CAS PRATIQUE 2
RAPPORT E3

Représentation vectorielle des mots dans les problématiques de Traitement Automatique du Langage Naturel

Marianne DEPIERRE

20/01/2023

Titre professionnel RNCP "Développeur en intelligence artificielle"

Table des matières

1. Fiabilité des sources utilisées 3
2. Etat de l'art : Représentations vectorielles de mots 4
   2.1. Introduction 4
      2.1.1. Définition du NLP 4
      2.1.2. Processus du NLP et définition de l'embedding 4
   2.2. One-Hot-Encoding 4
   2.3. Vecteurs basés sur la fréquence 4
      2.3.1. Vecteur d'occurrences (ou count vector) 5
      2.3.2. TF-IDF (Term Frequency-Inverse Document Frequency) 5
      2.3.3. Matrice de co-occurrence avec une fenêtre de contexte 5
   2.4. Vecteurs basés sur la prédiction 5
      2.4.1. Word2vec 5
   2.5. Embedding dans les modèles de Deep Learning 6
      2.5.1. Tensorflow (Keras) 6
      2.5.2. Transformer 7
   2.6. Limites de la représentation vectorielle basée sur le mot 7
   2.7. Conclusion 7
3. Bibliographie 8
   - Publications scientifiques 8
   - Autres sources 8

1. Fiabilité des sources utilisées

Dans le cadre du passage de mon titre professionnel de Développeur Intelligence Artificielle, j'ai réalisé une veille technologique afin de sélectionner une technique de représentation vectorielle des mots dans une problématique de Traitement Automatique du Langage Naturel (TALN). Pour effectuer cette veille, je me suis basée sur :

- majoritairement, des articles scientifiques, accessibles sur Google Scholar
- de la documentation de bibliothèques Python (Tensorflow, Scikit-Learn)
- des articles de blogs spécialisés en ligne (Medium)

Chacune de ces ressources a subi à sa manière une validation par ces paires, ce qui me garanti de la légitimité des propos tenus.

La majorité de mes sources sont des articles issus de revues scientifiques. Avant d'être publiés dans un journal, les articles de recherche sont évalués par d'autres scientifiques (« peer review »). Ceux-ci lisent l'article, relèvent les erreurs et les contradictions et soumettent leurs corrections à l'auteur. Celui-ci doit modifier son article avant d'être ré-évaluer, jusqu'à ce que l'article soit accepté. Cette validation par les pairs est un principe fondamental de la publication scientifique. Elle garantit l'intégrité des informations dispensées dans un article scientifique, et par conséquent de la connaissance.

Les bibliothèques de python sont open-source. Le code et la documentation sont donc contrôlés non seulement par les développeurs des bibliothèques, mais aussi par toute la communauté de développeurs python.

Dans une moindre mesure, les articles de vulgarisation sur la plateforme Medium subissent une étape de validation avant publication, les rendant plus fiables qu'un blog ordinaire.

2. Etat de l'art : Représentations vectorielles de mots

2.1. Introduction

2.1.1. Définition du NLP

Le NLP (Natural Language Processing) ou Traitement Automatique du Langage Naturel (TALN) est une section de l'Intelligence Artificielle (IA) qui porte sur la compréhension, la manipulation, la génération et la traduction du langage humain - parlé ou écrit - par les machines. Autrement dit, le NLP est l'intersection des domaines de l'IA et de la linguistique. Dans le domaine des NLP, la linguistique est traduite en programmes informatiques à l'aide de l'IA entre autres. Le champ d'application du NLP est vaste ; il solutionne entre autres des problématiques de traduction automatique, d'analyse de sentiments, de robots conversationnels (ou chatbots), d'autocomplétion des phrases, et de classement thématique de textes.

2.1.2. Processus du NLP et définition de l'embedding

Globalement, le processus du NLP se divise en deux parties. La première est celle de la linguistique, où les textes ou les vocaux d'origines sont prétraités (nettoyage, tokenisation, stop words retirés, etc.), et transformés en entrées exploitables par un modèle d'apprentissage. La deuxième partie concerne donc l'apprentissage automatique. Elle consiste à utiliser un modèle de Machine Learning ou de Deep Learning sur ces entrées.

