limitations de la représentation de documents dans l'espace des mots au moyen de la matrice documents-termes. On étudiera donc ici les méthodes qui proposent d'apprendre des représentations vectorielles des documents dans un espace de taille réduite. Dans ce nouvel espace, les vecteurs de deux documents dont le sens est proche auront une similarité élevée.

Nous résumons les approches présentées dans ce manuscrit dans le tableau 2.2.

On introduit ici quelques notations. $\mathcal{D}_i$, le i-ème document du corpus, est une séquence de tokens de taille l_i, c'est-à-dire $(w^{i,1}, w^{i,1}, ..., w^{i,l_i})$, où $w^{i,j}$ est le j-ème token du document. On note $d_i = f_{map}(\mathcal{D}_i, C_o; \theta) \in \mathbb{R}^r$ sa représentation vectorielle, avec C_o le contexte d'observation (un ensemble de mots, de documents, etc.). Dans le cas général, la représentation dépend d'un contexte et d'un ensemble de paramètres. Néanmoins, dans de nombreux cas, $d_i = f_{map}(\mathcal{D}_i; \theta)$. On note également $\mathcal{D}_i^w$ la séquence des représentations vectorielles des mots du document :

$$\mathcal{D}_i^w = (f_{map}(w^{i,1}, \mathcal{D}_i^w), \ldots, f_{map}(w^{i,l_i}, \mathcal{D}_i^w)), \tag{2.55}$$

et $\mathcal{D}_{i,j}^w$ le j-ème élément de cette séquence.

2.5.1 Approches historiques

La première approche proposée en 1988 (Dumais et al., 1988) puis en 1990 (Deerwester et al., 1990) propose d'appliquer une simple SVD à la matrice Tf ou Tf_{idf}. On peut ensuite tronquer

Nom	article	Vecteur	Distribution	Fonction d'agrégation
LSA/LSI	(Deerwester et al., 1990)	✓		
PV-DBOW	(Le and Mikolov, 2014)	✓		
PV-DM	(Le and Mikolov, 2014)	✓		Somme
SIF	(Arora et al., 2016)	✓		Moyenne
Skip-Thought	(Kiros et al., 2015)	✓		RNN
InferSent	(Conneau et al., 2017)	✓		RNN/Attention/Convolution
USE	(Cer et al., 2018)	✓		DAN/Transformer
SBERT	(Reimers and Gurevych, 2019)	✓		Pooling
G-LDA	(Das et al., 2015)		✓	
-	(Nikolentzos et al., 2017)		✓	

Tableau 2.2 – Tableau récapitulatif des approches de représentations de documents présentées dans ce chapitre

la SVD de façon à conserver les vecteurs singuliers les plus informatifs. Les vecteurs singuliers gauches représentent les documents. Néanmoins, la décomposition en valeurs singulières est coûteuse en termes de calcul, et passe difficilement à l'échelle. Plus précisément, la complexité de la SVD tronquée est en $\mathcal{O}(2JI^2 + I^3)$ (Li et al., 2019).

Profitant du succès des modèles Word2Vec, les mêmes auteurs ont proposé d'apprendre des représentations de documents suivant une procédure similaire (Le and Mikolov, 2014). La première approche, PV-DM, étend le modèle CBOW. Les représentations de documents sont des paramètres, la méthode n'est donc pas inductive. Le vecteur de contexte d'un mot cible est calculé en sommant les représentations des mots autour du mot cible à la représentation du document. La fonction objectif est ensuite la même que celle présentée en sous-section 2.4.1. Pour chaque occurrence, on maximise :

$$p(w_i|C_o, \mathcal{D}) = \frac{\exp\left(u_i.(\sum_{j\in C_o} v_j^\top + f_{map}(\mathcal{D}))\right)}{\sum_{i'=1}^{K} \exp\left(u_{i'}.(\sum_{j\in C_o} v_j^\top + f_{map}(\mathcal{D}))\right)}. \tag{2.56}$$

Le deuxième modèle, PV-DBOW, s'inscrit dans la continuité du modèle skip-gram. Les représentations des documents sont apprises de façon à ce que, lorsqu'on les passe dans un classifieur de type MLP, on maximise la probabilité des mots à l'intérieur du document. On obtient donc un modèle de langue plus simple défini par, avec θ les paramètres du réseau :

$$p(w_i|\mathcal{D}) = f_{mlp}(f_{map}(\mathcal{D}); \theta). \tag{2.57}$$

Le modèle est entraîné de façon à maximiser la probabilité des mots du document. Cette méthode a également le désavantage d'être transductive. Ces approches, même si elles fournissent de bons résultats sur différentes tâches d'évaluation par rapport aux approches Tf (analyse de sentiment et recherche d'information par exemple) ont rapidement été dépassées par des méthodes par agrégation.

2.5.2 Méthodes d'agrégation

Dans cette sous-section, nous présenterons les approches qu'on qualifie de méthodes d'agrégations. Ces travaux construisent des fonctions qui permettent d'agréger les séquences des représentations vectorielles des mots d'un document. Elles ont souvent deux intérêts majeurs : elles sont inductives, contrairement à Doc2Vec, et permettent de transférer l'information apprise par la construction de représentations vectorielles de mots sur de très gros corpus.

Dans (Le and Mikolov, 2014), les auteurs comparent leur approche à une simple moyenne de représentations vectorielles de mots. Cette dernière approche semble malgré tout fournir de bons résultats. Arora et al. (2017) proposent un modèle simple, qui représente le document comme une moyenne pondérée des vecteurs associés aux mots du document, légèrement altérée au moyen du résultat de la décomposition en valeurs propres de la matrice des représentations vectorielles des documents. Cette solution résulte de la modélisation choisie par les auteurs. L'hypothèse générative stipule que les mots du document sont tirés de façon à ce que la représentation vectorielle dans $\mathbb{R}^r$ d'un mot du document est proche du vecteur $d = f_{map}(\mathcal{D}) \in \mathbb{R}^r$ propre à chaque document du corpus. Les auteurs construisent d'abord une représentation intermédiaire :

$$\tilde{d} = \beta d_0 + (1 - \beta)d. \tag{2.58}$$

β est un hyperparamètre entre 0 et 1 et $d_0 \in \mathbb{R}^r$ un vecteur partagé par tous les documents du corpus. Cette représentation intermédiaire permet de rapprocher les représentations des mots centraux du corpus. Ensuite, la probabilité d'un mot du document est :

$$\begin{aligned} p(w_i|\mathcal{D}) =& \alpha p(w_i) + (1 - \alpha)\frac{\exp(< \tilde{d}, f_{map}(w_i) >)}{Z} \\ Z =& \sum_{j=1}^{J} \exp(< \tilde{d}, f_{map}(w_j) >). \end{aligned} \tag{2.59}$$

avec $< \tilde{d}, f_{map}(w_i) >$ le produit scalaire entre la représentation du document et celle du mot i. $\alpha \in [0, 1]$ est un hyperparamètre. $\alpha p(w_i)$ permet aux mots courants d'apparaître dans le document malgré une faible proximité à $\tilde{d}$, comme les mots vides. Sous certaines hypothèses, les auteurs montrent que la solution par maximum de vraisemblance pour $\tilde{d}$ est :

$$\begin{aligned} \tilde{d} =& \frac{1}{l} \sum_{w \in \mathcal{D}} \frac{a}{p(w) + a} f_{map}(w) \\ a =& \frac{1 - \alpha}{\alpha Z}. \end{aligned} \tag{2.60}$$

Les vecteurs d sont ensuite approximés en retranchant leur projection sur le premier axe factoriel calculé sur le corpus entier, qui approxime d_0. Soit s_c le premier vecteur propre :

$$d = \tilde{d} - s_c s_c^\top \tilde{d}. \tag{2.61}$$

Ces dernières approches ne prennent pas en compte l'ordre des mots, qui est pourtant important dans le sens d'une phrase. Dans (Kiros et al., 2015), les auteurs proposent d'utiliser une architecture Seq2Seq : $\mathcal{D}_i^w$ est passée dans un RNN pour produire une représentation vectorielle unique de la

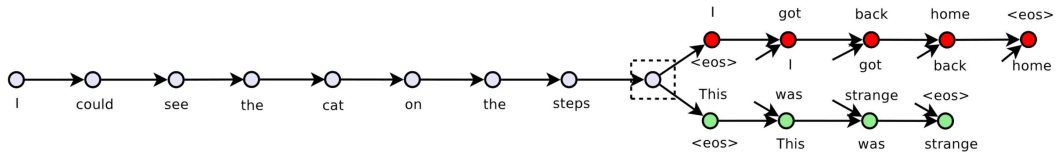


FIGURE 2.10 – Schéma de la méthode Skip-Thought issu de (Kiros et al., 2015). Pour un triplet $(s-1, s, s+1)$ de phrases contiguës, la phrase s est encodée, puis le réseau reconstruit la phrase précédente et la phrase suivante.

phrase (le dernier état caché), puis un décodeur reconstruit les séquences $\mathcal{D}^w$ du document précédent et du document suivant. C'est la méthode Skip-Thought, schématisée en figure 2.10. On appellera par la suite *encodeur* le réseau de neurones qui prend en entrée la matrice des représentations vectorielles des mots du document et ressort une représentation vectorielle du document $d \in \mathbb{R}^r$. Concrètement, le réseau est un cas particulier de la fonction de mapping telle que $d = f_{map}(\mathcal{D}^w; \theta)$.

Conneau et al. (2017) proposent la méthode InferSENT, qui utilise 4 architectures supplémentaires d'encodeurs, entraînées sur une tâche de classification du jeu de donnée SNLI (Bowman et al., 2015). Ce dataset contient 570 000 paires de phrases en anglais, permettant d'entraîner des modèles sur la tâche d'inférence en langage naturel, ou NLI (*Natural Language Inference*). Chaque paire est annotée comme *implication, contradiction, ou neutre*. En tableau 2.3, nous présentons quelques exemples de ce jeu de donnée. Chaque document est encodé, puis un vecteur contenant différentes mesures est construit à partir des représentations des documents. Le vecteur contient les représentations des deux phrases, la valeur absolue de leur différence et leur produit. Ce vecteur est passé dans un MLP avec activation softmax, de façon à prédire la bonne classe parmi les trois labels possibles du jeu de donnée. Cette procédure est présentée en figure 2.11 (deuxième schéma). Les 4 architectures sont les suivantes :

— Un RNN dont ils conservent seulement le dernier état caché,

— Un BiLSTM avec mean/max-pooling,

— Un BiLSTM additionné à un réseau d'attention. Soit $(h_1, ..., h_l)$ la séquence d'états cachés retournée par le BiLSTM, avec $h_i \in \mathbb{R}^r$, le self-encoder network (Liu et al., 2016) calcule :

$$\begin{aligned} \bar{h}_i &= \tanh\left(W h_i + b_c\right) \\ \alpha_i &= \frac{e^{h_i^T h_0}}{\sum_{i=1}^{l} e^{\bar{h}_i^T h_0}} \\ d &= \sum_{i=1}^{l} \alpha_i h_i \end{aligned} \tag{2.62}$$

avec $h_0 \in \mathbb{R}^r$ un vecteur de contexte appris et $\alpha \in \mathbb{R}^l$ le vecteur des poids d'attention,

— un réseau convolutif hiérarchique, composé de quatre couches de convolution. Chaque couche de convolution est agrégée par max pooling, et la concaténation des quatre vecteurs obtenus est utilisée comme représentation du document.

Le modèle USE (Cer et al., 2018) propose deux nouvelles architectures. De façon à construire des représentations plus générales des phrases, donc un encodeur "universel", les auteurs proposent également d'entraîner les encodeurs sur plusieurs tâches.

Texte	Hypothèse	Nature
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutre
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	contradiction
A soccer game with multiple males playing.	Some men are playing a sport.	implication
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	neutre

Tableau 2.3 – Exemples de paires du jeu de données SNLI (Bowman et al., 2015)

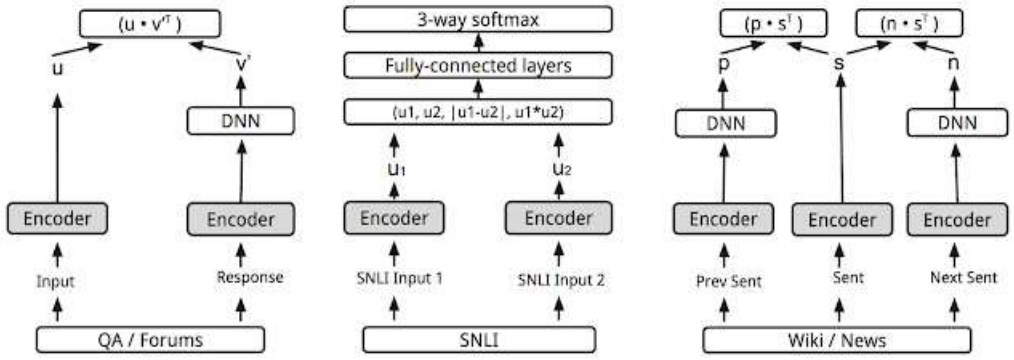


FIGURE 2.11 – Les trois tâches d'entraînement du modèle USE (Cer et al., 2018), illustration issue de `https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html`.

Dans la première tâche, les phrases sont encodées, et le réseau est entraîné à maximiser la proximité entre les représentations des phrases contiguës, et la représentation de la phrase source, similairement à (Kiros et al., 2015).

Dans la deuxième tâche, K réponses sont proposées pour une question source donnée. Le réseau calcule le produit scalaire entre la représentation de la phrase source et celles des réponses proposées. Les paramètres sont entraînés de façon à maximiser la probabilité de la bonne réponse.

Enfin, le réseau est entraîné sur la même tâche que le modèle InferSent, sur les données SNLI. Les trois tâches sont présentées en figure 2.11. Les encodeurs du modèle USE sont un Transformer, similairement à BERT, et un réseau DAN, pour Deep Averaging Network (Iyyer et al., 2015). La matrice des représentations vectorielles des mots est simplement moyennée puis passée dans un MLP. Le réseau DAN a l'avantage d'être beaucoup plus rapide à entraîner, même si ses performances sont légèrement moins bonnes que le Transformer.

Enfin, la méthode SBERT (Reimers and Gurevych, 2019) propose un procédé légèrement différent. De façon expérimentale, les auteurs montrent que le vecteur associé au token CLS tel qu'il est retourné par le modèle BERT/RoBERTa ne permet pas de résoudre efficacement des tâches nécessitant de calculer la similarité entre documents. Le modèle est donc affiner sur la tâche de NLI avec les jeux de données SNLI et multiNLI (Williams et al., 2018), en utilisant la même méthode que Conneau et al. (2017). L'encodeur est une simple fonction de pooling sur la matrice des représentations vectorielles contextualisées retournées par le modèle BERT.

Nous avons présenté jusque-là des méthodes d'apprentissage de représentations de documents comme des vecteurs dans $\mathbb{R}^r$. Comme pour les mots, certaines méthodes proposent d'apprendre des distributions. Elles ont l'avantage de révéler une forme d'incertitude sémantique et de variabilité dans

le contenu du document. Cette hypothèse est d'autant plus pertinente que les documents peuvent couvrir plusieurs thématiques.

2.5.3 Représentations de document comme des distributions

Dans cette section, nous développons les travaux d'apprentissage de représentations de documents comme des distributions. Plusieurs méthodes bien connues modélisent les documents comme des mélanges de *thématiques/topics*, autrement dit de distributions sur le vocabulaire, comme l'analyse sémantique latente probabiliste (Hofmann, 1999) ou l'allocation de Dirichlet latente (Blei et al., 2003). Néanmoins, elles ne modélisent pas de représentation vectorielle des mots et des documents dans un espace sémantique latent.

On peut cependant citer les travaux de Das et al. (2015), dans lesquels les documents sont modélisés comme des mélanges gaussiens. Les représentations vectorielles des mots d'un document sont traitées comme des réalisations de variables aléatoires, autrement dit, chaque élément de $\mathcal{D}_i^w$ suit une loi dont les paramètres dépendent du document i. Chaque document est ensuite associé à un vecteur π_i, et l'ensemble du corpus à un ensemble de K vecteurs $\mu_k \in \mathbb{R}^r$. Ces vecteurs sont interprétés comme des représentations vectorielles de thématiques dans le même espace que les mots. La probabilité d'un document devient :

$$p(\mathcal{D}_{i,l}^w) = \sum_{k=1}^{K} \pi_{i,k} \mathcal{N}(\mathcal{D}_{i,l}^w; \mu_k, \Sigma_k). \tag{2.63}$$

Les auteurs ajoutent les a priori suivant :

$$\begin{aligned} \Sigma_k &\sim \mathcal{W}^{-1}(\mu_0, \Sigma_0) \\ \mu_k &\sim \mathcal{N}(\mu_0, \frac{1}{\kappa}\Sigma_k) \\ \pi_i &\sim \mathcal{D}ir(\pi_0). \end{aligned} \tag{2.64}$$

Nikolentzos et al. (2017) proposent d'utiliser les estimateurs classiques en statistique inférentielle en modélisant un document comme une loi Normale, et les représentations vectorielles des mots du document comme des tirages de cette loi. On arrive donc facilement aux solutions suivantes : soient μ_i la moyenne et Σ_i la matrice de variance-covariance, les estimateurs par maximum de vraisemblance pour un document sont

$$\begin{aligned} \mu_i &= \frac{1}{l_i} \sum_{w \in \mathcal{D}_i^w} w \\ \Sigma_i &= \frac{1}{l_i} \sum_{w \in \mathcal{D}_i^w} (w - \mu_i)(w - \mu_i)^\top. \end{aligned} \tag{2.65}$$

Les paramètres sont donc dans ce cas très simple : la moyenne des mots du document et leur variance empirique.

2.6 Conclusion

Nous avons présenté dans ce chapitre les aspects théoriques fondamentaux de l'apprentissage statistique, des notions d'optimisation de mesures de perte, et une famille de fonctions particulières : les réseaux de neurones. Dans un second temps, nous avons détaillé la modélisation probabiliste et plusieurs algorithmes d'apprentissage, tels que l'EM, le VEM et le VIB. Ensuite nous avons donné plus de détails sur les méthodes fondatrices, mais aussi modernes d'apprentissage de représentations de mots, d'abord statique, puis probabiliste, avant de présenter les approches de représentations dynamiques et contextualisées. Dans la dernière partie, nous avons détaillé plusieurs méthodes d'apprentissage de représentations de documents, et présenté les méthodes d'agrégations de matrices de représentations vectorielles de mots, puis les méthodes qui représentent les documents comme des distributions.

Dans la prochaine partie, nous explorerons l'apprentissage de représentations de documents, lorsqu'on observe en plus du contenu textuel, des liens entre les documents, qui peuvent être des citations, ou d'autres types de relations formant un graphe. Nous présenterons également dans ce chapitre deux contributions permettant d'intégrer des représentations de mots pré-entraînées à l'apprentissage de représentations de documents en réseau. La deuxième contribution, GELD, permet d'apprendre des densités plutôt que de simples vecteurs, introduisant ainsi une mesure de l'incertitude sémantique des documents.

Chapitre 3

Apprentissage de représentations de graphes de documents

Dans la première partie, nous avons présenté plusieurs types de métadonnées qui peuvent guider l'apprentissage des représentations des documents. Dans ce chapitre, nous nous intéressons à l'information de structure du corpus de documents, c'est-à-dire qu'on peut construire un graphe en utilisant une information additionnelle au texte. Après une présentation des contextes d'observation des graphes de documents, nous définirons les fondements théoriques de cet objet mathématique. Dans une seconde section, nous présenterons les travaux existants qui permettent d'apprendre des

représentations des documents guidées par la structure du graphe dans un espace vectoriel de dimension réduite. Nous présenterons dans un dernier temps deux contributions au domaine, RLE (Gourru et al., 2020a) et GELD (Gourru et al., 2020b), ainsi que plusieurs résultats expérimentaux, avant de conclure.

3.1 Introduction

Dans cette section, nous allons présenter le contexte et définir notre objet d'étude : les graphes de documents. Nous présenterons donc également les fondements théoriques de la modélisation des graphes et quelques mesures et outils d'analyse qui leur sont spécifiques.

3.1.1 Contexte

Dans de nombreuses situations réelles, il est possible d'extraire des relations entre documents d'un corpus, qui peuvent être de natures multiples.

Le web forme un réseau, dans lequel il est possible de naviguer grâce aux liens hypertextes. Les articles scientifiques citent généralement d'autres travaux du même domaine, à titre de comparaison, d'exemple, de preuve, ou pour marquer la continuité scientifique. Enfin, les médias sociaux reposent sur une organisation des utilisateurs par différents systèmes les mettant en contact : on peut "suivre" une personne sur Instagram et Twitter (relation unidirectionnelle), ou être "ami" avec d'autres sur Facebook (relation bidirectionnelle). On peut utiliser cette information pour organiser un corpus de publications textuelles, mais aussi les interactions des utilisateurs avec les publications. Par exemple, on peut lier deux documents d'un même utilisateur, ou de deux utilisateurs "amis", mais aussi deux publications citées/commentées par le même utilisateur.

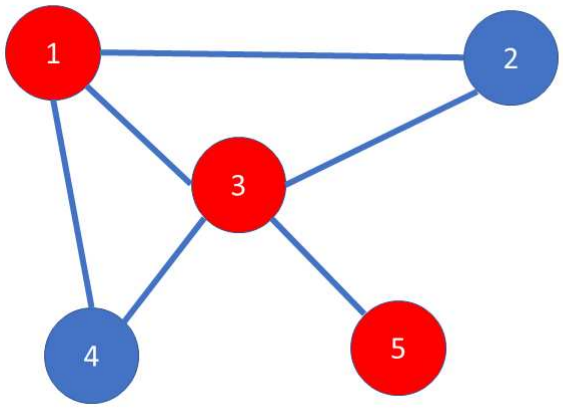
La structure du corpus fournit une information sémantique supplémentaire. Par exemple, dans les articles scientifiques, la limitation du nombre de pages conduit souvent à de courtes explications qui deviennent claires à la lecture des travaux cités. De nombreux travaux montrent que l'utilisation de cette information permet d'améliorer la qualité des représentations apprises, augmentant les performances en classification par exemple (Yang et al., 2015). La modélisation de l'information structurelle se fait au moyen d'un objet mathématique largement étudié : le graphe, qu'on va définir maintenant formellement.

3.1.2 Les Graphes

Definition 1. *Un réseau, ou graphe, G est une paire d'ensembles* (V, E)*, avec :*

- *V, un ensemble d'objets appelés sommets ou nœuds.*
- *E, un ensemble de paires de sommets, valuées ou non, appelées arêtes, ou arcs.*

Lorsque les arêtes prennent leur valeur dans $\mathbb{R}^*$, le graphe est dit "pondéré", il est "non pondéré" s'il ne peut prendre que la valeur 1 (absence ou présence d'arête). Lorsque les relations de E sont symétriques, c'est-à-dire que $(a, b) \in E$ implique que $(b, a) \in E$, il est "non dirigé", on parle alors d'arête pour le lien. Dans le cas contraire, c'est-à-dire que la relation a un sens, une orientation, on dit que le graphe est "dirigé", on parle alors d'arc.



0	0,33	0,33	0,33	0
0,5	0	0,5	0	0
0,25	0	0,25	0,25	0,25
0,5	0	0,5	0	0
0	0	1	0	0

FIGURE 3.1 – Un exemple de graphe composé de 5 nœuds et sa matrice de transition

Le degré $deg(i)$ est le nombre d'arêtes reliant un nœud au reste du graphe. La manière traditionnelle de représenter mathématiquement un graphe est la matrice d'adjacence A : c'est une matrice de dimension $n \times n$, n étant le nombre de sommets, où chaque entrée $A_{i,j}$ vaut 0 s'il n'y a pas d'arête entre le nœud i et le nœud j, le poids de l'arête sinon.

Nous considérons également la matrice de transition A_{l1}. L'entrée (i, j) de cette matrice contient la probabilité pour un marcheur aléatoire (ou une chaîne de Markov simple) de se déplacer du nœud i au nœud j. C'est aussi la probabilité de génération d'une arête entre les nœuds. Nous définissons $(A_{l1})_{i,j} = A_{i,j}/deg(i)$, avec la convention $(A_{l1})_{i,j} = 0$ si $deg(i) = 0$. La valeur de l'arête reliant les deux sommets (s'il y a une arête entre eux) donne la mesure de proximité de premier ordre.

La proximité de second ordre concerne la similarité du voisinage de deux nœuds. Deux nœuds non connectés mais ayant exactement le même voisinage sont d'une certaine manière similaires. Le moyen le plus simple de calculer cette proximité est la similarité cosinus $(A_{l1})(A_{l1})^\top$, qui compte simplement le pourcentage de voisins communs lorsque le graphe n'est pas pondéré.

Enfin, la proximité de marche d'ordre K est la probabilité pour une marche de longueur K de visiter le nœud b en partant du nœud a, donnée par $\sum_{k=1}^{K} (A_{l1})^k/K$. On peut obtenir une approximation de cette mesure en effectuant des marches aléatoires dans ce graphe. Dans les très gros graphes, cette approche est souvent utilisée pour accélérer les calculs de la similarité, comme dans les travaux de (Perozzi et al., 2014).

Une marche aléatoire consiste à tirer un nombre fini de séquences de nœuds en partant d'un nœud initial. A chaque étape, le nœud suivant est tiré en fonction des probabilités données par la ligne de la matrice de transition qui correspond au nœud actuellement visité par la marche. Par exemple, en figure 3.1, la marche $(1, 3, 5)$ commence au nœud 1, puis visite le nœud 3 avec une probabilité de $1/3$ puis le nœud 5 avec une probabilité de $1/4$. On peut réaliser un nombre fini de marches partant de chaque nœud du graphe. On obtient un ensemble de séquences de nœuds, sur lesquelles on peut faire passer une fenêtre de taille fixée, et on incrémentera l'entrée de la matrice de similarité correspondant à chaque paire de nœuds qui cooccurrent au sein de cette fenêtre. On obtient finalement une approximation de la proximité de marche dont l'ordre dépend de la taille de la fenêtre.

Une dernière mesure couramment utilisée est la longueur du plus court chemin séparant deux

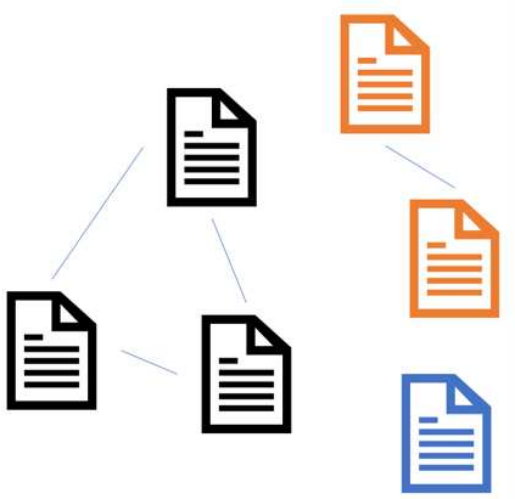


FIGURE 3.2 – Un réseau de documents est composé de documents liés entre eux. La structure peut être extraite de différentes informations, la principale étant le lien de citation.

nœuds du graphe. Elle compte le nombre minimal d'arêtes à emprunter pour aller d'un nœud a à un nœud b. Dans le cas d'un graphe qui n'est pas connexe, c'est-à-dire qu'au moins deux sous graphes ne sont pas connectés, la valeur du plus court chemin entre deux nœuds de sous graphes non connectés est fixée par convention à $+\infty$.

Dans notre cas, nous nous intéressons aux méthodes dans lesquelles les documents composent les nœuds, qu'on dit attribués, dans le sens où une information est associée à chaque nœud (le contenu textuel). D'autres travaux qu'on ne détaillera pas ici s'intéressent à la modélisation de liens entre mots et documents (Dupuy et al., 2021). La nature des arêtes/arcs dépend de l'information utilisée pour structurer le corpus. Dans les cas d'un réseau de citations (un corpus de littérature scientifique), le graphe de document est orienté et pondéré. On utilisera sans distinction les termes réseau de documents, graphe de documents, et documents liés pour qualifier ce type d'objet.

3.2 État de l'art

Dans le chapitre précédent (section 2.5), nous avons présenté les approches qui permettent d'apprendre des représentations de documents dans un espace sémantique $\mathbb{R}^r$, de façon à compresser l'information contenue dans la matrice documents-termes ou de cooccurrence des mots. Dans le cas où on ajoute une information de structure, l'information est encore plus complexe, le corpus étant associé à une matrice d'adjacence A en plus de l'information textuelle. Plusieurs travaux ont proposé de guider l'apprentissage de représentation des documents par la structure de graphe. Ils peuvent se diviser grossièrement en deux groupes, les approches basées sur la factorisation de matrice, et les méthodes qui utilisent un réseau de neurones.

3.2.1 Approches par factorisation

La première approche d'apprentissage de représentation de réseaux de document est la méthode TADW de Yang et al. (2015). Elle étend la méthode DeepWalk (Perozzi et al., 2014), développée à l'origine pour l'apprentissage de représentation des nœuds d'un graphe, en formulant le problème comme une tri-factorisation de matrices qui inclut l'information textuelle. Les auteurs proposent de factoriser la matrice de similarité de marche

$$S = \frac{A_{l1} + A_{l1}^2 + ... + A_{l1}^t}{t} \tag{3.1}$$

où t indique la longueur de marches considérée. La complexité du calcul des puissances de la matrice de transition étant cubique par rapport au nombre de nœuds, cette matrice est coûteuse à construire pour un t élevé. Les auteurs proposent donc de factoriser la matrice de marche avec $t = 2$: $S = \frac{A_{l1}+A_{l1}^2}{2}$, qui semble être un bon compromis entre performances expérimentales et temps de calcul.

Ils réduisent ensuite la dimension de la matrice Tf_{idf} au moyen d'une SVD tronquée pour obtenir la matrice de descripteurs textuels $T_t \in \mathbb{R}^{n \times r_t}$ (les r_t vecteurs singuliers gauches les plus informatifs). L'objectif de la méthode est de factoriser la matrice de similarité S, qui est issue du graphe. Cette factorisation est guidée par le contenu textuel. La fonction objectif à minimiser est :

$$\min_{W,H} ||S - WHT_t^\top||_F^2 + \frac{\lambda}{2}\big(||W||_F^2 + ||H||_F^2\big). \tag{3.2}$$

où $W \in \mathbb{R}^{n \times r}$ et $H \in \mathbb{R}^{r \times r_t}$ sont les paramètres libres. La norme de Frobenius des matrices de paramètres est régularisée. Cette régularisation permet d'empêcher le surapprentissage et a l'avantage de réduire le coût calculatoire (Yu et al., 2014). La forme de la fonction de perte force à déterminer une matrice WH particulière : le produit scalaire paire à paire de ses lignes et des lignes de T_t, donc dans l'espace sémantique, doit approximer la similarité issue du graphe. La matrice W des représentations des noeuds dans r est projetée par H dans l'espace sémantique de T_t. La représentation finale des documents est obtenue en concaténant W et $TH^\top$.

