Amélioration des méthodes de pondération pour le plongement de documents

Université Lyon 2
M2 Data Mining
Travaux d'étude et de recherche

Réalisé par :

Bich-Ngoc HOANG, Pierre PRABLANC

Encadrants :

Antoine GOURRU, Robin BROCHIER, Julien VELCIN

Contents

1 Introduction

Pour comprendre la différence entre des mots, il faut baser sur le context où ces mots sont utilisés. C'est une tâche, dites facile, pour nous, tout être humain. Pourtant, pour les machines, ce n'est pas si simple. Si on prend un exemple des articles, des journaux que nous lisons chaque jour, qui abordent de plusieurs topics. Dans la plupart de ces articles, nous pouvons apprendre des mots nouveaux, des nouveaux concepts en basant sur ce que nous avons déjà su. Comme par exemple, en anglais, nous comprennons que le mot *Apple* désigne un fruit: *La pomme* mais en lisant des articles de techologies, nous pouvons apprendre un autre sens de ce mot, qui désigne le nom d'une entreprise américaine. De la même façon, les machines ont besoin également une manière qu'elles peuvent apprendre des topics et qu'elles peuvent commprendre la différence entre des mots. A fin de réaliser cette tâche, plusieurs méthodes sont introduits. Ce sont des méthodes de représentations de mots (Word Embedding).

Tant qu'il existe déjà plusieurs solutions efficaces pour des mots, il reste encore compliqué de produire et apprendre les représentations pour une longue pièce de texte, comme une sentence, paragraphe ou même un document entier.

Dans le cadre de ce travail d'étude et de recherche, nous intéressons à étudier des méthodes récents qui représentent les documents dans une espace de l'embedding des mots et de proposer des nouveaux méthodes pour améliorer les méthodes d'apprentissage de représentations de documents mais en fixant des contraintes comme réutiliser des embeddings de mots déjà appris. Plus précisément, nous cherchons à trouver de nouvelles méthodes de pondération d'embeddings de mots. L'idée principale est d'utiliser le topic modeling dans cette pondération.

2 Etat de l'art

2.1 Les modèles de plongement de documents

L'apprentissage de la réprésentation (Representation learning) a l'objectif de trouver une représentation vectorielle de plusieurs objects à fin de découvrir la structure interne pour produire, par exemple, une meilleure visualisation des données de grande dimensions.

Le plongement des mots ou la représentation des mots (Word Embedding) semble être une méthode efficace pour les tâches de Natural Language Processing et Information Retrieval. Il essaie de trouver les représentations des mots où les mots qui donnent le même sens, ont des représentations similaires. Les méthodes de Word Embedding (word2vec [Tomas Mikolov, 2013] ou GloVe [Jeffrey Pennington, 2014]) aident à réaliser plusieurs tâches de NLP (TAL) comme l'analyse de sentiments, désambiguïsation sémantique des mots, etc. À part de la propriété de préserver la proximité sémantique des mots, ces méthodes nous

permettent également d'effectuer des opérations linéaires importanes. Par exemple, en soustrayant la représentation vectorielle du mots *Man* à celle du mot *King* et en ajoutant celle du mot *Woman*, nous trouvons un vecteur qui est proche du vecteur du mot *Queen* (Exemple Fig.1).

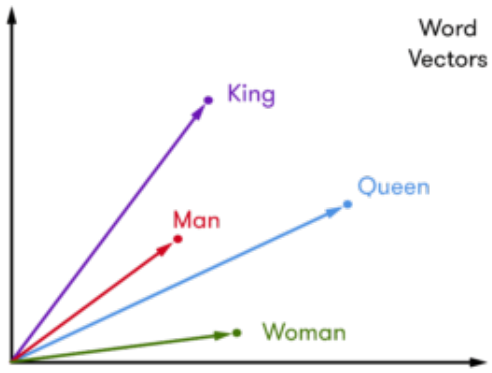

Figure 1: La réprésentation vectorielle des mots *King, Man, Queen, Woman.*

Pour aller plus loin, les recherches récentes ont essayé de capturer les sémantiques des séquences de mots, ce qu'on appelle, *Document Embedding*. Le plongement de documents signifie le plongement d'une séquence de mots (une phrase, un paragraphe, un document). Pourtant, il reste encore être une question, de représenter le sens d'une phrase complète (full sentence), de comment capturer les relations entre plusieurs mots et phrases dans un vecteur.

Plusieurs applications sont líées au plongement de documents, par exemple: Document retrieval, Web search, Spam filtering, etc.

Il existe jusqu'au moment des approches intéressants pour ce sujet comme *Paragraphe Vector* [Le and Mikolov, 2014], *Skip-Thought* [R. Kiros and Fidler, 2015], *FastSent* [Felix Hill, 2016], *Sent2vec* [Matteo Pagliardini, 2017]. Dans la partie qui suit, nous présenterons les trois grandes approches *Paragraphe Vector*, *Skip-Thought* et *Sent2vec*.

2.1.1 Paragraph Vector

Paragraph Vector (doc2vec) est proposé par [Le and Mikolov, 2014]. C'est un apprentissage non supervisé, qui accepte en input les sentences, paragraphes et documents. Un des objectifs de Paragraph Vector est de surpasser les faiblesses de Bag of words (BOW) bien qu'il soit bien simple, efficace et souvent donne une bonne précision, mais qui perd l'ordre des mots et qui ignore les sémantiques des mots. Ce modèle est éventuellement meilleure que Bag of n-grams puisque le dernier peut créer une représentation ayant une grande dimension et qui est

éparse, bien qu'il garde l'ordre des mots.

Un avantage important de Paragraph Vector est d'apprendre à partir des données non libellées et il peut également travailler sans avoir suffisamment de données libellées. Il hérite une propriété importante du plongement de mots (word2vec): la sémantique des mots. Il garde aussi l'ordre des mots, au moins, dans un petit context, de même manière que Bag of n-grams a fait.

Si vous connaissez word2vec, on apprend la représentation d'un mot en observant le contexte voisinage. Pareil avec Doc2vec, pourtant, il ajoute un vecteur dans ce context, représentant le paragraphe. De la même manière que word2vec récourt aux méthodes CBOW et Skip-gram, doc2vec récourt à deux méthodes: PV-DM et PV-DBOW. Chacun de ces deux ont leurs propres avantages et inconvénients.

Distributed Memory Model of Paragraph Vectors PV-DM : Chaque fois qu'on essaie de prédire un mot grâce à son contexte, on introduit un vecteur qui correspond au paragraphe où le contexte (la fenêtre) se situe. Avec ce méthode, on est capable de désambiguïser deux séquences de mots qui ont des mêmes mots mais qui veulent dire deux différentes choses. Le vecteur de paragraphe aide également à informer le modèle de, quel concept est le plus probable à discuter dans le context actuel.

Dans l'article [Le and Mikolov, 2014], les auteurs ont mentionné que ce vecteur de paragraphe joue un rôle comme une mémoire, car il changera en basant sur les mots qui existent dans des autres parties du même paragraphe ou en une façon, il se souvient ce qui manque dans le contexte actuel du paragraphe. C'est pour cette raison, ce méthode est appelé sous le nom Distributed Memory Model.

En observant la structure du modèle (Fig.2), chaque paragraphe est représenté comme un vecteur unique qui est une colonne de la matrice **D** et chaque mot est représenté comme un vecteur unique qui est une colonne de la matrice **W**. Le vecteur paragraphe et les vecteurs de mots sont moyennés ou concaténés pour prédire le mot suivant dans un context.

