Utilité d'un couplage entre Word2Vec et une analyse sémantique latente : expérimentation en catégorisation de données textuelles.

Oussama Ahmia*, Nicolas Béchet*, Pierre-François Marteau*, Alexandre Garel**

* IRISA, Université Bretagne Sud,
Rue Yves mainguy BP 573 56000 VANNES cedex
nom.prénom@irisa.fr, http ://www-expression.irisa.fr
** 2 Place Saint-Pierre, 44000 Nantes
a.garel@octopusmind.info, http ://www.octopusmind.info

Résumé. Nous réexaminons dans cet article les méthodes de vectorisation de textes dans le cadre d'une étude de classification de documents. Nous étudions les méthodes basées sur des plongements de mots (word2vec) ou de documents (analyse sémantique latente, ou sac de mots associées à diverses pondérations) ainsi que certaines combinaisons de ces méthodes. A cette fin, nous évaluons ces méthodes de vectorisation en utilisant trois modèles de classification (un perceptron multicouches, une machine linéaire à vecteurs supports optimisée par descente de gradient stochastique et un classifieur multinomial naïf de Bayes). Nos résultats montrent que le modèle proposé pour associer les méthodes word2vec et LSA, qui conjugue les deux caractérisations complémentaires du contexte d'occurrence des mots (local pour word2vec et global pour LSA), permet de produire une vectorisation robuste, en général plus discriminante que les autres approches testées.

1 Introduction

Avec la croissance rapide de l'information en ligne, la nécessité de développer des méthodes pour trouver, filtrer et gérer ces ressources de manière rapide et efficace devient d'autant plus prégnante. La classification de données textuelles consiste à classer de façon automatique des textes dans une ou plusieurs catégories. Cette dernière a déjà été appliquée à plusieurs problématiques notamment l'extraction d'information (Kushmerick et al., 2001), l'analyse de sentiments (Dey et Haque, 2009), la détection de SPAM (Jindal et Liu, 2007), etc.

Afin d'exploiter des algorithmes d'apprentissage automatique sur des données textuelles, il est souvent nécessaire de représenter le texte sous la forme d'un vecteur de taille fixe, ceci afin de plonger la donnée dans un espace métrique.

De nombreuse méthodes de "vectorisation" ont été développées au fil des années. La plus utilisée étant la méthode dite *sac de mots* (bag of words) (Harris, 1954) qui consiste à décrire un texte par les occurrences (fréquences) des mots qui le composent. Cette méthode considère que tous les mots dans un document donné ont le même poids ce qui est problématique pour

les mots peu discriminants. Dans le contexte d'une classification, afin de pallier ce problème, Karen Spärck Jones a introduit la notion de pondération *tf-idf* (fréquence du terme dans le document-proportion inverse de documents qui contiennent le mot) (Jones, 1972) qui consiste à donner plus de poids aux mots qui apparaissent dans moins de documents. Ainsi, si deux mots m_1 et m_2 ont une même fréquence dans un document, si $m1$ n'est présent que dans ce document, il aura un poids plus important que m_2. Toutefois, ces représentations engendrent des plongements dans des espaces vectoriels de très grande dimension et ne prennent pas en considération la sémantique des mots. Cela signifie par exemple que les mots "se nourrir", "manger" et "conduire" sont équidistants dans cet espace vectoriel malgré le fait que sémantiquement, "manger" devrait être plus proche de "se nourrir" que de "conduire".

Partant de l'hypothèse que les mots utilisés dans les mêmes contextes ont tendance à avoir une signification (sens) similaire, Scott Deerwester et al. ont introduit la notion d'analyse sémantique latente (LSA) (Landauer et al., 1998). LSA est une méthode statistique qui permet d'extraire des motifs dans les relations entre les termes et les concepts dans des documents en appliquant une décomposition en valeurs singulières (SVD) sur une matrice termes-documents.

Chaque composante (vecteur propre) générée représente un concept exprimé sous la forme d'une combinaison linéaire des vecteurs associés aux termes, ce qui permet de réduire grandement la dimensionnalité.