Par la suite, les auteurs de GVNR-t (Brochier et al., 2019) proposent d'étendre Glove (Pennington et al., 2014) en intégrant là aussi l'information structurelle du corpus. La matrice de similarité S est construite au moyen de marches aléatoires. Les auteurs tirent n_m marches par nœud de longueur n_l. Une fenêtre de taille n_f est passée sur chaque séquence tirée, si le nœud i et j sont séparés de n_q pas, l'entrée s_{ij} est incrémentée de $\frac{1}{n_q}$. Les valeurs en dessous d'une valeur seuil n_t sont ensuite fixées à zéro. Les auteurs proposent d'apprendre deux matrices : $U \in \mathbb{R}^{n \times r}$ et $W \in \mathbb{R}^{K \times r}$, avec K la taille du vocabulaire, qui forment des matrices de représentations respectivement des documents, et des mots. L'objectif proposé est que le produit scalaire entre la moyenne des représentations des mots dans un document j et la représentation du document i dans U soit proche du logarithme de leur similarité $s_{i,j}$. Avec n_k et c deux hyperparamètres, les auteurs proposent de résoudre :

$$\underset{U,W,b^u,b^h}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{n} \delta\left(s_{ij}\right)\Big(u_i \cdot Tf_j W + b_i^u + b_j^h - \log\left(c + s_{ij}\right)\Big)^2 \tag{3.3}$$

$$\delta(s_{ij}) = \begin{cases} 1 & \text{si } s_{ij} > 0 \\ m_i & \text{sinon, où } m_i \sim \text{Bernoulli}(\alpha_i) \end{cases}$$
$$\text{avec} \quad \alpha_i = \begin{cases} n_k \times \frac{p_i}{1-p_i} & \text{si } p_i \leq (n_k+1)^{-1} \\ 1 & \text{sinon.} \end{cases} \tag{3.4}$$

avec $b_i^u \in \mathbb{R}$, $b_j^h \in \mathbb{R}$ des biais. Les auteurs utilisent la concaténation de u_i et Tf_iW comme représentation finale du i-ème document.

Dans le modèle AANE, Huang et al. (2017) proposent quant à eux de factoriser la matrice de proximité des attributs textuels (Tf ou Td_{idf}) calculée par similarité cosinus pour chaque paire de documents, qu'on note S_t. La distance entre les représentations des documents est régularisée de façon à être basse entre documents connectés dans le graphe, élevée sinon. L'objectif final est :

$$\min_U ||S_t - UU^\top||_F^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} ||u_i - u_j||_2. \tag{3.5}$$

Cette fonction est optimisée au moyen de l'algorithme des multiplicateurs à direction alternée (Boyd et al., 2010), qui permet d'accélérer la phase d'apprentissage.

Dans la prochaine sous-section, on présentera les approches qui utilisent des architectures particulières de réseaux de neurones.

3.2.2 Approches Deep

Les travaux récents utilisent principalement des architectures profondes.

La méthode STNE (Liu et al., 2018) adapte l'architecture seq2seq présentée dans le chapitre précédent. Les auteurs proposent de générer un ensemble de séquences de documents par la réalisation de marches aléatoires de taille k_{st} dans le graphe. Cette ensemble forme les données d'entraînement du modèle. Pour une séquence de document $(d^1, .., d^{k_{st}})$, les représentations Tf_{idf} associées à chaque document d'une séquence sont passées dans un MLP, permettant d'extraire une séquence de vecteurs de dimension réduite $(d_s^{(1)}, ..., d_s^{(k_{st})})$. Cette nouvelle séquence est fournie à un Bi-LSTM, dont les auteurs conservent l'état finale seulement, c'est-à-dire la concaténation du dernier état caché de la passe forward et le dernier de la passe backward, noté w_{st}. C'est la phase d'encodage. Le décodeur génère une séquence de vecteur $(d_t^{(1)}, ..., d_t^{(k_{st})})$. Les auteurs utilisent une version simplifiée du LSTM :

$$d_t^{(t)} = \begin{cases} f_{lstm}(\mathbf{0}, w_{st}) & \text{si } t = 1 \\ f_{lstm}(\mathbf{0}, d_t^{(t-1)}) & \text{sinon,} \end{cases} \tag{3.6}$$

avec $\mathbf{0}$ un vecteur de zéros. Chaque vecteur $d_{st}^{(t)}$ est passé dans un MLP pour projeter la représentation latente dans l'espace des indices des documents. On obtient donc, après normalisation, un vecteur de probabilité (dans $\mathbb{R}^n$) sur l'ensemble des indices des nœuds du graphe (les documents), similairement aux réseaux seq2seq classiques. Les représentations étant contextualisées, les auteurs proposent de calculer la moyenne, sur toutes les occurrences du document à l'intérieur des marches des représentations issues du Bi-LSMT encodeur (la concaténation de la représentation de la phase

forward et de la phase backward). Une importante limitation de cette approche est que pour des marches de taille importante, toute la séquence du contenu des documents sera résumée par un unique vecteur w_{st}. Plusieurs approches ont donc introduit des mécanismes d'attention (présentés en chapitre 2) pour dépasser cette limitation.

Tu et al. (2017) proposent d'apprendre deux représentations par document : une représentation $u^s \in \mathbb{R}^{r/2}$ basée sur la structure du graphe et une seconde $u^t \in \mathbb{R}^{r/2}$ sur le contenu textuel. Une représentation unique u peut être construite en concaténant ces deux vecteurs. Les vecteurs u^s sont des paramètres, et les vecteurs u^t sont calculés au moyen d'un mécanisme d'attention. Pour chaque paire de nœuds (i, j), les mots de chaque document sont remplacés par leur représentation vectorielle, qui sont des paramètres du réseau, puis passés dans une couche de convolution. Le résultat de cette convolution fournit deux matrices P_i et P_j. A partir de ces deux matrices, le modèle construit une nouvelle matrice contextualisée, spécifique à la paire (i, j) telle qu'il existe un lien entre les nœuds i et j ($A_{i,j} \neq 0$) :

$$P_{cont,i,j} = tanh(P_i^\top A_{att} P_j), \tag{3.7}$$

avec A_{att} une matrice de paramètres. Cette matrice est ensuite agrégée au moyen d'un mean-pooling en ligne, suivi d'un softmax qui donne donc un vecteur de probabilité $u^{att}_{i|j}$. La représentation de i basée sur le contenu textuel est ensuite :

$$u^t_{i|j} = P_i u^{att}_{i|j} \tag{3.8}$$

Le mean-pooling en colonne suivi d'un softmax donne le vecteur de probabilité $u^{att}_{j|i}$. La représentation de j basée sur le contenu textuelle est ensuite :

$$u^t_{j|i} = P_j u^{att}_{j|i} \tag{3.9}$$

On obtient donc une représentation contextualisée par j du document i, et ce pour document j connecté à i dans le graphe. L'objectif à maximiser est composé de quatre sous objectifs :

$$\mathcal{L}_{cane}(i,j) = A_{i,j}\left(\log p(u^s_i|u^s_j) + \alpha \log p(u^t_{i|j}|u^t_{j|i}) + \beta \log p(u^t_{i|j}|u^s_j) + \gamma \log p(u^s_i|u^t_{j|i})\right), \tag{3.10}$$

où la probabilité conditionnelle $p(u|v)$ est calculée comme le produit scalaire entre u et v passé dans une fonction softmax. Cette dernière est approximée en utilisant la même approche par tirage négatif que (Mikolov et al., 2013b). Les auteurs proposent de construire une représentation unique des documents de la même façon que (Liu et al., 2018). Pour un document, ils calculent la moyenne des représentations contextualisées calculées pour chaque lien partant du document ou y aboutissant. Soit $\mathcal{E}_i = \{j | A_{i,j} \neq 0 \ \vee A_{j,i} \neq 0\}$:

$$u^t_i = \frac{1}{|\mathcal{E}_i|} \sum_{j|\mathcal{E}_i} u^t_{i|j} \tag{3.11}$$

IDNE (Brochier et al., 2020) intègre également un mécanisme d'attention. La particularité de cette approche est qu'elle apprend, conjointement aux représentations des noeuds, des vecteurs de thématiques, similairement à Das et al. (2015). Soient $T_{idne} \in \mathbb{R}^{n_{to} \times r}$ et $W_{idne} \in \mathbb{R}^{K \times r}$ deux

matrices de paramètres (les représentations des thématiques et des mots), une matrice $Z \in \mathbb{R}^{n_{to} \times K}$ est construite en calculant l'attention entre les représentations des mots d'un document et les représentations des thématiques. Soit $f_{diag}(x)$ la fonction qui retourne une matrice $r \times r$ à partir d'un vecteur de dimension x et dont la diagonale vaut x :

$$Z_i = g(T_{idne} W_{idne}^{\top} f_{diag}(tf_i)) \tag{3.12}$$

où g est une fonction d'activation ReLU suivie d'une normalisation en colonne. Cette matrice permet ensuite de calculer la représentation d'un document :

$$d_i = \sum_{j=1}^{n_{to}} \frac{z_{i,j} f_{diag}(tf_i) W}{n_i}. \tag{3.13}$$

La représentation du document dépend donc des poids d'attention entre les mots qui le composent et les vecteurs qui représentent les différentes thématiques du corpus. Les représentations obtenues sont ensuite entraînées de façon à ce qu'elles expliquent bien la structure du graphe. Le produit scalaire entre les représentations issues des thématiques de deux documents connectés devra être élevé. Les auteurs proposent donc de maximiser :

$$\mathcal{L}_{idne} = \sum_i \sum_j \delta\left(s_{ij}\right) \sigma(d_i^{\top} d_j) + (1 - \delta\left(s_{ij}\right)) \sigma(-d_i^{\top} d_j) \tag{3.14}$$

$$\delta\left(s_{ij}\right) = \begin{cases} 1 & \text{si } 2s_{ij} > 0 \\ 0 & \text{sinon} \end{cases} \tag{3.15}$$

IDNE, contrairement à la plupart des approches existantes est inductive, c'est-à-dire qu'elle permet de calculer la représentation d'un document en réseau qui n'a pas été vu pendant l'entraînement.

La méthode GraphSAGE (Hamilton et al., 2017) a été proposée quelque temps auparavant, elle aussi pour contrevenir à la transductivité des modèles de la littérature. Elle est basée sur des *fonctions d'agrégation* dont les paramètres sont appris : la représentation vectorielle d'un nouveau nœud est fonction des attributs de son voisinage local. Soit X la matrice contenant en ligne les attributs des nœuds. x_i est le vecteur associé à la i-ème observation, ici le document : les auteurs proposent de concaténer des représentations pré-entraînées des documents à partir de leur contenu textuel et des indicateurs issus du graphe (degré du document). Le modèle contient k_g fonctions d'agrégation $f_k()$ et k_g matrices de poids W_k. On rappelle que $\mathcal{E}_i$ est l'ensemble des documents connecté au i-ème document. Les auteurs proposent l'algorithme suivant pour encoder les documents :

Algorithm 1 Algorithme GraphSAGE "forward"

$h_v^0 = x_v, \forall v \in \mathcal{V}$
pour $k = 1$ à K_g **faire**
  pour $i = 1$ à n **faire**
    $h = f_k(\{h_j^{k-1}, \forall j \in \mathcal{E}_i\})$
    $h_i^k = \sigma(W_k.[h_v^{k-1}, h])$
  fin pour
fin pour
pour $i = 1$ à n **faire**
  $d_i = h_i^{K_g}/||h_i^{K_g}||_2$
fin pour

Les fonctions f_k peuvent prendre plusieurs formes : un MLP nommé Moyenne, qui étend le GCN, un LSTM, et un réseau MLP avec Max-Pooling.

Le voisinage est sous-échantillonné de façon à accélérer les calculs sur les très gros graphes, à partir d'un tirage uniforme de s_k nœuds (différent selon la "couche"). Enfin, les auteurs construisent une matrice de similarité S en utilisant des marches aléatoires, similairement à (Perozzi et al., 2014) : l'entrée $S_{i,j}$ compte le nombre de fois où deux documents ont cooccurré à l'intérieur d'une fenêtre de taille fixe dans une marche aléatoire. Soient r_{gs} le nombre de tirages aléatoires et $p_n(v)$ une distribution de tirages négatifs (qui associe à chaque document une probabilité). La fonction objectif non-supervisée de GraphSAGE est :

$$\mathcal{L}_{gs}(\cdot) = \sum_{(i,j)|S_{i,j}>0} \log \sigma(d_i^\top . d_j) + r_{gs} \cdot \mathbb{E}_{j' \sim p_n(v)}[\log \sigma(d_i^\top . d_{j'})]. \tag{3.16}$$

d_i est le résultat de l'algorithme 1. $\mathcal{L}_{gs}(\cdot)$ est optimisée par descente de gradient stochastique. Les paramètres libres sont les matrices W_k et les paramètres des fonctions d'agrégation f_k.

Toutes les méthodes que nous avons présentées jusque-là apprennent un vecteur unique par document. Cette hypothèse est limitée car les documents peuvent être sémantiquement riches.

Une première solution serait de représenter un document comme un mélange de vecteurs, qui, s'ils sont partagés par tous les documents du corpus, rendrait cette approche équivalente aux approches thématiques (Das et al., 2015; Blei et al., 2003).

La deuxième solution consiste à apprendre un vecteur d'incertitude associé à la représentation vectorielle, dans la continuité des travaux que nous avons présentés en sous-section 2.4.2. Cette deuxième approche a l'avantage de ne pas faire exploser le nombre de paramètres. Elle permet également de calculer simplement des similarités entre documents, soit au moyen de mesures de proximité entre distributions de probabilité (divergence de Kullback-Liebler), soit en utilisant des mesures de similarité classiques entre les moyennes des documents. Les moyennes peuvent également servir à partitionner/classifier les documents similairement aux approches de représentation vectorielle de documents présentées précédemment.

Le modèle VGAE (Kipf and Welling, 2016) intègre un Graph Convolutionnal Network (GCN) à deux couches dans un auto-encodeur variationnel. Le GCN est une architecture de réseau de neurones qui permet de modéliser les données en graphe au moyen d'une diffusion d'états latents au voisinage

des nœuds. Le GCN à deux couches, dans sa forme fonctionnelle, est défini, pour une matrice des attributs associés aux nœuds du graphe $X \in \mathbb{R}^{n \times f}$ et A la matrice d'adjacence, par :

$$GCN(X, A) = \tilde{A}ReLU(\tilde{A}XW_0)W_1 \tag{3.17}$$

où $\tilde{A}$ est la matrice d'adjacence normalisée symétrique et X une matrice d'attribut. L'architecture utilisée dans (Hamilton et al., 2017) qu'on a présentée plus haut est une extension du GCN.

Le modèle génératif de VGEA (Kipf and Welling, 2016) est défini par :

$$\begin{aligned} p(A|D) &= \prod_i^n \prod_i^n p(a_{i,j}|d_i, d_j) \\ p(a_{i,j} = 1|d_i, d_j) &= \sigma(d_i^\top d_j), \end{aligned} \tag{3.18}$$

associé à un a priori gaussien centré réduit pour les vecteurs d. Les auteurs proposent d'utiliser une approche variationnelle, c'est donc la ELBO qui est maximisée (cf section 2.3), définie par :

$$\mathcal{L}_{vgae} = \mathbb{E}_q[\log p(A|D)] + \mathbb{E}_q[\log p(D)] - \mathbb{E}_q[\log q(D|X, A)]. \tag{3.19}$$

Similairement aux VAE, $q(D|X, A)$ est une loi normale dont les paramètres sont donnés par un réseau de neurones, ici un GCN :

$$q(d_i|X, A) = \mathcal{N}(d_i; \mu_i, I\sigma_i^2), \tag{3.20}$$

avec $\mu_i = GCN_\mu(X, A)_i$, c'est-à-dire la i-ème ligne de la matrice retournée par un GCN et $\log \sigma_i = GCN_\sigma(X, A)_i$, $\tilde{A}$ est la matrice d'adjacence normalisée symétrique et X la matrice Tf.

Le modèle CAN (Meng et al., 2019) étend ce modèle en ajoutant un sous objectif qui prend la forme d'une loi conditionnelle sur la matrice d'attribut. Soit W_{can} la matrice des représentations vectorielles des mots, les auteurs définissent la vraisemblance du modèle comme :

$$\mathcal{L}_{can} = \mathcal{L}_{vgae} + \mathbb{E}_q[\log p(X|D, W)] - KL(\frac{q(W|X)}{p(W)}). \tag{3.21}$$

$p(W)$ est une normale centrée réduite qui se factorise sur tout le vocabulaire. $q(W|X)$ est une loi normale $q(w_i|X) = \mathcal{N}(w_i; \mu_{w_i}, I\sigma_{w_i}^2)$ dont les paramètres sont les sorties d'un MLP à deux couches :

$$[\mu_{w_i}, \sigma_{w_i}^2] = tanh(W_0 X_i^\top + b_0)W_1 + b_1. \tag{3.22}$$

Enfin, la probabilité des attributs $p(x_{i,j}|D, W)$, c'est-à-dire les entrées de la matrice Tf binarisée, est donnée par :

$$p(x_{i,j} = 1|d_i, w_j) = \sigma(d_i^\top w_j). \tag{3.23}$$

avec σ la fonction sigmoïde évaluée sur la similarité entre le vecteur document et le vecteur mot.

Enfin, Graph2Gauss (Bojchevski and Günnemann, 2018) propose un apprentissage à base d'énergie (LeCun et al., 2006). Chaque document est associé à une moyenne et une variance. L'objectif est que la similarité entre documents connectés soit plus grande qu'entre documents éloignés dans le graphe. La similarité est calculée au moyen d'une divergence de Kullback-Liebler

entre les gaussiennes qui représentent les documents. Soit Sp la matrice des plus courts chemins, les auteurs construisent un ensemble de triplets $E_{sp} = \{i, j_0, j_1 | Sp_{i,j_0} < Sp_{i,j_1}\}$. L'objectif à minimiser est :

$$\mathcal{L}_{g2g} = \sum_{(i,j_0,j_1 \in E_{sp}} KL(i,j_0)^2 + \exp(-KL(i,j_1)). \tag{3.24}$$

Soit elu la fonction Unité Linéaire exponentielle, définie pour $a > 0$ un hyperparamètre par :

$$elu(x) = \begin{cases} x & \text{si } x > 0 \\ a(e^x - 1) & \text{sinon} \end{cases} \tag{3.25}$$

, les moyennes et les variances sont retournées par des MLP, avec

$$\begin{aligned} h_i &= ReLU(x_i W_0 + b_0) \\ \mu_i &= h_i W_1 + b_1 \\ \sigma_i &= elu(h_i W_2 + b_2) + 1. \end{aligned} \tag{3.26}$$

A l'exception de CAN, aucune méthode ci-dessus ne peut, en même temps : i) représenter les documents et les mots dans le même espace, ii) apprendre une mesure de l'incertitude sémantique des représentations des documents. Elles ne permettent également pas d'intégrer des représentations pré-entraînées des mots dans la plupart des cas. Dans la prochaine section, nous présenterons nos propositions pour répondre à ces différentes problématiques.

3.3 Contributions

Dans cette section, nous présenterons deux contributions à l'apprentissage de représentation de documents en réseau : la méthode RLE (Gourru et al., 2020a) et la méthode GELD (Gourru et al., 2020b). Ces contributions présentent plusieurs avantages et sont plus performantes que l'existant sur plusieurs tâches d'évaluation. RLE est rapide et ne possède qu'un seul hyperparamètre à optimiser. GELD repose sur une hypothèse probabiliste qui permet d'associer chaque document à une mesure d'incertitude dans l'espace sémantique. Enfin, ces deux méthodes permettent d'intégrer des représentations de mots pré-entraînées.

3.3.1 Notations

Comme dans les chapitres précédents, nous considérons qu'il existe une fonction $f_{map}(w_i, C_o; \theta)$ avec des paramètres θ fixés qui permet d'associer chaque occurrence d'un mot dans un corpus avec une représentation dans un espace $\mathbb{R}^r$ en fonction de son contexte, c'est-à-dire un ensemble de mots noté C_o.

Dans les modèles comme Word2Vec (Mikolov et al., 2013b,a), après entraînement, les représentations vectorielles sont utilisées indépendamment du contexte, c'est-à-dire que $f_{map}(w_i, C_o; \theta) = f_{map}(w_i; \theta)$. Dans les modèles comme BERT (Devlin et al., 2019), les représentations vectorielles sont contextualisées en fonction des autres mots du document au moyen de mécanismes d'attention.

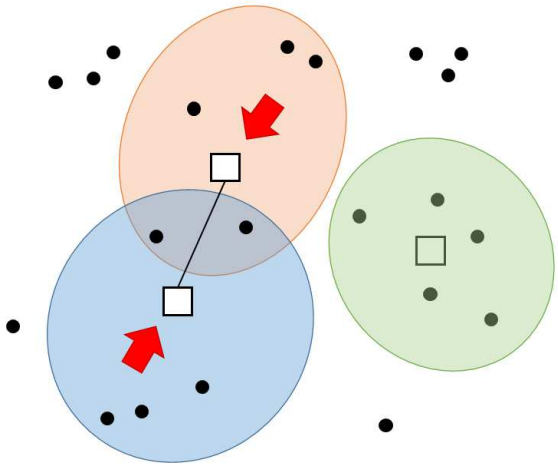


FIGURE 3.3 – La méthode RLE effectue un lissage (représenté par des flèches rouges) sur les moyennes des représentations des mots des documents (les blocs carrés). Le document dans le cercle bleu (les points sont des mots) est connecté au document orange, leurs représentations se rapprochent. Le document dans le cercle vert est isolé dans le graphe, sa représentation reste inchangée par l'effet de lissage.

Dans les deux cas, les représentations vectorielles peuvent être construites à l'avance (les paramètres étant fixés). Elles sont stockées dans une matrice $U \in \mathbb{R}^{m \times r}$. On note ici m le nombre total de mots du corpus et non la taille du vocabulaire. Pour encoder le contenu textuel du corpus, on utilise une matrice document terme normalisée Tf_{l1}.

3.3.2 Contribution 1 : RLE

Notre première proposition, "Regularized Linear Embedding" (RLE), est une approche simple, qui a plusieurs avantages : elle est faiblement paramétrée, très rapide et produit de bonnes performances sur plusieurs tâches et jeux de données.

Similairement à plusieurs méthodes existantes (Yang et al., 2015), on extrait une matrice de similarité $S \in \mathbb{R}^{n \times n}$, calculée comme $S = \frac{(A_{l1})+(A_{l1})(A_{l1})^\top}{2}$ avec A_{l1} la matrice de transition du graphe. De la même façon que TADW, les marches d'ordre 2 semblent offrir un bon compromis entre précision et temps de calcul.

Nous construisons, pour chaque document, un vecteur de poids $p_i \in \mathbb{R}^r$. Ces vecteurs sont stockés dans une matrice P. Nous définissons la représentation vectorielle d'un document par :

$$d_i = p_i^\top U. \tag{3.27}$$

d_i est donc une simple moyenne pondérée des représentations des mots encodées dans la matrice U. Nous construisons p_i comme suit : nous calculons d'abord une matrice de lissage $B \in \mathbb{R}^{n \times m}$ avec :

$$b_i = \frac{1}{\sum_j S_{i,j}} \sum_j S_{i,j} \cdot (t_{l1})_j. \tag{3.28}$$

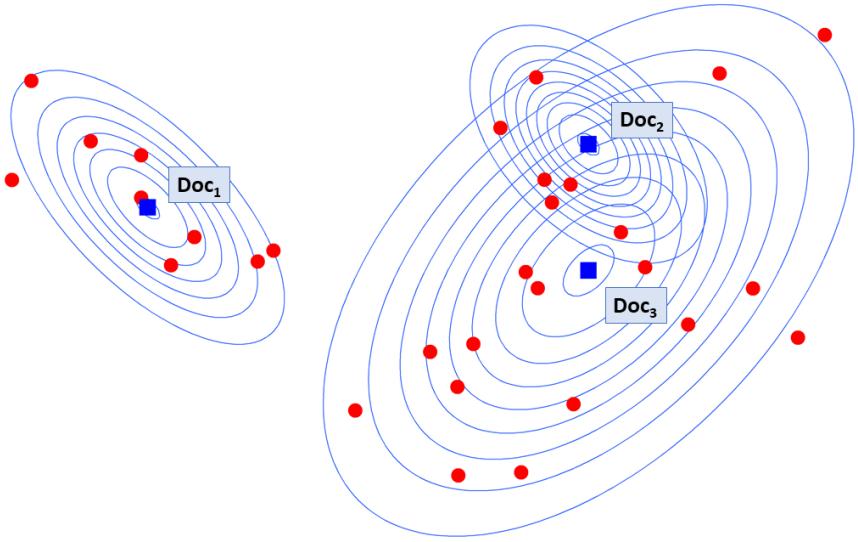


FIGURE 3.4 – Un document est représenté comme une distribution gaussienne (illustrée par les cercles bleus). Les carrés bleus représentent les moyennes. Les mots sont en rouge. Doc_2 et Doc_3 partagent des mots. Dans ce cas, Doc_3 cite Doc_2 mais Doc_2 ne cite pas Doc_3. Doc_1 ne partage aucun mot avec Doc_3 et Doc_2 et ne les cite pas.

Chaque ligne b_i de cette matrice est une moyenne de la matrice initiale de fréquence des termes du document Tf, pondérée par la similarité entre le document i et chacun des autres documents. Ensuite, nous calculons la matrice de poids P en fonction de Tf et B, en notation matricielle :

$$P = (1 - \lambda)Tf_{l1} + \lambda B, \tag{3.29}$$

où $\lambda \in [0, 1]$ contrôle l'intensité du lissage. Ensuite, nous calculons $D = PU$. Ainsi, le poids donné à la représentation d'un mot dépend d'une combinaison linéaire entre sa fréquence dans le document et sa fréquence dans les documents similaires (pondérée par les entrées de S). Si le mot est fréquent dans le document, mais aussi dans les voisins du document, son impact sera d'autant plus important. De façon équivalente, l'impact d'un mot courant dans le document, mais très peu dans les documents auxquels il est lié, sera réduit.

Notre méthode implique uniquement une multiplication et une normalisation de la matrice, ce qui la rend rapide et parallélisable. Lorsque $\lambda = 0$, $P = T_{l1}$, nous construisons une simple moyenne des représentations des mots du document. Lorsque $\lambda = 1$, nous obtenons $P = B$ et la représentation du document est fonction de l'information textuelle du voisinage uniquement (c'est-à-dire les documents similaires). Nous illustrons l'effet du lissage dans la figure 3.3.

3.3.3 Contribution 2 : GELD

Dans cette section, nous présentons un modèle original qui apprend à la fois une représentation vectorielle et un vecteur d'incertitude pour chaque document, appelé GELD pour Gaussian Embedding of Linked Documents. L'incertitude devra révéler à la fois la variance du réseau et du texte. Elle sera

plus élevée si le document cite des ensembles de documents très différents et s'il utilise des mots sémantiquement éloignés. En outre, les documents et les mots se trouvent dans le même espace : on peut calculer les similarités entre les documents et les mots dans cet espace sémantique, améliorer les requêtes ou décrire des groupes de documents en utilisant des mots proches.

On rappelle que chaque document est associé à une séquence de représentations vectorielles de mots

$$\mathcal{D}_i^w = \left(f_{map}(w^{i,1}, \mathcal{D}_i^w), f_{map}(w^{i,1}, \mathcal{D}_i^w), \ldots\right) \tag{3.30}$$

, où $w^{i,1}$ est le premier mot utilisé dans le document i. Notre objectif est d'apprendre les représentations des documents comme des distributions gaussiennes. Chaque document possède deux paramètres : une moyenne μ_i dans $\mathbb{R}^r$ et une variance diagonale $\sigma_i^2 I$, avec $\sigma_i^2 \in \mathbb{R}^r$, qui révèle l'incertitude du document.

Nous pouvons utiliser la moyenne du document comme représentation vectorielle lorsque c'est nécessaire, par exemple, dans les tâches de classification. Nous notons g la fonction qui fait correspondre le document d_i à sa moyenne $g(d_i) = \mu_i \in \mathbb{R}^r$. En utilisant l'information du réseau, nous avons donc, pour chaque document, un multiensemble de représentations vectorielles de documents cités par le document d_i, noté $\mathcal{D}_i^l = \{g(d^{i,1}), g(d^{i,2}), \ldots\}$, où $d^{i,1}$ est le premier document cité par le document i.

En prenant l'union de ces multiensembles, le corpus devient n multiensembles de vecteurs $\mathcal{D}_i = \mathcal{D}_i^w \cup \mathcal{D}_i^l$:

$$\mathcal{D}_i = \{f_{map}(w^{i,1}, \mathcal{D}_i^w), f_{map}(w^{i,2}, \mathcal{D}_i^w), \ldots, g(d^{i,1}), g(d^{i,2}), \ldots\} \tag{3.31}$$

dont le j-ième élément est noté $\mathcal{D}_{i,j} \in \mathbb{R}^r$.

Modèle and Optimisation Comme dans (Das et al., 2015; Nikolentzos et al., 2017), nous supposons que les vecteurs dans $\mathcal{D}_i$ sont tirés indépendamment d'une loi Normale de variance diagonale, ce qui signifie que :

$$\mathcal{D}_{i,j} \sim \mathcal{N}(\mu_i, \sigma_i^2 I). \tag{3.32}$$

La loi Normale est paramétrée par une moyenne et une variance diagonale caractérisant la représentation vectorielle du document et son incertitude. Les paramètres à apprendre pour chaque document sont $\sigma_i^2 \in \mathbb{R}^r$ et $\mu_i \in \mathbb{R}^r$. L'utilisation de variances diagonales permet de réduire nettement le nombre de paramètres de notre méthode et d'accélérer les calculs. Cette astuce est couramment utilisée dans la littérature (Bojchevski and Günnemann, 2018; Vilnis and McCallum, 2015). La log-vraisemblance du modèle proposé est, avec $|\mathcal{D}_i|$ la cardinalité de $\mathcal{D}_i$:

$$\mathcal{L}(\mathcal{D}; \mu, \sigma^2) \quad = \quad \sum_{i=1}^{n} \sum_{j}^{|\mathcal{D}_i|} \log \mathcal{N}(\mathcal{D}_{i,j}; \mu_i, \sigma_i^2 I) \tag{3.33}$$

où $\mathcal{D} = \{D_i\}_{i=1}^n$, $\sigma^2 = \{\sigma_i^2\}_{i=1}^n$ et $\mu = \{\mu_i\}_{i=1}^n$.