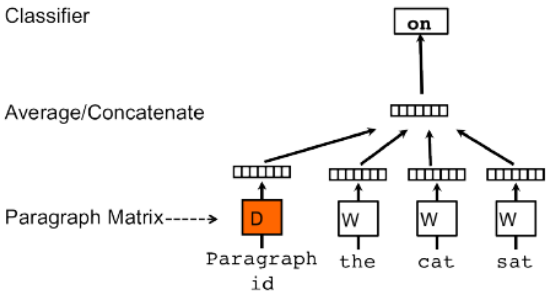

Figure 2: Structure pour le modèle Paragraphe Vector: Distributed Memory. [Le and Mikolov, 2014]

Distributed Bag of Words of Paragraph Vectors PV-DBOW : Une autre manière à trouver ces vecteurs est le méthode PV-DBOW. Avec ce méthode, le vecteur paragraphe est utilisé pour prédire un nombre de mots qui sont échantillonnés aléatoirement à partir du paragraphe. Bien que ce modèle commence par un seul élément et prédit un nombre, il semble être similaire au méthode Skip-gram de word2vec.

La structure de ce méthode est présentée dans Fig.3. Dans ce méthode, le vecteur paragraphe est entrainé à prédire les mots dans un petit context.

Pour comparer ces deux méthodes, [Le and Mikolov, 2014] a indiqué que le fait d'utiliser seulement le méthode PV-DM a tendance à fonctionner mieux que PV-DBOW, pourtant, en combinant ces deux méthodes, il produit un meilleur résultat.

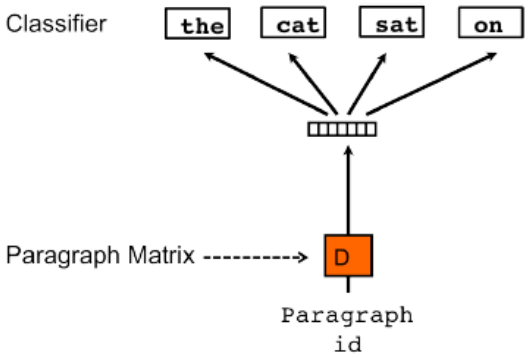

Figure 3: Structure pour le modèle Paragraphe Vector: Distributed Bag of Words. [Le and Mikolov, 2014]

2.1.2 Skip-Thought

Skip-Thought est proposé dans l'article de [R. Kiros and Fidler, 2015] en 2015. C'est un modèle pour l'apprentissage vecteur de sentence sans avoir une tâche supervisée particulière. Le seul signal de supervision que Skip-Thought utilise est l'ordre des sentences dans le corpus. Etant inspirés par les représentations vectorielles des mots, les auteurs proposent une fonction d'objectif qui résume le modèle Skip-gram pour un niveau de "sentence". Au lieu d'utiliser un mot pour prédire le context autour de lui, il encode une sentence pour prédire les sentences voisinages de celle-ci.

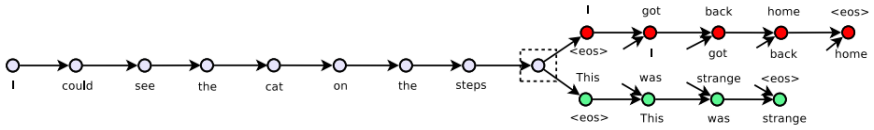

Figure 4: Modèle de Skip-Thought [R. Kiros and Fidler, 2015]

Donnant un tuple (x_{i1}, x_i, x_{i+1}) de sentences, avec x_i la i-ème sentence, la sentence x_i est encodée et essayer de reconstruire la sentence précédente x_{i-1} et celle suivante x_{i+1}. Dans l'example de Fig.4, l'input est le triplet de sentence: *I got back home. I could see the cat on the steps. This was strange.* Les flèches sans attaches sont connectées à l'encodeur d'ouput. Les couleurs indiquent quels components partagent les paramètres. *eos* désigne la fin du token de sentence.

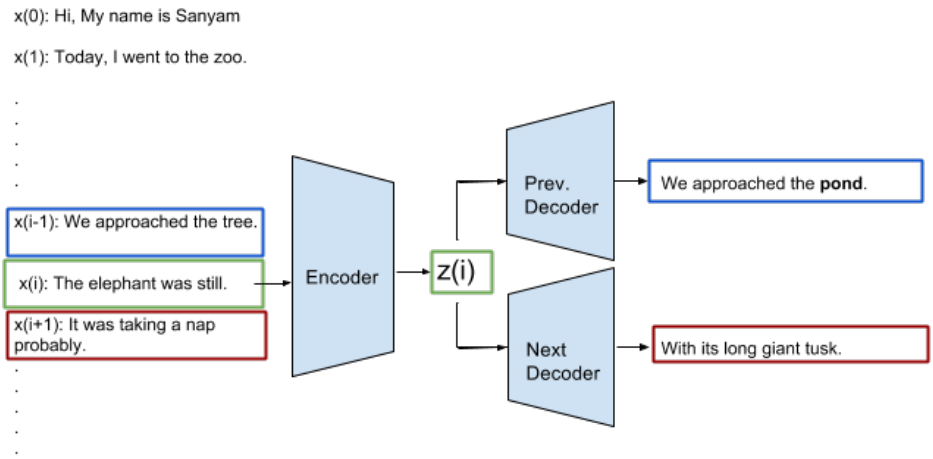

Figure 5: Modèle de Skip-Thought (Source: Medium)

- Skip-Thought est construit de trois parties:

- Encodeur: Prend la sentence x_i à l'index i et générer une représentation de taille fixée z_i.
- Décodeur précédent (Prev Decoder): Prend la représentation z_i et essayer de générer la sentence x_{i-1}
- Décodeur suivant (Next Decoder): Prend la présentation z_i et essayer de générer la sentence x_{i+1}

Un aspect intéressant de Skip-Thought est l'expansion du vocabulaire. Il peut gérer les mots qu'il n'a jamais vu pendant l'entrainement, en apprenant une transformation linéaire entre leur RNN word embedding et un autre word embedding plus large comme word2vec.

2.1.3 Sent2vec

Dans l'article [Matteo Pagliardini, 2017], un nouveau modèle pour sentence embedding - Sent2vec - est introduit. Sent2vec présente un simple méthode non supervisé mais efficace à entrainer les représentations des sentences. Ce modèle peut être considéré comme une extension de CBOW pour les sentences, au lieu des mots.

Formallement, Sent2vec apprend un embedding source (ou contexte) v_w et un embedding cible u_w pour chaque mot w dans le vocabulaire ν avec la dimension de l'embedding h et $k = |\nu|$. Donc "Sentence embedding" est défini comme la moyenne des embeddings sources des mots. On peut aller plus loin avec ce modèle en apprenant embedding source non pas seulement pour les unigrams mais aussi pour les n-grams qui existent dans chaque sentence, et moyenner le n-gram embeddings avec les mots. v_S - Sentence embedding pour sentence S est présenté comme ci-dessous:

$$v_S := \frac{1}{|R(S)|} V \iota_{R(S)} = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w$$

Où:

$R(S)$: liste des n-grams (unigrams également) qui existe dans la sentence S
ι : vecteur indicateur $\in 0, 1^k$: vecteur binaire encode S
V, U : matrices qui stockent les vecteurs des mots source et cible ($U \in \mathbb{R}^{k \times h}$, $V \in \mathbb{R}^{h \times k}$)

La complexité de Sent2vec est un des points forts pour ce modèle car elle n'égale que $O(1)$. Celui ci est fortement en contrast des modèles basant sur des réseaux neurons comme Skip-Thought, qui doit passer plus de temps à entrainer car il implique un entrainement avec le méthode de encoder-decoder où l'encodeur représente la sentence d'input comme un vecteur de sentence et le décodeur génère ses sentences voisinages. Les autres modèles basant sur les

réseaux neurons ont également des complexités de $O(n)$ et $O(n^2)$ où n est la taille du texte. Avec une faible complexité, Sent2vec peut travailler sur un dataset qui est extrêmement large.