Avec l'arrivée du modèle word2vec, proposé par Mikolov (Mikolov et al., 2013), de nouvelles opportunités sont apparues dans le domaine de la vectorisation des termes. Il existe deux architectures de modèle word2vec : la première, appelée CBOW prédit un mot en fonction de la connaissance seule de son contexte local d'occurrence (les mots qui l'entourent), et la seconde architecture, appelée skip-gram, prédit le contexte local d'occurrence d'un mot en fonction du mot lui-même.

word2vec permet de capturer une certaine sémantique lexicale, dans la mesure où les mots ayant une signification voisine auront une représentation similaire, plus précisément, ils seront proches dans l'espace vectoriel dans lequel ils seront plongés. Par exemple, "se nourrir" et "manger" sont proches, alors que "manger" et "conduire" sont plus distants. Le résultat de la soustraction ou l'addition de vecteurs de mots porte également une signification. Par exemple, les vecteurs de mots peuvent être utilisés pour retrouver des relations d'analogie entre les termes à l'aide d'une algèbre linéaire simple : "Roi" - "homme" + "femme" = "Reine" (Mikolov et al., 2013).

Nous présentons dans cet article, une méthode efficace et peu coûteuse de vectorisation, basée sur word2vec et LSA à des fins de catégorisation de documents. Elle assure la capture de la sémantique d'un document, en préservant au mieux l'intégralité des mots contenus dans le texte.

Notre méthode consiste, dans un premier temps à vectoriser un document en considérant la moyenne des vecteurs word2vec des mots qui le composent, puis, dans un deuxième temps à combiner cette vectorisation word2vec à une vectorisation obtenue par LSA pour ce même document. La combinaison proprement dite est le résultat de la concaténation des deux vectorisations précédentes.

2 Etat de l'art

De nombreux travaux ont expérimenté une combinaison de word2vec avec différentes variantes de vectorisation. Basé sur word2vec, le modèle doc2vec (Le et Mikolov, 2014) propose une représentation vectorielle au niveau du document. Le principe de doc2vec consiste à représenter chaque paragraphe et chaque mot par un vecteur de base unitaire (one hot vector). Le vecteur de paragraphe et les vecteurs de mots sont moyennés ou concaténés. Le vecteur résultant est utilisé pour prédire le mot suivant dans un contexte donné. Néanmoins cette approche nécessite des données peu bruitées, ainsi qu'un grand nombre d'itérations pour converger. Sa complexité dépend de la taille du vocabulaire (Chen, 2017). Ronghui Ju et al (Ju et al., 2015) ont développé un modèle qui combine i) une représentation basée sur LSA, ii) un sac-de-mot associé à la pondération tf-idf et iii) une vectorisation word2vec. Le principe consiste à créer une matrice tridimensionnelle $(m * n * v)$ où m est le nombre de mots, n le nombre de documents et v la dimension du vecteur word2vec. Dans ce tenseur tridimensionnel chaque document est représenté par une matrice, chaque ligne est le produit du sac-de-mot pondéré par tf-idf et de la représentation word2vec des termes. A l'issue de la création de cette matrice qui associe les termes aux documents, une décomposition en valeurs singulières (SVD) est appliquée sur chacune des matrices [termes x documents]. Le résultat est ensuite exploité pour entraîner un réseau neuronal convolutif (CNN) (LeCun et al., 1999). LDA2vec est également une approche combinant word2vec avec cette fois ci LDA (Allocation de Dirichlet latente). C'est un modèle développé par Christopher Moody (Moody, 2016) qui est inspiré de la variante skip-gram de word2vec. Il combine une analyse LDA (Blei et al., 2003) et une représentation word2vec. Le principe de cette méthode est de projeter les vecteurs word2vec des termes et les vecteurs de documents dans un même espace vectoriel. Cela consiste, d'une part, à représenter un document par une somme pondérée des vecteurs des thèmes. Celle-ci est obtenue en multipliant les "topic-vectors" générés par LDA et le vecteur de distribution de thèmes. Le résultat est ensuite combiné avec la représentation word2vec d'un mot pivot afin de générer un vecteur de contexte, qui sera utilisé pour prédire le contexte de ce mot pivot.