Ainsi, afin d'appliquer les modèles à des problématiques de langage naturel, les données textuelles sont transformées en données numériques, en vecteurs. Le processus utilisé s'appelle « embedding » (Zamani & Croft, 2017) ou « prolongement lexical » en français. La façon de représenter le langage est essentielle car elle impacte directement l'analyse réalisée par la suite. L'embedding est donc une étape critique dans le processus du NLP. Ce processus est classiquement employé à l'échelle d'un mot, ce qui est appelé « word embedding ». Il existe plusieurs approches de word embedding, décrites dans la suite de ce rapport.

2.2. One-Hot-Encoding

« the cat sat on the mat »	
the	[1 0 0 0 0]
cat	[0 1 0 0 0]
sat	[0 0 1 0 0]
on	[0 0 0 1 0]
the	[1 0 0 0 0]
mat	[0 0 0 0 1]

Habituellement dans le domaine du Machine Learning, les données qualitatives sont rendues interprétables par les modèles en les transformant en matrices par one-hot-encoding. Avec cette technique, à chaque mot ou symbole du corpus de texte est associé un vecteur de valeurs binaires (Pedregosa et al., 2011). Chaque vecteur est composé de 0 et d'un seul 1 qui permet de différencier un mot d'un autre. Voici un exemple de one-hot-encoding de la phrase « the cat sat on the mat ».

Dans cette approche il existe deux inconvénients majeurs. Premièrement, un vecteur est de dimension conséquente puisque celui-ci est de la longueur du dictionnaire du corpus. Cela implique un temps de calcul qui s'allonge excessivement, puisqu'un vecteur requiert beaucoup de mémoire. Deuxièmement la technique de one-hot-encoding n'apporte pas de sens sur le contexte des mots qu'elle encode. Ainsi des mots comme « chien » et « chat » sont tout aussi proches qu'avec « banane » car ils sont de même distance vectorielle. L'enjeu des techniques d'embedding est donc double : réduire la dimensionnalité des représentations vectorielles des mots de façon pertinente (Yin & Shen, 2018) et y associer un sens sémantique et/ou lexical.

2.3. Vecteurs basés sur la fréquence

Une première approche est celle du Bag-Of-Word (ou sac de mots) qui consiste à compter la présence et l'absence de chaque mots d'un texte. Le Bag-Of-Word peut se baser sur l'occurrence d'un mot dans un texte ou sur sa fréquence relatif à un corpus de textes.

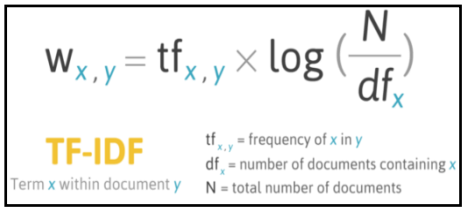
2.3.1. Vecteur d'occurrences (ou count vector)

Dans cette approche, la représentation vectorielle est basée sur la fréquence de chaque mot. Le nombre de chaque token du corpus de texte est compté (Pedregosa et al., 2011). Chaque texte du corpus est représenté par un vecteur d'occurrences.

Cependant, certains mots sont par nature plus utilisés que d'autres alors que la fréquence des mots importants est très faible : il s'agit de l'inconvénient majeur des vecteurs d'occurrences.

2.3.2. TF-IDF (Term Frequency-Inverse Document Frequency)

Dans cette approche, le nombre de tokens du corpus de texte est également compté mais il est divisé par le nombre d'occurrences total de ces mêmes tokens dans l'entièreté du corpus (Pedregosa et al., 2011). La partie TF mesure l'importance relative d'un mot dans un document, tandis que la partie IDF mesure la signification d'un terme en fonction de sa distribution et de son utilisation dans l'ensemble des documents. Plus un terme a de l'importance dans un document, plus le TF-IDF sera élevé. Souvent un seul texte est analysé, et pour appliquer le TF-IDF ce dernier est considéré comme le corpus de texte et une phrase comme un document.
Le calcul pour attribuer la valeur d'un mot dans le vecteur est affiché ci-contre (Mei 2019).



