Qualité des document embeddings

Mohamed BEN HAMDOUNE, Louis BOURQUARD, Lucas ISCOVICI

Paris, France

Université de Paris Descartes

Abstract

Le Text Mining est une branche du Data Mining qui se spécialise dans le traitement de corpus de textes pour en analyser le contenu puis en extraire des connaissances. Dans notre cas, l'objectif est d'évaluer la qualité d'un ensemble de matrices d'embeddings qui représentent des documents. On effectuera un clustering sur les documents et on comparera les valeurs de clustering avec différents critères d'évaluation.

Keywords: LSA, NMF, Doc2Vec, Indice De Rand, Information Mutuelle

Contents

1. Introduction

Les *paragraph vectors* (Doc2Vec) ont récemment été proposés comme méthode non supervisée pour l'apprentissage [2] de documents. Dans leurs travaux, les auteurs ont montré que la méthode peut apprendre à intégrer des textes de critiques de films pouvant être utilisés pour l'analyse des sentiments. Cette preuve de concept, tout en étant encourageante, était plutôt étroite.

En effet, le *word embedding* a ouvert de nouvelles perspectives à travers une réduction de dimensionnalité ainsi qu'une nouvelle vision des mots (une vision sémantique).
De la même manière le *document embedding* est une extension apportée lorsque le nombre de documents devient important et nous cherchons à avoir un résultat de similarité, puis pouvoir classifier et avoir une technique permettant de discrimer les documents (e.g., des livres de mécaniques et de botaniques).

Ici, nous considérons des tâches autres que l'analyse des sentiments, nous utiliserons deux jeux de données (Class3 et Reuteurs8) avec d'autres algorithmes d'analyse, tels que la NMF et la LSA afin d'évaluer la qualité de leurs *embeddings*. Ensuite nous évaluerons les performances afin de constater quels sont les points forts et faibles des différentes analyses.

1.1. Non-negative matrix factorization

La factorisation matricielle non négative est un modèle algébrique qui factorise les matrices de grande dimension en une représentation de faible dimensionnalité.
La NMF tire parti du fait que les matrices sont non négatives en les factorisant dans la forme de dimension inférieure puis la NMF force également les coefficients à être positifs.

Étant donné la matrice A originale, on peut obtenir deux matrices W et H telles que

$$A \approx W \times H$$

NMF a une propriété de compression, telle que W et H représentent les informations suivantes sur A: où W, H sont $n \times r$ et $r \times p$ matrices non-négatives respectivement.

- A (matrice de mots de documents) - entrée qui contient quels mots apparaissent dans quels documents.
- W (vecteurs de base) - les topics (clusters) découverts à partir des documents.
- H (matrice de coefficients) - la pondération des membres pour les sujets de chaque document.

Nous calculons W et H en optimisant une fonction objective, en mettant à jour les valeurs de W et H de manière itérative jusqu'à la convergence.

En pratique, le rang de factorisation r est souvent choisi de telle sorte que $r \ll \min(n, p)$. L'objectif derrière ce choix est de résumer et diviser l'information contenue dans X en facteurs r: les colonnes de W.

L'estimation se déroule de cette manière avec $F(W, H)$ comme fonction objective :

$$\underset{W,H\geq 0}{min} \underbrace{[D(X, WH) + R(W, H)]}_{=F(W,H)} \tag{1}$$

- D est une fonction coût qui mesure la qualité de l'approximation. Les fonctions de perte communes sont basées soit sur la distance de Frobenius

$$D : A, B \mapsto \frac{Tr(AB^t)}{2} = \frac{1}{2}\sum_{ij}(a_{ij} - b_{ij})^2,$$

  ou la divergence Kullback-Leibler.

$$D : A, B \mapsto KL(A||B) = \sum_{i,j} a_{ij} \log \frac{a_{ij}}{b_{ij}} - a_{ij} + b_{ij}.$$

- R est une fonction de régularisation facultative, définie pour appliquer les propriétés sur les matrices W et H, telle que la *smoothness* ou la *sparsité*.

Un paramétre critique dans la NMF est le rang de la factorisation r. Il définit le nombre de variables utilisées pour approcher la matrice cible. Étant données une méthode NMF et la matrice cible, une façon courante de décider de r est d'essayer différentes valeurs, de calculer une mesure de qualité des résultats et de choisir la meilleure valeur en fonction de ces critères de qualité.

Enfin un des plus grands points faibles de la NMF est son problème de convergence. Contrairement à la SVD, il n'y a pas de factorisation unique de la NMF. Les différents algorithmes de la NMF peuvent converger vers différents minima locaux (et même cette convergence vers des minima locaux n'est pas garantie), l'initialisation de l'algorithme devient critique. En pratique, la connaissance du domaine d'application peut aider à guider les choix d'initialisation.