3 Modèles utilisés

3.1 Word2vec

Word2vec (Mikolov et al., 2013) constitue une famille de modèles de plongement lexical (word embedding) permettant de créer des représentations vectorielles de mots à partir de grands corpus. Word2vec peut être utilisé via deux architectures différentes : CBOW (sac de mots continus) et Skip-gram (saut de gramme). CBOW construit la représentation vectorielle d'un mot via la prédiction de son occurrence à partir de la connaissance des mots avoisinants. La notion de sac de mots continus implique que l'ordre des mots n'est pas pris en compte. Par ailleurs, la deuxième architecture, Skip-gram, construit la représentation vectorielle d'un mot via la prédiction de son contexte d'occurrence. Les mots avoisinants les plus proches auront un poids plus important par rapport aux plus éloignés (c'est la notion de skip-gram). Pour les deux architectures, ces prédictions sont réalisées à l'aide d'un réseau de neurones à propagation avant (Zell, 1994) qui ne contient que 3 couches (couche d'entrée, couche cachée, couche de sortie) ce qui permet une exécution rapide.

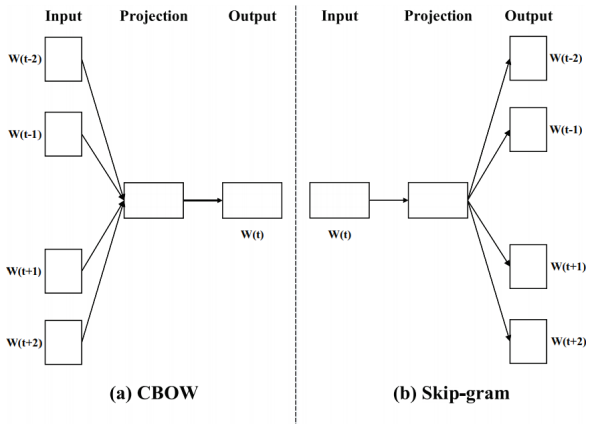

FIG. 1 – *Les deux architectures de word2vec.*

Pour les deux architectures, le nombre de mots avoisinants (ou la fenêtre) est défini par l'utilisateur. On désigne les mots à entraîner par $w_t, t \in \{1 \cdots T\}$. Dans la suite, nous nous focalisons sur l'architecture Skip-gram, que nous avions choisi pour nos expériences. L'objectif du Skip-gram est de maximiser la moyenne des log-probalités :

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\, p(w_{t+j}|w_t)$$

Où c est la taille du contexte. Une trop grande valeur de c augmente le temps d'entraînement.

La formulation de Skip-gram définit $p(w_{t+i}|wt)$ en utilisant la fonction *softmax*ă:

$$p(w_e|w_s) = \frac{exp({v'_{w_s}}^{\top} v_{w_e})}{\sum_{w=1}^{W} exp({v'_{w_s}}^{\top} v_{w_e})}$$

Où v et v' sont les représentations vectorielles "entrée" et "sortie" du mot w et w_e/w_s sont les mots entrée/sortie.

3.2 Analyse sémantique latente

L'analyse sémantique latente (LSA) est une méthode utilisée en traitement du langage naturel pour caractériser une forme de sémantique lexicale dite distributionnelle. Elle est basée sur la décomposition en valeurs singulières de la matrice [termes x documents] qui aboutit à la factorisation de cette matrice sous la forme :

$$X = T_k S_k P_k^T \tag{1}$$

où X est la matrice approchée [termes x documents], T_k est la matrice des k premiers vecteurs propres (vecteurs "terme") à gauche, P_k est la matrice des k premiers vecteurs propres (vecteurs "document") à droite et S_k est une matrice diagonale tronquée à l'ordre k.

LSA, comme word2vec, suppose que les mots dont les significations sont proches s'observent dans des parties similaires des textes qui composent le corpus (hypothèse distributionnelle). La différence entre les deux méthodes est que LSA construit une projection linéaire des vecteurs "terme" dans un espace de dimension réduite, le contexte de co-occurrence des termes étant global au document, tandis que word2vec propose une réduction de dimension via une projection non-linéaire en exploitant un contexte local de co-occurrence. Par ailleurs, la proximité sémantique des termes est appréhendée via la similarité cosinus associée aux pondérations *tf* et *tf-idf* pour LSA, tandis que les architectures word2vec proposent une distribution de probabilité associé à une pondération binaire des entrées du vocabulaire exploité.