On peut réécrire la log-vraisemblance en séparant les mots ($\mathcal{L}_w$) et les documents ($\mathcal{L}_d$).

$$\begin{aligned}\mathcal{L}(\mathcal{D};\mu,\sigma^2) &= \sum_{i=1}^{n} \sum_{f_{map}(w,\mathcal{D}_i^w)\in\mathcal{D}_i^w} \log\mathcal{N}(f_{map}(w,\mathcal{D}_i^w);\mu_i,\sigma_i^2 I) \\ &+ \sum_{i=1}^{n} \sum_{g(d)\in\mathcal{D}_i^l} \log\mathcal{N}(g(d);\mu_i,\sigma_i^2 I) \\ &= \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{m} T_{i,k}\log\mathcal{N}(u_k;\mu_i,\sigma_i^2 I)}_{\mathcal{L}_w} \\ &+ \underbrace{\sum_{i=1}^{n}\sum_{l=1}^{n} A_{i,l}\log\mathcal{N}(\mu_l;\mu_i,\sigma_i^2 I)}_{\mathcal{L}_d}.\end{aligned} \tag{3.34}$$

Comme les citations et les termes n'ont pas la même fréquence, nous observons un déséquilibre de cardinalité entre $\mathcal{L}_w$ et $\mathcal{L}_d$. Ce problème est fréquent lorsqu'on modélise des données hétérogènes à l'aide du même processus génératif (Wang, 2001). Nous optimisons donc une nouvelle fonction de vraisemblance cette fois-ci pondérée. En définissant $\eta \in [0,1]$, dénotant l'importance donnée à l'information du réseau, la fonction alternative à optimiser est :

$$\widetilde{\mathcal{L}} = (1-\eta)\mathcal{L}_w + \eta\mathcal{L}_d \tag{3.35}$$

En calculant et en annulant le gradient, on obtient les solutions optimales μ_i^* et $(\sigma_{i,r}^2)^*$:

$$\mu_i^* = \frac{\eta\sum_k \frac{T_{i,k}u_k}{\sigma_i^2} + (1-\eta)\sum_j \frac{A_{i,j}\mu_j}{\sigma_i^2} + (1-\eta)\sum_j \frac{A_{j,i}\mu_j}{\sigma_j^2}}{\eta\sum_k \frac{T_{i,k}}{\sigma_i^2} + (1-\eta)\sum_j \frac{A_{i,j}}{\sigma_i^2} + (1-\eta)\sum_j \frac{A_{j,i}}{\sigma_j^2}} \tag{3.36}$$

$$(\sigma_{i,r}^2)^* = \frac{\eta\sum_k T_{i,k}(\mu_{i,r}-u_{k,r})^2 + (1-\eta)\sum_j A_{i,j}(\mu_{i,r}-\mu_{j,r})^2}{\eta\sum_k T_{i,k} + (1-\eta)\sum_j A_{i,j}} \tag{3.37}$$

Nous maximisons $\widetilde{\mathcal{L}}$ en chaque paramètre et répétons le processus jusqu'à convergence. L'optimisation est itérative, et chaque mise à jour des paramètres dépend des valeurs courantes des autres paramètres. En raison de cette dépendance, nous proposons d'adopter la méthode de Robbins-Monro (Robbins and Monro, 1951) pour éviter d'aller trop vite vers une solution locale médiocre. Elle donne de bons résultats dans nos expériences. La valeur mise à jour du paramètre μ_i à l'itération k, étant données les valeurs optimales à l'itération k calculées à l'aide de l'équation 3.36 qu'on note $\mu_i^{*(k)}$, est :

$$\mu_i^{(k)} = \lambda^{(k)}\mu_i^{*(k)} + (1-\lambda^{(k)})\mu_i^{(k-1)} \tag{3.38}$$

Nous mettons à jour σ_i^2 de manière similaire. Nous proposons d'utiliser $\lambda^{(k)} = (\delta k)^{-\gamma}$, $\gamma \leq 1$ comme dans (Barkan, 2017) pour assurer les conditions de convergence. $\delta \in [0,1]$ est l'importance

Algorithm 2 GELD

Entrées : $\mathcal{D}, U$
Hyperparamètres : η, λ, k
Sortie : μ, σ^2

pour chaque : document i **faire**
  initialiser μ_i^0 et $(\sigma_i^2)^0$ d'après l'équation 3.39
fin pour
k=1
répéter
  pour chaque : document i **faire**
    calculer $\mu_i^{*(k)}$ et $(\sigma_i^2)^{*(k)}$ d'après l'équation 3.36 et 3.37
    mettre à jour $\mu_i^{(k)}$ et $(\sigma_i^2)^{(k)}$ d'après l'équation 3.38
  fin pour
  $k = k + 1$
jusqu'à convergence
retourner μ, σ^2

donnée à la solution optimale au début de l'optimisation. Dans nos expériences, nous obtenons de meilleurs résultats avec des valeurs faibles de δ. En d'autres termes, nous ne faisons pas *confiance* aux premières solutions optimales, l'optimisation est alternée en chaque paramètre.

On initialise les paramètres avec les moyennes et les variances empiriques, notées μ_i^0 et $(\sigma_i^2)^0$, calculées sur l'ensemble des représentations des mots uniquement :

$$\mu_i^0 = \frac{1}{|\mathcal{D}_i^w|} \sum_{f(w) \in \mathcal{D}_i^w} f(w)$$

$$N(\sigma_{i,r}^2)^0 = \frac{1}{|\mathcal{D}_i^w|} \sum_{f(w) \in \mathcal{D}_i^w} (f(w)_r - \mu_{i,r}^0)^2. \tag{3.39}$$

Le processus d'optimisation est résumé en algorithme 2. Nous avons présenté jusque-là nos contributions et les approches concurrentes. Dans la prochaine section, nous présenterons le cadre expérimental que nous avons utilisé pour les comparer et les évaluer.

3.4 Évaluation

3.4.1 Jeux de données

Nous expérimentons sur trois jeux de données. Cora (Tu et al., 2017) et Dblp (Tang et al., 2008; Pan et al., 2016) sont deux réseaux de citations. Cora contient 2 211 résumés de documents scientifiques étiquetés (7 classes) avec 5 001 arêtes. Dblp contient 60.744 titres de documents (4

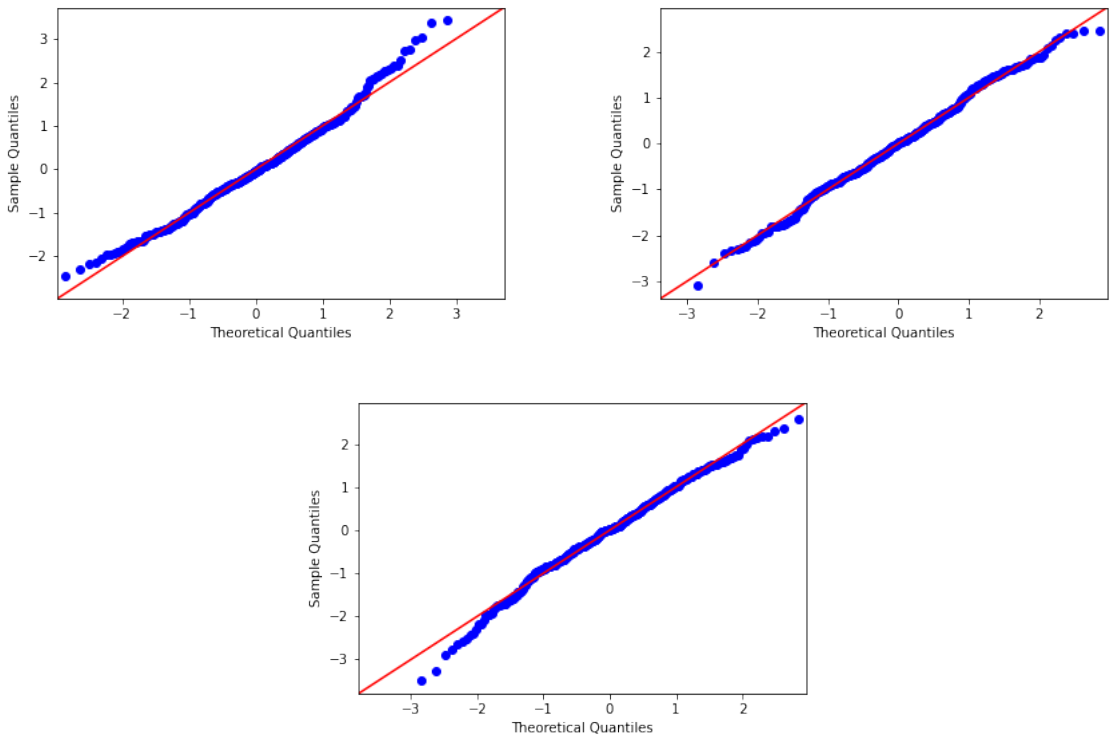


FIGURE 3.5 – Diagramme quantile-quantile des vecteurs BERT centrés réduits de trois documents du jeu de donné Cora (premier axe de représentation des trois documents les plus longs).

classes) et 52.914 arêtes entre eux. Nous utilisons également un jeu de données composé d'articles du New York Times (`https://www.nytimes.com/`) de janvier 2007. La classe correspond à la section de l'article ("politique", "sport", etc.). Nous créons un lien entre les paires d'articles partageant une étiquette commune [1]. Le jeu de données comporte 4 classes, 5 135 documents et 3 050 513 arcs. Pour chaque jeu de données, nous filtrons le vocabulaire en retirant les mots vides avec le package scikit-learn [2] et nous supprimons les mots communs et rares (les mots apparaissant moins de 4 fois et dans plus de 25% des documents). Nous obtenons une taille de vocabulaire de 4 390 sur Cora, 3 763 sur Dblp et 6 407 pour le jeu de données Nyt.

3.4.2 Tâches d'évaluation

La méthode GELD permet d'apprendre une moyenne et une variance pour chaque document. De nombreuses tâches d'application du monde réel nécessitent une représentation vectorielle unique pour un document. Nous évaluons donc la pertinence de l'utilisation de la moyenne des documents dans des tâches d'évaluation standard : la classification dans la section 3.5.1 et la prédiction de liens dans

1. Les articles sont associés à plusieurs *tags* décrivant leur contenu. Ces tags sont plus précis que la classe générale de l'article
2. `https://scikit-learn.org/`

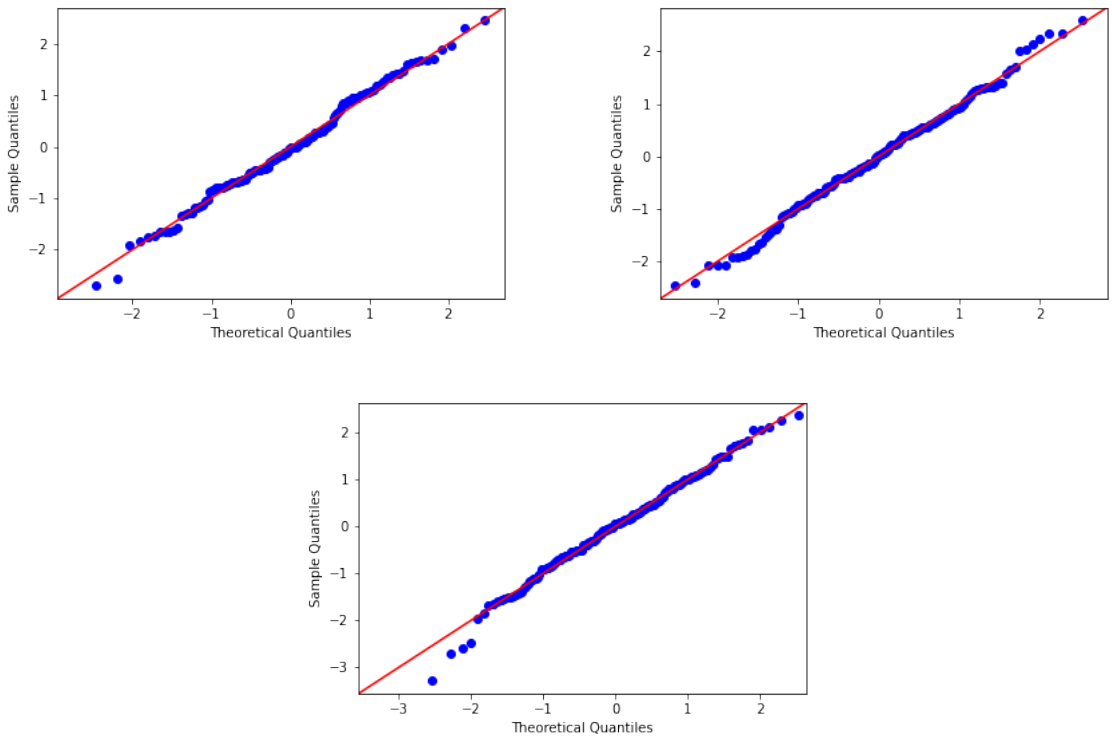


FIGURE 3.6 – Diagramme quantile-quantile des vecteurs SGNS centrés réduit de trois documents du jeu de donné Cora (premier axe de représentation des trois documents les plus longs).

la section 3.5.2. Nous évaluons RLE sur le même ensemble de tâches. Nous fournissons également des indications qualitatives sur l'utilisation possible de la variance pour GELD dans la section 3.5.4. En outre, nous démontrons également l'avantage d'intégrer les documents et les mots dans le même espace pour nos deux méthodes.

3.4.3 Compétiteurs et paramètres des méthodes

Nous comparons nos approches à des méthodes récentes. Nous utilisons trois approches basées sur la factorisation matricielle : TADW (Yang et al., 2015), AANE (Huang et al., 2017) et GVNR-t (Brochier et al., 2019) et trois modèles de réseaux de neurones profonds : VGAE (Kipf and Welling, 2016), Graph2Gauss (Bojchevski and Günnemann, 2018) et STNE (Liu et al., 2018). Nous comparons également notre méthode à DeepWalk, qui ne prend en compte que les informations du réseau, à LSA (Deerwester et al., 1990), qui construit les représentions des documents en tenant compte des informations textuelles, et à une concaténation des représentations obtenues avec ces deux méthodes que nous appelons "Concatenation" similairement à (Yang et al., 2015).

Nous utilisons toutes les implémentations originales fournies par les auteurs, excepté pour GraphSAGE (nous utilisons l'implémentation du package stellargraph `https://stellargraph.readthedocs.io`). Nous utilisons les hyperparamètres recommandés par les auteurs, pour ceux qui

	Cora		Dblp		Nyt	
ratio	10%	50%	10%	50%	10%	50%
DeepWalk	70.6 (2.0)	81.0 (0.7)	52.3 (0.4)	53.5 (0.2)	66.9 (0.7)	68.7 (0.9)
LSA	72.3 (1.9)	80.6 (0.7)	73.5 (0.2)	74.2 (0.2)	71.6 (1.0)	76.7 (0.7)
Concatenation	71.4 (2.1)	84.0 (1.1)	77.5 (0.2)	78.2(0.2)	77.9 (0.3)	81.1 (0.7)
TADW	81.9 (0.8)	87.4 (0.8)	74.8 (0.1)	75.5 (0.1)	75.8 (0.5)	79.4 (0.4)
AANE	79.8 (0.9)	84.4 (0.7)	73.3 (0.1)	74.2 (0.2)	71.7 (0.5)	76.9 (1.1)
GVNR-t	83.7 (1.2)	87.0 (0.8)	69.6 (0.1)	70.2 (0.2)	74.3 (0.4)	76.7 (0.6)
RLE	84.0 (1.3)	87.7 (0.6)	79.8(0.2)	81.2 (0.1)	77.7 (0.7)	80.0 (0.6)
VGAE	72.3 (1.7)	81.1 (0.7)	Memory Overflow		68.1 (0.8)	70.1 (0.6)
G2G	79.0 (1.5)	84.8 (0.7)	70.8 (0.1)	71.5 (0.2)	69.0 (0.5)	71.5 (0.8)
STNE	79.4 (1.0)	86.7 (0.8)	73.8 (0.2)	74.5 (0.1)	75.1 (0.7)	78.1 (0.6)
GELD	**84.3** (1.1)	**88.3** (0.4)	**81.63** (0.1)	**82.3** (0.1)	**78.5** (0.8)	**81.2** (0.3)

Tableau 3.1 – Comparaison des résultats en Micro-F1 sur une tâche de classification pour différents ratios entraînement/test. Nous indiquons l'écart-type entre parenthèses. GELD surpasse les méthodes les plus récentes pour chaque ratio entraînement/test. Sur Dblp, il surpasse TADW de 7 points.

utilisent des jeux de données similaires. Nous déterminons les autres hyperparamètres par *grid search* sur la tâche de classification de documents.

Pour TADW, nous utilisons $\lambda = 0.2$ (Yang et al., 2015) et la dimension 200 pour la représentation réduite du contenu des documents. Pour AANE, nous utilisons λ et ρ optimaux obtenus par recherche sur grille, comme x_{min} pour GVNR-t.

Pour VGAE, nous utilisons l'architecture issue de (Kipf and Welling, 2016), et $K = 1$ pour Graph2Gauss. VGAE ne permet pas de construire les représentations du jeu de données DBLP sur notre machine en un temps raisonnable.

Pour STNE, nous déterminons la profondeur en utilisant la recherche sur grille (*grid search*).

Pour DeepWalk, nous effectuons 40 marches de longueur 40 par nœud, et nous fixons la taille de la fenêtre à 10.

Pour GraphSage, nous utilisons une profondeur de 2, avec une taille d'échantillonnage du voisinage de 25 pour la première couche et 10 pour la deuxième, et 50 marches par nœud de taille 5 qui sont les paramètres utilisés dans (Hamilton et al., 2017). Nous évaluons les fonctions d'agrégation par moyenne et par max-pooling. Pour la matrice initiale d'attributs, nous utilisons (Arora et al., 2016), avec des vecteurs de mots entraînés sur le corpus avec SGNS pour construire une première représentation des documents, concaténée à leur degré, similairement à (Hamilton et al., 2017).

Nous exécutons toutes les expériences en parallèle avec 20 cœurs physiques (Intel® Xeon® CPU E5-2640 v4 @ 2.40GHz) et 96 Go de RAM de mémoire vive. Nous utilisons $r = 160$ comme dimension de représentation pour chaque méthode, similairement à (Yang et al., 2015).

De même, nous indiquons les paramètres optimaux de GELD obtenus par grid-search pour la tâche de classification : $\delta = 0,1$, $\gamma = 0,2$, $\eta = 0,99$ pour Cora, $\eta = 0,8$ pour Dblp et $\eta = 0,95$ pour Nyt. Pour RLE, nous utilisons $\lambda = 0,7$.

Pour apprendre les vecteurs de mots, nous utilisons Skip-gram avec échantillonnage négatif

	Cora		Dblp		Nyt	
% edges hidden	50%	25%	50%	25%	50%	25%
DeepWalk	73.2 (0.6)	80.9 (1.0)	89.7 (0.0)	93.2 (0.2)	88.1 (0.0)	88.1 (0.0)
LSA	87.4 (0.6)	87.2 (0.8)	54.2 (0.1)	54.8 (0.0)	54.9 (0.0)	54.9 (0.1)
Combination	77.9 (0.3)	83.7 (0.8)	88.8 (0.0)	92.6 (0.3)	88.2 (0.0)	88.3 (0.0)
TADW	90.1 (0.4)	93.3 (0.4)	61.2 (0.1)	65.0 (0.5)	**88.5 (0.0)**	**88.5 (0.0)**
AANE	83.1 (0.8)	86.6 (0.8)	67.4 (0.1)	66.5 (0.1)	58.9 (0.2)	61.2 (0.2)
GVNR-t	83.9 (0.9)	91.5 (1.1)	88.1 (0.3)	91.4 (0.1)	61.2 (0.2)	61.3 (0.3)
RLE	94.3 (0.2)	94.8 (0.2)	89.3 (0.1)	91.2 (0.2)	77.5 (0.3)	77.8 (0.2)
VGAE	87.1 (0.4)	88.2 (0.7)	Memory Overflow		88.4 (0.0)	88.4 (0.0)
Graph2Gauss	92.0 (0.3)	93.8 (1.0)	88.0 (0.1)	92.1 (0.5)	88.3 (0.0)	88.2 (0.0)
STNE	83.1 (0.5)	90.0 (1.0)	45.6 (0.0)	53.4 (0.1)	88.4 (0.0)	88.4 (0.0)
GELD	**95.3 (0.1)**	**95.8 (0.1)**	**92.6 (0.2)**	**94.7 (0.3)**	88.3 (0.0)	88.3 (0.0)

Tableau 3.2 – Comparaison de l'AUC moyenne sur une tâche de prédiction de liens pour différents pourcentages d'arêtes cachées. Nous supprimons aléatoirement les arêtes et répétons cette procédure 3 fois. Nous indiquons l'écart-type entre parenthèses. GELD surpasse les méthodes les plus récentes, jusqu'à 40 points pour STNE sur Dblp. Sur Nyt, il est comparable à TADW qui obtient la meilleure performance.

(Mikolov et al., 2013b) implémenté dans gensim [3]. Nous utilisons des fenêtres de taille 15 pour Cora, 10 pour Nyt, 5 pour DBLP (selon la taille des documents), et 5 exemples négatifs pour les trois. L'apprentissage des représentations des mots ne prend que 46 secondes sur Cora, 84 sur DBLP et 42 sur Nyt.

Nous avons également testé l'utilisation de vecteurs contextualisés avec BERT. Pour ce faire, nous avons affiné un modèle bert_base_uncased sur la tâche de Masked Language Model (cf sous-section 2.4.5) sur chacun de nos jeux de données. Le pas d'apprentissage initial est de 0.00005. Les performances en classification augmentent et atteignent un plateau au bout de 60 itérations.

3.5 Résultats

Sur l'hypothèse générative Notre modèle repose sur l'hypothèse que les représentations des mots d'un document suivent une loi Normale. Cela a de nombreux avantages, notamment calculatoires : le processus d'optimisation est facilité par l'existence de solutions optimales explicites et le nombre de paramètres n'est pas très élevé (nous apprenons une variance diagonale, donc $2 \times n \times r$ paramètres au total). Néanmoins, de nombreuses approches de représentation vectorielle des mots ne reposent pas sur une modélisation explicitement gaussienne des vecteurs, même si certains travaux intègrent des a priori et a posteriori gaussiens dans le cadre d'apprentissage de représentations probabilistes (cf. section 2.4.2).

En particulier, nous utilisons dans nos expérimentations des vecteurs SGNS et BERT. Ces modèles reposent sur une proximité basée sur le produit scalaire entre vecteurs. Nous analysons donc les

3. `https://radimrehurek.com/gensim/`

	Cora
GrapheSAGE_maxpool	78.65 (0.33)
GrapheSAGE_mean	83.08 (0.65)
RLE_w2v	87.7 (0.6)
RLE_bert	88.2 (0.7)
GELD_w2v	88.3 (0.4)
GELD_bert	88.7 (0.7)

Tableau 3.3 – Expérimentations additionnelles : comparaison entre vecteurs SGNS et BERT sur la tâche de classification, avec un ratio de 50%, et résultats pour la méthode GraphSage sur le jeu de données Cora. Les vecteurs BERT semblent donner de meilleurs résultats que les vecteurs obtenus avec SGNS. GraphSAGE donne des résultats moins bon que les compétiteurs testés.

distributions empiriques des mots des documents de nos jeux de données. En figure 3.6 et 3.5 nous présentons les diagrammes quantile-quantile pour plusieurs documents issus du jeu de données cora.

Nous avons conservé les trois documents possédant le plus de mots, et le premier axe de représentation. Les vecteurs sont centrés-réduits. En abscisse, on représente la valeur théorique des quantiles d'une loi normale, en ordonnée les quantiles observés. Dans le cas où les quantiles observés sont en parfaite adéquation avec la distribution théorique, les points bleus s'alignent avec la première bissectrice.

L'ajustement est satisfaisant pour SGNS et ne laisse pas apparaître de motif particulier (pas de courbe, pas de multimodalité, les distributions paraissent symétriques). On observe néanmoins que les queues des distributions sont plus légères qu'une loi Normale. Les mots semblent cependant suivre une loi unimodale et plutôt symétrique. Pour BERT, on observe des courbes qui montrent que les distributions sont biaisées à droite ou à gauche, même si l'alignement reste encore une fois raisonnable.

Notre hypothèse est donc simplificatrice mais pas irréaliste.

3.5.1 Résultats en classification

Nous adoptons des tâches d'évaluation classiques conformément aux travaux existants (Yang et al., 2015; Bojchevski and Günnemann, 2018). Nous effectuons une classification avec un classifieur SVM avec une régularisation L2. La régularisation optimale est fixée par grid-search pour chaque méthode et jeu de données. Nous exécutons les algorithmes 10 fois et reportons le score Micro-F1 et l'écart type dans le tableau 3.1. Le score Micro-F1 est la moyenne harmonique entre la précision (moyennée par classe) et le rappel (moyenné par classe). Nous évaluons également GraphSAGE et comparons l'utilisation de vecteurs BERT et SGNS en tableau 3.3 sur le jeu de données Cora.

À l'exception de VGAE, les méthodes de documents liés sont plus précises que Deepwalk et LSA sur Cora, mais elles ne sont pas plus performantes que la concaténation sur Dblp et Nyt. Il est intéressant de noter qu'avec une régularisation L2 optimale, TADW donne de meilleurs résultats que les méthodes plus récentes sur chaque jeu de données avec un rapport entraînement/test de 50%. Enfin, sur Cora avec un ratio de 50%, GraphSAGE donne de meilleurs résultats que DeepWalk, comme montré dans (Hamilton et al., 2017), mais est dépassé par les autres compétiteurs (excepté VGAE).

	Cora	Dblp	Nyt
TADW	10^{-1}	10^{1}	10^{1}
AANE	10^{-1}	10^{2}	10^{1}
GVNR-t	10^{1}	10^{2}	10^{2}
VGAE	10^{1}	–	10^{2}
G2G	10^{1}	10^{2}	10^{2}
STNE	10^{2}	10^{4}	10^{2}
RLE	10^{1}	10^{1}	10^{1}
GELD	10^{1}	10^{2}	10^{3}

Tableau 3.4 – Temps d'exécution des différentes méthodes de représentation de réseaux de documents sur les trois jeux de données, en secondes (en ordre de grandeur)

Sur la tâche de classification, RLE surpasse les méthodes existantes sur Cora et Dblp, et est la troisième meilleure méthode sur Nyt. Il est intéressant de noter que GVNR-t est performant avec peu d'exemples d'entraînement, tandis que TADW devient second avec 50% d'exemples d'entraînement. Soulignons que RLE s'exécute rapidement (c.f. tableau 3.4), il est même plus rapide que AANE sur Dblp. De plus, il est plus rapide que STNE sur Dblp de quatre ordres de magnitude.

Enfin, la Figure 3.8 montre que les valeurs optimales de λ sont similaires pour les deux ensembles de données. Sa valeur n'est pas si cruciale puisque RLE surpasse les méthodes existantes avec $\lambda \in [0.6, 0.85]$ sur Cora, $\lambda \in [0.15, 0.85]$ sur DBLP, et toutes les méthodes sauf Concatenation pour $\lambda \in [0.45, 0.8]$ sur Nyt .

GELD surpasse tous ses concurrents sur chaque jeu de données (Tableau 3.1), probablement en raison de l'impact de la variance pendant la phase d'apprentissage : en inspectant l'équation 3.36, on peut voir que la valeur optimale donne moins de poids aux documents à forte variance (c'est-à-dire les documents dont le sens est incertain ou trop général).

La figure 3.7 présente les résultats avec un rapport entraînement/test de 50% avec différentes dimensions pour les quatre meilleurs modèles. GELD est plus performant dans chaque dimension. TADW surpasse RLE en dimension 100, mais l'ajout de dimensions semble détériorer les résultats jusqu'à convergence.

Enfin, on peut observer en tableau 3.3 que l'utilisation de vecteurs contextualisés permet d'améliorer les résultats en classification, au prix d'un coût calculatoire élevé (dimension plus grande, coût du fine-tuning et de l'inférence).

3.5.2 Résultats en prédiction de lien

Pour la tâche de prédiction de liens, nous cachons un ensemble aléatoire d'arêtes pour apprendre les représentations. Ensuite, un ensemble aléatoire de paires de documents non connectés est tiré comme exemples négatifs. Nous calculons la similarité cosinus entre les paires de documents dans l'ensemble des arêtes cachées et l'ensemble des exemples négatifs. Nous reportons ensuite l'aire sous la courbe ROC (AUC) dans le tableau 3.2, obtenue sur trois essais, avec différents pourcentages d'arêtes cachées.