1.2. Singular Value Decomposition

La décomposition en une seule valeur (SVD) est un concept d'algèbre linéaire basé sur l'équation matricielle suivante:

$$A = USV^\intercal \tag{2}$$

qui indique qu'une matrice $A^{m \times n}$ rectangulaire (ici on n'impose pas en entrée une matrice carrée) peut être décomposée en 3 autres composantes de la matrice avec $U^{m \times m}$, $S^{m \times n}$, $V^{\intercal n \times n}$:

- U est constituée des vecteurs propres orthonormés de $AA^\intercal$, où $U^\intercal U = I$ (rappel d'algèbre linéaire, les vecteurs orthogonaux de longueur unitaires sont dits orthonormés).
- V est constituée des vecteurs propres orthonormés de $A^\intercal A$.
- S est une matrice diagonale constituée de la racine carrée des valeurs propres de U ou de V (qui sont égales).

Les valeurs de S décrivent la variance des composantes linéairement indépendantes le long de chaque dimension, de la même manière que les valeurs propres décrivent la variance expliquée par des composantes dans l'analyse en composantes principales (ACP).

Dans le domaine du Text Mining, la SVD fournit le fondement mathématique des techniques d'extraction de texte et de classification généralement connues sous le nom d'indexation sémantique latente (LSI que l'on nommera aussi LSA dans le cadre de notre projet). La matrice A est généralement une matrice

$$A^{document \times mot}$$

C'est un moyen de représenter notre texte sous la forme d'un modèle d'espace vectoriel de grande dimension.

En ce qui concerne l'extraction de texte, on peut interpréter la SVD de la manière suivante:

- Les documents sont représentés sous forme de lignes dans V.
- La similarité des documents peut être déterminée en examinant les lignes dans VS.
- Les mots sont représentés sous forme de lignes dans U.
- La similarité des mots peut être déterminée en examinant les lignes de la matrice US.

1.3. Latent Semantic Analysis

L'analyse sémantique latente, ou LSA, est l'une des techniques fondamentales de la topic modeling. L'idée centrale est de prendre une matrice document-termes et de la décomposer en une matrice document-topic et une matrice topic-terme.

La première étape consiste à générer notre matrice document-termes. Avec m documents et n mots dans notre vocabulaire, nous pouvons construire une matrice :

$$A^{m \times n}$$

dans laquelle chaque ligne représente un document et chaque colonne représente un mot.
Dans la version la plus simple de LSA, chaque entrée peut simplement être un comptage simple du nombre d'occurrences du j-ème mot dans le i-ème document. En pratique, toutefois, le comptage simple ne fonctionne pas particulièrement bien car il ne rend pas compte de la signification de chaque mot du document. Par exemple, le mot "moteur" nous en informe probablement davantage sur le(s) topic(s) d'un document que le mot "test".

Par conséquent, les modèles LSA remplacent généralement le comptage simple (dans notre cas on utilise la fonction CountVectorizer implémenter dans la librairie Scikit-Learn [7]) de la matrice document-termes par un score TF-IDF.

1.3.1. Term Frequency-Inverse Document Frequency

TF-IDF, ou fréquence de document inverse de fréquence terme, attribue une pondération au terme j dans le document i comme suit:

$$w_{ij} = tf_{i,j} \times \log \frac{N}{df_i}$$

Intuitivement, plus le terme apparaît fréquemment dans le document, plus son poids est faible et plus il apparaît rarement dans le corpus, plus son poids est important.

Une fois que nous avons notre matrice document-termes A, nous pouvons commencer à réfléchir à nos sujets latents. Par conséquent, pour rechercher les quelques sujets latents qui capturent les relations entre les termes et les documents, dans une matrice contenant beaucoup de colonnes, nous effectuons une réduction de la dimensionnalité sur A.

Cette réduction de dimensionnalité peut être effectuée à l'aide d'une *SVD tronquée* (c'est à dire que ce n'est pas une décomposition exacte). La SVD tronquée réduit la dimensionnalité en ne sélectionnant que les valeurs singulières (valeurs propres) les plus grandes, et en ne conservant que les c premières colonnes de U et V.

Dans ce cas, c est un hyperparamètre que nous pouvons sélectionner et ajuster pour refléter le nombre de topic que nous souhaitons trouver.
Intuitivement, nous pouvons considérer cela comme une conservation des dimensions les plus significatives de notre espace transformé.
Dans ce cas, $U \in \mathbb{R}^{m \times c}$ apparaît comme notre matrice document-topic et $V \in \mathbb{R}^{n \times c}$ notre matrice terme-topic.
Dans les colonnes U et V, les colonnes correspondent à l'un de nos c topics.
En U, les lignes représentent les vecteurs de documents exprimés en termes de topics.
Ensuite pour V, les lignes représentent des vecteurs de termes exprimés en termes de topics. Avec ces vecteurs de documents et vecteurs de termes, nous pouvons maintenant facilement appliquer des mesures telles que la similarité de cosinus pour évaluer:

1. la similitude de différents documents.
2. la similitude de mots différents.
3. la similitude des termes (ou requêtes effectuées) et des documents (ce qui devient utile dans la recherche d'informations).