3.3 Combinaison de Word2vec et LSA

Word2vec permet de créer des représentations vectorielles de mots dans un espace sémantique qui capture en partie le sens du mot. Partant de l'hypothèse que le sens général d'un texte est défini par la combinaison du sens des mots qui le composent, l'idée générale consiste à considérer la représentation vectorielle d'un texte sous la forme d'une combinaison des vecteurs des mots qui le forment. L'approche naïve se borne à effectuer la moyenne des vecteurs de mots obtenus par une représentation word2vec. Cette moyenne est une estimation de l'idée générale du texte. Cependant, toute approche basée sur un moyennage est sensible aux valeurs extrêmes, peu pertinentes dans le cas de données très asymétriques et souvent, la moyenne calculée ne correspond à aucune des valeurs observées. Par ailleurs, à partir de la moyenne word2vec, il est difficile, voire impossible, de remonter aux mots qui forment le texte. En contre-partie, une représentation vectorielle type LSA conserve la connaissance des occurrences des mots importants qui caractérisent les textes. C'est cette complémentarité des deux approches qui motive l'approche proposée. En concaténant la moyenne des vecteurs word2vec et la représentation vectorielle des textes fournie par LSA nous obtenons une représentation vectorielle en basse dimension au niveau du document qui capture la sémantique générale d'un texte tout en conservant une description lexicale/conceptuelle du document. A notre connaissance, cette combinaison n'a pas été explorée.

La concaténation du vecteur moyen word2vec d'un document à sa représentation LSA est illustrée en figure 1. Nous avons choisi d'appliquer LSA sur une matrice (termes x documents) pondérée par le *tf-idf*.

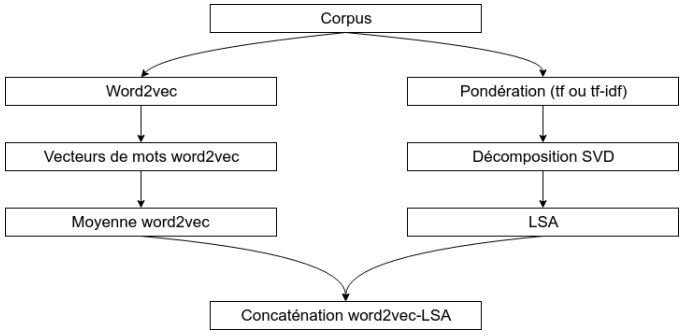

FIG. 2 – *Principe de notre combinaison de LSA avec de word2vec.*

4 Expérimentation

4.1 Données utilisées

Afin d'estimer les performances des différentes méthodes évaluées, nous avons choisi quatre jeux de données possédants les caractéristiques suivantes :

1. 20NewsGroup (Joachims, 1997), noté par la suite 20NG, est une base documentaire anglophone constituée de 20K documents contenants des commentaires extrait d'un forum de discussion, répartis dans 20 catégories. Ces documents contiennent du bruit sous la forme de méta-données présentes notamment dans les entêtes et les pieds de pages qui donnent des informations sur les auteurs des commentaires ou sur leur organisme de rattachement. Ces méta-données peuvent être bénéfiques lors d'une tâche de classification dans certains cas en biaisant l'apprentissage.
2. RCV1 (Lewis et al., 2004) est un corpus de 800K documents contenant des dépêches en anglais produites par l'agence de presse Reuters, classées dans 103 catégories. Les dépêches sont multi-étiquettes, i.e. *multi-label*, en ce sens qu'une même dépêche peut appartenir à plusieurs catégories.
3. TED-FR est un sous corpus issue du corpus fd-TED (Ahmia et al., 2018) [1]. contenant des appels d'offres se rapportant aux marchés publics européens. Les documents contiennent beaucoup de bruit qui prend la forme d'information juridico-commercial-administratif. Les documents sont classés en 45 catégories (également *multi-label*). Pour cette expérimentation, nous avons uniquement considéré les annonces en français traduites en entier ce qui constitue 800K documents.
4. TED-FILTRE, est un corpus de 2000K documents obtenus à partir du corpus complet fd-TED en filtrant les passages juridico-commercial-administratifs, ce qui réduit considérablement la taille des textes. Il comporte également 45 catégories.