Contrairement au vecteur d'occurrences, le TF-IDF rend compte de l'unicité d'un mot en donnant le poids et non la fréquence d'un mot. Cependant les mots étant séparés de leur positionnement dans la phrase, le TF-IDF ne prend pas en compte le contexte d'un mot. Un autre inconvénient réside dans la trop grande taille des vecteurs des corpus riches en vocabulaire pour entraîner un modèle de Machine Learning.

2.3.3. Matrice de co-occurrence avec une fenêtre de contexte

Dans un corpus de texte, la co-occurrence d'une sélection de mots se définit par le nombre de fois où ceux-ci sont apparus ensembles dans une phrase ou dans une fenêtre contextuelle (taille et direction fixes). Concernant une comparaison de paires de mot, la matrice de co-occurrence sera composée de la sélection de mots en colonne et de la ou des fenêtres contextuelles en lignes. La valeur attribuée à l'intersection de deux mots est la co-occurrence.

A l'opposé des méthodes précédemment décrites, la matrice de co-occurrence permet de déterminer les relations sémantiques entre les mots. Cependant elle devient rapidement de très haute dimension, augmentant ainsi la mémoire nécessaire et les temps de calculs lors de l'entraînement des modèles. Ce problème peut être partiellement résolu par une réduction de dimension, par PCA (Principal Component Analysis) ou SVD (Singular Value Decomposition).
Des modèles plus complexes reprennent le concept de la matrice de co-occurrence pour l'améliorer, comme le GloVe ou Global Vector (Pennington et al., 2014) qui utilise les valeurs non nulles de la matrice de co-occurrence.

2.4. Vecteurs basés sur la prédiction

2.4.1. Word2vec

Les approches basées sur la fréquence montrent leurs limitations dans la représentation de mots en faisant abstraction de leur contexte. En 2013, Mikolov et al. proposent un modèle, word2vec, qui prend en considération le contexte des mots en tenant compte de leurs ressemblances sémantiques, syntaxiques ou thématiques. Ainsi, des mots de contexte similaires ont des vecteurs de distance vectorielle proche. Ainsi, comme dans ce célèbre exemple « King – Man + Woman = ? », Word2vec est capable de déterminer que le résultat de l'équation est « Queen ». Word2vec est un réseau de neurones de seulement deux couches qui détermine la probabilité d'un

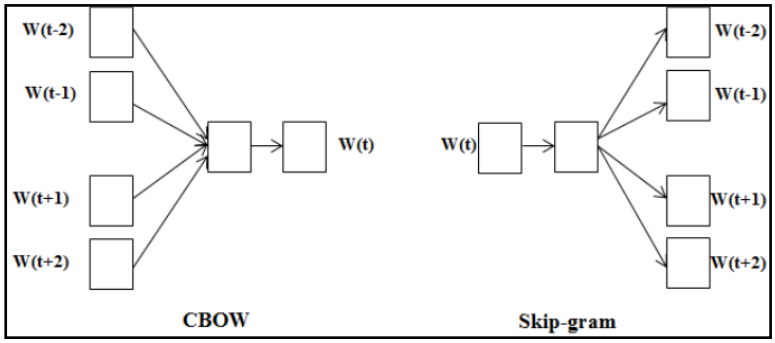
mot d'être proche de d'autres. Ce modèle possède deux variantes de son architecture : CBOW (Continuous Bag Of Words) et Skip-gram.

- ***CBOW***

L'algorithme de CBOW prédit un mot à partir d'une liste de mots. Celle-ci est composée d'une petite séquence de mots précédant le mot cible et d'une autre le suivant. Dans cette architecture, l'ordre des mots n'a pas d'importance, le modèle utilise les représentations de tous les mots de la liste (du contexte) pour calculer la probabilité d'un mot comme étant le mot cible. CBOW est plus rapide à entraîner que skip-gram, et est plus performant pour prédire des mots fréquents. Il rapproche plus facilement des mots de syntaxe proche.