1.4. Paragraph Vectors (Doc2Vec)

Dans ce projet, nous passerons en revue la méthode Doc2Vec, un concept présenté en 2014 par Mikilov et Le [4] puis il convient aussi de mentionner que Mikilov est également l'un des auteurs de Word2Vec. Pour les tâches de similarité de phrase, les vecteurs Doc2Vec peuvent raisonnablement fonctionner et comme son nom l'indique il s'appuie fortement sur Word2Vec puis contrairement aux modèles de séquence tels que RNN, où la séquence de mots est capturée dans les vecteurs de phrases générées, les vecteurs de phrases Doc2Vec sont indépendants de l'ordre des mots.

Les auteurs ont constaté que non seulement il est plus efficace, mais qu'il produit en fait de meilleurs vecteurs de mots en moyenne [6] et Word2Vec a été proposé comme une approche efficace pour les réseaux de neurones à apprendre à intégrer des mots encodés à partir d'un grand vocabulaire [5], avec quelques spécificités à préciser.

Pour commencer, les noms propres où autres mots combiner comme par exemple "Tour Eiffel" ou "Le Point" ont des sens différents pris séparément ou globalement, alors pour palier à ce problème les auteurs ont décidé de les représenter dans un seul vecteur encodé (*word pairs*). Ensuite la même chose s'applique pour les mots de combinaison supérieure à 2 (*Learning Phrases*), ce sont les détections de mots agrandissant la complexité de l'apprentissage et réduisant la précision comme par exemple "Université Paris Descartes". Ecrit de cette façon, il pourrait être représenté comme la combinaison de (Université, Paris-Descartes).

Maintenant, imaginons que nous avons 500 *features* et un vocabulaire de 100 000 mots.
Le réseau de neurones ayant deux matrices de pondération : une couche cachée et une couche de sortie. Ces deux couches auraient une matrice de poids de 500 x 10 0000 = 50 millions de poids.
Pour pallier à ce problème, le *negative sampling* a été introduite comme une alternative au Softmax (plus complexe) au niveau de la couche de sortie. Le problème avec Softmax est que le gradient est dépendant du résultat de la somme à travers toutes les classes. C'est à dire de tous les mots à travers le vocabulaire, et nous aurons un problème de complexité très lourd, donc cela

ralentira l'entraînement.

Pendant la phase d'apprentissage sur la paire de mots ("rédaction", "longue"), le label ou valeur correcte en sortie du réseau est un vecteur one-hot. C'est-à-dire que la neurone de sortie correspondant à «longue» produit un 1 et que tous les autres milliers de neurones de sortie génèrent un 0.
Avec un échantillonnage négatif, nous allons sélectionner au hasard un petit nombre de mots «négatifs» pour mettre à jour les poids. (Dans ce contexte, un mot «négatif» est un mot pour lequel nous voulons que le réseau fournisse un 0). Nous mettrons également à jour les poids pour notre mot «positif» (qui est le mot «longue» dans notre exemple actuel).

Les auteurs indiquent que la sélection de 5 à 20 mots convient parfaitement aux petits jeux de données et que la sélection de 2 à 5 mots sur un grand jeu de données rend l'apprentissage pratiquement infaisable.

Il existe deux approches au Word2Vec :

1. Skip-gram (S-G)
2. Continious Bag of Words (CBOW)

Le modèle S-G fonctionne de la manière suivante : nous utilisons un mot pour prédire tous les mots environnants («contexte»). La méthode S-G est beaucoup plus lente que CBOW, mais considérée comme plus précise avec un jeu de données ayant des problèmes de mots peu fréquents.

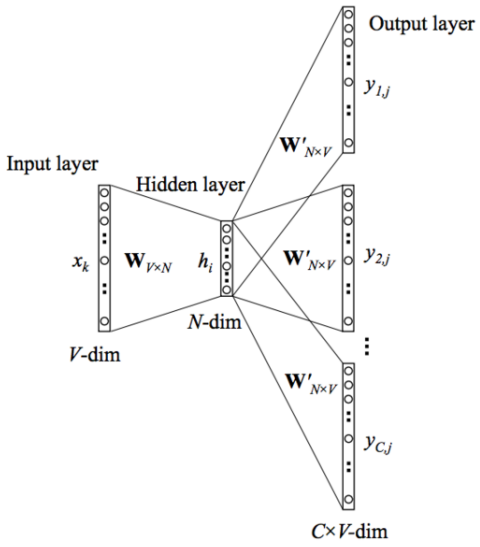

Figure 1: Réseau de neurones Skip-Gram

La fonction objective est de maximiser la probabilité du mot dans son contexte (on parle de *word context* dans la littérature anglaise) w_0 suivant le(s) mot(s) en entrée w_1, c'est à dire $P(w_0|w_1)$.
Soit un mot au milieu d'une phrase (mot d'entrée) noté w_i. Le réseau va nous indiquer la probabilité pour chaque mot de notre vocabulaire d'être le «mot voisin» que nous avons choisi (ce paramètre se retrouve implémenté généralement sous le nom de *window size*).
Exemple : "Il fait beau" avec une fenêtre de 1 produira comme échantillon avec en mot d'entrée "Il" : (Il,fait), ensuite pour "fait" : ((fait,Il),(fait,beau)) puis finalement pour le mot "beau" : (beau,fait).