4.2 Protocole expérimental

Nous décrivons dans cette section le protocole mis en œuvre afin de comparer notre méthode de vectorisation (combinaison LSA + word2vec) à d'autres types de vectorisation couramment employés dans la littérature sur les jeux de données mentionnés précédemment.

Pour toutes les méthodes testées les pré-traitements suivants s'appliquent. Nous retirons tout d'abord les mots vides (ou stop-words) adaptés aux différentes langues de nos jeux de données. Ensuite, dans le cas spécifique des corpus issus du fd-TED, les codes CPV (système de classification pour les marchés publics) contenus dans le texte sont remplacés par un mot-clé neutre (%digit%), car ces codes définissent la classe des documents. Par ailleurs, nous supprimons également les termes dont le nombre d'occurrences est inférieur à 5 dans un jeu de données.

Une fois les pré-traitements effectués, nous considérons les vectorisations suivantes : les pondérations *tf* et *tf-idf* sans et avec application de LSA (notés respectivement tf, tf-idf, LSA tf, LSA tf-idf par la suite), la moyenne des vecteurs word2vec (noté W2V par la suite), ainsi que la combinaison *tf-idf* avec LSA + word2vec (notée LSA+W2V par la suite).

1. https://github.com/oussamaahmia/TED-dataset

Dans notre expérimentation, nous avons choisi 100 dimensions pour LSA et W2V ce qui conduit à une vectorisation combinée en 200 dimensions.

— MLP : Un perceptron multicouches (Rumelhart et al., 1986) avec deux couches cachées de 200 neurones chacune, la fonction d'activation utilisée est la sigmoid : $\frac{e^x}{e^x+1}$.
— SGD : Une machine linéaire à vecteurs supports optimisée par descente de gradient stochastique (Zhang, 2004).
— NB : Un classifieur Bayésien multinomial naïf (Kibriya et al., 2004) avec $\alpha = 1$ pour les pondérations *tf* et *tf-idf*. Un classifieur Bayésien naïf gaussien (Hand et Yu, 2001) est utilisé pour les autres types de vectorisations car ces dernières peuvent contenir des valeurs négatives, avec une variance par défaut de $\sigma = 10^{-9}$.

La performance de ces différents algorithmes est évaluée en fonction des différentes vectorisations décrites précédemment. Toutes les combinaisons ne peuvent cependant pas être évaluées pour des raisons de complexité algorithmique. Il n'est en effet pas concevable en pratique de combiner par exemple une vectorisation en très grande dimension *tf-idf* à un classifieur MLP compte tenu de la combinatoire engendrée.

La classification '*multi-label*' a été effectuée en utilisant lapproche '*One v.s. Rest*', qui consiste à entraîner un modèle binaire par classe. Les individus dune classe sont considérés comme positifs et tous les autres comme négatifs. Un seuil de décision (0.5) est ensuite exploité (si le logarithme de la probabilité d'un individu par rapport à une classe est supérieure au seuil) pour décider de laffectation.

4.3 Résultats expérimentaux

Cette section présente les résultats expérimentaux obtenus pour les différents couples (vectorisation ; algorithme) présentés.

Les algorithmes de classification sont évalués sous l'angle de quatre métriques :

— l'exactitude empirique (*accuracy*) qui représente le pourcentage des observations étiquetées de manière identique à l'annotation : $A = \frac{vp+vn}{vp+fp+vn+fn}$,
— La précision $P = \frac{vp}{vp+fp}$,
— Le rappel $R = \frac{vp}{vp+fn}$ où vp est le nombre de vrais positifs, fp est le nombre de faux positifs et fn est le nombre de faux négatifs,
— La F1-mesure $F = \frac{(1+\beta^2).P.R}{\beta^2.P+R}$ (avec β=1),

où vp, fp, vn, fn représentent respectivement les nombres de vrais positifs, faux positifs, vrais négatifs, faux négatifs.