RLE surpasse les méthodes existantes sur Cora, tandis que DeepWalk donne de meilleurs résultats

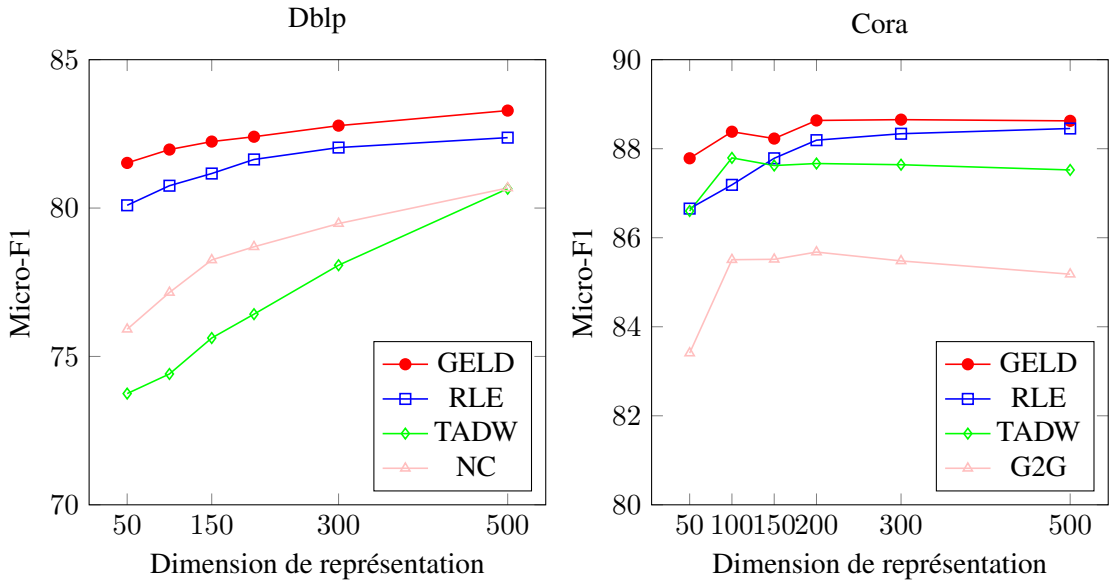


FIGURE 3.7 – Micro-F1 pour un ratio d'entraînement de 50% par rapport à la dimension des représentations pour les quatre meilleures méthodes sur la tâche de classification sur Dblp à gauche et Cora à droite. NC est la méthode Concaténation. GELD obtient les meilleures performances pour chaque dimension testée.

que les autres approches sur Dblp, à l'exception de GELD. Cela peut être dû à la brièveté des documents (la longueur moyenne est de 6 alors qu'elle est de 49 pour Cora) : les informations textuelles peuvent ne pas être aussi informatives que les informations du réseau pour la prédiction de liens.

GELD surpasse les méthodes de référence sur Cora et Dblp, mais est battu par TADW sur Nyt (Tableau 3.2). Néanmoins, chaque méthode, à l'exception de LSA, AANE et GVNR-t, obtient une AUC comprise entre 88,2 et 88,5, pour $\%25$ et $\%50$ d'arêtes cachées. Ceci est dû à la nature du réseau : le degré moyen est d'environ 500, soit 11% du réseau. Même avec 50% d'arêtes cachées, l'information du réseau est bien représentée. De plus, TADW ne parvient pas à produire de bonnes représentations pour la prédiction des liens sur Dblp qui est moins dense alors que les performances de GELD restent constantes pour différentes topologies de réseau.

3.5.3 Aspects qualitatifs - explicabilité des classes

Comme indiqué précédemment, GELD et RLE représentent les mots et les documents dans le même espace. Cette propriété permet d'expliquer le contenu des classes des documents. Nous calculons une représentation vectorielle pour une classe en calculant le barycentre des représentations des documents de cette classe (les vecteurs μ pour GELD). Nous présentons les mots les plus proches de cette représentation en termes de similarité cosinus, ce qui fournit une description générale de la classe. Dans le Tableau 3.5, nous présentons une description utilisant cette méthode pour les quatre premières classes du jeu de données Cora. Nous fournissons également les termes les plus pondérés lors du calcul de la moyenne des documents $tf \cdot idf$ de la classe. La méthode $tf \cdot idf$ produit des

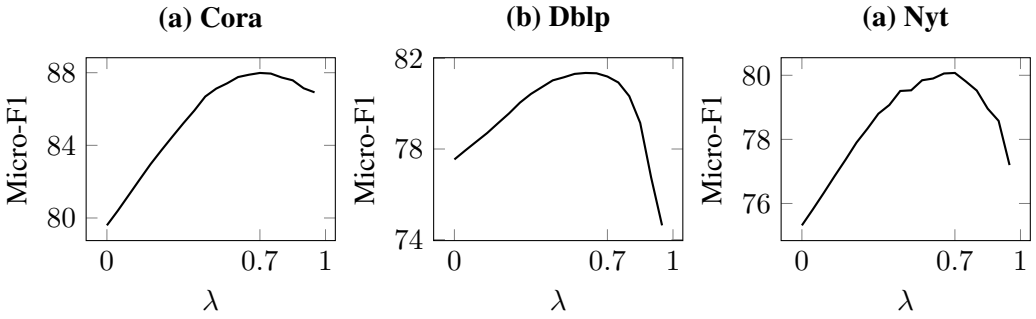


FIGURE 3.8 – Impact du paramètre λ sur les performances du modèle RLE sur la tâche de classification, avec $r = 160$. L'optimum semble se situer autour de 0.7 pour chacun des jeux de données (Cora, Nyt : 0.7, Dblp : 0.65).

mots trop généraux, tels que "apprentissage", "algorithme" et "modèle". Les méthodes RLE et GELD semblent fournir des mots plus spécifiques, ce qui rend les descriptions plus pertinentes.

3.5.4 Étude qualitative de la variance - GELD

Dans le Tableau 3.6, nous présentons les six documents les plus proches de la représentation apprise avec GELD de l'article "Latent Dirichlet Allocation", ainsi que leur variance (plus précisément la somme de la variance de chaque axe). Ces articles sont inclus dans le jeu de données Dblp (titre et articles cités). Les articles "A perceptual hashing algorithm using latent Dirichlet allocation" et "Spatial Latent Dirichlet Allocation" appliquent tous deux la méthode LDA aux images. Par conséquent, ils ont une plus grande variance car ils doivent citer des articles de différents domaines. Au contraire, l'article "Latent Dirichlet Co-clustering" est dans la même classe que LDA, et il est plus probable de citer des documents de cette classe. Il est intéressant de noter que les documents ayant une variance plus faible sont tous dans la même classe que l'"allocation latente de Dirichlet" (classe 3).

3.6 Conclusion et Perspectives

Dans ce chapitre, nous avons présenté les travaux existants permettant de guider l'apprentissage de représentations de documents par la structure du corpus (un graphe de documents). Nous avons ensuite présenté deux contributions. RLE est une méthode rapide et obtenant de bons résultats sur différentes tâches d'évaluation, qui repose sur des simples opérations de multiplications matricielles. GELD est basé sur une modélisation gaussienne des documents. On obtient ainsi, en plus d'un vecteur de représentation, une mesure de variance, ou d'incertitude, pour chaque document. Enfin, nos deux méthodes permettent d'intégrer des représentations vectorielles pré-entraînées des mots. Elles peuvent donc évoluer avec la découverte de nouvelles approches de représentation vectorielle de mots plus efficaces, comme nous l'avons montré avec l'utilisation de représentations vectorielles construites grâce à un modèle BERT pré-entrainé.

Plusieurs axes de recherches sont restés inexplorés :

Class 1			Class 2		
RLE	$tf \cdot idf$	GELD	RLE	$tf \cdot idf$	GELD
hebbian	neural	network	reinforcement	**learning**	reinforcement
network	network	networks	discounted	reinforcement	rl
layers	networks	neural	qlearning	control	barto
multilayer	**learning**	feedforward	rl	state	qlearning
filters	**model**	multilayer	multiagent	policy	multiagent

Class 3			Class 4		
RLE	$tf \cdot idf$	GELD	RLE	$tf \cdot idf$	GELD
posterior	bayesian	posterior	pac	**learning**	pac
gibbs	**model**	bayesian	learnability	**algorithm**	schapire
bayesian	models	gibbs	polynomialtime	algorithms	error
mcmc	**algorithm**	models	dnf	**model**	queries
sampler	belief	model	queries	decision	set

Tableau 3.5 – Description des classes pour nos méthodes et en utilisant $tf \cdot idf$. Les mots qui sont répétés dans toutes les classes sont en gras. RLE et GELD produisent une description plus discriminante du contenu des classes.

Title	Variance	Class
Collective Latent Dirichlet Allocation	545	3
Spatial Latent Dirichlet Allocation	605	1
Distributed Inference for Latent Dirichlet Allocation	590	1
Fast collapsed gibbs sampling for latent dirichlet allocation	513	3
Latent Dirichlet Co-Clustering.	398	3
A perceptual hashing algorithm using latent dirichlet allocation	604	2

Tableau 3.6 – Six plus proches voisins de la représentation de l'article "Latent Dirichlet Allocation" de Blei et al., obtenue en appliquant GELD sur Dblp. On présente également la variance totale (la somme des variances sur chaque axe). Les articles qui utilisent cette méthode, initialement proposée pour le texte, aux images ont une variance plus grande.

— La modélisation des documents par une loi Normale, même si elle n'est pas totalement irréaliste, n'est pas complètement satisfaisante. Il conviendrait de tester d'autres lois de probabilité, même si ce changement se fera au prix d'un coût calculatoire plus élevé. A l'inverse, nous n'avons pas testé l'utilisation de représentations explicitement gaussiennes des mots, comme (Vilnis and McCallum, 2015; Barkan, 2017) qui se prêteraient mieux à ce cadre de modélisation.

— Nous n'avons pas intégré la variance pour résoudre nos différentes tâches d'évaluation. Dans de prochains travaux, on pourrait s'intéresser à l'impact de la variance sur des tâches de recommandation. Dans nos expérimentations préliminaires, une approche de concaténation des deux vecteurs (la moyenne et la variance) ne permet pas d'améliorer les résultats comme c'était le cas dans (Bojchevski and Günnemann, 2018). De même, lorsqu'on calcule une distance

euclidienne plutôt qu'une similarité cosinus, nous n'avons pas observé de meilleurs résultats. Néanmoins, on pourrait s'intéresser au calcul de similarité entre documents qui utilisent des mesures permettant de comparer des distributions, comme la divergence de Kullback Liebler.

— Nous pourrions également développer un modèle Bayésien complet, par l'introduction d'a priori sur les représentations des documents. Par exemple, de façon à ce que les documents écrits par les mêmes auteurs soient proches, on pourrait ajouter un a priori gaussien sur la moyenne dont les paramètres dépendent de l'auteur, mais également un a priori de Wishart inverse (Das et al., 2015) par auteur pour la variance.

Chapitre 4

Apprentissage de représentations d'auteurs

4.1 Introduction

Dans les chapitres précédents, nous avons vu que l'apprentissage de représentations de mots et de documents permettait de résoudre de nombreuses tâches en TAL, telles que la recherche d'information (Kuzi et al., 2016), la génération de texte (Brown et al., 2020) et la classification de documents (Yang et al., 2015). On connaît souvent l'auteur (parfois les auteurs) d'un document.

Cette métadonnée est importante : elle donne une information supplémentaire qui structure le corpus. En effet, les documents écrits par la même personne sont souvent proches. L'information textuelle permet également de caractériser l'auteur lui-même, par rapport aux sujets qu'il ou elle aborde, mais aussi vis-à-vis de son style d'écriture.

Ces informations permettent de résoudre plusieurs tâches : l'identification de l'auteur d'un document non signé, la recommandation de liens entre auteurs, pour des collaborations scientifiques par exemple, ou la recommandation de lecture en fonction de la production d'un auteur (par exemple, Google Scholar). Il est possible de résoudre ces tâches au moyen de méthodes classiques de machine learning. Comme précédemment, il devient nécessaire de représenter mathématiquement les auteurs. De la même façon que les documents et les mots dans les chapitres précédents, nous nous intéressons donc aux approches d'apprentissage de représentations d'auteurs dans un espace vectoriel de dimension réduite.

Très peu de travaux existants s'intéressent à l'apprentissage de représentations d'auteurs. Un grand nombre de méthodes ont été proposées pour représenter les utilisateurs d'outils informatiques ou de médias sociaux. Elles utilisent un ensemble hétérogène de données associées aux utilisateurs, et notamment leur production textuelle. On différencie donc dans ce manuscrit l'apprentissage de représentations d'auteurs, qui utilise seulement le contenu textuel, de ce type d'apprentissage de représentations d'utilisateurs, ou *user embedding*.

Différentes propriétés nous semblent importantes lorsqu'il s'agit d'apprendre des représentations d'auteurs.

Premièrement, il semble nécessaire, dans de nombreux cadres applicatifs, que les documents et les auteurs soient représentés dans le même espace. De cette façon, il est possible d'effectuer des calculs de similarité entre ces objets, pour résoudre des tâches de recommandation ou d'identification d'auteurs (c.f. figure 4.1).

Deuxièmement, la représentation des auteurs ou des documents comme un seul point dans l'espace latent est limitée puisqu'ils peuvent traiter plusieurs sujets. Similairement à GELD, présenté en sous-section 3.3.3, on s'intéresse donc à des représentations probabilistes des auteurs.

Enfin, les modèles existants sont transductifs et ne permettent donc pas de représenter des documents et des auteurs qui n'ont pas été vus pendant la phase d'apprentissage (cf. sous-section 1.2.4). C'est un problème majeur : les documents sont produits de manière continue par les auteurs, notamment dans les réseaux sociaux.

Après avoir présenté les travaux connexes, nous présenterons l'application du cadre VIB, introduit en sous-section 2.3.4, à l'apprentissage de représentation, qui est à la base de notre contribution, puis nous décrirons notre modèle. Dans la dernière section, nous présenterons des résultats expérimentaux sur plusieurs tâches : identification d'auteur, classification d'auteurs et de documents.

Nos expériences démontrent que notre modèle surpasse les méthodes existantes d'apprentissage de représentations d'auteurs, en plus d'être capable d'inférer la représentation de documents qui n'ont pas été vus pendant l'entraînement, et de mesurer l'incertitude sémantique des auteurs et des documents.

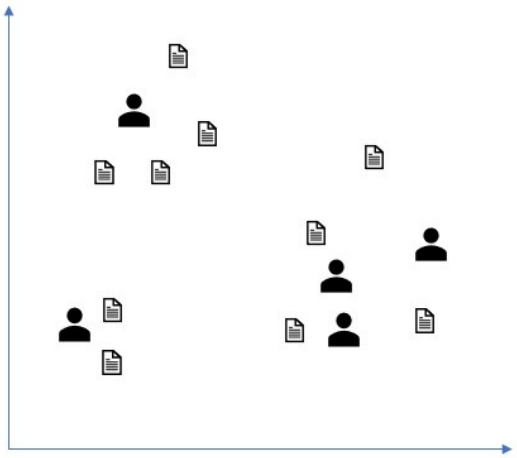


FIGURE 4.1 – Apprentissage de représentations des auteurs et des documents dans le même espace. Sur la base de ces vecteurs, nous pouvons facilement effectuer de nombreuses tâches, telles que l'identification des auteurs, la recommandation, la classification, le partitionnement, etc.

4.2 État de l'art

On introduit la matrice auteurs-documents Ta, de dimension $n_a \times n$ où l'entrée $Ta_{i,j} = 1$ si le i-ème auteur a écrit le j-ème document, 0 sinon. L'apprentissage de représentations d'auteurs consiste à déterminer les paramètres d'une fonction $a_i = f_{map}(a^i, D_o, \theta) \in \mathbb{R}^r$ qui associe l'auteur a^i à un vecteur a_i, où D_o est un ensemble de données de contexte dont la nature dépend de la méthode et θ l'ensemble des paramètres.

Une des premières approches proposant d'associer les auteurs d'un corpus à un vecteur de faible dimension est l'Author Topic Model (ATM) (Rosen-Zvi et al., 2004). Elle étend la méthode de l'Allocation de Dirichlet Latente (LDA) (Blei et al., 2003). Ces modèles probabilistes proposent d'apprendre des vecteurs de thématiques, qui sont des distributions sur les mots du vocabulaire. Ici, chaque auteur est associé à une distribution sur l'ensemble des thématiques. La forme de l'espace de représentation est donc particulière : les auteurs sont représentés sur le simplexe, c'est-à-dire que chaque coordonnée est positive, et que la norme L1 des représentations est de 1. L'hypothèse générative (simplifiée) est la suivante :

— Pour chaque mot d'un document, on tire une thématique selon la distribution sur les thématiques associées à l'auteur du document. Ce paramètre est un vecteur dans $\mathbb{R}^r$, où r est le nombre de thématiques. Ce vecteur suit un a priori de Dirichlet symétrique paramétré par un vecteur α partagé par tous les auteurs.

— Chaque thématique est associée à un vecteur dans $\mathbb{R}^K$, qui suit un a priori de Dirichlet de paramètre β. Une fois la thématique tirée, on tire un mot en fonction de ce vecteur.

La maximisation de la vraisemblance de ce modèle hiérarchique n'a pas de solution algébrique : la vraisemblance est difficile à intégrer. Les auteurs ont donc recours à une approche approximée : l'algorithme de l'échantillonneur de Gibbs (Griffiths and Steyvers, 2004).

Suite aux avancées en apprentissage de représentations de mots et de documents, des modèles récents ont adapté les cadres de modélisation de travaux comme (Le and Mikolov, 2014; Mikolov et al., 2013b). Le modèle Aut2vec (Ganguly et al., 2016) permet d'apprendre des représentations des auteurs et des documents en utilisant une fonction objectif similaire à Word2Vec.

Plus précisément, le modèle "Content Info" qui considère l'information textuelle uniquement, consiste à maximiser la probabilité des couples (i, j) dans le cas où l'auteur i a écrit le document j et la minimiser sinon. Cette probabilité est calculée au moyen d'un réseau de neurones qui mesure la distance entre les représentations vectorielles de l'auteur i (a_i) et du document j (d_j). Un premier vecteur h_c est calculé de façon à prendre en compte deux mesures de distance entre les vecteurs a_i et d_j. Les auteurs motivent empiriquement ce choix, à partir de travaux plus anciens et de leurs expériences. On obtient :

$$\begin{aligned} h_C^{(\times)} &= a_i \odot d_j \\ h_C^{(+)} &= |a_i - d_j| \\ h_C &= tanh(W_C^{\times} h_C^{(\times)} + W_C^{+} h_C^{(+)} + b_C^{(h)}). \end{aligned} \tag{4.1}$$

Les paramètres du réseau sont $W_C^{\times} \in \mathbb{R}^{r_0 \times r}$, $W_C^{+} \in \mathbb{R}^{r_0 \times r}$, $b_C^h \in \mathbb{R}^{r_0}$. Ensuite :

$$p(Ta_{i,j}) = softmax(U_C.h_C + b_C^{(p)}), \tag{4.2}$$

avec $U_C \in \mathbb{R}^{2 \times r_0}, b_C^{(p)} \in \mathbb{R}^2$. La vraisemblance est calculée sur l'ensemble des paires positives et un ensemble d'exemples négatifs tirés aléatoirement de façon à réduire l'impact des 0 de la matrice Ta. Cette approche est complètement transductive, et prend seulement en compte le lien entre un auteur et les documents qu'il a écrits, sans considérer le contenu textuel. Dans l'article, les représentations des documents sont néanmoins initialisées par les représentations obtenues par la méthode Doc2Vec.

Le modèle Usr2vec (Amir et al., 2017) apprend les représentations des auteurs à partir de vecteurs de mots pré-entraînés. Ils utilisent un objectif équivalent à (Le and Mikolov, 2014). Soit u_i la représentation du i-ème mot du vocabulaire et soit a_j la représentation du j-ème auteur. On rappelle qu'on note $\mathcal{D}_k^w$ la séquence des représentations vectorielles des mots du k-ème document. Les auteurs proposent de maximiser

$$\mathcal{L}_{usr2vec} = \sum_{(i,k)|Ta_{i,k}=1} \sum_{j \in \mathcal{D}_k^w} \log p(u_j|a_i). \tag{4.3}$$

Cette fonction nécessite de calculer une vraisemblance, dont le calcul du terme de normalisation est souvent coûteux et n'a même parfois pas de forme close (cf section 2.3). Cette fonction est donc approximée de façon à réduire le coût calculatoire. Pour une paire (i, j), telle que le j-ème mot du vocabulaire a été observé dans un document écrit par le i-ème auteur, un exemple négatif j' est tiré. Similairement à (Mikolov et al., 2013b), la probabilité de tirage dépend de la fréquence empirique observée du mot dans le corpus. Les auteurs proposent ensuite de minimiser, pour chaque triplet (i, j, j') :

$$\mathcal{L}_{usr2vec}(i, j, j') = -max(0, 1 - u_j^{\top} a_i + u_{j'}^{\top} a_i). \tag{4.4}$$

Le modèle est équivalent à PV-DBOW, mais sans fenêtre et utilise des représentations vectorielles des mots pré-entraînées.

Le modèle Author2Vec (Maharjan et al., 2019) reprend également la fonction objectif de Doc2Vec en intégrant un token identifiant l'auteur plutôt que le document. Cependant, au lieu des mots, les auteurs proposent d'utiliser les trigrammes de caractères. Ces trigrammes sont également annotés comme préfixe, suffixe, ou mot entier/trigramme interne. Chaque trigramme correspond donc à trois unités distinctes dans le vocabulaire final.

Les auteurs motivent ce choix par le fait que cette granularité permettrait de capturer des aspects linguistiques et stylistiques de la production des auteurs. Cependant, même si des travaux antérieurs semblent confirmer cette hypothèse, la capacité des représentations apprises à intégrer ces aspects n'a pas été évaluée expérimentalement.

On peut noter qu'aucune méthode présentée ici ne permet : a) d'apprendre des représentations gaussiennes des auteurs et des documents dans un espace de faible dimension, b) de construire des représentations de documents non vus pendant la phase d'entraînement (elles ne sont pas inductives), c) d'intégrer des représentations pré-entraînées (à l'exception toutefois de (Amir et al., 2017)).

Dans la prochaine section, nous présentons le modèle VADE (Variational Author and Document Embedding), basé sur le cadre du Variational Information Bottleneck (VIB) (Alemi et al., 2017), qui étend l'architecture proposée par (Oh et al., 2019). Une première version plus simple de ce modèle a été publiée dans les actes de la conférence EGC 2021 (Gourru et al., 2021).

4.3 Contribution 3 : VADE

4.3.1 Apprentissage de représentations par VIB

Nous traitons un ensemble de documents. Chaque document peut être écrit par un ou plusieurs auteurs (par exemple, dans la littérature scientifique). Chaque document d est associé à une matrice W_d de $n_d \times r$, avec n_d le nombre de mots dans d. La i-ème ligne de cette matrice est la représentation vectorielle du i-ème mot du document. C'est donc la matrice qu'on obtient en concaténant les vecteurs de $\mathcal{D}_d^w$.

Notre objectif est triple : i) nous voulons construire des représentations des auteurs et des documents *dans le même espace* $\mathbb{R}^r$ de telle sorte que leur proximité capture leur similarité sémantique, ii) nous voulons apprendre une mesure d'incertitude sémantique pour chaque objet, et iii) nous voulons que notre modèle exploite des vecteurs de mots pré-entraînés, afin qu'il puisse transférer des connaissances à partir de sources de données plus riches.

Nous construisons une architecture basée sur (Oh et al., 2019). Cette architecture adapte le cadre VIB à l'apprentissage de représentations d'images. Nous proposons une modification du modèle, notamment des lois d'encodage, de façon à pouvoir modéliser les auteurs et les documents. On rappelle que la fonction objectif générale est :

$$-L_{vib} = \mathbb{E}[\log q(y|z)] - \beta KL(p(z|x)||q(z)) \tag{4.5}$$

où $q(y|z)$ est une approximation variationnelle de $p(y|z)$ (que nous appelons la "loi de décodage") et $q(z)$ qui approxime $p(z)$. $\beta \geq 0$ est un hyperparamètre qui contrôle l'équilibre entre les deux sous-objectifs. (Oh et al., 2019) proposent d'utiliser ce cadre pour apprendre des représentations probabilistes d'images, en séparant des exemples positifs ($y = 1$) et négatifs ($y = 0$).

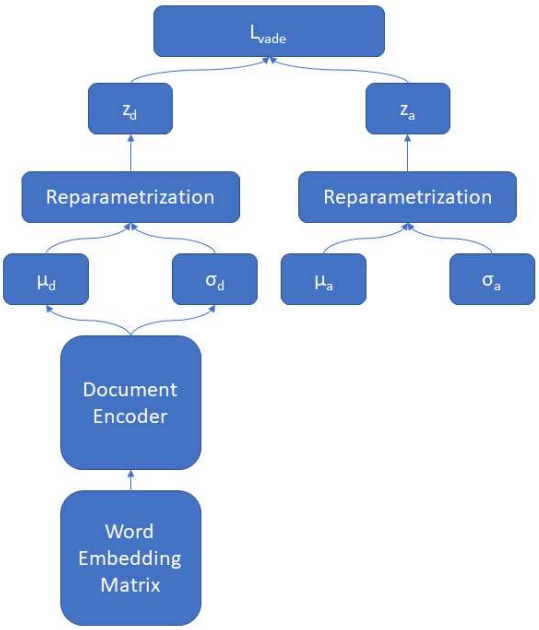


FIGURE 4.2 – Schéma du modèle VADE (Variational Author and Document Embedding). La matrice des représentations vectorielles des mots du document produit une moyenne et une variance. La représentation z_d est tirée en utilisant l'astuce de reparamétrisation de façon à permettre la rétropropagation du gradient. La moyenne et la variance de l'auteur sont des paramètres appris (couches de représentations vectorielles). L_{Vade} calcule la probabilité que la paire auteur/document soit observée, plus un terme de régularisation, voir l'équation (4.7).

Nous proposons d'appliquer ce cadre à l'apprentissage de représentations d'auteurs et des documents. Nous construisons un ensemble de paires (a, d) avec l'étiquette $y = 1$ si a a écrit d. Nous tirons ensuite k paires négatives (a', d) pour chaque paire observée, associées à l'étiquette $y = 0$, où a' n'est pas un auteur de d. Les lois d'encodage des auteurs et des documents sont des lois normales.

Nous apprenons les paramètres suivants pour chaque auteur a : moyenne μ_a et matrice de variance diagonale $\sigma_a^2 I$. Pour un document d, nous construisons sa moyenne $\mu_d = f_{vm}(W_d) \in \mathbb{R}^r$ et sa matrice de variance qui a comme diagonale $\sigma_d^2 = f_{vv}(W_d) \in \mathbb{R}^r$, qui sont fonction de la matrice W_d. Nous présenterons plus loin f_{vm} et f_{vv} (que nous appelons les "fonctions d'encodage").

Suivant (Oh et al., 2019), la probabilité d'une étiquette est :

$$q\left(y = 1 | z_a, z_d\right) = \sigma\left(-c||z_a - z_d||_2 + e\right) \tag{4.6}$$

où σ est la fonction sigmoïde, z_a (resp. z_d) la représentation vectorielle de a (resp. d), $c > 0$ et $e \in \mathbb{R}$ des paramètres libres. Maintenant que nous avons entièrement défini notre modèle, nous pouvons expliciter la fonction de perte basée sur le cadre VIB comme suit :

$$\begin{aligned} L_{vade} = -\mathbb{E}_{p(z_a|x_a),p(z_d|x_d)}[\log q\left(y|z_a, z_d\right)] \\ +\beta KL(p(z_a|x_a)||q(z_a)) + \beta KL(p(z_d|x_d)||q(z_d)). \end{aligned} \tag{4.7}$$

La loi d'encodage des documents ($p(z_d|x_d)$) dépend de paramètres issus de fonctions complexes rendant le premier terme de L_{vade} difficilement intégrable en les paramètres. Nous estimons donc l'espérance par échantillonnage. Pour un nombre de tirages L :

$$\mathbb{E}[\log q\left(y|z_a, z_d\right)] \approx \frac{1}{L}\sum_{l=1}^{L} \log q(y|z_a^{(l)}, z_d^{(l)}). \tag{4.8}$$

En utilisant l'astuce de reparamétrisation (Kingma and Welling, 2014) :

$$z_a^{(l)} = \mu_a + \sigma_a \odot \epsilon,\ \epsilon \sim \mathcal{N}(0,1)$$
$$z_d^{(l)} = \mu_d + \sigma_d \odot \epsilon,\ \epsilon \sim \mathcal{N}(0,1).$$

Cette perte peut être minimisée à l'aide d'un réseau neuronal, de la même manière que les VAE.

4.3.2 Encodeur de documents

Dans la sous-section précédente, nous avons défini deux fonctions, f_{vm} et f_{vv}, qui permettent de déterminer μ_d et σ_d^2 à partir de la matrice des vecteurs mots W_d. Ces fonctions sont des réseaux de neurones. La première est un réseau DAN, et la deuxième utilise un mécanisme d'attention. Ces deux architectures ont été présentées en sous-section 2.4.5 et 2.5.2.

Deep Averaging Cette première approche incorpore un DAN (Iyyer et al., 2015). La moyenne de la matrice W_d est passée dans un réseau neuronal à deux couches avec activation ReLU. Pour f_{vv}, nous utilisons un deuxième réseau neuronal à deux couches avec activation ReLU, qui prend également la moyenne de W_d en entrée. Cette approche obtient les meilleurs résultats sur les tâches d'évaluation. L'intuition est que certaines zones de l'espace latent sont associées à une variance plus élevée, par exemple si le vecteur se situe entre deux clusters sémantiques. Nous avons expérimenté un DAN associé à la variance empirique, mais cette approche était constamment surpassée par la première méthode.

Attention Averaging Nous appliquons une couche d'attention de type *dot product* à W_d. Dans l'esprit des connexions résiduelles utilisées dans les Transformers (cf section 2.4.5), nous additionnons la sortie de la couche d'attention à la matrice originale pour construire une matrice contextualisée W'_d. La moyenne de W_d donne μ_d. f_{vv} est un réseau neuronal à une couche avec activation ReLU prenant μ_d comme entrée. Similairement à l'architecture précédente, cette approche simple et rapide fournit de meilleurs résultats.

Ces deux fonctions d'encodage ont l'avantage d'être rapides et peu profondes. Dans ce qui suit, nous évaluons notre modèle sur deux ensembles de données et utilisons deux ensembles de représentations des mots pré-entrainées avec Glove (Pennington et al., 2014) et BERT (Devlin et al., 2019).

Utilisation de représentations de documents pré-entrainées Nous évaluons également notre approche en utilisant des représentations pré-entrainées des documents. Nous utilisons la représentation pré-entrainée comme moyenne dans notre modèle. VADE nécessite néanmoins qu'une variance soit

Dataset	Auteur(s)	Texte	Catégorie
Nyt	J. Sexton	"Lemieux Scores 4 Against Lackluster Rangers"	sports
Nyt	G. Vecsey	"Sports of The Times ; Antonio Gaudi Is Ready for the 1992..."	sports
Nyt	J. Burns	"On Mandela's Walk, Hope and Violence"	world
Nyt	L. Rohter	"U.S. Charges in Drug Agent's Death : New Friction"	world
S2G	T. Yang	"Adaptive SVRG Methods under Error Bound Conditions..."	neurips
S2G	M. Jordan, G. Hinton	"An Alternative Model for Mixtures of Experts"	neurips

Tableau 4.1 – Exemples de documents des jeux de données NYT et S2G.

associée à chaque document. Nous utilisons donc une fonction pour faire correspondre la moyenne du document à sa variance, de manière similaire aux dernières architectures. Nous utilisons un réseau neuronal feed-forward à deux couches avec activations ReLU et linéaire prenant la représentation du document pré-entrainée (qui devient la moyenne dans notre modèle) comme entrée. Nous évaluons cette architecture avec SBERT (Reimers and Gurevych, 2019) et USE (Cer et al., 2018).