Le modèle CBOW est une méthode qui prend le contexte de chaque mot comme entrée et tente de prédire le mot correspondant au contexte. Prenons notre exemple: Il fait beau aujourd'hui.

Soit "beau", le mot d'entrée au réseau. Ici le but est de prédire un mot target ("aujourd'hui") en utilisant un seul word context ("beau"). Plus spécifiquement, nous utilisons le vecteur one hot en entrée et mesurons l'erreur de sortie par rapport au vecteur en target ("aujourd'hui"). Au cours du processus de prédiction du mot en target, nous apprenons la représentation vectorielle du mot en target.

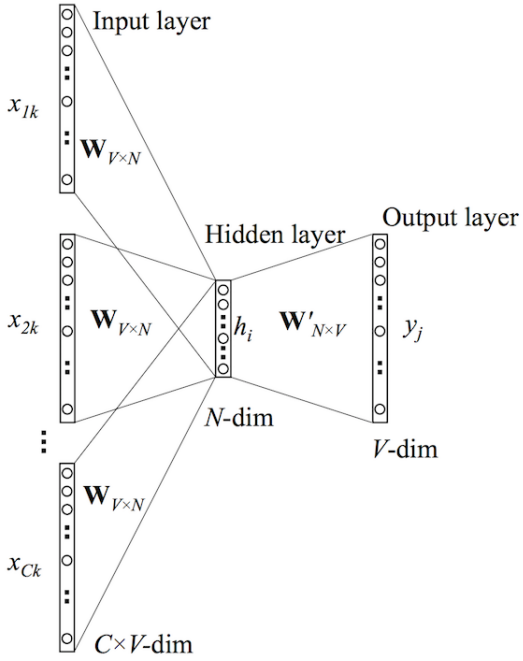

Figure 2: Réseau de neurones CBOW

Le modèle ci-dessus prend C *word context*. Lorsque $W_{V\times N}$ est utilisé pour calculer les entrées de couche masquées, nous prenons une moyenne pour toutes ces entrées C.

Le Doc2Vec étant une extension de Word2Vec, nous nous retrouvons avec deux façons de procéder qui sont *PV-DBOW* (Distributed Bag of Words version of Paragraph Vector, une extension du Skip-Gram) et aussi *PV-DM* (Distributed Memory version of Paragraph Vector) de la même manière une extension de CBOW. Nous utiliserons la librairie gemsim [8] pour réaliser le Doc2Vec.

1.5. K-means et Spherical K-means

Les deux méthodes de clustering que nous utiliserons seront *K-means* et *Spherical K-means*.
K-means est une méthode de partitionnement de données qui, à partir d'un ensemble de points, va pouvoir déterminer pour un nombre de classes fixé une répartition de points qui minimise un critère appelé *inertie* ou variance *intra-classe*.
Plus formellement étant donnée k, nous allons donc chercher à répartir les points $x_1, x_2, .., x_n$ en k groupes $C_1, C_2, \ldots, C_K$ de telle sorte que :

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{x \in C_k} ||x - (\frac{1}{|C_k|} \sum_{x \in C_k} x)||^2$$

soit la plus minimale.

En 2001, Dhillon et Modha ont introduit la méthode [3] Spherical K-Means pour regrouper de grands ensembles de données textuelles éparses.
Cette méthode a été établie en tirant parti de certains pré-traitements souvent effectués lorsque l'on travaille avec du texte dans un modèle d'espace vectoriel. Spherical K-means, contrairement à K-means qui cherche à minimiser une distance euclidienne, va plutôt définir le centre de chaque groupe de sorte à uniformiser et minimiser l'angle entre les composantes (en utilisant la similarité cosinus (ou mesure cosinus) comme métrique).
Notre objectif est d'avoir des mots avec un contexte similaire occupant des positions spatiales proches. Mathématiquement, le cosinus de l'angle entre ces vecteurs doit être proche de 1, c'est-à-dire un angle proche de 0.