- ***Skip-gram***

A l'opposé du CBOW, le modèle skip-gram prédit un ou plusieurs mots à partir d'un seul mot. Le ou les mots cibles sont des séquences de taille définie (fenêtre fixe) positionnées avant ou après le mot en entrée. En résumé, un mot est utilisé pour prédire son contexte.
Ce modèle est préféré à CBOW lorsque l'on dispose d'une petite quantité de données d'entraînement, et est plus performant pour prédire les mots ou les séquences de mots rares. Il est plus enclin à attribuer des vecteurs proches pour des mots de même sémantique.



Les méthodes de word2vec ont plusieurs limites. Tout d'abord, elles basent l'embedding sur un mot individuel, ce qui est problématique pour les représentations vectorielles des phrases : il ne suffit pas de combiner les vecteurs de chaque mot composant cette phrase. De plus, lorsque des petites fenêtres sont utilisées, des antonymes comme « bon » et « mauvais » obtiennent un même embedding car ils sont de sémantiques similaires (Socher et al., 2011). Cela est très problématique dans le cas d'une analyse de sentiments, puisque la notion polarité est perdue (Wang et al., 2015) et la performance du modèle est amoindrie. Cet exemple illustre à quel point le word embedding est lié à son application.

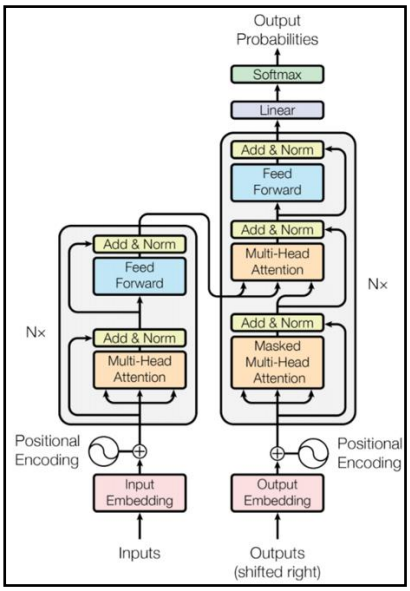
2.5. Embedding dans les modèles de Deep Learning

2.5.1. Tensorflow (Keras)

Pour ses modèles de Deep Learning, Abadi et al. (2015) proposent dans leur bibliothèque Tensorflow une couche appelée « Embedding ». Auparavant, les données doivent avoir été transformées : chaque mot est représenté par un identifiant. La couche d'Embedding reçoit une phrase d'un certain nombre de mots numérotés dans un vecteur d'une taille définie arbitrairement. Si la phrase est composée de moins de mots que la longueur de vecteur d'entrée, un padding est appliqué, c'est-à-dire que chaque valeur ajoutée pour complétion du vecteur est égale à 0. La couche remplace chacun de ces mots par un vecteur dense d'un nombre de composantes également définies arbitrairement. La couche va maintenir une structure de données, un dictionnaire, où chaque mot est associé à son vecteur dense. Initialisées aléatoirement, ces composantes sont mises à jour durant l'apprentissage de manière à ce que la valeur de ces vecteurs denses permettent d'avoir de bons résultats en sortie du réseau de neurone. Ainsi, à la fin de l'entraînement, les mots de signification proches ont des distances vectorielles proches.

2.5.2. Transformer

Le transformer est un modèle de type encodeur-décodeur qui effectue des prédictions grâce à un mécanisme nommé Attention (Vaswani et al., 2017). Dans une architecture encodeur-décodeur classique, la sortie de l'encodeur est utilisée seule en entrée, ce qui engendre une perte de l'information. Dans les transformers, l'information qui arrive dans le décodeur n'est pas seulement la sortie de l'encodeur. Avant d'effectuer une prédiction, par exemple le prochain mot d'une phrase à traduire, toutes les informations sur l'état de toutes les cellules de l'encodeur sont également prises en compte, en plus de la sortie de l'encodeur.