4.4 Évaluation

4.4.1 Jeux de données

Nous utilisons deux jeux de données en anglais, préparés par (Delasalles et al., 2019). S2G est un corpus d'articles sur l'apprentissage automatique provenant de (Ammar et al., 2018). Les articles ont été publiés entre 1985 et 2017. Chaque article est associé à ses auteurs, sa conférence et son année de publication. Le contenu textuel est composé des titres uniquement. Il compte 45 496 documents et 1117 auteurs. NYT est un ensemble d'articles de presse du New York Times écrits entre 1990 et 2015. Il a été initialement rassemblé par (Yao et al., 2018). Il est composé du titre de l'article, de l'auteur et du sujet (par exemple : sport, art, affaires). Il compte 41 249 documents et 542 auteurs. Nous présentons en table 4.1 six exemples de documents de ces jeux de données.

4.4.2 Compétiteurs

Nous comparons notre approche, VADE à plusieurs méthodes existantes récentes.

Nous évaluons les performances des modèles ATM (Rosen-Zvi et al., 2004), la version "Content-Info" d'Aut2Vec (Ganguly et al., 2016) et Usr2Vec (Amir et al., 2017), ainsi qu'une approche naïve de moyenne de vecteurs BERT.

Pour notre modèle, nous utilisons deux méthodes d'apprentissage de représentation de mots : Glove et BERT. Nous évaluons également les deux encodeurs de mots introduits précédemment, que nous appelons VADE$_D$ (pour Deep Averaging) et VADE$_A$ (pour Attention Averaging). Enfin, la dernière version de notre modèle intègre les représentations de documents pré-entrainés. Nous utilisons deux méthodes récentes : SBERT (Reimers and Gurevych, 2019) et USE (Cer et al., 2018).

4.4.3 Tâches d'évaluation

Identification d'auteurs Nous utilisons les documents de la dernière année de la période de production de chaque auteur comme ensemble de test, et les documents précédents comme ensemble

d'entraînement. Comme S2G contient des documents écrits par plusieurs auteurs, nous utilisons l'erreur de couverture (CE) et le score de précision moyen du classement des étiquettes (LRAP). Nous proposons d'utiliser ces mesures, qui sont bien adaptées aux tâches d'évaluation *multi-label* (pour lesquelles on a plusieurs étiquettes par observation). L'erreur de couverture CE mesure le nombre moyen de voisins les plus proches à prendre en compte pour couvrir tous les vrais auteurs d'un document donné. Nous normalisons l'erreur par le nombre d'auteurs (le pire des cas) et nous l'exprimons en pourcentage. Nous obtenons un score entre 0 et 1, $1/n_a$ étant le meilleur score possible pour un corpus à auteur unique. La précision LRAP calcule, pour chaque vrai auteur du document, la proportion de voisins de rang supérieur qui sont également des auteurs du document. Nous calculons les plus proches voisins d'un document en utilisant la similarité cosinus.

Classification des documents et des auteurs Chaque document est associé à une classe, la conférence pour S2G et la catégorie de l'article pour NYT. Nous utilisons une méthode d'évaluation classique en apprentissage de représentations issue de (Yang et al., 2015) : nous entraînons un SVM avec une régularisation L2 avec différents rapports entraînement/test et nous rapportons le score Micro-F1 et son écart-type. La régularisation optimale est déterminée par une recherche sur grille (*grid search*). Nous faisons de même pour évaluer les différentes méthodes en classification d'auteurs. La classe de l'auteur est la classe la plus fréquente observée dans ses productions. Comme le nombre de documents est important, nous sous-échantillonnons 1000 documents 5 fois, et nous rapportons les résultats moyens. Nous n'effectuons pas ce sous-échantillonnage pour les auteurs.

4.4.4 Paramètres

Pour VADE, nous testons plusieurs alternatives.

Premièrement nous implémentons VADE$_D$ et VADE$_A$ avec deux types de représentations de mots : des représentations contextualisées, apprises avec BERT, et des représentations pré-entraînées avec Glove [1]. Pour BERT, nous utilisons le modèle "bert-base-uncased" et l'implémentation de BERT par HuggingFace (Wolf et al., 2019) pour construire les représentations contextualisées des mots.

Deuxièmement, nous évaluons l'encodeur utilisant des représentations pré-entraînées des documents. Nous avons testé les représentations inférées par SBERT (Reimers and Gurevych, 2019), plus précisément le modèle 'bert-base-nli-mean-tokens', et par USE, version 4, disponible sur le Hub de Tensorflow (`https://tfhub.dev/google/universal-sentence-encoder/4`.

Nous déterminons les hyperparamètres optimaux par grid-search sur la tâche d'identification d'auteurs et avec les représentations vectorielles Glove. Nous utilisons Adam, avec un pas d'apprentissage initial de $5e-5$ et tirons 10 exemples négatifs. Nous utilisons $L = 20$ échantillons pour estimer l'équation (4.7). Nous obtenons les meilleures performances avec $\beta = 1e-12$.

Le premier compétiteur est une approche simple, notée $Average - BERT$. Nous construisons les représentations des documents en moyennant les représentations inférées avec BERT des mots qui les composent. La représentation des auteurs est obtenue en moyennant les représentations de documents obtenues.

Pour ATM, nous utilisons Gensim [2]. Nous utilisons le nombre de thématiques qui maximise la valeur de cohérence c_v (Röder et al., 2015), comme fait dans la littérature, c'est-à-dire 201 pour NYT

1. `http://nlp.stanford.edu/data/glove.840B.300d.zip`
2. `https://radimrehurek.com/gensim/`

	S2G		NYT	
	CE	LRAP	CE	LRAP
Average-BERT	42.6	1.59	38,80	4,68
ATM	41.29	2.24	41,93	2,92
VADE-SBERT	29.71	4.33	18.42	13.33
VADE-USE	25.1	5.93	14.73	18.82
VADE$_D$-Glove	**18.33**	**8.57**	14.45	18.44
VADE$_D$-BERT	19.4	7.99	16.15	15.97
VADE$_A$-Glove	22.59	6.65	**13.18**	18.88
VADE$_A$-BERT	20.5	8.14	14.93	13.88

Tableau 4.2 – Erreur de couverture (ou CE, le meilleur modèle est celui avec la valeur la plus faible) et score de précision moyenne de classement des étiquettes (ou LRAP, mesure à maximiser) sur les jeux de données NYT et S2G sur l'ensemble de test (production de la dernière année des auteurs). Notez que nous comparons VADE aux méthodes inductives uniquement. Nous indiquons le meilleur score en gras.

et 229 pour S2G.

Dans (Ganguly et al., 2016), les auteurs initialisent la couche des représentations vectorielles des documents en utilisant la méthode PV-DBOW (Le and Mikolov, 2014). Cette approche fournit de mauvaises performances dans nos expériences. Nous utilisons donc les représentations des documents obtenues en appliquant la méthode Universal Sentence Embedding, qui produisent les meilleurs résultats. De plus, nous avons pu observer que la méthode Aut2Vec est très sensible à l'initialisation de la couche des représentations vectorielles des auteurs. Nous avons d'abord expérimenté avec une initialisation aléatoire, mais nous obtenons de bien meilleurs résultats en initialisant avec la moyenne de représentations des documents écrits par l'auteur. Par exemple, sur NYT (tâche d'identification d'auteur, rapport train/test de 50%), nous obtenons 12% avec PV-DBOW, 16% avec USE et initialisation aléatoire, et 58% en initialisant avec les moyennes de documents. Notez que ce problème d'initialisation n'est pas étudié dans (Ganguly et al., 2016). La dimension de représentation est 512, nous utilisons 256 neurones dans la couche intermédiaire, et nous optimisons les paramètres avec Adam et un taux d'apprentissage initial de $1e-3$. Nous tirons 10 exemples négatifs par paire observée. Ces paramètres permettent d'obtenir les meilleurs résultats sur nos tâches d'évaluation.

Pour Usr2Vec, nous utilisons un taux d'apprentissage de $5e-5$, une marge de 1 et 25 itérations, et une patience de 5 (nombre nécessaire d'itérations sans décroissance de la perte pour arrêter l'entraînement), similairement aux recommandations des auteurs (Amir et al., 2017). Nous utilisons des représentations de mots apprises avec SGNS avec un échantillonnage négatif (Mikolov et al., 2013b) en dimension 300. Nous supprimons également les mots rares (apparaissant moins de 5 fois). Cet élagage du vocabulaire a conduit à une augmentation significative des performances dans nos expériences.

4.5 Résultats

Train/Test ratio	10%	30%	50%
Average-BERT	52.48 (1.96)	60.58 (2.0)	64.24 (2.33)
ATM	30.74 (3.72)	38.58 (1.78)	39.7 (2.56)
Aut2Vec	42.01 (2.31)	53.53 (2.12)	57.38 (2.42)
Usr2Vec	51.56 (3.14)	58.87 (1.51)	60.44 (1.67)
VADE-SBERT	55.84 (4.05)	63.05 (1.63)	66.05 (1.49)
VADE-USE	**60.94 (3.34)**	**68.11** (1.52)	**69.63** (0.72)
VADE_D-Glove	58.89 (2.64)	63.11 (1.31)	65.54 (1.86)
VADE_D-BERT	59.63 (3.32)	65.39 (1.33)	66.64 (1.84)
VADE_A-Glove	56.97 (2.35)	60.13 (1.95)	58.89 (2.57)
VADE_A-BERT	50.23 (2.3)	54.89 (1.57)	55.57 (1.64)

Tableau 4.3 – Résultats en classification d'auteur sur le jeu de données NYT. Nous indiquons le F1-score et l'écart type entre parenthèses. La version de VADE intégrant des vecteurs de documents appris grâce à la méthode USE donne de meilleurs résultats, 9 points meilleurs que la meilleure méthode d'apprentissage de représentations d'auteur.

4.5.1 Comparaison des paramètres de VADE

Il est intéressant de noter que l'encodeur de document Deep Averaging surpasse l'encodeur basé sur l'attention, en plus d'être plus rapide. Dans la plupart des cas, les représentations vectorielles BERT fournissent de meilleures performances que les représentations obtenues avec Glove. La version de VADE utilisant des représentations de documents pré-entraînées avec SBERT obtient de moins bons résultats que les versions basées sur les représentations de mots.

Néanmoins, la version utilisant les vecteurs USE obtient de bons résultats sur plusieurs tâches. USE semble construire des représentations de documents performantes pour construire des représentations d'auteurs. Le modèle obtient de meilleures performances sur les tâches de classification d'auteurs.

4.5.2 Résultats en identification d'auteur

Sur les deux jeux de données, toutes les versions de VADE sont largement plus performantes que les modèles auxquels nous nous sommes comparés. Average-BERT et ATM sont les seuls compétiteurs sur cette tâche : à l'exception de VADE et ATM, aucune méthode de représentation d'auteur ne peut construire la représentation de documents qui n'ont pas été vus pendant l'entraînement.

Il est intéressant de noter que, même dans la meilleure configuration, le rang moyen des vrais auteurs reste élevé, même pour VADE. Cela démontre que l'identification de l'auteur, en utilisant uniquement le titre et des représentations continues, reste une tâche difficile pour les articles de presse et les articles scientifiques.

4.5.3 Résultats pour la classification des auteurs

VADE surpasse constamment les méthodes de représentation d'auteurs sur NYT. VADE-USE dépasse de plus de 10 points la méthode Aut2Vec et de 9 points la méthode Usr2Vec. Average-BERT est plus performante que Usr2Vec et Aut2Vec. Sur ce jeu de données, de meilleures représentations

Train/Test ratio	10%	30%	50%
Average-BERT	52.85 (1.87)	61.19 (0.7)	63.24 (1.07)
ATM	42.64 (2.61)	51.15 (1.22)	53.85 (1.02)
Aut2Vec	50.0 (1.99)	53.96 (1.59)	54.83 (1.62)
Usr2Vec	54.99 (2.67)	64.19 (1.05)	**66.62** (1.27)
VADE-SBERT	57.06 (1.88)	60.03 (0.98)	61.0 (0.93)
VADE-USE	61.65 (1.35)	64.07 (1.43)	63.77 (0.88)
VADE$_D$-Glove	58.45 (1.43)	57.12 (1.2)	56.37 (1.37)
VADE$_D$-BERT	**62.19** (1.67)	**64.92** (1.68)	65.31 (1.87)
VADE$_A$-Glove	57.46 (1.48)	54.83 (1.15)	52.86 (1.1)
VADE$_A$-BERT	60.49 (1.82)	60.54 (0.87)	59.34 (1.17)

Tableau 4.4 – Résultats en classification d'auteur sur le jeu de données S2G. Nous indiquons le F1-score et l'écart type entre parenthèses. La version de VADE avec un encodeur DAN et les vecteurs BERT obtient les meilleurs résultats pour des faibles ratios entraînement/test mais est battue par Usr2Vec pour un ratio de 50%.

semblent donner de meilleurs résultats. Sur S2G, Usr2Vec surpasse les versions de VADE pour un ratio de 50%. Cela peut s'expliquer par le fait que Usr2Vec utilise des vecteurs obtenus par SGNS directement sur le corpus d'entraînement, qui contient un vocabulaire spécifique (lié au machine learning). Néanmoins, VADE$_D$-BERT est plus performant pour des ratios plus faibles.

4.5.4 Résultats pour la classification de documents

Sur le jeu de données NYT, VADE surpasse toutes les méthodes existantes. Il est à noter qu'Average-BERT donne de mauvais résultats. On observe également que le modèle VADE$_D$-BERT obtient un F1-score qui dépasse le meilleur compétiteur de 13 points avec un ratio de 10%. Sur S2G, toutes les méthodes obtiennent des scores très bas. Les classes peuvent être difficiles à séparer dans ce cas : en effet, déterminer la conférence de publication d'un article avec son seul titre est une tâche difficile. Il est intéressant de noter qu'ATM produit de meilleurs résultats sur ce jeu de données, même s'ils obtiennent de mauvais résultats pour toutes les autres tâches.

4.5.5 Visualisation

Sur la Figure 4.3, nous fournissons une visualisation des représentations des auteurs apprises par VADE$_D$-Glove (les moyennes), réduites à 2 dimensions avec T-SNE (Maaten and Hinton, 2008). Nous observons une séparation claire des auteurs par classe (conférence dans laquelle l'auteur publie le plus fréquemment). La proximité des auteurs semble également pertinente : nous voyons que l'espace est bien séparé entre les communautés : Geoffrey Hinton, Yoshua Bengio et Ian Goodfellow sont très proches les uns des autres dans le centre supérieur de l'image.

Il est intéressant de noter que la communauté ACL est légèrement séparée du reste des auteurs en bas à droite. Nous fournissons une visualisation des représentations de Christopher Manning, Ming Zhou et Jianfeng Gao et de trois de leurs articles les plus proches (dans l'espace sémantique) dans la

Train/Test ratio	10%	30%	50%
Average-BERT	19.62 (1.07)	23.79 (1.13)	25.08 (1.72)
ATM	19.53 (0.42)	19.27 (0.78)	18.9 (1.48)
Aut2Vec	36.49 (1.68)	42.23 (1.84)	44.6 (1.11)
VADE-SBERT	28.27 (2.2)	33.73 (1.4)	35.85 (1.51)
VADE-USE	35.24 (1.83)	39.8 (1.68)	41.44 (1.78)
VADE$_D$-Glove	42.23 (1.52)	43.81 (1.86)	42.82 (2.3)
VADE$_D$-BERT	**50.19** (1.3)	**51.56** (1.05)	**50.86** (2.08)
VADE$_A$-Glove	34.76 (1.46)	36.57 (1.05)	36.54 (2.05)
VADE$_A$-BERT	36.6 (1.94)	38.07 (1.73)	38.3 (1.15)

Tableau 4.5 – Résultats en classification de documents sur le jeu de données NYT. Nous indiquons le F1-score et l'écart type entre parenthèses. VADE surpasse toutes les méthodes existantes. Plus précisément, la version utilisant l'encoder DAN avec les représentations vectorielles Glove obtient les meilleurs scores pour chaque ratio entraînement/test.

Figure 4.4. L'article (Gao et al., 2002), écrit par M. Zhou et J. Gao est exactement entre les deux auteurs.

4.5.6 Analyse de la variance

Nous calculons la variance (plus précisément la somme des variances de chaque axe) de trois auteurs célèbres de la communauté ACL. La représentation de Christopher D. Manning possède une variance de 112, Jianfeng Gao de 154 et Ming Zhou de 111. Jianfeng Gao a la variance la plus élevée. Cet auteur s'est intéressé dans sa carrière à des domaines divers tels que l'image et le texte, tandis que Ming Zhou et Christopher D. Manning se sont concentrés sur le traitement automatique des langues.

Cet exemple semble montrer que VADE apprend des variances qui sont corrélées avec la diversité des sujets d'intérêt des auteurs. Cependant, il faudrait consolider ces premières observations avec des expériences additionnelles d'analyse de la variance.

4.6 Conclusion et perspectives

Dans ce chapitre, nous avons présenté les méthodes existantes permettant d'apprendre des représentations d'auteurs. Nous avons ensuite présenté notre contribution, VADE (Variational Author and Document Embedding), une méthode originale qui utilise des représentations vectorielles de mots et de documents pré-entraînés pour construire des représentations d'auteurs comme des distributions gaussiennes.

Notre approche peut construire la représentation d'un document qui n'a pas été vu pendant la phase d'entraînement et surpasse les méthodes existantes en classification d'auteurs et de documents. Les deux meilleurs encodeurs utilisent 1) un DAN (Iyyer et al., 2015) mixé à des représentations BERT, 2) les représentations obtenues avec le modèle USE.

Elle permet de mieux identifier l'auteur d'un document non-signé que les méthodes existantes de représentation d'auteurs, même si la tâche reste difficile à résoudre. En effet, en moyenne, dans

Train/Test ratio	10%	30%	50%
Average-BERT	8.02 (0.78)	8.39 (0.98)	8.64 (0.81)
ATM	**11.92** (3.36)	**13.56** (1.91)	**13.72** (0.79)
Aut2Vec	8.56 (0.86)	8.07 (0.84)	8.2 (1.18)
VADE-SBERT	7.54 (0.69)	7.54 (0.86)	7.57 (1.05)
VADE-USE	8.92 (1.08)	8.69 (0.93)	8.88 (1.2)
VADE_D-Glove	8.37 (0.66)	8.79 (1.38)	9.18 (1.65)
VADE_D-BERT	8.77 (1.09)	8.41 (1.03)	8.96 (1.94)
VADE_A-Glove	7.67 (1.34)	7.29 (0.76)	7.2 (1.09)
VADE_A-BERT	7.68 (0.54)	7.21 (0.88)	8.04 (1.2)

Tableau 4.6 – Résultats en classification de documents sur le jeu de données S2G. Nous indiquons le F1-score et l'écart type entre parenthèses. Aucune méthode ne permet d'obtenir de bons résultats en classification : déterminer la conférence de publication à partir du titre d'un document demeure une tâche difficile.

le meilleur des cas, il faut considérer environ 10% des plus proches voisins d'un document dans l'espace qu'on construit pour couvrir son vrai auteur sur NYT, et environ 20% sur S2G.

Le gain de performance obtenu avec notre méthode par rapport aux méthodes existantes est plus important sur le jeu de données NYT que S2G, le deuxième contenant un vocabulaire et une syntaxe plus spécifique.

Concernant les améliorations de notre modèle, plusieurs pistes sont envisageables :

— Une des limites de notre protocole expérimental est que nous évaluons notre méthode seulement sur des documents courts. Néanmoins, USE et SBERT sont des méthodes pertinentes pour ce format de texte. Nous pourrions évaluer notre méthode sur des documents plus longs en utilisant des architectures comme le Longformer (Beltagy et al., 2020) à la place de nos encodeurs, ou encore le modèle de (Pappagari et al., 2019).

— Nous n'avons également pas évalué notre méthode avec des modèles de représentations affinés sur nos jeux de données. L'intérêt était de montrer que notre modèle était performant, même avec des représentations qui ne sont pas spécifiques au jeu de données traité. De cette façon, les utilisateurs du modèle peuvent faire l'économie de la phase de sur-entraînement, qui peut être coûteuse d'un point de vue calculatoire.

— De plus, nous pourrions également étendre notre méthode afin qu'il puisse traiter l'information de liens entre auteurs ou documents. Pour ce faire, on pourra s'inspirer du modèle complet proposé dans (Ganguly et al., 2016). On pourra également étendre notre modèle GELD, présenté en chapitre 3.

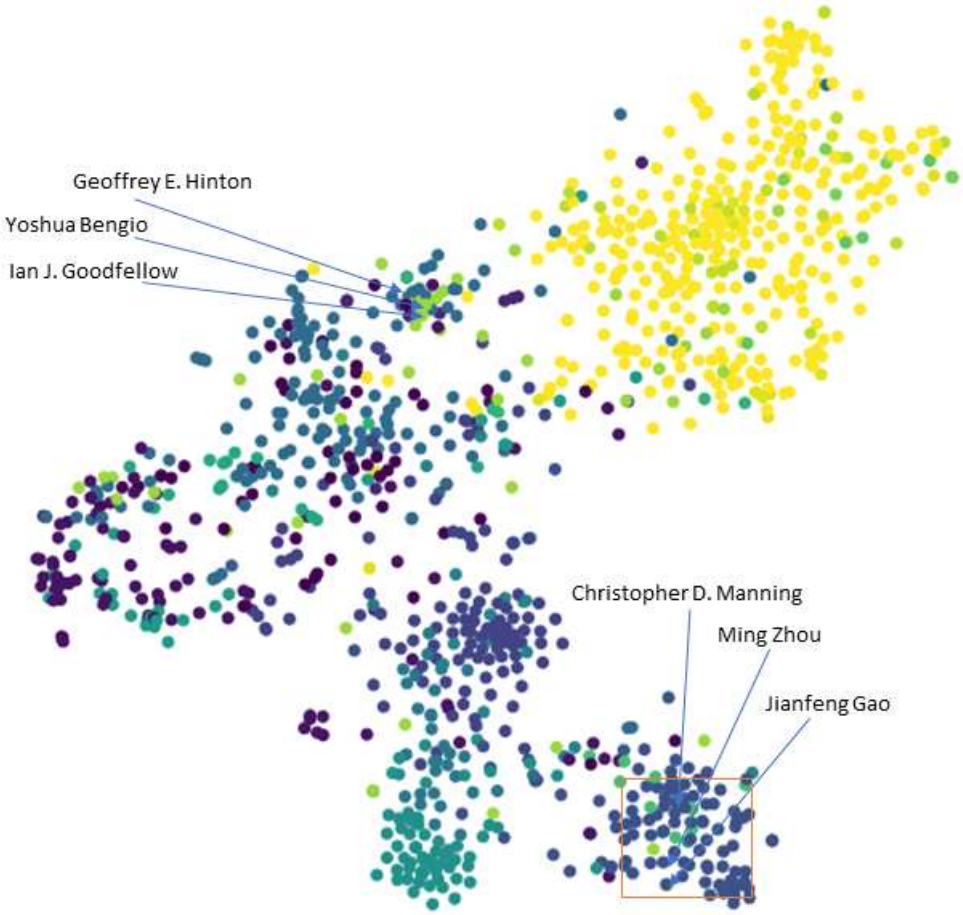


FIGURE 4.3 – Représentations des auteurs obtenues avec notre modèle VADE$_D$-Glove sur le jeu de données S2G. Nous réduisons la dimension à 2 en utilisant la méthode T-SNE (Maaten and Hinton, 2008). Les couleurs correspondent aux classes des auteurs (conférence dans laquelle ils publient le plus souvent). Nous fournissons un zoom de la zone dans le carré orange pour trois auteurs de la communauté ACL en Figure 4.4.

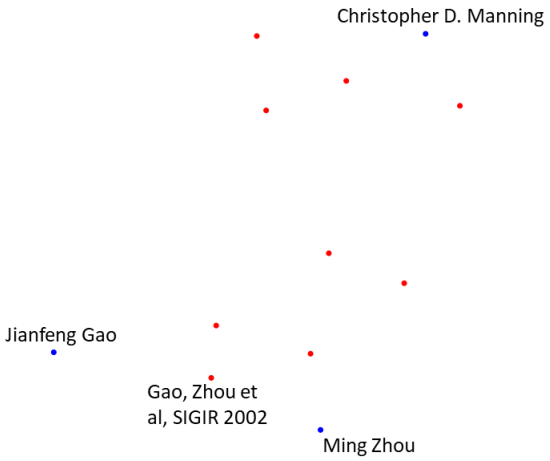


FIGURE 4.4 – Représentations de trois auteurs et de leurs trois plus proches articles, obtenues avec VADE$_D$-Glove. Nous réduisons la dimension à 2 en utilisant la méthode T-SNE (Maaten and Hinton, 2008). L'article (Gao et al., 2002), écrit par J. Gao and M. Zhou, apparaît entre les représentations des deux auteurs.

Chapitre 5

Apprentissage de représentations dynamiques d'auteurs

5.1 Introduction

5.1.1 Contexte

Dans le dernier chapitre, nous nous sommes intéressés à l'apprentissage de représentations de documents d'un corpus et de leurs auteurs. Le modèle VADE permet d'associer chaque auteur et chaque document à une distribution gaussienne dans un espace sémantique de faible dimension. Dans de nombreux cas, on possède l'historique de publications d'un auteur sur une période temporelle importante. Or, leurs sujets d'intérêt et leur style d'écriture peuvent changer au fil du temps.

Par exemple, dans la littérature scientifique, les chercheurs ont tendance à publier dans des domaines précis lorsqu'on considère des fenêtres de temps courtes, bien que leurs intérêts de recherche varient considérablement à l'échelle d'une décennie en raison de l'évolution de leurs domaines scientifiques, ou en raison de nouvelles collaborations.

De façon à illustrer l'effet de cette évolution sur les espaces de représentation, on considère le jeu de données S2G présenté dans le chapitre précédent, composé d'articles scientifiques. Il contient par exemple un article de Geoffrey Hinton écrit en 2017 ("Outrageously Large Neural Networks : The Sparsely-Gated Mixture-of-Experts Layer") et un en 1985 ("Shape Recognition and Illusory Conjunctions"), dont les thématiques sont très éloignées.

On peut évaluer quantitativement cette évolution avec quelques expériences simples pour motiver l'intérêt de l'apprentissage de représentations dynamiques. Nous présentons deux expériences simples dans la sous-section suivante.

5.1.2 Évolution temporelle des représentations

Nous avons d'abord construit les représentations de chacun des documents du jeu de données avec la méthode USE (Cer et al., 2018). Nous avons ensuite calculé, pour chaque auteur, la similarité cosinus entre les représentations des documents qu'il a écrits dans sa dernière année de publication T et les représentations des documents qu'il a écrits dans l'année précédente (T-1), dans celle qui précède cette dernière (T-2) et dans sa toute première année de publication (T1). Nous calculons ensuite la similarité moyenne, pour chaque auteur, puis la moyenne globale sur le corpus.

Nous avons également entraîné un classifieur simple (MLP à deux couches, avec 700 unités, activation ReLU et dropout à 0,1) qui prédit l'auteur du document à partir des représentations USE. Nous utilisons les documents à T comme données d'entraînement, et évaluons les performances à T-1, T-2 et T1 (en termes de micro-F1). Pour comparaison, le micro-F1 sur un ensemble de validation de 10% est d'environ 13% sur les données au temps T. On obtient les résultats présentés en tableau 5.1.

En moyenne, les représentations des documents d'un auteur sont plus proches entre T et T-1 et entre T et T-2 qu'entre T et T1. Nous réalisons un test de Student sur les moyennes par auteur et obtenons que les similarités T vs T-1 sont significativement plus grandes que T vs T1 (p-valeur : 1.6e-24), plus grande à T vs T-2 que T vs T1 (p-valeur : 8.6e-20) mais pas significativement plus grande entre T vs T-1 et T vs T-2 (p-valeur :0.059). Enfin, on observe une chute importante de la performance en prédiction de l'auteur entre T-1/T-2 et T1.

Ces résultats montrent que les représentations sont bien plus proches entre T, T-1 et T-2 qu'entre T et T1. Sur ce jeu de donnée, les représentations des documents produits par les auteurs ont donc visiblement évolué.

	T-1	T-2	T-0
Similarité Cosinus moyenne (T vs ...)	0,233	0,228	0,201
Micro-F1 en prédiction (en%)	4,74	3,13	0,87

Tableau 5.1 – Comparaison des similarités cosinus moyennes entre les représentations des documents des auteurs de S2G T vs T-1, T vs T-2 et T vs T-0. On présente également les résultats d'un classifieur simple pour identifier l'auteur d'un document entraîné sur les représentations des documents des auteurs à T.

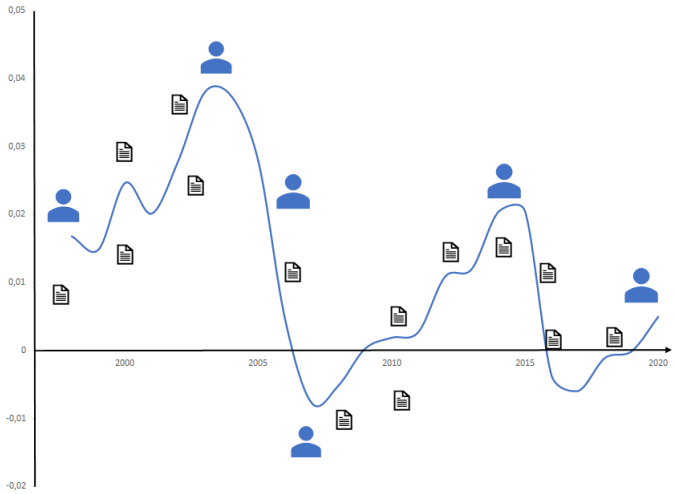


FIGURE 5.1 – Apprentissage de représentations d'auteurs et de documents dans le même espace (ici en 1 seule dimension, l'axe des x est la dimension temporelle). La représentation de l'auteur évolue dans le temps.