Les métriques (précision, rappel, F1-score) utilisées sont présentées sous la forme dune moyenne pondérée par rapport au support de chaque classe. Il sagit donc dune macro-moyenne pour tenir compte du déséquilibre des classes.

Par ailleurs, 80% du jeu de chacune des bases de données présentées en section 4.1 sont utilisés en tant que données d'apprentissage et 20% en tant que données de test, en procédant à une validation croisée (k=5).

Les tableaux 1, 2, 3 et 4 présentent les résultats de classification obtenus sur l'ensemble des quatre corpus tels que définis en section 4.1.

	Exactitude	Précision	Rappel	F1-score
SGD (tf-idf)	92,92	92,8	92,8	92,8
NB (tf-idf)	90,07	90,6	90	89,8
MLP (LSA+W2V)	84,46	84,6	84,6	84,6
NB (tf)	83,96	86,2	83,8	83,4
MLP (LSA tf-idf)	81,86	82	81,8	81,6
SGD (LSA tf-idf)	78,99	81,2	79	79
NB (LSA tf-idf)	74,1	75,4	74,4	74,4
MLP (W2V)	73,43	73,6	73,6	73,6
SGD (LSA+W2V)	71,49	80,8	71,4	73
SGD (tf)	71,12	83,2	71	74,6
NB (LSA+W2V)	68,46	71,6	68,4	69
MLP (LSA tf)	59,37	59,2	59,4	58,4
SGD (LSA tf)	54,16	68,4	54,2	54,4
NB (W2V)	51,95	55	51,8	52
SGD (W2V)	51,16	65	51,2	52,6
NB (LSA tf)	39,23	51,2	39,2	41,6

TAB. 1 – *Résultats obtenus pour les méthodes testées sur le jeu de données 20NG.*

Pour le corpus 20NG, la meilleure exactitude est obtenue avec le classifieur SGD pour la vectorisation *tf-idf*. Ce résultat s'explique par le fait que la pondération *tf-idf* arrive à mieux représenter un document en considérant tous les mots qui le forme, ce qui lui permet de séparer de manière plus efficace les classes les plus similaires. En effet, 20NG est une base contenant des classes assez similaires sémantiquement (exemple : religion.misc et religion.christian), et d'autres classes plus éloignées (exemple : religion.misc et sci.electronics).

Pour les autres jeux de données, notre modèle LSA+W2V obtient les meilleurs résultats. Pour le TED, le rappel est très largement supérieur à celui obtenu avec les autres méthodes. Ce jeu de donnés se révèle constituer une tâche difficile de classification, notamment par le fait qu'il contienne beaucoup de bruit, et que les mots réellement discriminants par rapport aux classes ne représentent qu'une partie du texte. LSA (*tf-idf*) se classe en deuxième position avec une exactitude empirique de 49,45% et word2vec en quatrième position avec une exactitude empirique de 45,23% comme indiqué dans le tableau 2.

	Exactitude	Précision	Rappel	F1-score
MLP (LSA+W2V)	55,27	78,2	57	64
MLP (LSA tf-idf)	49,65	76,8	49,4	55,4
SGD (tf)	48,32	78,2	59,6	62,8
MLP (W2V)	45,23	76,6	45	53
MLP (LSA tf)	33,13	64,4	31,8	37
SGD (LSA+W2V)	32,8	66,6	34,4	41
SGD (W2V)	29,9	64,4	31,8	38,4
SGD (LSA tf-idf)	17,63	53	16,6	22
NB (tf)	16,65	34,6	81,2	45,6
SGD (tf-idf)	16,35	73,8	15	20,4
NB (LSA tf-idf)	14,6	32	45,6	35,2
NB (tf-idf)	13,99	68,8	13,4	19,8
NB (LSA tf)	11,41	28,2	41,8	30,4
NB (LSA+W2V)	6,08	26,4	60,8	33,2
SGD (LSA tf)	6,06	30,2	5,8	9
NB (W2V)	5,33	19,8	48,2	25,6

TAB. 2 – *Résultats obtenus pour les méthodes testées sur le jeu de données TED-FR.*

Sur la base RCV1 word2vec se classe en deuxième position avec une exactitude empirique de 57,86%. LSA (tf-idf) se classe en troisième position avec une exactitude empirique de 54,64% comme indiqué dans le tableau 3.