2. Évaluation du clustering

2.1. Précision

Le mot "Accuracy" est ce que nous entendons habituellement lorsque nous utilisons le terme précision. Il désigne le rapport entre le nombre de prédictions correctes et le nombre total d'échantillons d'entrée. Soit A la classe définissant un terme appartenant au document. De suite, nous définissons B, la classe définissant un terme n'appartenant pas au document. Il s'ensuit 4 cas différents:

- TP (True Positive) : Le nombre de A correctement identifié.
- FP (False Positive) : Le nombre de A incorrectement identifié.
- TN (True Negative) : Le nombre de B correctement identifié.
- FN (False negative) : Le nombre de B incorrectement identifié.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

2.2. Information mutuelle

L'information mutuelle (MI) de deux variables aléatoires est une mesure de la dépendance mutuelle entre les deux variables. Plus précisément, elle quantifie la "quantité d'informations" (en bits) obtenue à propos d'une variable aléatoire en observant l'autre variable aléatoire. Le minimum est de 0 si le clustering est aléatoire par rapport à l'appartenance à une classe. Dans ce cas, le fait de savoir qu'un document se trouve dans un groupe particulier ne nous fournit aucune information nouvelle sur sa classe. Un maximum d'informations mutuelles est atteint pour un cluster qui recrée parfaitement les classes, mais également si les clusters sont subdivisés en clusters plus petits. Nous utilisons une version normalisée, c'est à dire que les valeurs sont comprises entre 0 et 1 d'où le nom de NMI (Normalized Mutual Information).

2.3. Indice de Rand

On appelle *indice de Rand* la proportion de paires d'observations qui sont soit de même classe et dans le même cluster, soit de classe différente et dans deux clusters différents. Nous utilisons une version ajustée, c'est à dire que les valeurs sont comprises entre 0 et 1 d'où le nom de ARI (Adjusted Rand Index).

3. Jeux de données

3.1. Classic3

Un ensemble de données de référence bien connu utilisé dans l'exploration de texte est la collection Classic, qui peut être obtenue à partir du site internet de l'Université de Cornell. Cet ensemble de données comprend 4 collections de documents différentes: CACM, CISI, CRAN et MED. Ces collections peuvent être téléchargées sous la forme d'un fichier par collection. Dans notre projet nous utilisons seulement 3 classes, c'est un corpus de 3891 documents appartenant à 3 différentes collections de documents: CISI, CRAN et MED.

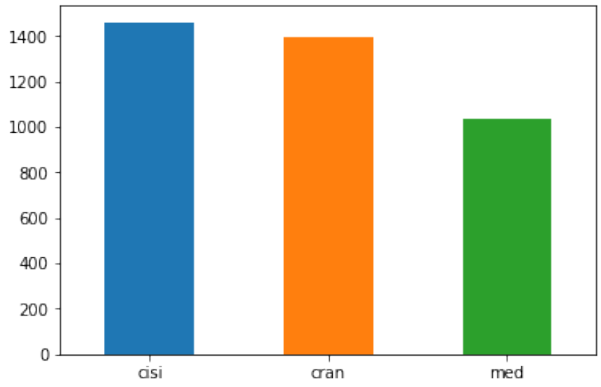

Figure 3: Répartition des classes sur Classic3

Nous avons 3891 documents pour un total de 21469 mots, et une matrice sparse à 99.7325%.

D'abord, nous regardons la proportion de chaque classe afin de vérifier si il existe un disproportion. Le pourcentage respectif de chaque classe est de CISI (sujets autour autour de la science) avec 37,52% ensuite CRAN (sujets autour de l'aérodynamique) avec 35,92%, et MED (sujets autour de la médecine) avec 26,54%.

Il n'y pas de grande disproportion à noter parmi les classes, nous allons vérifier les 5 mots les plus fréquents dans tout le dataset et ensuite regarder à quelles classes elles appartiennent.

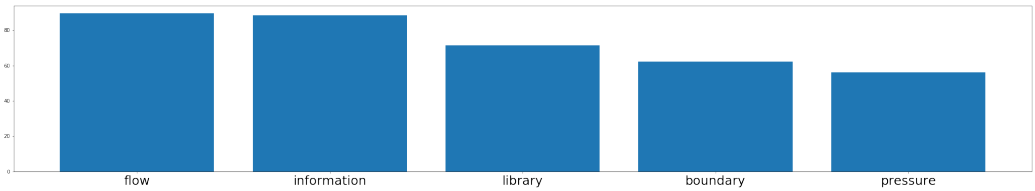

Figure 4: Fréquence des 5 premiers mots sur Classic3

Il est intéressant de voir que ce sont des mots autour de la classe CISI représentant 37,52% qui sont finalement dans le top 5 des mots les plus fréquent. Nos résultats sont issus d'une TD-IDF, ce qui signifie finalement que dans sa globalité les termes appartenant à la plus grande classe apparaissent rarement.

3.2. Reuters8

Reuters [9] est un ensemble de données de référence pour la classification de documents. Il compte 90 classes, 10 788 documents.
Dans notre cas nous avons 8 classes qui sont très disportitionnés, nous pouvons affirmer que classe *ear* a une majorité avec une présence de 51.77% puis en deuxième position *acq* avec 29.09%.