5.1.3 Représentations dynamiques d'auteurs

A la lumière de ces observations, nous proposons donc d'apprendre des représentations dynamiques des auteurs, c'est-à-dire que la représentation d'un auteur évolue dans le temps. Ces représentations devraient démontrer des trajectoires temporelles lisses comme le montre la figure (5.1). Similairement aux chapitres précédents, nous proposons d'entraîner un modèle à partir de représentations pré-entraînées, de façon à transférer la connaissance issue de leur entraînement sur des grands corpus de données textuelles. Nous représentons donc les auteurs dans le même espace que les documents.

Très peu de travaux s'intéressent à l'apprentissage de représentations dynamique d'auteurs. On peut citer deux travaux : Sarkar et al. (2006) et Delasalles et al. (2019). Ces modèles n'utilisent que les relations auteurs-mots, et apprennent des vecteurs et non des distributions. Aucun modèle existant ne permet d'apprendre des représentations d'auteurs comme des distributions gaussiennes qui évoluent dans le temps et dans le même espace que les documents.

Dans ce chapitre, nous décrivons d'abord les travaux existants, ainsi que le cadre mathématique des équations de Kalman, utilisé dans une de nos contributions. Nous proposons ensuite un cadre

général original, que nous appelons DGEA (Dynamic Gaussian Embeddings of Authors). Ce cadre est général dans le sens où il peut traiter n'importe quelle représentation de document, et qu'il intègre une modélisation générique de la dépendance temporelle. Plus précisément, nous formulons une hypothèse simple qui exploite les représentations de documents pré-entraînées : les vecteurs des documents dans $\mathbb{R}^r$ sont générés par une distribution gaussienne, dont les paramètres dépendent de l'auteur et du temps.

Nous introduisons ensuite deux types de dépendances temporelles. Les paramètres (moyennes et variances) sont soit 1) lissés, de sorte que les paramètres de tranches de temps consécutives sont proches, en modélisant la dépendance comme un a priori au niveau des paramètres, soit 2) une fonction de tout l'historique de publication, c'est-à-dire que les moyennes et variances au temps T sont conditionnellement indépendantes des valeurs des moyennes et variances des tranches temporelles précédentes. Nous proposons ensuite plusieurs instances de ce cadre général : K-DGEA, basé sur les filtres de Kalman (approche par lissage), et R-DGEA, qui utilise un réseau de neurones récurrent (la représentation de l'auteur au temps T est fonction de tout son historique de publication).

Enfin, nous comparons ces modèles aux travaux précédents. Nous montrons que l'utilisation de l'information dynamique permet d'améliorer l'identification d'auteurs, la prédiction de co-auteurs et la classification d'auteurs sur les jeux de données S2G et NYT présentés dans le chapitre précédent.

Notre modèle a de nombreux avantages par rapport aux travaux existants : 1) le modèle R-DGEA (la deuxième instance de DGEA) peut inférer des représentations pour des auteurs non vus, ce qui évite de ré-entraîner le réseau (notre approche est la seule méthode existante avec cette propriété inductive), 2) les auteurs sont représentés comme des distributions gaussiennes, ils sont donc associés à une mesure de variance permettant de donner une information supplémentaire dans un processus de recommandation par exemple, 3) il incorpore facilement les représentations pré-entraînées de documents, de sorte que notre méthode peut évoluer au gré de la découverte de nouvelles approches d'apprentissage de représentations de documents, et 4) notre modèle obtient de très bonnes performances sur les tâches évaluées.

5.2 État de l'art

Dans cette section, nous présenterons d'abord la modélisation de données séquentielles avec un modèle de Markov simple. Nous détaillerons dans ce cadre les équations de Kalman associées à l'algorithme EM pour optimiser la vraisemblance d'un modèle probabiliste qui contient une dépendance temporelle. Cette modélisation et ce procédé d'optimisation sont utilisés dans notre modèle K-DGEA. Nous présenterons en second temps les travaux existants qui apprennent des représentations dynamiques d'auteurs.

5.2.1 Modèles de Markov et équations de Kalman

Dans cette sous-section, nous nous basons sur le chapitre 13 de Bishop (2006b).

Les données séquentielles sont des observations qui dépassent le cadre i.i.d., c'est-à-dire qu'il existe une dépendance entre les observations. Ces données sont ordonnées, on peut donc les représenter comme des suites d'observations. Nous nous intéressons ici aux données temporelles. Chaque observation est indexée par un nombre réel qui date la donnée.

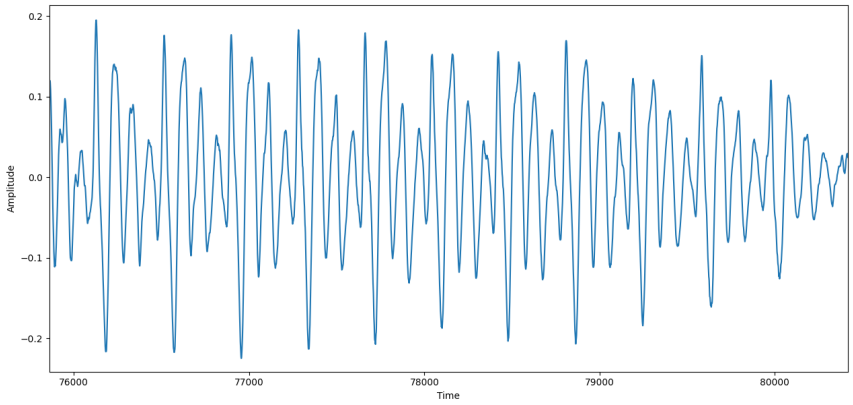


FIGURE 5.2 – Exemple de données séquentielles : extrait d'un enregistrement audio du mot "représentation". En ordonnée l'amplitude, en abscisse le temps. Les données sont dépendantes sur l'axe temporel.

Très souvent, on a recours à une discrétisation du temps, tel que l'indice de l'observation correspond à l'indice d'une tranche temporelle. Nous ne détaillerons pas les procédés de discrétisation, qui font l'objet d'une littérature abondante. Dans ce manuscrit, nous nous concentrerons sur les données discrétisées.

De nombreuses informations du monde réel sont temporelles, comme les signaux ou le langage. En figure 5.2, on montre un extrait d'un enregistrement du mot "représentation" (nous affichons l'amplitude par rapport au temps). Dans ces contextes, la valeur de l'observation au temps t est largement déterminée par ses valeurs précédentes. L'approche classique de modélisation de cette propriété est le modèle de Markov (Bishop, 2006b). Soit $(x_1, ..., x_T) = x_{1:T}$ une séquence de vecteurs, la probabilité jointe devient :

$$p(x_1, ..., x_T) = p(x_1) \prod_{t=2}^{T} p(x_t|x_1, .., x_{t-1}). \tag{5.1}$$

La chaîne de Markov de premier ordre permet de simplifier l'hypothèse, dans le sens où l'observation au temps t dépend seulement de l'observation au temps précédent.

$$p(x_1, ..., x_T) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1}). \tag{5.2}$$

Lorsque chaque observation est déterminée par un état latent non observé, qui évolue lui-même selon un modèle de Markov, on parle de modèle d'état (*state-space model*). Par exemple, dans le modèle DTM de Blei and Lafferty (2006), le contenu textuel publié au temps t dépend d'un ensemble de thématiques latentes qui évoluent au cours du temps. Soit $(z_1, ..., z_T) = z_{1:T}$ la séquence d'états

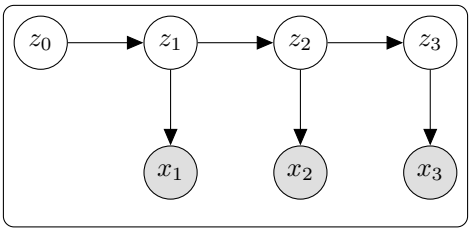


FIGURE 5.3 – Illustration d'un modèle d'état (*state-space model*) pour $T = 3$.

latents, on obtient la probabilité jointe :

$$p(x_1, ..., x_T, z_1, ..., z_T) = p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \prod_{t=1}^{T} p(x_t|z_t) \tag{5.3}$$

et la log-vraisemblance :

$$\mathcal{L}(x_{1:T}) = \log \int p(x_1, ..., x_T, z_1, ..., z_T) d_{z_{1:T}}. \tag{5.4}$$

L'état latent est une variable discrète ou continue. Nous nous intéressons seulement au cas précis où :

— Les variables latentes sont des vecteurs dans $\mathbb{R}^r$

— $p(x_t|z_t)\ p(z_t|z_{t-1})$ sont des lois Normales.

On parle alors de système linéaire dynamique gaussien. Il peut être entièrement déterminé par trois lois, la loi de transition et la loi d'émission, et par la loi de l'état initial, elle aussi gaussienne :

$$\begin{aligned} p\left(z_t|z_{t-1}\right) &= \mathcal{N}\left(z_t|Az_{t-1}, \Gamma\right) \\ p\left(x_t|z_t\right) &= \mathcal{N}\left(x_t|Cz_t, \Sigma\right) \\ p\left(z_1\right) &= \mathcal{N}\left(z_0, \Gamma_0\right). \end{aligned} \tag{5.5}$$

Dans ce cas-là, il est possible d'optimiser la vraisemblance présentée en équation 5.4 en utilisant un algorithme EM (cf. section 2.3). L'ensemble des paramètres est $\theta = [z_0, \Gamma_0, \Gamma, \Sigma, A, C]$.

On rappelle qu'à l'étape E, il faut déterminer $p(z_{1:T}|x_{1:T}; \theta^{old})$ (qu'on note $p(\mathbf{z}|\mathbf{x})$), où θ^{old} est la valeur courante des paramètres. Il est possible de déterminer les paramètres de cette loi grâce à un algorithme somme-produit (Bishop, 2006b), qui nécessite d'alterner une résolution en avant, ou *forward* $(z_t|z_{1,t-1})$ et une résolution en arrière, ou *backward* $(z_t|z_{t+1,T})$.

Dans le cadre du système linéaire dynamique gaussien, on parle du filtre de Kalman pour la récursion forward (Kalman, 1960) et des équations de Rauch-Tung-Striebel (Rauch et al., 1965) pour la récursion backward (ou lissage de Kalman).

Dans l'étape M, on maximise l'espérance (en $p(\mathbf{z}|\mathbf{x})$) de la vraisemblance complétée, c'est-à-dire :

$$\begin{aligned}\mathbb{E}_{p(\mathbf{z}|\mathbf{x})}[\log p(x_{1:T}, z_{1:T}|\theta)] = \mathbb{E}_{p(\mathbf{z}|\mathbf{x})}[\log p(z_1|z_0, \Gamma_0) + \sum_{t=2}^{T} \log p(z_t|z_{t-1}, A, \Gamma) \\ + \sum_{t=1}^{T} \log p(x_t|z_t, C, \Sigma)].\end{aligned} \tag{5.6}$$

On obtient finalement les étapes suivantes à alterner :

Etape E forward Pour chaque tranche temporelle, on calcule successivement :

$$\begin{aligned}\mu_t &= A\mu_{t-1} + K_t(x_t - CA\mu_{t-1}) \\ V_t &= (I - K_t C)P_{t-1}\end{aligned} \tag{5.7}$$

avec

$$\begin{aligned}K_t &= P_{t-1}C^\top \left(CP_{t-1}C^\top + \Sigma\right)^{-1} \\ P_{t-1} &= AV_{t-1}A^\top + \Gamma\end{aligned} \tag{5.8}$$

et

$$\begin{aligned}\mu_1 &= \mu_0 + K_1(\mathrm{x}_1 - \mathrm{C}\mu_0) \\ V_1 &= (\mathrm{I} - K_1\mathrm{C})V_0 \\ K_1 &= V_0C^\top \left(CV_0C^\top + \Sigma\right)^{-1}\end{aligned} \tag{5.9}$$

Etape E backward Une fois que l'étape forward est réalisée, on calcule, de nouveau pour chaque tranche temporelle :

$$\begin{aligned}\widehat{\mu}_t &= \mu_t + J_t(\widehat{\mu}_{t+1} - A\mu_t) \\ \widehat{V}_t &= V_t + J_t\left(\widehat{V}_{t+1} - P_t\right)J_t^\top \\ J_t &= V_tA^\top (P_t)^{-1}\end{aligned} \tag{5.10}$$

Étape M On réalise ensuite la maximisation en les paramètres de

$$\mathbb{E}_{p(\mathbf{z}|\mathbf{x})}[\log p(x_{1:T}, z_{1:T}|\theta)] \tag{5.11}$$

avec :

$$\begin{aligned}\mathbb{E}[z_t] &= \widehat{\mu}_t \\ \mathbb{E}\left[z_t z_{t-1}^\top\right] &= J_{t-1}\widehat{V}_t + \widehat{\mu}_t\widehat{\mu}_{t-1}^\top \\ \mathbb{E}\left[z_t z_t^\top\right] &= \hat{V}_t + \widehat{\mu}_t\widehat{\mu}_t^\top\end{aligned} \tag{5.12}$$

Cet algorithme a été utilisé dans de nombreux modèles, à l'image des modèles thématiques dynamiques (Blei and Lafferty, 2006; Dieng et al., 2019). Il est également utilisé pour optimiser les paramètres de notre contribution, K-DGEA, qu'on présentera dans la section suivante.

5.2.2 Apprentissage de représentations dynamiques d'auteurs

Dans le chapitre précédent, nous avons présenté les approches récentes d'apprentissage de représentations d'auteurs.

L'Author Topic Model (ATM) (Rosen-Zvi et al., 2004), étend le modèle LDA (Blei et al., 2003). Aut2vec (Ganguly et al., 2016), apprend des représentations d'auteurs et de documents au moyen d'un réseau neuronal profond. Nous avons également introduit Usr2vec (Amir et al., 2017) qui utilise des représentations de mots pré-entraînées et une fonction objectif proche de Le and Mikolov (2014).

Très peu de méthodes prennent en compte la structure temporelle des publications. Blei and Lafferty (2006) proposent un modèle graphique pour apprendre des sujets avec une évolution temporelle régulière. Notre cadre de travail partage également des similitudes avec des travaux plus récents sur les représentations dynamiques de thématiques tels que (Dieng et al., 2019). Néanmoins, cette littérature ne modélise pas de représentation au niveau de l'auteur.

Dans (Sarkar et al., 2006) (et son extension (Sarkar et al., 2007)), les auteurs proposent un modèle d'état permettant d'apprendre conjointement des représentations dynamiques d'auteurs et des mots. Le modèle étend l'approche de (Globerson et al., 2007), nommée CODE.

CODE établit un modèle de langue basé sur la distance entre les représentations latentes des auteurs et des mots. Soient $f_{map}(w^i) = \mu_i$ la représentation du i-ème mot et $f_{map}(a^j) = a_j$ la représentation du j-ème auteur, et $p(w^i)$ la probabilité du mot w^i dans le corpus. Le modèle devient :

$$p(w^i|a^j) = \frac{p(w^i)e^{-||\mu_i - a_j||_2^2)}}{\sum_{v=1}^{V} p(w^v)e^{(-||\mu_v - a_j||_2^2)}}. \tag{5.13}$$

Sarkar et al. (2006) proposent d'apprendre des représentations dynamiques pour expliquer les cooccurrences à chaque tranche temporelle. Le temps est discrétisé, et les mots et auteurs sont associés à une représentation par tranche temporelle, qu'on note $w_{i,t}$ et $a_{j,t}$. La probabilité devient :

$$p(w^i, a^j|t) = \frac{p(w^i|t)p(a^i|t)e^{-||\mu_{i,t} - a_{j,t}||_2^2)}}{\sum_{v=1}^{V} p(w^v)e^{(-||\mu_{v,t} - a_{j,t}||_2^2)}}. \tag{5.14}$$

$p(a^j|t)$ est la probabilité empirique d'observer l'auteur a^j au temps t et $p(w^i|t)$ d'observer le mot w_i. Les auteurs proposent ensuite une probabilité de transition simple :

$$\begin{aligned} p\left(a_{j,t}|a_{j,t-1}\right) &= \mathcal{N}\left(a_{j,t-1}, \Gamma\right) \\ p\left(\mu_{i,t}|\mu_{i,t-1}\right) &= \mathcal{N}\left(\mu_{i,t-1}, \Gamma\right). \end{aligned} \tag{5.15}$$

Dans ce modèle, la variance de transition Γ est partagée par tous les objets, et est fixée a priori. Les auteurs effectuent ensuite une série d'approximations de $p(w^i, a^j|t)$ de façon à identifier une borne inférieure gaussienne à la loi d'émission. Ayant explicité un modèle linéaire gaussien, les auteurs proposent ensuite d'utiliser les équations du filtre de Kalman (récursion *forward* uniquement,

présentée plus haut) pour déterminer les valeurs des variables μ et a qui maximisent la vraisemblance de leur modèle probabiliste.

Le modèle DAR (Dynamic Author Representation), proposé par Delasalles et al. (2019), utilise un modèle de langue dynamique paramétré par des représentations dynamiques et statiques des auteurs. Chaque mot est associé à une représentation vectorielle au moyen d'une fonction $f_{map}(w)$.

Similairement à la modélisation du modèle GELD (chapitre 3), chaque document i est donc une séquence de vecteurs, cette fois-ci indexée par a l'indice de l'auteur du document et t l'indice de la tranche temporelle du document :

$$\mathcal{D}^w_{i,a,t} = \left(f_{map}(w^{i,1}, \mathcal{D}^w_{i,a,t}), f_{map}(w^{i,1}, \mathcal{D}^w_{i,a,t}), \ldots\right). \tag{5.16}$$

Pour simplifier les notations, on note $f_{map}(w^{i,k}, \mathcal{D}^w_{i,a,t}) = w_k$. La représentation de l'auteur a au temps t est notée $h_{a,t} \in \mathbb{R}^r$. Les auteurs introduisent également une représentation statique générale de l'auteur, notée $h_a \in \mathbb{R}^r$. La probabilité du document est :

$$p(\mathcal{D}^w_{a,t}) = \prod_{k=0}^{|\mathcal{D}^w_{a,t}|} p(w_{k+1}|w_{0:k}, h_{a,t}, h_a). \tag{5.17}$$

La probabilité $p(w_{k+1}|w_{0:k}, h_{a,t}, h_a)$ est la sortie d'un réseau de type LSTM (cf. section 2.4.4). En plus de la séquence $w_{0:k}$, les auteurs proposent de fournir la séquence $([h_{a,t}, h_a], w_{0:k})$ au LSTM. La représentation au temps t est déterminée au moyen d'un MLP $f_{dar}()$ avec activation ReLU :

$$h_{a,t} = h_{a,t-1} + f_{dar}(h_{a,t-1}, h_a) \tag{5.18}$$

et la représentation au temps 1 est déterminée au moyen d'un deuxième MLP $g_{dar}()$ avec activation ReLU :

$$h_{a,1} = g_{dar}(h_a). \tag{5.19}$$

La vraisemblance du corpus est ensuite maximisée en les paramètres au moyen d'une descente de gradient stochastique par lot (*batch*).

Ces deux approches modélisent les cooccurrences auteur-mot et ne peuvent pas inférer des représentations pour des auteurs non vus. En outre, elles n'apprennent pas une mesure de la variance associée aux auteurs, ni de représentation au niveau du document.

5.3 Modélisation DGAE

5.3.1 Données

Nous considérons un corpus de documents pour lesquels on connaît la date de publication. Chaque document a une représentation fixe dans $\mathbb{R}^r$. Le modèle peut utiliser n'importe quelle approche pour représenter les documents. Dans nos expériences, les représentations des documents des corpus sont inférées en utilisant USE (Cer et al., 2018), InferSent (Conneau et al., 2017) et SBERT (Reimers and Gurevych, 2019). Pour d'autres jeux de données plus spécifiques, d'autres approches sont envisageables. Par exemple, nous pouvons traiter les liens entre les documents (citations) en utilisant

les méthodes d'apprentissage de représentations de documents en réseaux (Yang et al., 2015; Gourru et al., 2020b).

Comme dans la plupart des travaux modélisant des données dynamiques (Rudolph and Blei, 2018; Bamler and Mandt, 2017), nous découpons le corpus en T tranches temporelles indexées par $t \in \{1, \dots, T\}$. Chaque auteur i ($1 \leq i \leq N$) est associé aux ensembles de documents $\mathcal{D}_i^{(t)} = \{d_{i,1}^{(t)}, d_{i,2}^{(t)}, \dots\}$ qu'il a écrits au temps t. Dans nos notations, $d_{i,1}^{(t)}$ est la représentation du premier document écrit par l'auteur i au temps t. Nous introduisons la notation $|\mathcal{D}_i^{(t)}| = n_{i,t}$, le nombre de documents publiés par l'auteur i au temps t. Nous utilisons également la notation suivante pour les séquences de vecteurs : $x_{1:T} = (x_1, x_2, ..., x_{T-1}, x_T)$.

5.3.2 Modèle génératif

Nous faisons l'hypothèse que les vecteurs dans $\mathcal{D}_i^{(t)}$ sont des réalisations de variables indépendantes tirées d'une distribution gaussienne de variance diagonale, ce qui signifie que le j-ème élément de cet ensemble $d_{i,j}^{(t)}$ est tiré d'une loi Normale de variance diagonale : $\mathcal{N}(\phi_{i,t}, \sigma_{i,t}^2 I)$, avec $\phi_{i,t}, \sigma_{i,t}^2 \in \mathbb{R}^r$.

Nous avions déjà utilisé cette hypothèse en chapitre 3 pour le modèle GELD. Nous choisissons d'utiliser une distribution gaussienne pour plusieurs raisons : cela permet d'apprendre une mesure de l'incertitude sémantique, et simplifie le calcul. Une hypothèse forte est que les documents d'une même tranche temporelle sont échangeables. Nous estimons que cette hypothèse est raisonnable si l'on choisit une résolution temporelle pertinente.

Évidemment, la moyenne et la variance ne sont pas indépendantes de l'historique de publication de l'auteur. Nous devons introduire une certaine forme de *dépendance temporelle*. Dans la figure (5.1), nous montrons à quoi devrait ressembler l'espace de représentation : les moyennes des auteurs sont représentées comme des trajectoires lisses dans l'espace de représentation de documents pré-entraîné.

Nous proposons deux approches différentes pour modéliser les dépendances temporelles : la dépendance stochastique et la dépendance fonctionnelle.

5.3.3 Dépendance stochastique : le modèle K-DGEA

La manière la plus simple de modéliser l'évolution temporelle est d'affirmer que les représentations des auteurs évoluent selon un modèle de Markov, c'est-à-dire que les séquences de moyennes et de variances suivent :

$$\begin{aligned} p(\phi_{i,1:T}) &= p(\phi_{i,1}) \prod_{t=2}^{T} p(\phi_{i,t}|\phi_{i,1:t-1}) \\ p(\sigma_{i,1:T}^2) &= p(\sigma_{i,1}^2) \prod_{t=2}^{T} p(\sigma_{i,t}^2|\phi_{i,1:t-1}). \end{aligned} \tag{5.20}$$

Avec $\mathcal{D}$ l'ensemble des observations, i.e. $\{\{\mathcal{D}_i^{(t)}\}_{i=0}^N\}_{t=1}^T$, et $\mathcal{D}_i = \{\mathcal{D}_i^{(t)}\}_{t=1}^T$, on obtient la vraisemblance suivante :

$$\begin{aligned} L(\mathcal{D}) = \prod_i \int \int & p(\sigma_{i,1:T}^2) p(\phi_{i,1:T}) \\ & \times p(\mathcal{D}_i|\phi_{i,1:T}, \sigma_i^2) d_{\phi_{i,1:T}} d_{\sigma_{i,1:T}^2}. \end{aligned} \tag{5.21}$$

Cette vraisemblance est généralement difficile à maximiser en raison de la double intégration et de la dépendance temporelle. Ici, il n'existe pas de forme explicite contrairement à ce qui a été présenté en section 5.2.1. Nous considérons donc un modèle plus simple. Nous utilisons un modèle de Markov du premier ordre sur les moyennes des auteurs. Nous apprenons une variance unique par auteur, c'est-à-dire que la variance ne dépend pas du temps. Cette deuxième hypothèse permet de maximiser efficacement la vraisemblance : nous pouvons la réécrire de façon à faire usage des équations de Kalman (cf. section 5.2.1).

A chaque étape, la moyenne de l'auteur évolue suivant une distribution gaussienne dépendant uniquement du pas de temps précédent, et avec une variance diagonale :

$$\phi_{i,t} \sim \mathcal{N}(\phi_{i,t-1}, \delta_i^2 I). \tag{5.22}$$

La première représentation de l'auteur suit l'a priori

$$\phi_{i,1} \sim \mathcal{N}(\phi_{i,0}, \delta_{i,0}^2 I) \tag{5.23}$$

où $\phi_{i,0}$ et $\delta_{i,0}^2$ sont des paramètres d'état initial à estimer. On obtient :

$$\begin{aligned} L(\mathcal{D}) &= \prod_i \int p(\phi_{i,1:T}) p(\mathcal{D}_i|\phi_{i,1:T}, \sigma_i^2) d_{\phi_{i,1:T}} \\ &= \prod_i \int p(\phi_{i,1:T}) \prod_t \prod_j p(d_{i,j}^{(t)}|\phi_{i,t}, \sigma_i^2) d_{\phi_{i,1:T}} \end{aligned} \tag{5.24}$$

où $\psi = \{\phi_{i,0}, \delta_{i,0}^2, \delta_i^2, \sigma_i^2\}_{1 \leq i \leq N}$. Pour chaque auteur, nous devons identifier les valeurs des paramètres σ_i^2, δ_i^2, $\phi_{i,0}$ et $\delta_{i,0}^2$ qui maximisent $\mathcal{L}(\mathcal{D})$. Comme nous le montrons en annexe 5.8, $p(d_{i,j}^{(t)}|\phi_{i,t}; \sigma_i^2)$ peut être réécrit :

$$\prod_j p(d_{i,j}^{(t)}|\phi_{i,t}; \sigma_i^2) = C_t \zeta(\sigma_i^2, \mathcal{D}_i^{(t)}) \mathcal{N}(\bar{d}_{i,t}; \phi_{i,t}, r_{i,t}^2 I) \tag{5.25}$$

avec C_t une constante positive. $\bar{d}_{i,t}$, $r_{i,t}^2$ et $\zeta(\sigma_i^2, \mathcal{D}_i^{(t)})$ sont définis comme :

$$\bar{d}_{i,t} = \frac{1}{n_{i,t}} \sum_l d_{i,j}^{(t)}, \tag{5.26}$$

$$r_{i,t}^2 = \frac{\sigma_i^2}{n_{i,t}} \tag{5.27}$$

et

$$\zeta(\sigma_i^2, \mathcal{D}_i^{(t)}) = \exp\left(-\frac{n_{i,t}}{2} \sum_r \frac{S_{i,t,r}^2}{\sigma_{i,r}^2}\right) \left(\prod_r \sigma_{i,r}^2\right)^{-\frac{n_{i,t}-1}{2}} \tag{5.28}$$

,avec

$$S_{i,t,r}^2 = \frac{1}{n_{i,t}} \sum_j (d_{i,j,r}^{(t)} - \bar{d}_{i,t,r})^2. \tag{5.29}$$

Dans cette notation, r fait référence à la dimension de représentation du document et de l'auteur. De plus, $\zeta(\sigma_i^2, \mathcal{D}_i^{(t)})$ ne dépend pas de $\phi_{i,t}^2$. La vraisemblance devient :

$$\mathcal{L}(\mathcal{D}) = \prod_i^N \underbrace{\prod_{t=1}^T C_t \zeta(\sigma_i^2, \mathcal{D}_i^{(t)})}_{\text{Facteur de normalisation}} \underbrace{\int p(\phi_{i,1:T}) \prod_{t=1}^T \mathcal{N}(\bar{d}_{i,t}; \phi_{i,t}, r_{i,t}^2 I) d_{\phi_{i,1:T}}}_{\text{Système linéaire dynamique gaussien simple}} \tag{5.30}$$

Nous obtenons un système dynamique plus simple, pour lequel au lieu de plusieurs observations par pas de temps, nous pouvons agréger les observations et ne considérer que la moyenne. A un facteur de normalisation près, notre objectif est maintenant similaire à la maximisation de la vraisemblance du modèle présenté en sous-section 5.2.1 :

$$\begin{aligned} \phi_{i,t} &\sim \mathcal{N}(\phi_{i,t-1}, \delta_i^2 I) \\ \bar{d}_{i,t} &\sim \mathcal{N}(\phi_{i,t}, r_{i,t}^2 I) \end{aligned} \tag{5.31}$$

$\phi_{i,t}$ est l'état latent qui évolue avec le bruit δ_i^2 et $\bar{d}_{i,t}$ la mesure, avec l'erreur $r_{i,t}^2$. On peut donc appliquer les calculs standards de Kalman (Kalman, 1960). Similairement à ce que nous avons présenté en sous-section 5.2.1, nous utilisons un modèle EM avec récursions forward et backward.

Dans l'étape E, nous évaluons la probabilité des états cachés $\phi_{i,t}$ pour $1 \leq i \leq N$ et $1 \leq t \leq T$ en utilisant les équations avant-arrière, et dans l'étape M, nous maximisons l'espérance de la log-vraisemblance complétée. On obtient, pour l'étape E forward :

$$\begin{aligned} m_{i,t} &= m_{i,t-1} + \frac{v_{i,t-1} + \delta_i^2}{v_{i,t-1} + \delta_i^2 + r_{i,t}^2} \left(\bar{d}_{i,t} - m_{i,t-1}\right) \\ v_{i,t} &= \frac{\delta_i^2 r_{i,t}^2 v_{i,t-1}}{v_{i,t-1} + \delta_i^2 + r_{i,t}^2} \end{aligned} \tag{5.32}$$

avec comme condition initiale

$$m_{i,0} = \phi_{i,0} + \frac{\delta_{i,0}^2 \left(\bar{d}_{i,1} - \phi_{i,0}\right)}{\delta_{i,0}^2 + r_{i,1}^2} \tag{5.33}$$

et

$$v_{i,1} = \left(\frac{\delta_{i,0}^2 r_{i,1}^2}{\delta_{i,0}^2 + r_{i,1}^2}\right). \tag{5.34}$$

L'étape E backward donne :

$$\begin{aligned} \widehat{m}_{i,t} &= m_{i,t} + \frac{v_{i,t}}{v_{i,t} + \delta_i^2} \left(\widehat{m}_{i,t+1} - m_{i,t}\right) \\ \widehat{v}_{i,t} &= v_{i,t} + \left(\frac{v_{i,t}}{v_{i,t} + \delta_i^2}\right)^2 \left(\widehat{v}_{i,t+1} - v_{i,t} - \delta_i^2\right) \end{aligned} \tag{5.35}$$

avec $\widehat{m}_{i,T} = m_{i,T}$ et $\widehat{v}_{i,T} = v_{i,T}$.