Nous retrouvons des résultats similaires avec TED-FILTRE, comme le montre le tableau 4. Notre modèle LSA+W2V, qui exploite les deux vectorisations complémentaires, facilite la discrimination des classes.

En associant LSA à word2vec, notre modèle, grâce à LSA, tient compte des fréquences d'occurrence des mots importants qui composent le document et, grâce à la moyenne W2V, prend en compte la sémantique générale des documents, tout en préservant une dimension réduite pour la représentation finale du document.

En tirant partie de la complémentarité des deux approches, LSA+W2V atteint le meilleur score sur les bases RCV1 et TED. Pour 20NG, qui comporte des catégories sémantiquement très proches, la vectorisation LSA est potentiellement réalisée avec une dimension trop faible pour préserver une bonne séparation de telles catégories.

	Exactitude	Précision	Rappel	F1-score
MLP (LSA+W2V)	58,51	87	81	83,6
MLP (W2V)	57,86	87,4	80	83
MLP (LSA tf-idf)	54,64	86	77	80,4
SGD (tf-idf)	49,67	91,4	69,2	76,2
MLP (tf)	49,41	84,2	72,2	76,6
SGD (tf)	44,84	81,6	75,6	77,8
SGD (LSA+W2V)	38,32	79	71	73,4
SGD (LSA tf-idf)	36,42	82,8	60,6	67,2
SGD (W2V)	32,99	76,6	67,4	70
SGD (tf)	18,81	78,4	44	53
NB (LSA tf-idf)	4,85	40,8	77,2	50,2
NB (LSA+W2V)	2,58	40,6	85,2	50,4
NB (W2V)	1,54	38,6	85,2	48,4
NB (tf)	0,55	27,4	72,8	36,2
NB (tf)	0,29	4,4	0	0
NB (tf-idf)	0,29	0	0	0

TAB. 3 – *Résultats obtenus pour les méthodes testées sur le jeu de données RCV1.*

	Exactitude	Précision	Rappel	F1-score
MLP (LSA+W2V)	77,47	95,2	75	83,2
MLP (W2V)	75,6	95,2	72,6	81,4
MLP (LSA tf-idf)	69,38	94,4	66,4	76,6
NB (tf-idf)	66,07	88	66,2	74,6
SGD (LSA+W2V)	61,22	91	59,2	69,8
SGD (W2V)	56,16	88	55,2	65,6
SGD (tf)	55,74	89,4	53,8	65
NB (tf)	51,9	62,8	78,2	68,8
SGD (tf-idf)	49,39	93,4	46	59
SGD (LSA tf-idf)	44,73	83,2	42,4	53,4
SGD (tf)	31,82	74,4	30,2	39,2
NB (W2V)	5,72	23,8	77,6	33,8
NB (tf)	2,38	19,8	73	27,8
NB (LSA+W2V)	1,85	22,8	81,2	33,8
NB (LSA tf-idf)	1,63	20	76,8	30,2

TAB. 4 – *Résultats obtenus pour les méthodes testées sur le jeu de données TED-FILTRE.*

Notre modèle obtient respectivement une exactitude empirique de 55.27%, 58.51% et 77.47% sur les jeux de données TED-FR, RCV1, TED-FILTRE, comme le montrent les tableaux 2, 3 et 4. Les résultats plus faibles obtenus sur le TED-FR s'expliquent en grande partie

par la nature de la base, qui est très bruitée notamment par la présence d'information juridico-administrative (laquelle représente la majorité des textes), et la difficulté de la classification multi-étiquettes en 45 classes.

5 Conclusion

Nous avons introduit dans cet article un modèle simple de vectorisation en basse dimension des textes d'un corpus qui combine les méthodes LSA et word2vec. Nous avons comparé cette vectorisation aux autres approches classiques de vectorisation proposées dans l'état de l'art.