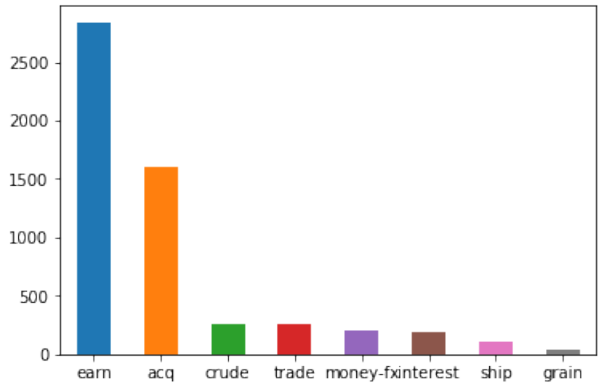

Figure 5: Répartition des classes sur Reuters8

Nous avons 5485 documents pour un total de 19759 mots, et une matrice

sparse à 99.7914%.
Dans sa globalité, voici les 5 premiers mots venant d'une TD-IDF :

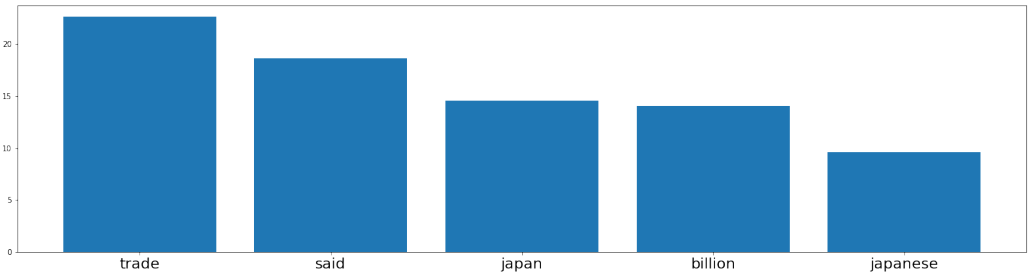

Figure 6: Fréquence des 5 premiers mots sur Reuters8

Nous avons effectué une vérification des 5 premiers mots pour les 8 classes afin de vérifier d'où proviennent principalement ces 5 mots. Les mots trade, japan, billion, japanese appartiennent à la classe *trade* qui a un pourcentage de 4.57% sur l'ensemble du jeux de données.
Le deuxième mot *said* provient principalement de la classe *ship* sachant qu'elle a un poids de 1.96% mais ce mot apparait aussi dans le top 5 pour les *acq, crude, grain, interest, money-fx, trade* mais il n'apparait pas finalement dans la classe la plus imposante (earn).

Nous sommes ici face à un jeu de données plus complexes de par sa répartition des classes, ainsi que de ses fréquences des mots dans chaque classe.

4. Résultats

Nous utiliserons Spherical K-Means et K-means afin d'effectuer le clustering sur les nouvelles représentations des données issues de la LSA, NMF et Doc2Vec. Afin d'attribuer un poids égal à chacun des n points d'un ensemble de données, les vecteurs de colonnes $x_1, x_2, .., x_n$ de la matrice de données $M \times N$ originale X sont normalisés pour être de longueur unitaire dans la norme euclidienne. Pour Spherical K-means la conséquence est de ne prendre en compte que la direction de chaque vecteur et non la longueur, puis pour les deux méthodes éviter le poids de la variation de la variance.

Ensuite nous utiliserons un comptage simple mais aussi un TD-IDF pour la LSA et NMF. En ce qui concerne Doc2Vec, nous allons effectuer un tag sur les différents documents.

4.1. 3 clusters sur Classic3

Tout d'abord, nous allons réaliser le clustering sur les matrices embeddings suivants les dimensions suivantes : (25,50,150,200). Pour éviter de surcharger le rapport, nous avons mis dans le tableau les résultats de K-means, mais comme expliqué précédemment, les résultats sont inférieurs à Spherical K-means.

Dimension	ARI	NMI	Accuracy
25	4,09%	3,66%	43,45%
50	12,63%	10,57%	56,82%
150	16,58%	13,78%	59,90%
200	**20,70%**	**18,65%**	**61,75%**

Malgré une très bonne précision, la qualité du clustering reste inférieure puisque les valeurs de la NMI et ARI. On compare les résultats obtenus à partir de la LSA avec une TD-IDF en entrée (ARI: 96.75% NMI: 94.04% Accuracy: 98.92%) qui sont meilleurs en tout points.