Dans l'étape M, nous maximisons :

$$\begin{aligned}\mathbb{E}_p[\log \mathcal{L}_i(\mathcal{D}_i, \phi_{i,1:T})] = \sum_{t=1}^{T} \log \zeta(\sigma_i^2, \mathcal{D}_i^{(t)}) + \mathbb{E}_p[\log p(\phi_{i,1}|\phi_{i,0}, \delta_{i,0}^2)] \\ + \mathbb{E}_p[\sum_{t=2}^{t} \log p(\phi_{i,t}|\phi_{i,t-1}, \delta_i^2 I) + \sum_{t=1}^{t} \log p(\bar{d}_{i,t}|\phi_{i,t}, r_{i,t}^2)]\end{aligned} \tag{5.36}$$

par rapport aux paramètres du modèle (c.f. sous-section 5.2.1).

5.3.4 R-DGAE : Deep Filtering

Au lieu de traiter les informations temporelles comme un a priori structurant les séquences de variables latentes, nous supposons que la représentation de l'auteur est une fonction de l'ensemble de l'historique de ses publications passées :

$$\begin{aligned}\phi_{i,t} &= f(\mathcal{D}_i^{(1)}, ..., \mathcal{D}_i^{(t-1)}) \\ \sigma_{i,t} &= h(\mathcal{D}_i^{(1)}, ..., \mathcal{D}_i^{(t-1)}).\end{aligned} \tag{5.37}$$

La vraisemblance devient :

$$L(\mathcal{D}) = \prod_{i,t,j} p(d_{i,j}^{(t)}|f(\mathcal{D}_i^{(1:t-1)}), h(\mathcal{D}_i^{(1:t-1)})). \tag{5.38}$$

Nous proposons d'utiliser des réseaux neuronaux capables de modéliser des données séquentielles, tels que les réseaux neuronaux récurrents présentés en section 2.4.4 (par exemple, LSTM (Hochreiter et al., 1997), GRU (Cho et al., 2014)). Nous proposons ce qui suit :

$$\phi_{i,t} = f_\theta(g(\mathcal{D}_i^{(1)}), ..., g(\mathcal{D}_i^{(t-1)})) \tag{5.39}$$

où f_θ est un réseau neuronal récurrent ou un Transformer et g est une fonction d'agrégation qui projette une matrice de $\mathbb{R}^{n_j \times r}$ dans $\mathbb{R}^r$. Les paramètres θ sont partagés entre les auteurs.

Pour la variance, nous proposons d'utiliser un MLP faisant correspondre la moyenne au vecteur de variance. La fonction est donc $\log \sigma_{i,t}^2 = h_{\theta'}(\phi_{i,t})$. Cette fonction est essentiellement similaire au DAN utilisé dans Cer et al. (2018). Similairement au chapitre précédent, l'intuition est que certaines zones de l'espace sémantique latent seront associées à une variance plus élevée. En effet, l'espace latent reste difficilement interprétable. On peut donc imaginer qu'il existe des zones entre clusters de documents par exemple, ou que certaines portions de l'espace peuvent être associées à des combinaisons de thématiques, et donc produire une incertitude plus élevée. Enfin, entraîner un réseau à fournir une mesure de variance à partir d'une observation ponctuelle rappelle le mécanisme utilisé dans les auto-encodeurs variationnels (Kingma and Welling, 2014) : une observation est passée dans deux MLP qui calculent une moyenne et une variance pour l'observation.

Les paramètres du modèle R-DGEA peuvent être optimisés en utilisant une descente de gradient stochastique par batch.

5.4 Différences entre les modèles

Le modèle K-DGEA est une approche de lissage. Il ne peut pas prédire les représentations des auteurs au temps $t + 1$: la valeur la plus probable de ϕ_{t+1} sera ϕ_t, avec une incertitude déterminée par la variance de transition, δ_i.

D'un autre côté, R-DGEA prédit la représentation au temps $t + 1$ étant donné l'historique des publications jusqu'à t. Il est donc capable de déduire les représentations d'auteurs qui n'ont pas été vus pendant la phase d'apprentissage. Une autre différence importante est que, dans R-DGEA, les paramètres de transition θ sont partagés par tous les auteurs (ce sont les paramètres du RNN).

5.5 Protocole expérimental

5.5.1 Jeux de données

Nous utilisons les deux ensembles de données présentés en chapitre précédent (S2G et NYT), préparés par Delasalles et al. (2019). On rappelle que S2G est un corpus d'articles sur l'apprentissage automatique et NYT un ensemble d'articles de presse du New York Times. Nous n'avons accès qu'à l'année de publication de chaque document. Nous utilisons donc cette information comme intervalle temporel.

Dans nos expériences, seul le titre des documents est disponible. Nous utilisons donc InferSent (Conneau et al., 2017), USE (Cer et al., 2018) et SBERT (Reimers and Gurevych, 2019), qui fonctionnent bien pour les documents courts. Néanmoins, notre cadre peut être appliqué à n'importe quelle représentation de documents, y compris les représentations de documents longs obtenues à l'aide d'architectures telles que le CNN (Collobert et al., 2011) ou le Longformer (Beltagy et al., 2020).

5.5.2 Méthodes évaluées

Nous comparons notre approche, DGEA (Dynamic Gaussian Embedding for Authors) aux méthodes les plus récentes. Nous comparons notre méthode à trois méthodes statiques de représentations d'auteur, ATM (Rosen-Zvi et al., 2004), le modèle "Content-Info" de Aut2Vec (Ganguly et al., 2016) et Usr2Vec (Amir et al., 2017). Enfin, nous comparons notre approche à DAR (Delasalles et al., 2019), qui produit une représentation de faible dimension de l'auteur à chaque pas de temps. Nous utilisons la représentation statique, la représentation dynamique, et une concaténation des deux.

Nous utilisons trois encodeurs de texte distincts de façon à évaluer l'impact des représentations de documents sur les résultats. Nous fournissons les résultats avec SBERT (Reimers and Gurevych, 2019), USE (Cer et al., 2018) et InferSent (Conneau et al., 2017)..

N-DGEA est la configuration la plus simple dans laquelle tous les intervalles de temps sont considérés comme indépendants. Pour un auteur, la solution est donc simplement de calculer les moyennes et variances empiriques des documents à chaque pas de temps. Cette approche naïve n'utilise pas l'information temporelle. Elle permet de démontrer l'avantage de lisser les représentations des auteurs dans le temps.

K-DGEA est le modèle optimisé en utilisant les équations de Kalman décrites dans la sous-section 5.3.3. R-DGEA est le modèle qui utilise un réseau neuronal profond décrit dans la sous-

section 5.3.4.

L'indice _U fait référence aux représentations vectorielles USE, _I à InferSent et _S à SBERT. Nous n'avons pas utilisé la variance dans l'évaluation pour être équitables avec les approches qui apprennent uniquement un vecteur.

5.5.3 Paramètres des méthodes évaluées

Pour SBERT (Reimers and Gurevych, 2019), nous utilisons le modèle 'bert-base-nli-mean-tokens'. Pour InferSent, nous utilisons le modèle pré-entraîné basé sur Glove [1] et les représentations vectorielles Glove pré-entraînées en 300 dimensions [2]. Pour USE, nous utilisons la version 4, disponible sur le Hub Tensorflow [3].

Pour Aut2Vec, Usr2Vec et ATM, nous utilisons les mêmes paramètres que dans le chapitre précédent (en sous-section 4.4.4).

Pour les méthodes dynamiques, nous divisons les documents par année. Pour le modèle DAR (Delasalles et al., 2019), nous utilisons les paramètres recommandés par les auteurs, car ils évaluent leur modèle sur les mêmes jeux de données.

Pour K-DGEA, nous utilisons 100 itérations et notre propre implémentation de l'algorithme EM et de l'algorithme forward-backward. Pour R-DGEA, nous utilisons un LSTM. La fonction d'agrégation est une simple moyenne : cet agrégateur produit les meilleurs résultats sur ces jeux de données. La dimension de représentation est la même que celle des documents, et nous utilisons le dernier état caché comme moyenne ϕ. $h_{\theta'}()$ est un MLP à deux couches avec activation $relu()$ et linéaire. Nous utilisons une taille de batch de 32 auteurs avec Adam (Kingma and Ba, 2015) et un taux d'apprentissage de 1e-4.

5.6 Résultats

Nous choisissons une tâche pour évaluer un des avantages de notre méthode : elle peut inférer des représentations pour des documents qui n'ont pas été vus pendant l'entraînement. Nous utilisons ensuite les tâches d'évaluation classiques pour évaluer l'apprentissage de représentations d'auteurs : la prédiction de liens a été utilisée dans (Ganguly et al., 2016) et la classification dans (Amir et al., 2017).

Nous présentons donc les résultats sur les jeux de donnée NYT et S2G en identification d'auteurs en tableau (5.2), en prédiction de co-auteurs en tableau (5.3) pour S2G, et en classification d'auteurs en tableau (5.4) et en tableau (5.5).

5.6.1 Identification d'auteur

Nous utilisons les documents de la dernière année de production de chaque auteur comme ensemble de test (T+1), et les documents précédents comme ensemble d'entraînement. Comme S2G présente des auteurs multiples, nous utilisons l'erreur de couverture. Cette métrique est bien adaptée à l'évaluation multi-label.

1. https://github.com/facebookresearch/
2. http://nlp.stanford.edu/
3. https://tfhub.dev/google/

	NYT			S2G		
Method	SBERT	InferSent	USE	SBERT	InferSent	USE
Average	19.07	13.70	12.51	26.98	27.83	23.54
N-DGEA	22.72	21.22	17.39	28.93	28.72	26.99
K-DGEA	**17.54**	**12.65**	12.16	**26.35**	**24.71**	**22.74**
R-DGEA	17.83	13.41	**11.09**	26.89	26.57	23.42

Tableau 5.2 – Erreur de couverture (à minimiser). L'erreur de couverture calcule le pire rang (en pourcentage) du voisin le plus proche qui est un véritable auteur du document. ATM, Usr2Vec, Aut2Vec et DAR ne sont pas présentés ici car ils ne modélisent pas les représentations au niveau du document. N-DGEA est le modèle gaussien sans lissage temporel (on calcule simplement la moyenne pour chaque tranche, indépendamment). R-DGEA avec USE est le plus performant, avec 11,09% sur NYT, K_DGEA avec USE sur S2G avec 22,74%. L'ajout de l'information temporelle améliore les performances en identification d'auteur : R-DGEA et K-DGEA sont plus performants que N-DGEA et Average

On rappelle que l'erreur de couverture mesure le nombre moyen de voisins les plus proches qu'il faut prendre en compte pour couvrir tous les vrais auteurs d'un document donné. Nous normalisons l'erreur par n_a, le nombre total d'auteurs dans le corpus (le pire scénario) et nous fournissons ce résultat en pourcentage. Nous pensons que l'erreur de couverture est une mesure intéressante car elle fournit également une quantification du niveau de classement des vrais auteurs par les modèles (contrairement à la perte de Hamming). Nous calculons les plus proches voisins d'un document en utilisant la similarité cosinus. Cette similarité est calculée entre les représentations des auteurs au temps T (pour les approches dynamiques) et celles des documents de l'ensemble de test, c'est-à-dire à $T + 1$.

Comme les méthodes existantes de représentation d'auteurs (ATM, Usr2Vec, Aut2Vec, DAR) ne modélisent pas de représentation au niveau du document, nous comparons notre approche à plusieurs méthodes simples. Nous calculons la moyenne des représentations des documents écrits par un auteur pour construire sa représentation. Cette approche est équivalente à N-DGEA dans la configuration naïve pour laquelle tous les documents sont dans la même tranche temporelle.

K-DGEA et R-DGEA surpassent les moyennes de représentation de documents sans lissage temporel. USE donne de meilleurs résultats sur cette tâche : l'approche semble capable de construire des représentations de documents qui discriminent mieux les auteurs. Le modèle est entraîné sur deux tâches supplémentaires par rapport à SBERT et InferSent : une tâche de question-réponse et une de prédiction des phrases qui cooccurrent. Cette dernière pourrait permettre au modèle de capter une information pertinente pour identifier l'auteur d'un document, comme le style d'écriture.

On observe que les résultats moyens sont moins bons sur S2G : on rappelle que les modèles ne sont pas affinés (*fine-tuned*) sur le jeu de données. Or S2G contient un vocabulaire plus spécialisé que NYT.

Enfin, même dans le meilleur des cas, il faut considérer 10% des plus proches voisins pour couvrir les vrais auteurs d'un document. Sur ces jeux de donnée, la tâche reste donc difficile à résoudre (même dans le cas supervisé, cf. tableau 5.1) : on rappelle que l'on considère le titre du document

ATM		40.78	
Aut2Vec		30.36	
Usr2Vec		23.22	
DAR (static)		**21.49**	
DAR (dynamic)		42.25	
DAR (concat)		33.17	
Method	SBERT	InferSent	USE
N-DGEA	31.39	33.82	30.69
K_DGEA	24.91	26.97	23.96
R-DGEA	22.67	22.89	21.55

Tableau 5.3 – Erreur de couverture (à minimiser) pour la prédiction de liens entre auteurs sur le jeu de données S2G, c'est-à-dire le pourcentage de voisins les plus proches que nous devons conserver pour couvrir tous les co-auteurs du véritable auteur.

.

seulement, pour des jeux de données dans lesquels le langage est formel et stéréotypé : il existe un "style" journalistique et un "style" scientifique qui sont consensuels et partagés par tous les auteurs.

5.6.2 Classification d'auteurs

Similairement au chapitre précédent, la classe d'un auteur est la classe la plus fréquente observée dans ses productions. Par exemple, Geoffrey E. Hinton a publié le plus souvent à la conférence NeurIPS, tout comme Yann LeCun, alors que Christopher D. Manning a principalement publié à ACL, et Jure Leskovec à WWW.

Ici, nous pouvons construire la classe d'un auteur à chaque tranche temporelle. Nous utilisons la dernière année de publication pour l'évaluation. Nous entraînons le classifieur sur les représentations de la dernière tranche temporelle pour DGEA.

Le classifieur est le même que dans le chapitre précédent et que dans les travaux existants (Yang et al., 2015) : un SVM avec une régularisation par norme L2. Nous rapportons la moyenne Micro-F1 et l'écart-type.

La meilleure méthode DGEA est jusqu'à 10 points plus performante que toutes les méthodes de référence sur le jeu de données NYT. Sur le jeu de données S2G, ses performances sont proches de celles de DAR.

On observe également que les performances sont globalement meilleures pour les approches de représentations dynamiques : les jeux de données contiennent une période temporelle de production assez longue (environ 30 ans), et les thématiques d'intérêt des auteurs changent au cours du temps. Cette observation confirme l'intérêt d'apprendre des représentations qui évoluent.

5.6.3 Prédiction de lien

Nous construisons le graphe des co-auteurs sur le jeu de données S2G pendant la dernière année de publication de chaque auteur. Nous calculons dans quelle mesure la similarité entre les représentations des auteurs permet de prédire les liens.

Train/Test ratio	10%	30%	50%	70%
ATM	23.7 (4.48)	31.7 (1.69)	34.4 (1.3)	35.5 (2.5)
Aut2Vec	27.2 (2.57)	32.3 (2.13)	34.5 (1.6)	36.2 (2.21)
Usr2Vec	34.7 (4.04)	41.7 (2.81)	43.8 (2.36)	45.4 (2.41)
DAR (static)	34.9 (3.09)	42.9 (2.23)	43.3 (1.72)	45.21 (2.55)
DAR (dynamic)	18.6 (2.27)	24.9 (2.67)	27.1 (3.03)	29.6 (2.7)
DAR (concat)	34.5 (2.1)	43.6 (1.37)	44.1 (1.48)	47.4 (2.44)
N-DGEA_I	34.0 (2.36)	41.6 (1.11)	44.9 (2.24)	46.7 (2.17)
K-DGEA_I	41.7 (2.97)	51.3 (1.92)	53.1 (2.34)	**56.0** (3.34)
R-DGEA_I	**43.75** (2.84)	50.76 (1.92)	53.17 (1.81)	53.19 (2.53)
N-DGEA_S	29.3 (2.36)	34.8 (2.23)	36.49 (2.59)	37.2 (2.53)
K-DGEA_S	37.2 (3.01)	42.1 (2.56)	44.8 (2.05)	45.3 (3.33)
R-DGEA_S	39.9 (2.25)	46.58 (2.13)	49.52 (2.51)	51.35 (2.66)
N-DGEA_U	34.1 (2.16)	41.3 (2.53)	45.1 (2.38)	46.8 (2.3)
K-DGEA_U	42.3 (2.46)	50.8 (2.71)	**54.2** (2.36)	54.9 (3.41)
R-DGEA_U	42.83 (2.08)	**51.05** (1.17)	52.36 (2.95)	53.87 (4.0)

Tableau 5.4 – Résultats en classification d'auteurs sur le jeu de données NYT. Nous affichons le score Micro-F1 pour différents ratios train/test et fournissons l'écart-type entre parenthèses. Nos approches sont plus performantes que toutes les méthodes existantes.

De la même manière que pour l'identification de l'auteur, nous utilisons l'erreur de couverture qui est définie comme la proportion moyenne de plus proches voisins que nous devons calculer pour couvrir tous les co-auteurs du véritable auteur.

R-DGEA avec USE égale la version statique de DAR (avec une différence de 0,06 points). En outre, chaque modèle R-DGEA surpasse tous les autres modèles de base, y compris la version dynamique de DAR.

Enfin, Usr2Vec et la version statique de DAR produisent de bons résultats : la prise en compte de la dynamique des auteurs ne semble pas si pertinente sur cette tâche. Cette observation pourrait signifier que les collaborations scientifiques sont pérennes dans le temps et que : soit elles restent peu sensibles aux changements de thématiques de recherche, soit elles ne sont pas indépendantes de ces changements, dans le sens où la collaboration peut elle-même modifier la trajectoire thématique des auteurs.

5.6.4 Analyse de la variance des auteurs

La modélisation des auteurs comme des distributions gaussiennes présente de nombreux avantages, tels que ceux soulignés par Bojchevski and Günnemann (2018); Gourru et al. (2020b) : elle permet de capturer l'incertitude de la représentation et peut aider à interpréter les résultats en recommandation. En tant qu'hypothèse de modélisation, elle permet d'utiliser le cadre théorique des modèles linéaires dynamiques gaussiens, et donc l'algorithme EM ainsi que les équations forward-backward.

Nous présentons en tableau 5.6 la corrélation moyenne entre les variances apprises par nos modèles et différentes mesures de variance empirique calculées sur NYT et S2G et les représentations

Train/Test ratio	10%	30%	50%	70%
ATM	31.4 (0.93)	32.4 (1.7)	33.5 (1.62)	33.5 (2.78)
Aut2Vec	23.2 (1.39)	24.0 (1.32)	24.6 (1.94)	24.9 (1.97)
Usr2Vec	34.6 (1.22)	36.9 (1.67)	36.9 (1.4)	36.1 (2.27)
DAR (static only)	36.4 (1.29)	38.8 (1.07)	40.4 (1.08)	40.3 (2.27)
DAR (dynamic only)	31.0 (1.13)	32.4 (0.74)	33.2 (1.48)	32.1 (2.28)
DAR (concatenated)	35.6 (1.48)	38.0 (0.79)	39.8 (1.48)	40.2 (1.7)
N-DGEA_I	30.9 (1.42)	34.5 (1.34)	35.2 (1.34)	35.6 (1.64)
K-DGEA_I	34.5 (2.11)	39.0 (0.85)	40.5 (1.22)	40.7 (1.51)
R-DGEA_I	**37.37** (1.14)	**40.28** (1.64)	**42.75** (1.9)	**42.62** (2.42)
N-DGEA_S	26.1 (1.62)	26.6 (0.83)	27.5 (1.4)	27.6 (1.79)
K-DGEA_S	28.1 (1.75)	30.4 (0.78)	30.9 (1.86)	31.3 (1.91)
R-DGEA_S	29.41 (0.86)	33.03 (0.95)	33.72 (1.53)	35.83 (1.59)
N-DGEA_U	27.2 (2.36)	30.5 (1.12)	31.4 (1.41)	32.5 (1.61)
K-DGEA_U	31.9 (2.14)	36.3 (1.29)	37.2 (1.33)	38.6 (2.56)
R-DGEA_U	36.4 (1.28)	38.85 (1.11)	39.7 (1.57)	40.57 (2.83)

Tableau 5.5 – Résultats en classification d'auteurs sur le jeu de données S2G. Nous affichons le score Micro-F1 pour différents ratios train/test et fournissons l'écart-type entre parenthèses.

	VE	#topics	mean #topic	#topics t+1	mean #docs	#docs
			NYT			
K-variance	**0.86**	0.15	0.18	0.06	0.02	0.09
K-δ	0.09	0.17	-0.12	-0.03	**-0.72**	-0.5
R-variance	0.20	-0.01	0.01	-0.01	-0.04	-0.10
VE	1	0.39	0.24	0.05	-0.16	0.11
			S2G			
K-variance	**0.85**	0.14	0.21	0.18	0.29	0.27
K-δ	0.12	-0.37	**-0.75**	-0.45	**-0.81**	-0.56
R-variance	0.41	-0.13	-0.22	-0.15	-0.13	-0.14
VE	1	0.02	-0.08	-0.04	0.02	0.20

Tableau 5.6 – Corrélation moyenne entre les variances apprises par nos modèles et différentes mesures calculées sur NYT et S2G.

USE. L es mesures sont les suivantes.

La variance empirique (noté VE) est la variance empirique des représentations des documents écrits par l'auteur sur tout son historique de publication. Le nombre de thématiques, noté #topics est le nombre total de labels différents observés sur tout l'historique de l'auteur. Le nombre moyen de labels observés (noté mean #topic) est obtenu en calculant la moyenne du nombre de labels différents par tranche temporelle. Le nombre de labels sur la dernière année de production est noté #topics t+1.

Le nombre moyen de documents écrits par un auteur par tranche temporelle est noté mean #docs et le nombre total de documents écrits par un auteur est noté #docs.

La variance d'un auteur apprise avec K-DGEA montre une forte corrélation avec la variance empirique des représentations de ses documents (0,86 pour K-DGEA$_U$ sur NYT, et 0,85 sur S2G).

Le paramètre δ (la variance temporelle) est négativement corrélé avec le nombre moyen de documents par tranche de temps (-0.72 pour K-DGEA$_U$ sur NYT et -0.81 sur S2G) et avec le nombre moyen de sujets (-0.75 sur S2G) : plus de publications et de sujets réduisent l'incertitude temporelle d'un auteur, c'est-à-dire que les auteurs publiant beaucoup et dans beaucoup de domaines évoluent moins dans le temps.

5.6.5 Comparaison des modèles DGEA

K-DGEA et R-DGEA sont constamment plus performants que N-DGEA : le traitement indépendant de chaque tranche de temps est généralement 10 points moins performant sur nos tâches d'évaluation. Le lissage temporel améliore donc clairement la performance des représentations d'auteurs pour résoudre les tâches évaluées.

R-DGEA est plus performant que K-DGEA en classification sur S2G, et en prédiction de liens. Néanmoins, K-DGEA produit de meilleurs résultats sur le jeu de données S2G sur la tâche d'identification d'auteur et pour NYT en classification d'auteurs. En moyenne, les représentations vectorielles USE semblent produire de meilleurs résultats. Les résultats des deux modèles restent comparables à méthode de représentation de documents similaires. Les performances semblent donc plus déterminées par notre modèle général que par la nature de l'algorithme d'optimisation.

5.6.6 Visualisation

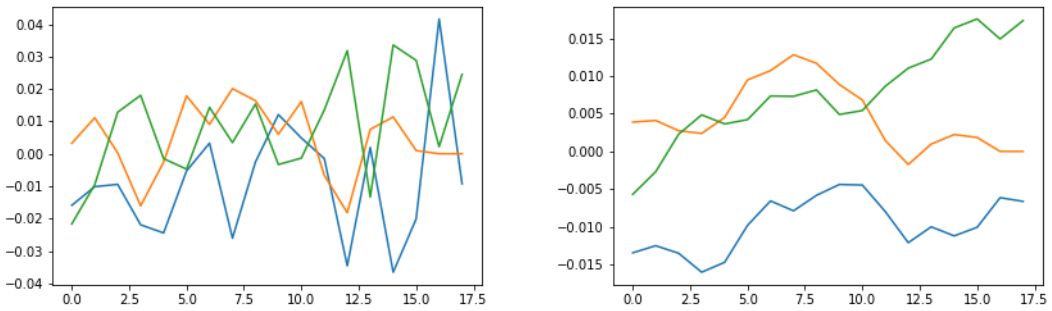


FIGURE 5.4 – Premier axe de représentation pour la moyenne de trois auteurs (codés par couleur) par rapport au temps. Nous fournissons leurs représentations à partir de N-DGEA$_U$ (à gauche ; sans lissage temporel), et K-DGEA$_U$ (à droite ; avec lissage temporel). L'utilisation de l'information temporelle permet de séparer facilement les *trajectoires* d'auteur. Il est difficile de visualiser les représentations dynamiques : les graphiques avec beaucoup d'auteurs sont rapidement illisibles, c'est pourquoi nous avons limité la visualisation à 3 auteurs et une dimension.

Nous fournissons la visualisation de la *trajectoire* de trois auteurs dans la figure (5.4). Nous avons

	NYT			S2G		
Method	SBERT	InferSent	USE	SBERT	InferSent	USE
mean only	17.60	13.18	11.34	26.62	25.68	**22.99**
LSTM variance	32.34	15.16	11.75	47.64	30.97	27.29
single variance	17.62	13.14	11.15	26.94	25.75	23.16
R-DGEA	17.83	13.41	**11.09**	26.89	26.57	23.42

Tableau 5.7 – Erreur de couverture (à minimiser). L'erreur de couverture calcule le pire rang (en pourcentage) du voisin le plus proche qui est un véritable auteur du document.

Method	SBERT	InferSent	USE
mean only	22.55	22.77	22.24
LSTM variance	29.24	28.52	23.13
single variance	22.75	22.52	22.33
R-DGEA	22.67	22.89	**21.55**

Tableau 5.8 – Erreur de couverture (à minimiser) pour la prédiction de liens entre auteurs sur le jeu de données S2G, c'est-à-dire le pourcentage de voisins les plus proches que nous devons conserver pour couvrir tous les co-auteurs du véritable auteur.

sélectionné un auteur aléatoirement, puis avons effectué un tirage aléatoire de deux autres auteurs parmi les auteurs ayant publié autant d'années que le premier. En ordonnée, nous traçons la valeur sur le premier axe de représentation de l'auteur (plus précisément sa moyenne), en abscisse le temps. A droite, nous montrons le résultat K-DGEA_U pour 3 auteurs sur le jeu de données NYT et à gauche, la version N-DGEA_U (sans lissage temporel).

Les trajectoires formées par les représentations dynamiques des auteurs sont faciles à séparer et fortement entrelacées sans lissage temporel, démontrant l'intérêt de notre approche.

5.6.7 Comparaison entre R-DGEA et différentes architectures concurrentes

Le modèle R-DGEA apprend une variance par tranche temporelle au moyen d'un réseau de type DAN (Iyyer et al., 2015). Nous évaluons la pertinence de ce choix de modélisation en évaluant trois alternatives au modèle R-DGEA original. Nous commençons par supprimer la variance et entraînons le réseau en utilisant une simple erreur quadratique moyenne (ce qui équivaut à fixer la diagonale de la variance de chaque auteur à un). Nous appelons ce modèle "mean only". Ensuite, nous utilisons un LSTM dans lequel nous faisons passer les séquences de variances empiriques à la place du DAN. Nous ajoutons en sortie du LSTM un MLP à deux couches, avec activation tanh et relu. Cela revient à utiliser :

$$\sigma^2_{i,t} = h_{\theta'}(g'(\mathcal{D}_i^{(0)}), ..., g'(\mathcal{D}_i^{(t-1)})) \tag{5.40}$$

où $h_{\theta'}$ est un LSTM et $g'(\mathcal{D}_i^{(t)})$ calcule la variance observée de $\mathcal{D}_i^{(t)}$. Nous appelons cette méthode "LSTM variance". Dans la troisième variante, nous apprenons les variances des auteurs comme des

Train/Test ratio	10%	30%	50%	70%
mean only (I)	44.0 (3.47)	51.0 (1.54)	52.88 (1.49)	53.31 (3.23)
LSTM variance (I)	40.12 (3.15)	47.84 (1.61)	48.19 (2.48)	48.71 (1.51)
single variance (I)	43.48 (3.28)	50.16 (1.53)	52.66 (1.7)	53.5 (3.75)
R-DGEA (I)	43.75 (2.84)	50.76 (1.92)	**53.17** (1.81)	53.19 (2.53)
mean only (S)	41.02 (2.1)	46.47 (1.81)	48.08 (2.77)	48.47 (2.86)
LSTM variance (S)	33.73 (2.36)	40.42 (1.52)	42.03 (2.95)	44.36 (2.73)
single variance (S)	39.2 (3.02)	46.74 (1.84)	48.01 (2.52)	50.31 (2.72)
R-DGEA (S)	39.9 (2.25)	46.58 (2.13)	49.52 (2.51)	51.35 (2.66)
mean only (U)	**44.61** (2.66)	49.32 (1.66)	52.03 (2.17)	**54.36** (2.3)
LSTM variance (U)	43.52 (2.86)	49.79 (1.52)	52.25 (1.94)	52.82 (3.6)
single variance (U)	44.28 (1.58)	49.95 (1.72)	51.33 (1.64)	53.19 (1.71)
R-DGEA (U)	42.83 (2.08)	**51.05** (1.17)	52.36 (2.95)	53.87 (4.0)

Tableau 5.9 – Résultats en classification d'auteurs sur le jeu de données NYT. Nous affichons le score Micro-F1 pour différents ratios train/test et fournissons l'écart-type entre parenthèses.

paramètres (c'est-à-dire une couche de représentations vectorielles). Nous appelons cette méthode "single variance".

Nous entraînons le modèle avec une taille de batch de 32 auteurs avec Adam (Kingma and Ba, 2015) et un taux d'apprentissage de 1e-4, à l'exception du modèle "LSTM variance" que nous optimisons avec un taux d'apprentissage de 1e-3. Nous présentons les résultats sur NYT et S2G en identification d'auteur en tableau (5.7), en prédiction de lien en tableau (5.8), et en classification d'auteurs en tableau (5.9) et tableau (5.10).