En particulier, notre méthode obtient de meilleurs résultats comparativement à toutes les autres méthodes testées pour trois des quatre jeux de données traités sur des tâches de classification supervisées. Sur le quatrième jeu de données qui comporte des classes très similaires du point de vue sémantique, notre méthode en basse dimension est pénalisée par rapport aux approches qui représentent les documents sous la forme de sac-de-mots.

L'exploitation de cette approche de vectorisation sur une tâche de clustering de documents textuels constitue une perspective à ce travail.

Références

Ahmia, O., N. Béchet, et P.-F. Marteau (2018). Two multilingual corpora extracted from the tenders electronic daily for machine learning and machine translation applications. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Blei, D. M., A. Y. Ng, et M. I. Jordan (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*(Jan), 993–1022.

Chen, M. (2017). Efficient verctor representation for documents through corruption. *arXiv preprint arXiv :1707.02377*, 13.

Dey, L. et S. M. Haque (2009). Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition (IJDAR) 12*(3), 205–226.

Hand, D. J. et K. Yu (2001). Idiot's bayesnot so stupid after all ? *International statistical review 69*(3), 385–398.

Harris, Z. S. (1954). Distributional Structure. *WORD 10*(2-3), 146–162.

Jindal, N. et B. Liu (2007). Review spam detection. In *Proceedings of the 16th international conference on World Wide Web - WWW '07*, Banff, Alberta, Canada, pp. 1189. ACM Press.

Joachims, T. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, San Francisco, CA, USA, pp. 143–151. Morgan Kaufmann Publishers Inc.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28*, 11–21.

Ju, R., P. Zhou, C. H. Li, et L. Liu (2015). An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis. In *2015 IEEE International Conference*

on Computer and Information Technology ; Ubiquitous Computing and Communications ; Dependable, Autonomic and Secure Computing ; Pervasive Intelligence and Computing, LIVERPOOL, United Kingdom, pp. 2276–2283. IEEE.

Kibriya, A. M., E. Frank, B. Pfahringer, et G. Holmes (2004). Multinomial naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence*, AI'04, Berlin, Heidelberg, pp. 488–499. Springer-Verlag.

Kushmerick, N., E. Johnston, et S. McGuinness (2001). Information Extraction By Text Classification. In *In The IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*.

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

Le, Q. V. et T. Mikolov (2014). Distributed Representations of Sentences and Documents. *arXiv :1405.4053 [cs]*. arXiv : 1405.4053.

LeCun, Y., P. Haffner, L. Bottou, et Y. Bengio (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pp. 319–345. Springer.

Lewis, D. D., Y. Yang, T. G. Rose, et F. Li (2004). Rcv1 : A new benchmark collection for text categorization research. *Journal of machine learning research 5*(Apr), 361–397.

Mikolov, T., K. Chen, G. Corrado, et J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv :1301.3781 [cs]*. arXiv : 1301.3781.

Moody, C. E. (2016). Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *arXiv :1605.02019 [cs]*. arXiv : 1605.02019.

Rumelhart, D. E., G. E. Hinton, et R. J. Williams (1986). Parallel distributed processing : Explorations in the microstructure of cognition, vol. 1. Chapter Learning Internal Representations by Error Propagation, pp. 318–362. Cambridge, MA, USA : MIT Press.

Zell, A. (1994). *Simulation neuronaler netze*, Volume 1. Addison-Wesley Bonn.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA, pp. 116–. ACM.

Summary

We present in this article a study on text vectorization methods for document classification. We study methods based on word embedding (word2vec), and document embedding (latent semantic analysis and bag of words associated with various weightings) as well as some combinations of this methods. To this end, we evaluate these vectorization approaches by using three classification models (a multilayer perceptron, a linear vector-support machine based on stochastic gradient descent optimization and multinomial or Gaussian naïve Bayes classifiers). Our results clearly show that the straightforward combination of word2vec and LSA methods that we propose, which achieves the association of two complementary definitions of the context (local for word2vec and global for LSA) of word occurrences, makes it possible to produce a robust vectorization for texts that is, in general, significantly more discriminating than the other tested vectorization approaches.