2.2 Méthodes de pondération pour le plongement de documents

Nous avons vu précédemment qu'il était possible d'apprendre des représentations de documents à partir d'un corpus. Ces techniques nécessitent cependant beaucoup de données et restent coûteuses en temps de calcul ou en matériel. Il existe pourtant un autre moyen de construire des vecteurs de documents. En combinant les vecteurs d'embedding de mots, il est possible d'obtenir un embedding de documents. Étant donné qu'il existe déjà des vecteurs d'embedding de mots pré-entrainés et performants, il n'est pas nécessaire d'effectuer un apprentissage depuis le départ. Ce type de méthode est rapide, économique et ne nécessite pas un apprentissage lorsqu'on souhaite obtenir des vecteurs pour de nouveaux documents. Il suffit alors de trouver les coefficients pondération α_j des vecteurs de mots pour un document:

$$\mathbf{v}_d = \frac{1}{Z} \sum_{w_j \in d} \alpha_j \mathbf{u}_{w_j} \tag{1}$$

où Z est un coefficient de normalisation du vecteur de document, $\mathbf{v}_d$ est le vecteur du document et $\mathbf{u}_{w_j}$ est un des vecteurs mots du document.

La méthode du barycentre est la méthode la plus simple et consiste à calculer la moyenne des vecteurs de mots d'un document. Ici les coefficients α_j sont tous égaux à 1:

$$\mathbf{v}_d = \frac{1}{|d|} \sum_{w \in d} \mathbf{u}_w \tag{2}$$

Malgré sa simplicité, cette solution offre malgré tout des résultats intéressants comparées à des méthodes plus sophistiquées. Cette baseline est d'ailleurs utilisée pour comparer Doc2Vec [Lau and Baldwin, 2016]. La méthode du barycentre présente pourtant quelques inconvénients. Elle considère que tous les mots ont la même importance dans un document, ce qui est une hypothèse discutable, nous reviendrons sur ce point plus tard. Par ailleurs, elle ne tient pas compte de la polysémie.

La méthode de TF-IDF : D'autres pondérations ont déjà été testées telles que la pondération TF-iDF et SIF (*Smooth Inverse Frequency*) dans [Sanjeev Arora, 2017]. La pondération des vecteurs mots avec TF-iDF s'écrit:

$$\mathbf{v}_d = \frac{1}{|d|} \sum_{w \in d} iDF_w \mathbf{u}_w \tag{3}$$

Ici, on n'utilise que la partie iDF_w pour la pondération, la partie TF se retrouve dans $w \in d$.

La méthode SIF : La méthode proposée dans [Sanjeev Arora, 2017] modélise la probabilité d'un mot dans un document étant donné le vecteur de discours dans cette phrase. Les détails sur la définition de ce vecteur de discours sont assez flous. Une solution pour les paramètres de ce modèle est obtenue par un estimateur du maximum de vraisemblance. L'algorithme se déroule ainsi en deux temps (Algorithme 1). Dans un premier temps, on calcule une première estimation des vecteurs de documents comme une somme pondérée des vecteurs mots du document en utilisant une estimation de la probabilité d'occurence des mots:

$$\mathbf{v}_d \leftarrow \frac{1}{|d|} \sum_{w \in d} \frac{a}{a + p(w)} \mathbf{u}_w \tag{4}$$

où $p(w)$ est l'estimation de la probabilité d'occurence du mot w et a est un hyperparamètre fixé entre $0,0001$ et $0,001$. Dans un deuxième temps, on effectue une SVD sur la matrice $\mathbf{X}$ de l'estimation des vecteurs de documents et on extrait le premier vecteur singulier. Enfin, on retire à la première estimation du vecteur de document la projection de ce vecteur sur sa première composante principale. Il est dit dans [Sanjeev Arora, 2017] que cette méthode obtient des résultats similaires à TF-iDF.

Algorithm 1: Pondération SIF

Input : Vecteurs de mots $\{\mathbf{u}_w : w \in V\}$, un ensemble de documents D, le paramètre a et l'estimation des probabilités $\{p(w) : w \in V\}$ des mots.

Output: Vecteurs de documents $\{\mathbf{v}_d : w \in D\}$

for *document* $d \in D$ **do**

  $\mathbf{v}_d \leftarrow \frac{1}{|d|} \sum_{w \in d} \frac{a}{a+p(w)} \mathbf{u}_w$

end

Créer la matrice $\mathbf{X}$ dont les colonnes sont formées par $\{\mathbf{v}_d : d \in D\}$ et en extraire le premier vecteur singulier noté $\mathbf{g}$.

for *document* $d \in D$ **do**

  $\mathbf{v}_d \leftarrow \mathbf{v}_d - \mathbf{g}\mathbf{g}^T \mathbf{v}_d$

end

2.3 La modélisation thématique

La modélisation thématique désigne une manière de modéliser une collection de documents permettant d'en extraire les thèmes ou topics. Un thème est défini comme une "idée sur lesquels portent une réflexion, un discours, une œuvre, autour desquels s'organise une action" (définition Larousse). C'est donc une

notion abstraite qui n'est pas toujours exprimée de façon explicite. Un roman peut présenter un titre n'exprimant pas le ou les thèmes principaux du livre. C'est également une notion subjective puisque deux individus peuvent associer à un corpus des thèmes différents ou dans des proportions différentes. Selon la définition du dicitonnaire, on comprend bien que le problème de modélisation thématique semble mal posé. Assez étrangement, l'objet de la modélisation, ici le thème, est rarement étudié dans la littérature scientifique. Au lieu de cela, on en vient directement aux considérations du modèle. On peut néanmoins extrapoler une définition de [Hofmann, 2013] qu'un topic se situe au niveau sémantique, se différenciant du niveau lexical, et cherche à répondre à la question "à quoi fait-on référence ?". Nous pensons qu'une analyse plus approfondie de la notion de thème pourrait être intéressante afin de mieux comprendre ce que l'on cherche à modéliser. Néanmoins puisqu'il ne s'agit pas de l'objet premier de ce rapport, nous reprenons la définition du thème au sens de la modélisation thématique probabiliste. Un topic est une variable latente (ou cachée) qui est à l'origine du processus de génération de la sémantique du document. Les modèles thématiques reposent sur les hypothèses suivantes. Un document est composé par un mélange de topics. Un topic est représenté par une liste de mots. Les méthodes de modélisation thématique cherchent ainsi à retrouver ces variables cachées.

Les applications de la modélisation thématique sont nombreuses. De façon non exhaustive, on peut citer la recherche d'information, le filtrage collaboratif pour les systèmes de recommandation, la bio-informatique ou bien l'analyse d'articles de presse [M. Blei et al., 2001], [M Wallach, 2008].

À présent, nous nous posons la question des motivations de la modélisation thématique (automatique). On pourrait imaginer une extraction des topics d'un document de façon manuelle. Or un thème n'est généralement pas exprimé de façon explicite dans un document ou un corpus. Et lorsque ce thème est explicité (par exemple un titre d'article), on peut vouloir chercher à connaitre des thèmes plus précis, plus spécifiques. La tâche peut donc s'avérer longue. De plus, les quantités de données textuelles ont dépassé la capacité de traitement humaine. D'où l'intérêt d'un traitement automatique.

En conséquence, plusieurs modèles ont été développés depuis les années 1990 tels que LSA [Deerwester et al., 1990], pLSA [Hofmann, 2013], LDA [M. Blei et al., 2001] et ses variantes [Li and Mccallum, 2006]. Nous ne présentons dans ce rapport que les modèles LSA, pLSA et LDA.