Ensuite pour la NMF, nous effectuons une initialisation *NNDSVD* puisque les avantages de la SVD (Singular Value Decomposition) comprennent une propriété d'optimalité avec le fait que la SVD tronquée produit la meilleure approximation k (en termes de distances au carré), car le calcul est rapide et robuste. De plus, l'unicité de la factorisation car l'initialisation n'affecte pas les algorithmes SVD [1].
La décomposition en valeur singulière double non négative ($NNDSVD$) tente d'ajuster la SVD d'une manière légèrement différente en produisant directement une décomposition non négative. Les variables W et H sont toutes deux positives, mais elles sont généralement éparses, elles sont donc modifiées en ajoutant ϵ plus petit que la moyenne des éléments aux entrées nulles. Dans le cas symétrique, cette méthode peut être ajustée pour obtenir $W = H$, et elle gagne en coût de calcul, puisque la SVD peut être remplacée par un processus de diagonalisation. Cette méthode d'initialisation n'a pas de partie aléatoire, donc un algorithme NMF ne produira qu'un seul minimum local.

Les meilleurs résultats obtenus avec la NMF ont été possible avec l'utilisation de la TD-IDF en entrée mais sans effectuer une normalisation L2 avant d'appliquer le Spherical K-Means (ARI: 90,23%, NMI: 84,70% Accuracy: 96,65%). Ces résultats restent inférieur à la LSA.

En ce qui concerna le Doc2Vec, nous avons utiliser un nombre d'epochs à 100 avec un vecteur de features égale à 20 puis un pas d'apprentissage fixé à 0.025 et décroit chaque epoch de 0.0002 (un minimum fixé à 0.00025). L'algortihme pour le Doc2vec est celui du PV-DM.
En pratique, les auteurs montrent que le modèle Skip-Gram/DM fonctionne mieux si nous avons beaucoup de données et que le modèle CBOW / DBOW fonctionne mieux avec moins.

La précision pour Doc2Vec est de 99,69% puis ARI: 99.03% et NMI: 98.19%.

Finalement, le grand vainqueur sur le jeux de données Classic3 est Doc2Vec grâce à K-Means sans normalisation et il est intéressant de noter que le clustering n'est pas affecté suivant si on met en entrée les données avec une normalisation L2 pour Spherical K-Means et K-Means.

4.2. 8 clusters sur Reuteurs8

La matrice embeddings de Reuteurs8 contient 200 colonnes, nous commençons à effectuer un clustering simple sur ce jeux de données : ARI: 7.51% NMI: 21.02% Accuracy: 29.60%. Le clustering sur les matrices d'embedding nous donne de faibles résultat, cela reste sans surprise en soi dû à la complexité du jeu de données comme on a pu le constater.

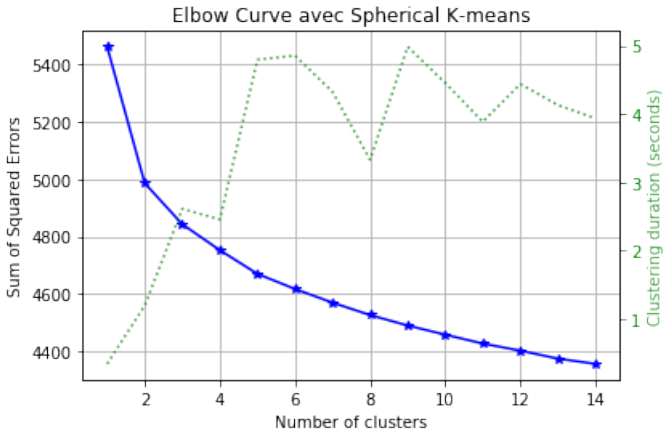

Figure 7: Critère du coude

La courbe nous indique que le nombre a prendre est 2, ce qui est loin d'être le compte puisqu'il y a 8 classes. Afin de comprendre la compléxité du jeu de données, nous effectuons un *world cloud* sur les 100 premiers.

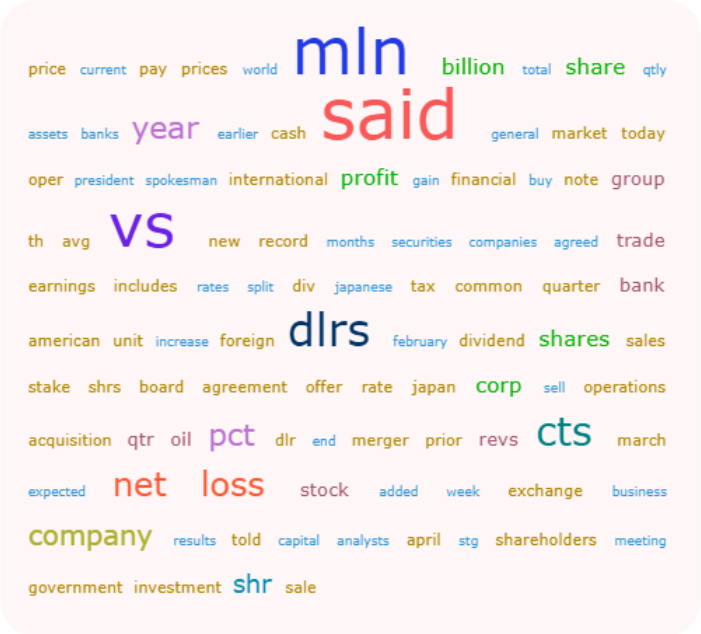

Figure 8: World Cloud sur Reuters8

Nous constatons qu'il n'y a beaucoup de mot avec une faible proportion en soi des mais ils sont équivalents entre eux. De plus, le vocabulaire reste le même entre, ce qui rend la tâche du clustering à partir de la matrice embeddings très complexe.