R-DGEA avec USE est plus performant en identification d'auteurs, tandis que le modèle "mean only" avec USE est plus performant que les autres modèles sur S2G. R-DGEA avec USE produit les meilleurs résultats en prédiction de liens entre auteurs.

Il est intéressant de noter que le modèle "LSTM variance" ne parvient pas à produire de bons résultats en identification d'auteurs avec les représentations SBERT. Le modèle "mean only" produit généralement les meilleurs résultats en classification d'auteurs sur NYT, tandis que R-DGEA surpasse les concurrents sur S2G.

Néanmoins, il n'y a pas de différence significative entre tous les modèles. Comme les différents modèles ont des performances assez équivalentes, nous avons choisi de conserver le modèle R-DGEA présenté précédemment pour différentes propriétés intéressantes lors de l'application du modèle à des situations réelles.

En effet, le modèle original présente plusieurs avantages par rapport aux autres instances : il fournit une mesure de l'incertitude sémantique contrairement à l'apprentissage de la moyenne uniquement, il a moins de paramètres que l'apprentissage de la variance basé sur le LSTM et il permet d'inférer des représentations d'auteurs non vus pendant l'entraînement, contrairement à l'apprentissage d'une seule variance comme paramètre.

Train/Test ratio	10%	30%	50%	70%
mean only (I)	34.59 (1.27)	38.17 (2.14)	39.71 (1.51)	38.96 (1.33)
LSTM variance (I)	31.11 (1.71)	34.54 (1.4)	36.76 (1.66)	37.32 (1.51)
single variance (I)	36.09 (1.23)	36.82 (1.46)	38.5 (1.46)	38.78 (0.92)
R-DGEA (I)	**37.37** (1.14)	**40.28** (1.64)	**42.75** (1.9)	**42.62** (2.42)
mean only (S)	32.64 (1.52)	36.21 (1.18)	36.37 (1.23)	37.8 (2.51)
LSTM variance (S)	26.45 (1.63)	28.89 (1.15)	30.29 (1.65)	30.68 (1.78)
single variance (S)	28.94 (1.53)	30.35 (1.23)	31.79 (1.62)	33.21 (2.59)
R-DGEA (S)	29.41 (0.86)	33.03 (0.95)	33.72 (1.53)	35.83 (1.59)
mean only (U)	36.29 (1.45)	37.94 (1.03)	38.84 (1.51)	39.4 (1.26)
LSTM variance (U)	31.18 (1.96)	33.79 (1.37)	35.08 (2.13)	35.57 (2.1)
single variance (U)	34.72 (0.97)	37.81 (0.62)	38.84 (1.27)	39.32 (1.55)
R-DGEA (U)	36.4 (1.28)	38.85 (1.11)	39.7 (1.57)	40.57 (2.83)

Tableau 5.10 – Résultats en classification d'auteurs sur le jeu de données S2G. Nous affichons le score Micro-F1 pour différents ratios train/test et fournissons l'écart-type entre parenthèses.

5.7 Conclusion

Nous avons proposé le cadre de modélisation DGEA, qui permet d'apprendre des représentations dynamiques d'auteurs à partir de n'importe quelles représentations vectorielles pré-entraînées des documents. Nous avons proposé deux instances de ce cadre : K-DGEA, basé sur un modèle de Markov du premier ordre, et R-DGEA, un modèle profond intégrant une architecture neuronale récurrente.

Les deux modèles obtiennent des performances similaires dans nos expériences. La principale différence est que l'optimisation de K-DGEA est plus rapide et peut être parallélisée, et que R-DGEA peut inférer des représentations pour des auteurs qui n'ont pas été vus pendant l'entraînement. L'utilisateur peut choisir les deux en fonction de ses ressources informatiques et des tâches visées, tout en sachant que les performances seront similaires.

L'information temporelle permet d'améliorer l'identification d'auteurs, la prédiction de liens entre auteurs et la classification des auteurs par rapport aux travaux existants.

Plusieurs pistes et limitations pourraient être étudiées à l'avenir :

— Dans nos expériences, la fonction d'agrégation f_θ la plus performante était une simple moyenne par tranche. Néanmoins, il conviendrait d'identifier une architecture plus complexe qui pondérerait différemment les documents de la tranche en fonction de leur pertinence pour la tâche étudiée. Nous avons expérimenté différentes architectures basées sur des mécanismes simples d'attention, mais aucune n'a donné de meilleurs résultats que l'agrégation moyenne.

— Nous n'avons pas exploré l'ajout de l'information de graphe qu'on peut construire à partir de certains corpus de documents, comme S2G (deux auteurs sont liés s'ils ont co-écrit un document). Cette information pourrait être pertinente, d'autant plus dans un contexte dynamique où les collaborations se font et se défont au fil des années. Pour ce faire, on pourrait reprendre la fonction objectif du modèle GELD présentée en chapitre 3. On rappelle que DGEA étend simplement l'hypothèse générative de GELD aux auteurs et documents. Si l'auteur a^i et a^j ont

co-écrit un article au temps t, on pourrait introduire : $\phi_{i,t} \sim \mathcal{N}(\phi_{j,t}, \sigma^2_{i,t} I)$ similairement à GELD.

— Nous modélisons les auteurs comme des distributions Normales et non comme de simples vecteurs, contrairement aux approches existantes. Néanmoins, même cette hypothèse peut paraître limitée : les auteurs publient parfois dans des thématiques éloignées. Les modéliser comme un mélange de gaussiennes pourrait bénéficier au modèle et réduire le nombre de paramètres. On pourrait par exemple considérer le modèle suivant : soient L thématiques qui évoluent dans le temps. Chaque thématique est une gaussienne similairement à nos travaux, ou à (Das et al., 2015), associées aux paramètres suivants : $\phi_{l,t}$ la moyenne de la thématique l au temps t et $\sigma^2_{l,t} I$ sa variance. On peut introduire la loi de transition suivante : $\phi_{l,t} \sim \mathcal{N}(\phi_{l,t-1}, \Gamma)$. La probabilité d'un document devient : $p(d^{(t)}_{i,j}) = \sum_{l=1}^{L} \pi_{i,l,t} \mathcal{N}(\phi_{l,t}, \sigma^2_{l,t} I)$. Chaque auteur est associé à un vecteur de probabilité sur les thématiques qui évolue lui aussi dans le temps. On peut introduire des a priori supplémentaires permettant de lisser l'évolution de π_i, similairement à (Blei and Lafferty, 2006), ou une simple régularisation similairement à (Rudolph and Blei, 2018).

— Enfin, nous avons formulé une hypothèse forte de modélisation : les représentations des documents écrits par un auteur dans une tranche temporelle sont des variables gaussiennes. Or les modèles testés intègrent des mesures de distances complexes dans leur phase d'optimisation (produit scalaire, distance euclidienne, produit et concaténation de ces mesures). On pourrait tester d'autres lois de probabilité : ce changement serait plus simple pour R-DGEA (dans le cas ou la vraisemblance est bien déterminée) que pour K-DGEA : même si les filtres à particules (cf. (Bishop, 2006b)) permettent d'optimiser des modèles linéaires avec des lois d'émission non gaussiennes, ils nécessitent de réaliser des tirages, et sont plus coûteux en termes de calcul.

— L'information supplémentaire de la variance peut être utilisée pour améliorer des tâches telles que l'identification d'experts, en pénalisant les auteurs ayant une grande variance. Nous n'avons pas évalué ce cadre expérimental, qui nécessiterait des investigations supplémentaires.

5.8 Appendice A : Produit de densités gaussiennes diagonales.

Theorem 1. *Le produit de densités gaussiennes de variance diagonale, sur des observations aléatoires indépendantes, avec une moyenne similaire ϕ et une variance diagonale $\sigma^2 I$, est une densité gaussienne diagonale sur la moyenne empirique des vecteurs, avec une moyenne ϕ et une variance diagonale $\frac{\sigma^2}{n} I$, où n est le nombre de variables aléatoires.*

Pour prouver ce résultat, nous suivons la preuve de (Murphy, 2007). Soient $x = \{x_i\}_{i=1}^n$, $x_i, \phi, \sigma^2 \in \mathbb{R}^r$.

$$\begin{aligned}
\prod_i \mathcal{N}(x_i; \phi, \sigma^2 I) &= \frac{1}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^n} \exp\left(\sum_i -\frac{1}{2} \sum_r \frac{(x_{i,r} - \phi_r)^2}{\sigma_r^2}\right) \\
&= \frac{1}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^n} \exp\left(-\frac{1}{2} \sum_r \frac{1}{\sigma_r^2} \sum_i (x_{i,r}^2 - 2x_{i,r}\phi_r + \phi_r^2)\right) \\
&= \frac{1}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^n} \exp\left(-\frac{1}{2} \sum_r \frac{n}{\sigma_r^2} (s_r^2 + \bar{x}_r^2 - 2\bar{x}_r\phi_r + \phi_r^2)\right)
\end{aligned} \tag{5.41}$$

avec

$$\bar{x} = \frac{1}{n} \sum_i x_i \qquad s_r^2 = \frac{1}{n} \sum_i (x_{i,r} - \bar{x}_r)^2. \tag{5.42}$$

On obtient ensuite

$$\begin{aligned}
\prod_i \mathcal{N}(x_i; \phi, \sigma^2 I) &= \frac{1}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^n} \exp\left(-\frac{1}{2} \sum_r \frac{n s_r^2}{\sigma_r^2}\right) \exp\left(-\frac{1}{2} \sum_r \frac{n}{\sigma_r^2} (\bar{x}_r - \phi_r)^2\right) \\
&= \frac{\exp\left(-\frac{1}{2} \sum_r \frac{n s_r^2}{\sigma_r^2}\right)}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^{n-1}} \cdot \frac{1}{\sqrt{2\pi^r \prod_r \sigma_r^2}} \exp\left(-\frac{1}{2} \sum_r \frac{n}{\sigma_r^2} (\bar{x}_r - \phi_r)^2\right) \\
&= \frac{\exp\left(-\frac{1}{2} \sum_r \frac{n s_r^2}{\sigma_r^2}\right)}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^{n-1} n^{\frac{r}{2}}} \cdot \frac{1}{\sqrt{2\pi^r \prod_r \frac{\sigma_r^2}{n}}} \exp\left(-\frac{1}{2} \sum_r \frac{n}{\sigma_r^2} (\bar{x}_r - \phi_r)^2\right) \\
&= \frac{\exp\left(-\frac{1}{2} \sum_r \frac{n s_r^2}{\sigma_r^2}\right)}{(\sqrt{2\pi^r \prod_r \sigma_r^2})^{n-1} n^{\frac{r}{2}}} \cdot \mathcal{N}(\bar{x}; \phi, \frac{\sigma^2}{n} I).
\end{aligned} \tag{5.43}$$

Le premier terme de cette équation est également gaussien, on peut donc écrire :

$$\prod_i \mathcal{N}(x_i | \phi; \sigma^2 I) \propto \mathcal{N}(s; 0, \frac{\sigma^2}{n} I) \mathcal{N}(\bar{x}; \phi, \frac{\sigma^2}{n} I) \tag{5.44}$$

Chapitre 6

Conclusion

6.1 Conclusion

Dans ce manuscrit, nous avons présenté l'apprentissage statistique, ses fondements et plusieurs cadres de modélisation : les réseaux de neurones et les modèles probabilistes.

Nous avons ensuite décrit un large éventail d'approches d'apprentissage de représentations de mots et de documents. Nous avons présenté les travaux qui apprennent des représentations statiques, contextualisées ou non, des représentations dynamiques, mais aussi des méthodes d'apprentissage de distributions dans un espace sémantique latent. Ces approches ne prennent pas en compte les métadonnées associées aux documents. Or ces informations additionnelles permettent souvent d'améliorer la qualité des représentations des mots et des documents (Yang et al., 2015). Elles ne permettent également pas de représenter les auteurs des documents dans l'espace sémantique.

Dans un second temps, nous nous sommes intéressés à l'apprentissage de représentations de documents liés, c'est-à-dire qu'on peut construire un graphe des documents pour lesquels on veut construire une représentation. Les approches existantes ne peuvent pas : 1) représenter les documents et les mots dans le même espace 2) représenter les documents comme des distributions. Nous avons ainsi proposé deux contributions originales qui permettent d'intégrer des représentations de mots pré-entraînées : la méthode RLE et la méthode GELD. Ces deux méthodes sont plus performantes que les approches existantes sur plusieurs tâches d'évaluation. Le modèle RLE est rapide et ne nécessite de régler qu'un seul paramètre. La méthode GELD permet d'apprendre des densités gaussiennes

pour chaque document plutôt qu'un simple vecteur, de façon à construire un vecteur d'incertitude sémantique.

Ensuite, nous avons exploré l'apprentissage de représentations d'auteurs d'un corpus de documents. Cette problématique a été peu étudiée, et les approches existantes ont de nombreuses limitations : 1) elles ne représentent pas les auteurs et les documents dans le même espace, 2) elles n'apprennent que des représentations vectorielles et 3) elles sont généralement transductives. Nous avons proposé une contribution, VADE, qui permet, grâce au cadre VIB, d'apprendre des représentations des documents et des auteurs dans le même espace. VADE utilise un réseau de neurones qui encode les représentations vectorielles des mots du document et permet donc de représenter un document qui n'a pas été vu pendant l'entraînement. Cette méthode dépasse ou égale les approches existantes en classification de documents, d'auteurs et en identification d'auteur (pour un document dont on ne connaît pas l'auteur).

Nous avons ensuite étudié l'apprentissage de représentations dynamiques d'auteurs. On effet, lorsque la production des auteurs est observée sur un temps long, nous avons montré que leurs thématiques d'intérêt évoluaient (au moins dans l'espace de représentation des documents). Nous avons proposé un nouveau cadre général, DGEA, permettant d'apprendre, pour chaque tranche temporelle, une représentation de l'auteur comme une distribution gaussienne. Nous avons ensuite proposé deux instances de DGEA : K-DGEA, qui utilise un modèle de Markov de premier ordre, et R-DGEA, un modèle profond basé sur un réseau récurrent. Ces modèles sont plus performants que l'existant en classification d'auteurs, en identification d'auteur et obtiennent des résultats équivalents au meilleur modèle existant en prédiction de liens entre auteurs.

6.2 Perspectives

Dans cette section, nous présentons un certain nombre de perspectives à nos contributions.

6.2.1 Modélisation gaussienne

Un premier point a déjà été discuté en fin de chapitres 3, 4 et 5. Notre hypothèse de modélisation des documents et des auteurs comme des distributions gaussiennes implique nécessairement une approximation. Ce choix a néanmoins plusieurs avantages (existence d'une forme explicite, rapidité de l'optimisation, etc...). Dans chacune de nos contributions, il est possible d'étendre le modèle à d'autres distributions, au prix d'une optimisation plus complexe.

Pour GELD, on pourrait étendre la modélisation à tous les membres de la famille exponentielle, tout en conservant un gradient bien défini (Rudolph et al., 2016). Pour VADE, il est possible d'intégrer n'importe quelle loi de probabilité qui est reparametrisable (voir (Kingma and Welling, 2014)).

Pour R-DGEA, nous n'utilisons pas un modèle bayésien complet. Il est donc possible de maximiser le modèle pour de nombreuses lois (tant que la vraisemblance est bien définie). Enfin, pour K-DGEA, l'approche des filtres à particules (Bishop, 2006b) permet d'étendre le modèle linéaire dynamique gaussien à d'autres distributions. Il serait particulièrement intéressant d'explorer les modèles de mélanges ou les approches thématiques dans la continuité de (Das et al., 2015; Dieng et al., 2019).

6.2.2 Supervision

Nos travaux proposent de construire des représentations des documents et des auteurs à partir de fonctions objectifs auto-supervisées. Nous n'utilisons pas d'information additionnelle (comme la classe). Cela permet d'apprendre des représentations polyvalentes.

Néanmoins l'ajout de contraintes ou de sous objectifs supervisés ou semi-supervisés permet souvent d'améliorer les performances des représentations sur des tâches spécifiques (Hamilton et al., 2017; Wang et al., 2016). Il serait intéressant d'évaluer l'impact de cette supervision sur les autres tâches que nous avons évaluées (par exemple l'identification d'auteur ou la prédiction de liens entre documents).

6.2.3 Fairness

Enfin, une perspective intéressante en apprentissage de représentations est l'étude de l'équité des modèles : le modèle de représentation lui-même contient-il des biais, par exemple de genre, tels que présentés dans (Bolukbasi et al., 2016) ? L'utilisation des représentations dans des cadres d'application sensibles peut-elle mener à la sous-représentation de certaines classes de personnes ?

Par exemple, notre modèle VADE permettrait d'apprendre des représentations d'auteurs de lettre de recommandation. Si un recruteur décide d'utiliser les représentations des auteurs pour entraîner un modèle lui permettant de déterminer automatiquement les personnes à recruter, VADE sera-t-il juste ? Cet aspect éthique du machine learning est de plus en plus étudié (Hardt et al., 2016; Davidson, 2021) car central lors de l'application de ces modèles dans le monde réel.

6.2.4 AR de documents long

La majorité des travaux que nous avons présentés en apprentissage de représentations de documents modélisent des documents relativement courts. L'apprentissage de représentations de documents longs pose plusieurs problèmes, notamment pour les approches basées sur l'agrégation de représentations de mots (par exemple (Arora et al., 2016), voir (Gupta et al., 2020)). On a déjà cité des approches qui étendent BERT, comme le Longformer (Beltagy et al., 2020), ou les modèles hiérarchiques (Pappagari et al., 2019).

Nos travaux présentés en chapitres 4 et 5 pourraient permettre de contourner ce problème : on peut découper les documents longs en plusieurs unités plus courtes, et appliquer nos modèles en considérant qu'un document est un auteur et les sous-unités sont ses publications (qui sont dynamiques, car séquentielles). L'apprentissage de représentations de documents longs en utilisant un modèle d'apprentissage de représentations d'auteurs n'a pas encore été étudié à notre connaissance.

6.2.5 Style ou thématiques ?

Nous avons présenté plusieurs travaux de représentation d'auteurs. L'utilisation d'encodeurs qui prennent en compte les séquences de mots dans nos modèles et dans certains compétiteurs pourrait permettre au réseau d'apprendre certaines informations relatives au style d'écriture des auteurs. Dans le contexte de données de littérature par exemple, il est important que deux auteurs qui abordent des thématiques différentes, mais au style proche (par exemple, deux auteurs spécialistes des sonnets) soient proches dans l'espace de représentation.

Nous avons évalué les différentes approches, dont nos contributions, sur des tâches de classification, d'identification et de prédiction de liens entre auteurs. Malheureusement, ces tâches ne permettent pas d'identifier ce que capturent les représentations des auteurs.

Dans un récent papier (Terreau et al., 2021), nous avons proposé un nouveau protocole expérimental pour évaluer les représentations d'auteurs. Il permet de quantifier la capacité des représentations à capturer efficacement un ensemble de caractéristiques stylistiques. Nous montrons que les modèles récents (Ganguly et al., 2016; Cer et al., 2018) capturent principalement le contenu sémantique, c'est-à-dire les thématiques des auteurs.

Ce résultat montre qu'il est nécessaire de modéliser explicitement l'information de style de façon à la capturer lors de la construction des représentations d'auteurs. Cette problématique est encore très peu étudiée et reste une question largement ouverte : comment séparer automatiquement ce qui relève du style et ce qui relève du contenu ?

Bibliographie

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2017. Deep variational information bottleneck. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Silvio Amir, Glen Coppersmith, Paula Carvalho, Mario J Silva, and Byron C Wallace. 2017. Quantifying mental health from social media with neural user embeddings. *arXiv preprint arXiv :1705.00335*.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *International conference on Machine learning*, pages 380–389. PMLR.

Oren Barkan. 2017. Bayesian neural word embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 3135–3143.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer : The long-document transformer. *arXiv :2004.05150*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3 :1137–1155.

CM Bishop. 2006a. Pattern recognition and machine learning (information science and statistics), chapter 11.

CM Bishop. 2006b. Pattern recognition and machine learning (information science and statistics), chapter 13.

CM Bishop. 2006c. Pattern recognition and machine learning (information science and statistics), chapter 9.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.

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

Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep gaussian embedding of graphs : Unsupervised inductive learning via ranking. In *Proceeding of the International Conference on Learning Representations*, ICLR.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker ? debiasing word embeddings. *Advances in neural information processing systems*, 29 :4349–4357.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2010. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1) :1–122.

Arthur Bražinskas, Serhii Havrylov, and Ivan Titov. 2018. Embedding words as distributions with a bayesian skip-gram model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1775–1789.

Robin Brochier, Adrien Guille, and Julien Velcin. 2019. Global vectors for node representations. In *Proceedings of the World Wide Web Conference*, WWW, pages 2587–2593.

Robin Brochier, Adrien Guille, and Julien Velcin. 2020. Inductive document network embedding with topic-word attention. In *European Conference on Information Retrieval*, pages 326–340. Springer.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, pages 169–174.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE) :2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 795–804.

Ian Davidson. 2021. Fairness and explanation in clustering and outlier detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4037–4037.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391–407.

Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer. 2019. Learning dynamic author representations with temporal language models. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 120–129.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2019. The dynamic embedded topic model. *arXiv preprint arXiv :1907.05545*.

Susan Dumais, George Furnas, Thomas Landauer, and Scott Deerwester. 1988. Using latent semantic analysis to improve information retrieval. *Proceedings of CHI'88 Conference on Human Factors in Computing Systems*, pages 281—-285.

Jean Dupuy, Adrien Guille, and Julien Jacques. 2021. Contextual-rtm : un cadre général pour la modélisation de thématiques dans les réseaux de documents. *Extraction et Gestion des Connaissances : Actes EGC'2021*.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Soumyajit Ganguly, Manish Gupta, Vasudeva Varma, Vikram Pudi, et al. 2016. Author2vec : Learning author representations by combining content and link information. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 49–50. International World Wide Web Conferences Steering Committee.

Jianfeng Gao, Ming Zhou, Jian-Yun Nie, Hongzhao He, and Weijun Chen. 2002. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8 :2265–2295.

Antoine Gourru, Adrien Guille, Julien Velcin, and Julien Jacques. 2020a. Document network projection in pretrained word embedding space. In *European Conference on Information Retrieval*, pages 150–157. Springer.

Antoine Gourru, Julien Velcin, and Julien Jacques. 2020b. Gaussian embedding of linked documents from a pretrained semantic space. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*.

Antoine Gourru, Rohit Yadav, and Julien Velcin. 2021. Apprentissage conjoint de représentations d'auteurs et de documents. *Extraction et Gestion des Connaissances : Actes EGC'2021*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1) :5228–5235.

Vivek Gupta, Ankit Saw, Pegah Nokhiz, Praneeth Netrapalli, Piyush Rai, and Partha Talukdar. 2020. P-sif : Document embeddings using partition averaging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7863–7870.

Michael Gutmann and 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*, pages 297–304.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29 :3315–3323.

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

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

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57.

Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *Proceedings of the SIAM International Conference on Data Mining*, SDM, pages 633–641.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1 : Long papers)*, pages 1681–1691.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2) :183–233.

Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam : A method for stochastic optimization. In *ICLR (Poster)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop*, BDL-NeurIPS.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3294–3302.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932. ACM.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. 2020. Flaubert : Unsupervised language model pre-training for french. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2479–2490.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553) :436–444.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries : how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Xiaocan Li, Shuo Wang, and Yinghao Cai. 2019. Tutorial : Complexity analysis of singular value decomposition and its variants. *arXiv e-prints*, pages arXiv–1906.

Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to node : Self-translation network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1794–1802. ACM.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv e-prints*, pages arXiv–1605.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137.

Minh-Thang Luong, Hieu Pham, and 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*, pages 1412–1421.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov) :2579–2605.

Suraj Maharjan, Deepthi Mave, Prasha Shrestha, Manuel Montes-Y-Gómez, Fabio A. González, and Thamar Solorio. 2019. Jointly learning author and annotated character N-gram embeddings: A case study in literary text. *International Conference Recent Advances in Natural Language Processing, RANLP*, 2019-Septe.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation : contextualized word vectors. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6297–6308.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec : Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61.

Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 393–401.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kevin P Murphy. 2007. Conjugate bayesian analysis of the gaussian distribution. *def*, $1(2\sigma 2)$:16.

Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate gaussian document representation from word embeddings for text categorization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, pages 450–455.

Seong Joon Oh, Kevin Murphy, Jiyan Pan, Joseph Roth, Florian Schroff, and Andrew Gallagher. 2019. Modeling uncertainty with hedged instance embedding. In *Proceedings of the International Conference on Learning Representations*.

Alberto Paccanaro and Geoffrey E Hinton. 2000. Learning distributed representations by mapping concepts and relations into a linear space. In *ICML*, pages 711–718.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, IJCAI, pages 1895–1901.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Jacques Prévert. 1949. *Paroles*. Gallimard Paris.

Herbert E Rauch, F Tung, and Charlotte T Striebel. 1965. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8) :1445–1450.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert : Sentence embeddings using siamese bert-networks. *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1003–1011. International World Wide Web Conferences Steering Committee.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11) :613–620.

Purnamrita Sarkar, Sajid M Siddiqi, and Geoffrey J Gordon. 2006. Approximate kalman filters for embedding author-word co-occurrence data over time. In *ICML Workshop on Statistical Network Analysis*, pages 126–139. Springer.

Purnamrita Sarkar, Sajid M Siddiqi, and Geogrey J Gordon. 2007. A latent space approach to dynamic embedding of co-occurrence data. In *Artificial Intelligence and Statistics*, pages 420–427. PMLR.

Hinrich Schütze. 1993. Word space. In *Advances in neural information processing systems*, pages 895–902.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer : extraction and mining of academic social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 990–998.

Enzo Terreau, Antoine Gourru, and Julien Velcin. 2021. Writing style author embedding evaluation. In *To appear in the Proceedings of the Second Workshop on Evaluation and Comparison of NLP Systems (@EMNLP 2021)*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 1999. The information bottleneck method. *The 37th annual Allerton Conference on Communication, Control, and Computing*, page 368–377.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong 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)*, volume 1, pages 1722–1731.

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*, pages 5998–6008.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding.

Steven Xiaogang Wang. 2001. *Maximum weighted likelihood estimation*. Ph.D. thesis, University of British Columbia.

Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Linked document embedding for classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 115–124. ACM.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1) :36–45.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers : State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. 2014. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601. PMLR.

Table des figures

Liste des tableaux

Long Abstract in English

What is sometimes referred as the digital revolution led to an exponential growth of the quantity of long term stored information. What used to be registered in stone, paper, or in our hippocampus, is now mainly saved in electronic devices. Most of this information is in the form of textual data. From digital books, web pages, Wikipedia, to social media and SMS, text is ubiquitous in human numeric activity.

Since decades, researcher and scientists developed automatic methods to handle, store, manipulate and treat the information. Most recently, a new discipline has arised. Machine learning - a branch of artificial intelligence - consists in "training" a model by feeding it data, therefore it can learn "intelligent" behavior, i.e., make decision. Natural Language Processing (NLP) is a sub domain of machine learning that focuses on textual data and oral language. ?

Most frequent tasks in NLP are text classification (labeling documents with a topic), clustering (grouping similar documents/words/author together), recommendation, sentiment analysis, and automatic text generation. The models that solve these tasks need the text to be represented as a mathematical object.

The historical approach represents documents using the document-term matrix (a document is a vector in the vocabulary space). This modeling has several limitations. It is sparse, leading to efficiency issues. The representation is in very high dimensions, leading to many unwanted properties (the curse of dimensionality). A new domain has arised to circumvent these problems. It is called representation learning. The aim of representation learning is to make use of machine learning approaches to learn representations for textual objects (word, document, author) in a low dimensional vectorial space. The similarity between the vectorial representations of these objects should relates with their semantic content or stylistic similarity.

In addition to the text itself, documents are often associated with meta data. They are often linked (e.g. hypertext references, citations in scientific production), tagged with their author and timestamped. This information was shown to improve the quality of the documents representations and can help to guide the author representation in a relevant vector space. Nevertheless, incorporating these meta data is not trivial.

Additionally, NLP evolved rapidly during last decades. Representation models are now trained on massive amount of textual data and fine-tuned on specific tasks. These models are of high interest when working with small datasets, allowing to transfer knowledge from relevant information sources. Developing representation learning models that can incorporate these pre-trained representations is therefore crucial.

Finally, most of prior works learn pointwise representations. This is a serious limitation as language is often ambiguous. Words are often polysemic, and documents mostly multi topic. A branch

of the literature proposes to learn probabilistic distributions in a semantic space to tackle this issue.

In this thesis, we first introduce the theoretical of machine learning, and a general overview of existing works in representation learning for words and documents (without meta data).

We then focus on representation learning for linked documents. We present prior works of the domain and propose two contributions : the RLE model (Regularized Linear Embedding) and the GELD model (Gaussian Embedding of Linked Documents). RLE builds a smoothed probability vector over the vocabulary for each document by combining its term frequency vector and its neighbors' term frequency vectors. These probabilities are used as weight when computing the mean of pre-trained word vectors. The GELD model uses a gaussian generative hypothesis. The document is a gaussian distribution in the pre-trained word vector space that "generates" the vector representation of the words of the documents, and the vector representations of the document it cites. We show that these two approaches are more accurate than prior work on a document classification task and on a link prediction task.

Then, we explore representation learning for authors and documents in the same vector space. We present most recent works and our contribution VADE (Variational Authors and Documents Embedding). VADE uses the Variational Information Bottleneck framework : it jointly learns representations of documents from pre-trained representations of words, and representations of authors. The representations of authors should be close to the representation of the documents it wrote. Both authors and documents representations are gaussians and make use of several encoders to map the word vectors of a document to its mean and variance. We evaluate this model in author classification, document classification and author identification, showing that this approach outperforms or matches existing methods.

Finally, we study the problematic of representation learning for authors in a dynamic context : their representations should evolve over time. We first review previous models, and propose an original contribution, DGEA (Dynamic Gaussian Embedding of Authors), a general framework for dynamic embedding of authors as evolving gaussian distributions. It makes use of pre-trained document representations, and it is based on the same generative hypothesis than GELD. We propose two instances of this general framework : K-DGEA, which uses a first-order Markov model for author mean evolution, and R-DGEA, which is a deep approach based on a LSTM. These models have several relevant properties : K-DGEA is fast to optimize and parallelizable, while R-DGEA can infer representations for unseen authors. Both models outperform or match prior models in author classification, link prediction, and author identification.

Additionally, we propose several scientific axes to improve our contributions, and some open questions for future research.