2.3.1 Analyse sémantique latente

L'analyse sémantique latente, plus connue dans l'appellation anglophone "Latent Semantic Analysis" (LSA) ou même "Lantent Semantic Indexing" (LSI), est une méthode introduite par Deerwester et al. [Scott Deerwester et al., In-

dexing by Latent Semantic Analysis]. Dans ce modèle, on fait l'hypothèse qu'un document est une combinaison linéaire de plusieurs topics et qu'un topic est une combinaison linéaire de plusieurs mots. Pour cela, cette méthode se base sur la décomposition en valeurs singulières (SVD) de la matrice document-terme qui utilise généralement la pondération tf-idf. De façon formelle, la SVD permet de factoriser une matrice rectangulaire en un produit de 3 matrices tel que dans notre cas:

$$\mathbf{X} = \mathbf{D} \cdot \mathbf{S} \cdot \mathbf{W}^T \tag{5}$$

où $\mathbf{X} \in \mathbb{R}^{m \times n}$ est la matrice document-terme (tf-idf), $\mathbf{D} \in \mathbb{R}^{m \times m}$ est la matrice document-topic, $\mathbf{S} \in \mathbb{R}^{m \times n}$ est une matrice diagonale et $\mathbf{W} \in \mathbb{R}^{n \times n}$ est la matrice topic-terme. En tronquant les matrices, on peut obtenir une approximation de la matrice document-terme en tronquant les matrices de sorte que:

$$\mathbf{X} \approx \mathbf{D}_t \cdot \mathbf{S}_t \cdot \mathbf{W}_t^T \tag{6}$$

où t correspond au nombre de vecteurs propres que l'on conserve. On associe également t au nombre de topics que l'on souhaite conserver. Chaque ligne de $\mathbf{D}_t \in \mathbb{R}^{m \times t}$ correspond à la représentation d'un document en termes de topics. De même, chaque ligne de $\mathbf{W}_t \in \mathbb{R}^{m \times t}$ correspond à la représentation d'un mot en termes de topics. Cette méthode permet d'effectuer des opérations de comparaisons dans l'espace des topics entre mots, entre documents ainsi qu'entre une paire mot-document.
En revanche, elle présente plusieurs inconvénients. D'abord, il n'est pas possible de connaître les noms des topics car les matrices $\mathbf{D}_t$ et $\mathbf{W}_t$ sont composées de valeurs positives et négatives non normalisées. Il n'est donc pas facile d'interpréter ces valeurs de coefficients. Par ailleurs, la SVD repose sur une hypothèse de normalité des données qui n'est pas nécessairement vérifiée.

2.3.2 Analyse sémantique lantente probabiliste

En 1999 T. Hofmann introduit une nouvelle approche nommée probabilistic Latent Semantic Analysis (pLSA). Contrairement à LSA, cette méthode ne repose pas sur une SVD de la matrice document-terme mais sur une modélisation probabiliste. Dans cette méthode, on cherche à modéliser la probabilité jointe $P(w_j, d_i)$ des mots et des documents. Les nouvelles hypothèse sur ce modèle sont les suivantes. Pour un document donné, chaque topic z_k a une probabilité $P(z_k|d_i)$ d'être présent dans un document. Pour un topic donné, chaque mot a une probabilité $P(w_j|z_k)$ d'être généré par ce topic. De manière plus formelle, on peut écrire la probabilité jointe d'un mot et d'un document comme:

$$P(w_j, d_i) = \sum_{k=1}^{K} P(z_k) P(w_j, d_i|z_k) \tag{7}$$

qui est obtenue en marginalisant par rapport aux topics et en utilisant la règle de Bayes. En faisant une hypothèse d'indépendance entre les documents et les

mots conditionnellement aux topics, on obtient:

$$P(d_i, w_j) = P(d_i) \sum_{k=1}^{K} P(z_k|d_i)P(w_j|z_k) \tag{8}$$

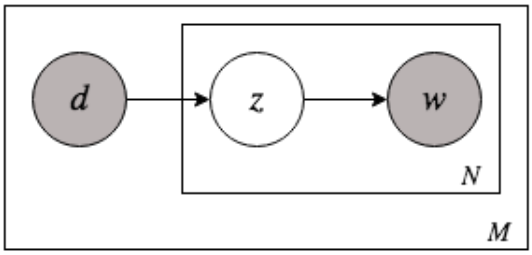

Figure 6: Diagramme en assiettes du modèle pLSA, formulation asymétrique.

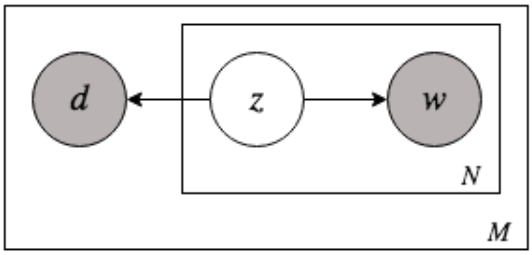

Figure 7: Diagramme en assiettes du modèle pLSA, formulation symétrique.

Sous cette forme, $P(w_j, d_i)$ peut être vu comme un modèle de mélange de distributions multinomiales dont les proportions de chaque composante sont les probabilités a priori des topics. Cette relation peut se formuler en utilisant un modèles graphique. Pour représenter ces modèles, on utilise un diagramme en assiettes (*plate notation*). La représentation du modèle provenant de (8) dite asymétrique est visible sur la figure 6. L'estimation de ce modèle peut se faire en utilisant un algorithme EM.
En utilisant une nouvelle fois la règle de Bayes, on peut écrire:

$$P(w_j, d_i) = \sum_{k=1}^{K} P(z_k)P(d_i|z_k)P(w_j|z_k) \tag{9}$$

qui correspond à la formulation symétrique de (8) représenté par le diagramme en assiette de la figure 8. Cette formulation permet d'établir une relation entre LSA et pLSA. En effet, on peut faire une analogie entre $P(w_j, d_i)$ et la matrice document-terme X. De même, on peut établir une correspondance entre $P(z_k)$,

$P(d_i|z_k)$, $P(w_j|z_k)$ et $\mathbf{S}_t$, $\mathbf{D}_t$, $\mathbf{W}_t$ respectivement.

pLSA présente également plusieurs limitations. D'abord, il n'y a pas de modélisation probabiliste au niveau des documents. Dans un sens, on capture la possibilité qu'un document puisse contenir plusieurs topics puisque $p(z|d_i)$ peut être vu comme une combinaison de poids des topics pour un document d_i. Pourtant, il est important de noter que d_i est un index virtuel puisqu'on ne numérote pas les documents dans le training set. Par conséquent, d_i est une variable multinomiale aléatoire avec autant de valeurs possibles qu'il y a de documents (dans le training set) et le modèle apprend le mélange de topics $p(z|d_i)$ seulement pour les documents pour lesquels on l'a entrainé. Pour cette raison, pLSI n'est pas un bon modèle pour générer des documents. Il n'y a aucune façon d'assigner des probabilités à des documents non observés dans l'entrainement sans sortir du cadre théorique dans lequel pLSA est défini.
Par ailleurs, le nombre de paramètres augmente linéairement avec le nombre de documents à entrainer. Les paramètres pour un mélange de K topics proviennent de K distributions multinomiales de taille V (taille vocabulaire) avec M composantes sur les K topics cachés. On a donc $kV + kM$ paramètres à apprendre. La taille du vocabulaire augmente peu si on augmente le nombre de documents (sauf quand on a peu de documents). À nombre de topics fixé, le nombre de paramètres évolue linéairement avec le nombre de documents à entrainer.