La LSA nous donne de meilleurs résultats avec CountVectorizer que la TD-IDF (toujours avec comme méthode de clustering la Spherical K-means) un ARI fixé à 32,94% avec une NMI à 44,51% puis une Accuracy de 49,11%. Cela s'explique par le fréquence des mots vu lors de l'analyse exploratoire.
Ensuite, la NMF indique un meilleur clustering cette fois tout en utilisant comme matrice un CountVectorizer : ARI à 34,97%, et une NMI: 47,19%, Accuracy: 50,10%.
La NMF obtient de meilleurs résultat que la LSA dans notre cas car elle tient compte de la pondération de chaque membre lors de sa décomposition dans la matrice H qu'elle essaye d'optimiser à chaque itération jusqu'à la convergence.
Ensuite nous avons deux résultats intéressant à analyser avec Doc2Vec.

Méthode	ARI	NMI	Accuracy
Spherical K-means	55,52%	75,67	61,16%
K-means	**80,42%**	**78,24%**	**76,73%**

K-means (on a normalisé la matrice embeddings de Doc2Vec) possède une meilleure précision, ce qui est une nette différence avec une meilleur l'ARI ce qui signifie une meilleur homogénéité des clusters.
Nous utilisons l'algortihme PV-DBOW car notre jeu de données contient peu de donnéees et il est disproportionnée assez de données et que nous avons constaté une grosse chute de performance avec PV-DM.

5. Conclusion

Dans le cadre de ce projet nous avons beaucoup appris sur les méthodes tels que la LSA, NMF ainsi que Doc2Vec.
Chaque méthode vue lors de ces projets présentent des points forts et faibles comme par exemple la rapidité de la LSA fasse à Doc2Vec et la NMF, ou bien la capacité à la NMF d'avoir une décomposition prenant en compte le poids de chaque cluster et ainsi d'être plus facilement interprétable suivant chaque colonne de la matrice W puis la LSA avait une capacité a récuperer les différents sens des mots.

Ensuite nous avons constaté sans réel surprise que Doc2Vec a été la méthode apportant les meilleurs résultats sachant que nous pouvons charger un modèle pré-entrainé sur un très grand jeu de données pour pouvoir transférer un savoir (en d'autres termes initialiser les poids du réseau de neurones), ce qui nous laisse penser que c'est une méthode très prometteuse.
Doc2Vec apparait comme une méthode idéale en soi, même si ce sont des jeux de données classiques, on a pu voir que les défauts de Reters8 ont pu être dépasser en modifiant l'algorithme grâce aux deux approches de cette méthode.

Ce projet nous a permis d'utiliser plusieurs bibliothèque python populaire, avec leurs lots de variétés et de pouvoir analyser et modéliser nos jeux de données ainsi que l'apprentissage de plusieurs manières.
Nous aurions pu améliorer les résultats sur les jeux de données en effectuant un pre-processing plus poussé, mais le but de ce projet était plutôt d'évaluer la qualité de manière général sans être spécifique suivant le contexte des données.

References

[1] Boutsidis, C., Gallopoulos, E., Apr. 2008. Svd based initialization: A head start for nonnegative matrix factorization. Pattern Recogn. 41 (4), 1350–1362.
URL http://dx.doi.org/10.1016/j.patcog.2007.09.010

[2] Dai, A. M., Olah, C., Le, Q. V., 2015. Document embedding with paragraph vectors. In: NIPS Deep Learning Workshop.

[3] Hornik, K., Feinerer, I., Kober, M., Buchta, C., 2012. Spherical k-means clustering. Journal of Statistical Software, Articles 50 (10), 1–22.
URL https://www.jstatsoft.org/v050/i10

[4] Le, Q. V., Mikolov, T., 2014. Distributed representations of sentences and documents. CoRR abs/1405.4053.
URL http://arxiv.org/abs/1405.4053

[5] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. CoRR abs/1301.3781.
URL http://arxiv.org/abs/1301.3781

[6] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. CoRR abs/1310.4546.
URL http://arxiv.org/abs/1310.4546

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[8] Řehůřek, R., Sojka, P., May 2010. Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. ELRA, Valletta, Malta, pp. 45–50, http://is.muni.cz/publication/884893/en.

[9] Thoma, M., Jul. 2017. The reuters dataset.
URL https://martin-thoma.com/nlp-reuters