Dans le transformer, il y a deux étapes d'embedding, une pour l'encodeur et une autre pour le décodeur, mais il s'agit de la même technique utilisée deux fois. Dans un premier temps, une opération d'embedding du même genre que celle des modèles de Deep Learning de Tensorflow est effectuée. Une étape appelée « Positional encoding » est ajoutée à la suite de l'embedding. De base, les vecteurs générés par l'embedding ne tiennent pas compte de la position du mot dans la phrase, provoquant la perte de contexte d'un mot. Le Positionnal Encoding rajoute à chaque vecteur des informations sur la position du mot. Ainsi, lorsque les différentes opérations sont effectuées par la suite, cette information est présente et sera prise en compte pour effectuer une prédiction.

2.6. Limites de la représentation vectorielle basée sur le mot

Comme vu précédemment, les méthodes d'embedding les plus couramment utilisées sont basées sur les mots. Ces techniques ont un inconvénient majeur : elles ne prennent pas en compte la polysémie. En effet, la représentation vectorielle du mot est globale à tout le corpus de texte, elle ne distingue pas une différence de sens ou de comportement syntaxique d'un même mot au sein de ce corpus. Par exemple, dans les phrases « She went to the bank to deposit her check » et « This plant grows on the river bank » le mot « bank » possède deux contextes distincts. L'idéal serait de pouvoir créer deux vecteurs pour le représenter. De nouveaux modèles, comme ELMo (Peters et al., 2018), ne représentent plus les mots de façon globale mais en proposant des word embedding contextuels. Dans l'exemple d'ELMo, des représentations différentes sont créer pour chacun des sens d'un mot, plus précisément il est généré un vecteur par phrase/document.

Il existe d'autres méthodes d'embedding qui se basent sur le caractère. Elles sont intéressantes pour des tâches de POS tagging et NER, puisqu'elles ont la capacité de capturer des informations morphologiques à l'intérieur du mot. Pour une tâche de NER, cette technique s'est même montrée plus performante pour des langues morphologiquement riches telles que l'espagnol et le portugais (Santos & Guimaraes, 2015).

2.7. Conclusion

Le domaine de l'embedding est complexe et relève deux défis majeurs pour le traitement du langage naturel. Majoritairement à l'échelle du mot, sa représentation vectorielle doit refléter son sens sémantique et son sens lexical, mais aussi son importance dans la tâche de NLP à effectuer. Le choix de la méthode d'embedding est donc crucial pour les performances de l'analyse réalisée.

Suite à cette veille technologique, mon choix de technique d'embedding s'est porté pour un premier temps sur l'embedding avec Tensorflow. En effet, j'effectue une analyse de sentiment avec des méthodes de Deep Learning et cette méthode est adaptée à ma problématique. Elle est simple à mettre en place puisqu'elle est importée avec la bibliothèque Tensorflow que j'utilise pour ma modélisation. Elle est aussi moins coûteuse en termes de performances de calculs pour l'ordinateur que des méthodes plus complexes, comme les transformers ou les méthodes d'embedding basées sur la prédiction.

3. Bibliographie

• Publications scientifiques

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Wang, X., Liu, Y., Sun, C. J., Wang, B., & Wang, X. (2015, July). Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1343-1353).

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011, July). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 conference on empirical methods in natural language processing* (pp. 151-161).

Santos, C. N. D., & Guimaraes, V. (2015). Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp 2227–2237), New Orleans, Louisiana. Association for Computational Linguistics.

Yin, Z., & Shen, Y. (2018). On the dimensionality of word embedding. *Advances in neural information processing systems*, *31*.

Zamani, H., & Croft, W. B. (2017, August). Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 505-514).

• Autres sources

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Disponible sur : https://www.tensorflow.org/.

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*. Disponible sur https://scikit-learn.org/.

Mei, T. (2019). *Demystify TF-IDF in Indexing and Ranking.* Disponible sur : https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0 .