2.3.3 Allocation de Dirichlet latente

Afin de palier à ces limitations, un autre modèle allant plus loin dans la modélisation probabiliste a été proposé par David M. Blei, Andrew Y. Ng et Michael I. Jordan en 2003. LDA se différencie du modèle précédent puisqu'on ne suppose plus que les documents sont observés. En effet, on modélise la distribution de topics pour un document donné par θ qui est un échantillon aléatoire d'une distribution de Dirichlet. Pour comprendre le principe du modèle, on imagine un processus

génératif qui permet de générer des documents décrit par l'algorithme suivant:

Algorithm 2: Pseudo-algorithme de LDA

1. Pour chaque document $i \in (1, ..., M)$, on tire un échantillon aléatoire d'une distribution de Dirichlet $Dir(\alpha)$ qui représente la distribution θ_i de topics pour chaque document.
2. Pour chaque topic $k \in (1, ..., K)$, on tire un échantillon aléatoire d'une distribution de Dirichlet $Dir(\beta)$ qui représente la distribution φ_k de mots pour chaque topic.
3. Pour chaque document $i \in (1, ..., M)$, on tire au hasard un topic $z_{i,j}$ de la distribution $Multinomial(\theta_i)$
4. Pour chaque topic tiré $z_{i,j}$, on tire au hasard un mot $w_{i,j}$ de la distribution $Multinomial(\varphi_{z_{i,j}})$ avec $j \in (1, ..., N_i)$

On cherche ainsi à estimer les variables latentes suivantes:

- $\theta \sim Dir(\alpha)$: distribution des topics pour un document
- $\varphi \sim Dir(\beta)$: distribution des mots pour un topic
- $z_{i,j}$: identité du topic dans le document i pour le mot j

Les paramètres de ce modèle peuvent être estimés de plusieurs manières (inférence variationnelle, *Gibbs sampling*, expectation propagation). L'estimation par inférence variationnelle est généralement plus rapide que la méthode par *Gibbs sampling*. En revanche, cette dernière est plus précise [Wenig, 2018].

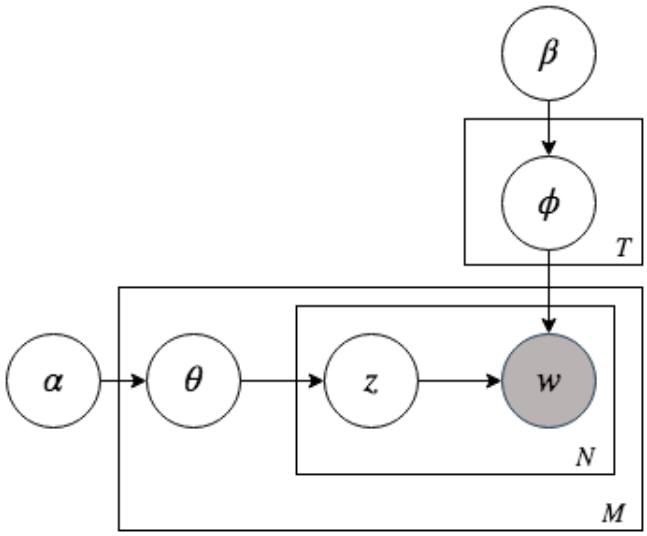

Figure 8: Diagramme en assiettes du modèle pLSA, formulation symétrique.

3 Contribution

Dans cette partie nous proposons deux approches de pondération de terme basée sur la modélisation thématique. Dans un premier temps, nous présentons les méthodes et les problèmes qu'elles cherchent à adresser. Par la suite, nous décrivons les modalités d'évaluation de ces méthodes. Une description des configurations des méthodes proposées et des baselines est donnée dans la partie suivante. Enfin nous comparons les méthodes proposées avec plusieurs baselines. Nous présentons ensuite les résultats et analyses.

3.1 Méthodes proposées

Les méthodes de pondération à partir d'embedding de mots, citées précédemment, sont assez restrictives. Prenons l'exemple de la pondération barycentrique. Cette méthode fait l'hypothèse que tous les mots ont la même importance. Dans ce cas là, seule le nombre d'occurence d'un mot permettra de donner plus d'importance à un mot en particulier. Or un mot, même s'il présente une faible fréquence d'occurence peut être plus représentatif d'un document qu'un autre mot plus fréquent dans ce document. Les méthodes de pondération comme TF-iDF et SIF devraient pouvoir remédier à ce problème. Mais qu'en est-il dans le cas où tous les mots ont environ la même fréquence d'occurence ? Il semble pourtant intuitif que certains mots apportent plus d'information que d'autre sur le sens du document. Par exemple le mot "HMM" renseigne plus que le mot "algorithm" qui est d'ordre plus général.

En effet, nous pensons que la modélisation thématique peut nous permettre de gérer les problèmes que nous venons d'évoquer. La connaissance du thème d'un document et les mots qui lui sont fortement associés doit pouvoir nous permettre de pondérer les mots qui influent fortement sur le document. Ceci doit permettre de répondre à la question de l'importance d'un mot dans un document même s'il est peu fréquent dans le document.

3.1.1 Pondération par topic majoritaire

La première méthode proposée repose sur l'hypothèse que la majeure partie de l'information est contenue dans le topic majoritaire du document. La distribution de probabilité des mots dans ce topic constitue ainsi la pondération des termes d'embedding de mots. On calcule donc un modèle thématique sur le corpus étudié. Pour chaque document, on extrait l'identifiant du topic majoritaire et on applique au vecteur d'embedding la probabilité de génération du mot dans le topic. Pour un document donné d_i, nous définissons les poids des vecteurs mots de la façon suivante:

$$k^* = \underset{k}{\operatorname{argmax}}\, P(z = k|d = d_i) \tag{10}$$

Nous avons ensuite pensé à utiliser logiquement $P(z = k^*|d = d_i)$ pour la pondération. Cependant, pour une raison que ne n'avons pas encore pu déterminer, les résultats sont très mauvais. Pour une deuxième raison que nous ignorons également, nous avons utilisé une fonction softmax qui permet de donner une raison d'être à notre méthode:

$$\alpha_j = \frac{\exp P(w_j|z = k^*)}{\sum_l \exp P(w_l|z = k^*)} \tag{11}$$

De cette manière, on donnera plus d'importance aux mots qui ont une forte probabilité d'être généré dans le topic majoritaire.
Cependant, il peut arriver que le document soit gouverné par plusieurs topics aux proportions comparables. Imaginons qu'on se situe dans le cas suivant: 50% pour le premier topic, 40% pour le second et 10% réparti sur les autres topics. Des mots du topic majoritaire seront pondérés fortement, mais des mots qui auraient eu une forte probabilité d'occurence dans le 2nd topic ont peut être une faible probabilité dans le topic 1. L'hypothèse du topic majoritaire dans ce cas n'est pas respectée et pénalise les topics secondaires avec une proportion relative élevée. Nous proposons donc une deuxième solution pour prendre en compte ce cas.

3.1.2 Deuxième méthode de pondération

L'idée de cette deuxième méthode est de ne pas se restreindre sur le topic majoritaire mais de considérer les proportions de chaque topic conjointement avec la probabilité de génération du mot dans le topic. De cette manière, on ne rejette pas les topics secondaires. De manière pratique, on extrait le topic qui maximise le produit de la proportion du topic dans le document et de la probabilité de génération du mot dans le topic:

$$k^* = \underset{k}{\operatorname{argmax}}\, P(z = k|d = d_i)P(w|z = k) \tag{12}$$

De même, on utilise une fonction softmax pour calculer les poids:

$$\alpha_j = \frac{\exp P(z = k^*|d = d_i)P(w_j|z = k^*)}{\sum_l \exp P(z = k^*|d = d_i)P(w_l|z = k^*)} \tag{13}$$

Nous allons à présent évaluer la qualité de embeddings de documents générés.

3.2 Expériences

Pour évaluer la qualité des embeddings, nous décidons de comparer nos 2 méthodes proposées avec 4 baselines dans une tâche de classification de documents. Pour cela, nous utilisons un sous ensemble du jeu de données annotées CORA qui contient 2211 résumés d'articles scientifiques. Chaque document est associé à une classe parmi 7 (Case Based, Genetic Algorithms, Neural Networks,

Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory). Les documents contiennent en moyenne 90 mots.

Les 4 baselines sont construites sur les pondérations barycentrique, TF-iDF, Okapi-BM25 et SIF. Trois de ces baselines ont déjà été expliquées précédemment. Nous avons décidé de tester également la pondération Okapi-BM25 qui est une alternative à TF-iDF.
Pour la pondération TF-iDF, aucun pré-traitement n'est effectué et nous utilisons les paramètres suivant pour Okapi-BM25: $k = 2$ et $b = 0.75$. La méthode SIF nécessite de calculer la probabilité d'occurence des mots $p(w)$ et de fixer l'hyperparamètre a. Nous utilisons donc les jeux de données *text8* et CORA pour estimer les probabilités des mots $p(w)$ et fixons a à 0.0001. Nous conservons tous les mots du vocabulaire pour estimer ces probabililités.

Pour les deux méthodes proposées, nous effectuons au préalable un prétraitement des données pour l'extraction des topics. On retire ainsi les mots vides et on ne conserve que les mots ayant une fréquence supérieure à 10. Nous utilisons LDA pour la modélisation thématique et testons avec 5 et 10 topics. Pour le reste, les méthodes sont implémentées telles qu'elles sont décrites précédemment.

Nous testons deux embeddings construits avec GloVe. Le premier est constitué des 50000 premiers vecteurs d'embedding en dimension 300, pré-entrainés sur les données de *Wikipedia 2014* et *Gigaword 5*, soit 6 milliards de tokens et 400000 mots de vocabulaire. Le deuxième embedding est une extension du premier en effectuant un deuxième apprentissage sur le dataset CORA. Nous utilisons le premier embedding en initialisation et fixons les valeurs des autres paramètres du modèle aléatoirement entre -0.5 et 0.5. Nous conservons une taille de vecteur égale à 300. Le paramètre x_max de saturation de la fonction de pondération est fixé à 100. On fixe le nombre d'itération pour AdaGrad à 100.
La taille du nouvel embedding est d'environ 50300. Les tailles des deux embeddings sont donc comparables. Lorsque nous utilisons le premier embedding, nous ignorons les mots ne figurant pas dans le vocabulaire. Nous pouvons ainsi tester s'il est pertinent d'utiliser des vecteurs d'embedding de mots estimés sur un dataset relativement réduit (environ 200000 mots pour CORA) ou s'il est préférable d'ignorer les mots ne figurant pas dans l'embedding initial.

Nous utilisons les classifieurs bayésien naïf (NB) et Machine à Vecteur Support (SVM). Il est intéressant de tester ces 2 classifieurs puisqu'ils sont très différents. Le premier classifieur est une méthode générative donnant des résultats généralement intéressant compte tenu de sa simlicité tandis que le deuxième est une méthode discriminative réalisant de la sélection de variables. Pour les deux classifieurs, nous utilisons les paramètres par défaut des méthodes du package *e1071*. Notamment, le classifieur bayésien naïf n'utilise pas de lissage de laplace. Pour SVM, on utilise un kernel radial sans régularisation particulière

(on n'utilise pas nu-SVM).

Nous mesurons la qualité de prédiction des classifieurs en utilisant le F1-score qui correspond à la moyenne harmonique du rappel et de la précision. Nous répartissons les données de test et d'entrainement dont la proportion de données pour l'entrainement varie de 10% à 50% par pas de 10%. Pour chaque paramétrage, nous répétons 10 fois cette tâche de classification afin d'avoir des résultats plus fiables.

3.3 Résultats et analyse

Une partie des résultats obtenus est visible sur la table 1 et sur les graphiques Fig.9, Fig.10, Fig.11 et Fig.12. Nous pouvons comparer les résultats obtenus avec les 2 classifieurs. En moyenne, les SVM donnent généralement de meilleurs résultats que NB. Le classifieur SVM semble donc plus adapté aux données d'embedding de document.

Dans toutes les configurations, les deux baselines TF-iDF et Okapi donnent les moins bons résultats. Étonnamment, nous n'avons pas réussi à retrouver les résultats obtenus dans [Sanjeev Arora, 2017] indiquant que TFi-DF et SIF obtiennent des résultats similaires. Il faudrait peut être approndir les prétraitements et la paramétrisation utilisée.

Si on compare à présent les 2 embeddings de mots (GloVe et GloVe extension CORA), on constate que les configurations ne sont pas toutes affectées de la même manière. Dans le cas où l'on utilise les embeddings GloVe non modifiés, on constate que la baseline barycentrique donne presque tout le temps les meilleurs résultats. Elle semble d'autant plus performante que le pourcentage de données d'entrainement est grand. En revanche, lorsque les données d'entrainement sont en faible quantités (inférieur à 20%), la baseline SIF obtient de meilleurs résultats. Les méthodes que nous proposons ne parviennent pas à battre les résultats des autres baseline lorsqu'on utilise les embeddings GloVe.
À présent, lorsqu'on utilise des embeddings ré-entrainés sur le corpus CORA, on constate que nos propositions permettent d'obtenir presque tout le temps de meilleurs résultats que les autres baselines. Il n'est cependant pas possible de dire s'il est préférable d'utiliser ce nouvel embedding par rapport à l'embedding original car les résultats sont très proches et pas assez systématiques.

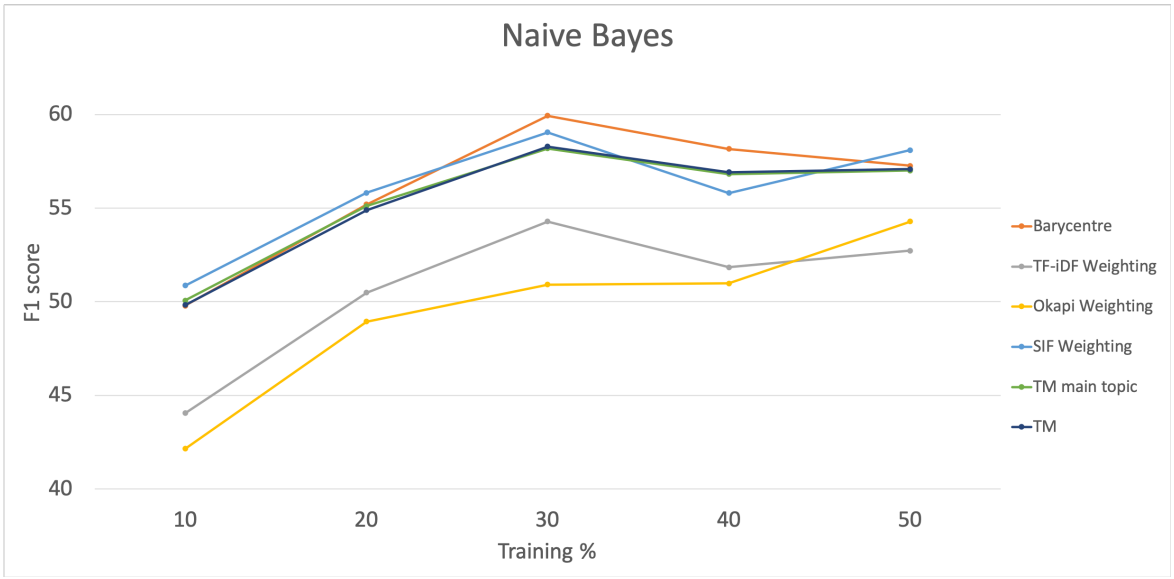

Figure 9: Graphique représentant les résultas en F1 score, obtenus par Naive Bayes en fonction du nombre de données pour l'entrainement (Sans apprentissage sur CORA)

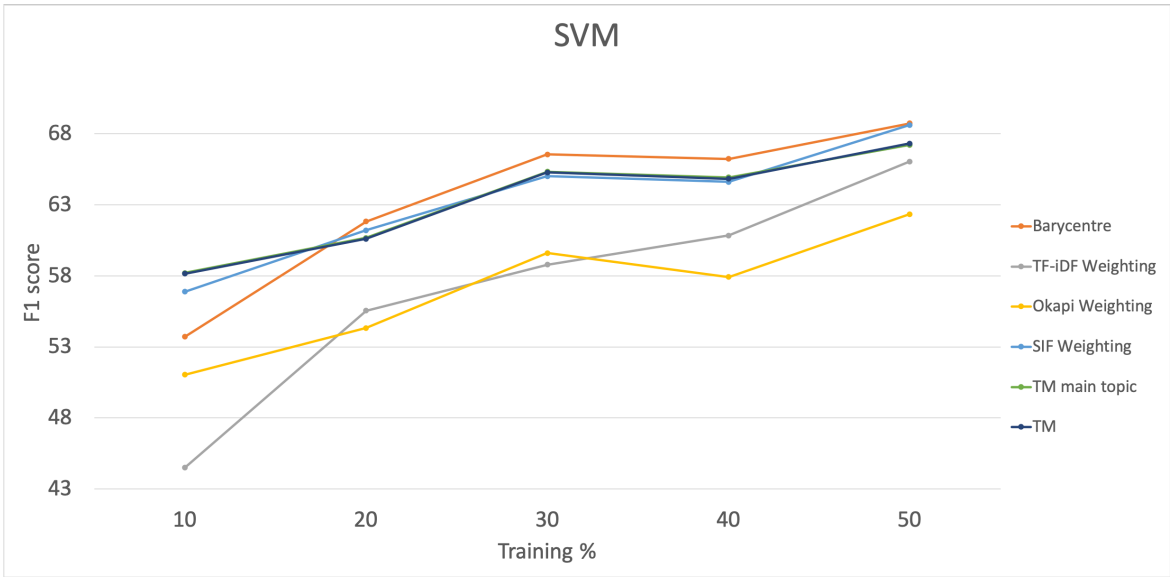

Figure 10: Graphique représentant les résultas en F1 score, obtenus par SVM en fonction du nombre de données pour l'entrainement (Sans apprentissage sur CORA)

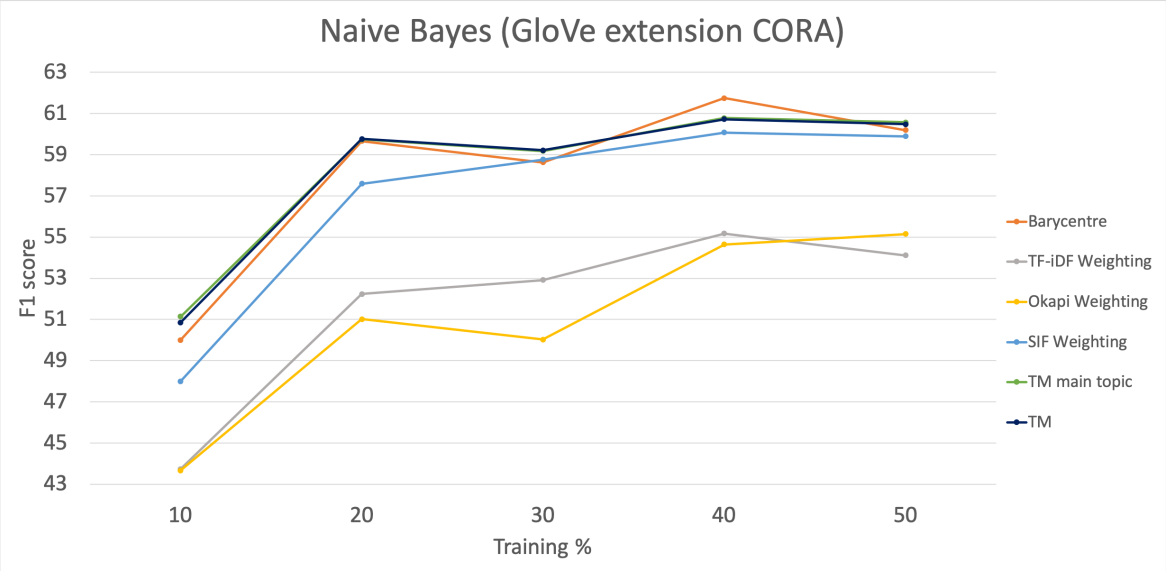

Figure 11: Graphique représentant les résultas en F1 score, obtenus par Naive Bayes en fonction du nombre de données pour l'entrainement (Avec apprentissage sur CORA)

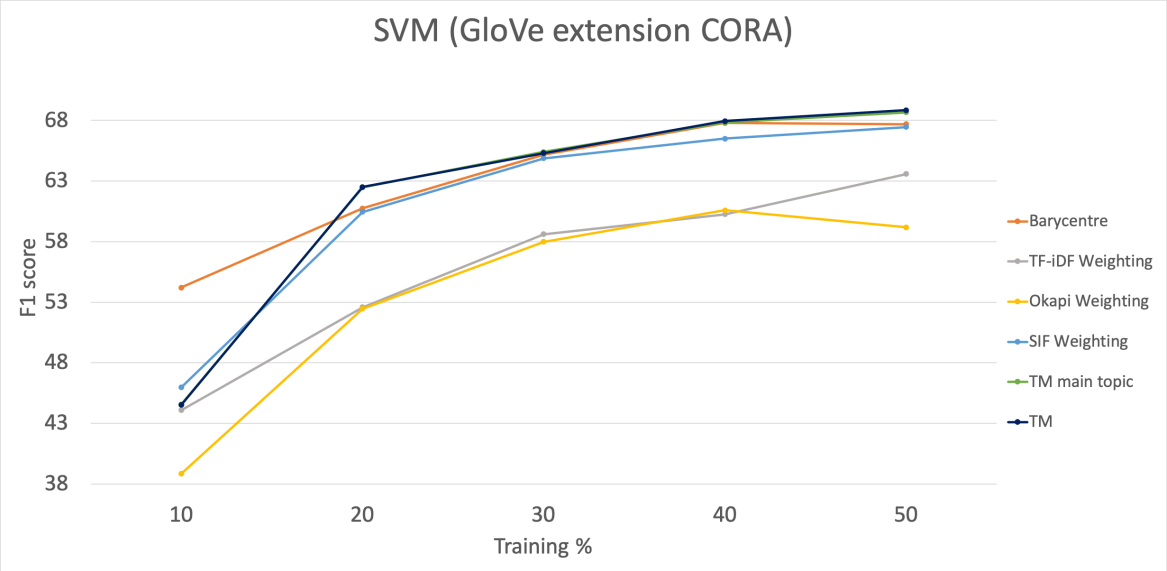

Figure 12: Graphique représentant les résultas en F1 score, obtenus par SVM en fonction du nombre de données pour l'entrainement (Avec apprentissage sur CORA)

Classifier	Measure	F1-score				
	Training%	10%	20%	30%	40%	50%
	DE method					
Naive Bayes GloVe	Barycentre	49.79	55.19	**59.93**	**58.16**	**57.26**
	TF-IDF Weighting	44.05	50.48	54.28	51.84	52.72
	OKAPI Weighting	42.14	48.93	50.91	50.98	54.28
	SIF Weighting	**50.86**	**55.81**	59.05	55.80	58.09
	TM main topic	50.06	55.10	58.18	56.81	57.00
	TM	49.82	54.89	58.28	56.92	57.08
SVM GloVe	Barycentre	53.71	**61.82**	**66.56**	**66.23**	**68.74**
	TF-IDF Weighting	44.49	55.54	58.79	60.84	66.06
	OKAPI Weighting	51.04	54.32	59.61	57.92	62.34
	SIF Weighting	56.89	61.21	65.02	64.61	68.63
	TM main topic	**58.20**	60.68	65.33	64.94	67.22
	TM	58.16	60.61	65.28	64.84	67.32
Naive Bayes GloVe extension (CORA)	Barycentre	50.00	59.66	58.63	**61.75**	60.20
	TF-IDF Weighting	43.74	52.24	52.92	55.17	54.11
	OKAPI Weighting	43.67	51.01	50.03	54.64	55.15
	SIF Weighting	48.00	57.59	58.77	60.08	59.90
	TM main topic	**51.14**	59.75	59.18	60.78	**60.57**
	TM	50.85	**59.77**	**59.22**	60.72	60.48
SVM GloVe extension (CORA)	Barycentre	**54.21**	60.75	65.17	67.79	67.70
	TF-IDF Weighting	44.09	52.58	58.62	60.26	63.58
	OKAPI Weighting	38.84	52.47	57.99	60.58	59.18
	SIF Weighting	45.97	60.46	64.88	66.50	67.47
	TM main topic	44.56	62.49	**65.41**	67.83	68.67
	TM	44.52	**62.50**	65.31	**67.96**	**68.86**

Table 1: Comparaison des résultats sur dataset CORA en testant les 4 baselines: Barycentre, TF-IDF, OKAPI, SIF et 2 méthodes proposées (TM main topic et TM). Nous comparons aussi les résultats entre GloVe normal (sans apprentissage sur CORA) et GloVe amélioré (avec apprentissage sur CORA) avec les 2 classifieurs: Naive Bayes et Machine à vecteur de support (SVM). Les résultats affichés pour les méthodes proposées sont obtenus pour un nombre de topic fixé à 10.

Pour les deux méthodes proposées, nous avons essayé 3 valeurs pour le nombre de topics (5, 10 et 15) lors du calcul des modèles thématiques. Parmi ces 3 valeurs, nous avons constaté que nous obtenions de meilleurs résultats avec 10 topics. Ce nombre peut être comparé avec le nombre de classes. On peut alors penser qu'il est nécessaire d'avoir un nombre de topic au moins supérieur au nombre de classes pour obtenir un effet significatif dans la tâche de classification.

Si on compare à présent les deux méthodes entre elles, on constate que la

deuxième proposition donne généralement de meilleurs résultats. Les résultats n'étant pas suffisamment significatifs, nous ne pouvons pas donner d'explication avec certitude. Cependant, on peut penser que les hypothèse de la deuxième proposition sont moins restrictives et permettent d'obtenir de meilleurs résultats.

Un autre problème qui n'a pas été adressé par les méthodes proposées concerne la polysémie des mots. Puisqu'on ne dispose pas d'un embedding de mot polysémique, un mot aura la même représentation quelque soit sa valeur sémantique. Ceci peut avoir pour effet d'induire en erreur l'embedding du document. Par exemple, si on rencontre le mot "Apple", on ne sait pas s'il s'agit du fruit ou de l'entreprise. Supposons que ce vecteur soit situé dans une zone de l'espace des "nouvelles technologies". Si le document parle davantage de nourriture et que ce vecteur est pondéré par un poids non négligeable, la qualité de représentation du vecteur document risque de diminuer. Or, l'utilisation du contexte doit pouvoir nous permettre de diminuer l'erreur d'ambiguïté que le vecteur d'embedding de mot ne peut pas gérer.

Dans de futurs travaux, il pourrait être intéressant de proposer une façon de prendre en compte la polysémie. Notamment, on pourrait utiliser la modélisation thématique d'une autre manière pour résoudre le problème de la polysémie. Si on prend l'exemple de Apple, on pourrait comparer (avec une mesure de distance par exemple) les vecteurs d'embeddings voisins (dans l'espace d'embedding des mots) avec les principaux mots des topics majoritaires. Si ces vecteurs sont fortement distants, alors l'embedding du mot ne correspond pas au topic (polysémie) et le poids qui lui est accordé devrait être faible pour minimiser l'erreur. Cette idée pourrait être évaluée dans de prochains travaux.

4 Conclusion

Dans ce rapport de travail d'étude et de recherche, nous avons réalisé un état de l'art sur les méthodes d'apprentissage de représentations de documents, sur les méthodes de pondérations pour la construction d'embedding de documents et sur la modélisation thématique. Dans nos contributions, nous avons proposé deux méthodes de pondérations basées sur la modélisation thématique et les avons évalué dans une tâche de classification. Nous avons montré que nos méthodes proposées pouvaient présenter un intérêt lorsque l'embedding de mot est ré-entrainé sur le corpus de travail. Nous avons également proposé une nouvelle piste à explorer pour tenir compte de la polysémie. Dans de futurs travaux, nous pensons qu'il pourrait également être intéressant de nous inspirer d'autres méthodes telles que LDA2vec [E Moody, 2016] et TADW [Yang et al., 2015].

References

[Deerwester et al., 1990] Deerwester, S., T. Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[E Moody, 2016] E Moody, C. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec.

[Felix Hill, 2016] Felix Hill, Kyunghyun Cho, A. K. (2016). Learning distributed representations of sentences from unlabelled data.

[Hofmann, 2013] Hofmann, T. (2013). Probabilistic latent semantic analysis.

[Jeffrey Pennington, 2014] Jeffrey Pennington, Richard Socher, C. M. (2014). Glove: Global vectors for word representation. pages 1532–1543.

[Lau and Baldwin, 2016] Lau, J. H. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Rep4NLP@ACL*.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. pages 1188–1196.

[Li and Mccallum, 2006] Li, W. and Mccallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. pages 577–584.

[M. Blei et al., 2001] M. Blei, D., Y. Ng, A., and Jordan, M. (2001). Latent dirichlet allocation. volume 3, pages 601–608.

[M Wallach, 2008] M Wallach, H. (2008). *Structured Topic Models for Language*. PhD thesis.

[Matteo Pagliardini, 2017] Matteo Pagliardini, Prakhar Gupta, M. J. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features.

[R. Kiros and Fidler, 2015] R. Kiros, Y. Zhu, R. S. R. S. Z. A. T. R. U. and Fidler, S. (2015). Skip-thought vectors.

[Sanjeev Arora, 2017] Sanjeev Arora, Yingyu Liang, T. M. (2017). A simple but tough-to-beat baseline for sentence embedding.

[Tomas Mikolov, 2013] Tomas Mikolov, Ilya Sutskever, K. C. G. S. C. J. D. (2013). Distributed representations of words and phrases and their compositionality. pages 3111–3119.

[Wenig, 2018] Wenig, P. (2018). *Creation of Sentence Embeddings Based on Topical Word Representations*. PhD thesis.

[Yang et al., 2015] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. (2015). Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2111–2117. AAAI Press.