COMMENT APPLIQUER DE L'IA A DES PROBLEMATIQUES DE TRAITEMENT DU LANGAGE: TECHNIQUES D'EMBEDDING

OU

REPRESENTATIONS VECTORIELLES DES MOTS DANS DES PROBLEMATIQUE DE TRAITEMENT DU LANGAGE

PREMIERE PARTIE

Dans le cadre du passage de mon titre professionnel de Développeur Intelligence Artificielle, j'ai dû réaliser une veille technologique, afin de sélectionner une technique d'embedding pour tester des modèles de Machine Learning. Pour effectuer cette veille, je me suis basée sur :

* des articles scientifiques sur Google Scholar
* des articles de presse spécialisée en ligne (Towards Data Science, Medium )
* des vidéos de vulgarisation sur la plateforme Youtube
* de la documentation de librairies Python (Tensorflow, )

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

DEUXIEME PARTIE

INTRODUCTION

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 traduit 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.

(IMAGE ?)

Processus du NLP et définition de l'embedding

Globalement, le processus du NLP se divisent 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 automatique. 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 de Machine Learning à des problématiques de langage naturel, les données textuelles sont transformées en données numériques, en vecteurs. Le processus utilisé s'appelle « embedding » ou « prolongement lexical » en français. Le terme français n'étant jamais employé dans la bibliographie, celui qui est utilisé dans ce rapport est toujours « embedding ». 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. L'embedding 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.

ONE-HOT-ENCODING

Habituellement dans le domaine du Machine Learning (ML), 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. 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 » :

|  |  |
| --- | --- |
| « 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] |

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, puisque 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. La finalité des techniques d'embedding est donc double : réduire la dimensionnalité des représentations vectorielles des mots et y associer un sens sémantique et/ou lexical.

VECTEURS BASES SUR LA FREQUENCE

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.

* Vecteurs d’occurrences (ou count vectors)

Dans cette approche, le word embedding est basé sur la fréquence de chaque mot. Le nombre de chaque tokens du corpus de texte est compté. Chaque texte du corpus est représenté par un vecteur d’occurrences.

Voici un exemple issu d'un article sur medium.com :

« the weather was wonderful today and I went outside to enjoy the beautiful and sunny weather »

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| the | weather | was | wonderful | today | and | I | went | outside | to | enjoy | beautiful | sunny |
| 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Ici les mots « the », « weather » et « and » apparaissent deux fois dans le corpus, tandis que les autres n'apparaissent qu'une seule fois.

L'inconvénient des vecteurs d’occurrences c'est que certains mots sont par nature plus utilisés que d’autres alors que la fréquence des mots importants est très faible.

* 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. 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, 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 donc le suivant :

Voici un exemple de vecteur TF-IDF généré à partir d'un corpus de texte.

Dans cet exemple, le mot « Messi » possède quatre occurrences dans le premier document et aucune dans le suivant. Ainsi : TF-IDF(Messi, Document1) = (4/8)\*log(2/1) = (4/8)\*0.301 = 0.15

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 prends 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.

* Matrice de co-occurence avec une fenêtre de contexte

Dans un corpus de texte, la co-occurrence d'une sélection de mots se défini 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-occurence sera composé de la sélection de mots en colonne et de la ou des fenêtres contextuelles en lignes et la valeur attribuée à l'intersection de deux mot est la co-co-occurence. Dans l'exemple ci dessous :

Le mot « digital » est entouré des mots en en-tête de colonnes (« aardvark », « computer », « data », « result », « pie », « sugar »). D'après cette matrice de co-occurence le mot « digital » est encadré 85 fois par le mot « result ».

A l'opposé des méthodes précédemment décrites, la matrice de co-occurence 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 de machine learning. 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-occurence pour l'améliorer, comme le GloVe (Global Vector) qui utilise les valeurs non nulles de la matrice de co-occurence.

VECTEURS BASES SUR LA PREDICTION

* Word2vec

Les approches basées sur la fréquence montre leurs limitations dans la représentation de mots en faisant abstraction de leur contexte. En 2013, Mikolov propose un modèle, word2vec, ~~qui~~ prend en considération le contexte des mots en tenant compte des ressemblances sémantiques, syntaxiques ou thématiques des mots. 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 réseaux 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 mot 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~~, il converge plus facilement,~~ 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'un 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.

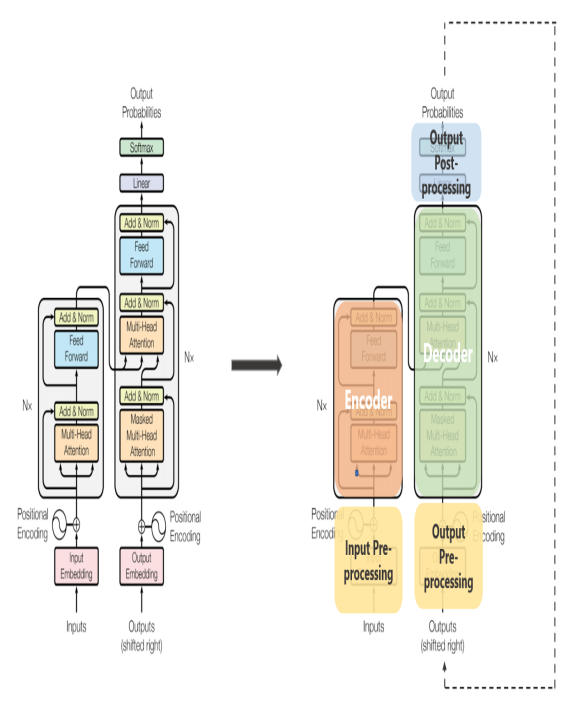
Les méthodes de word2vec ont plusieurs limites. Tout d'abord, elles basent le l'embedding sur un mot individuel, ce qui est problématiques pour les représentations vectorielles des phrases : il ne suffit pas de combiner les vecteurs de chaque mot composant cette phrase. De plus, lorsque des petites fenêtres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

EMBEDDING DANS LES MODELES DE DEEP LEARNING

* Tensorflow (Keras)

Pour ses modèles de Deep Learning, Tensorflow propose 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 définis arbitrairement. Si la phrase est composé 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 bon résultats en sortie du réseau de neurone. Ainsi, à la fin de l’entraînement, les mots de signification proches ont des distances vectorielles proches.

* Transformer

Le transformer est un modèle de type encodeur-decodeur qui effectue des prédictions grâce à un mécanisme nommé Attention. 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 decodeur 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 trasnformer, il y a deux étapes d'embedding, une pour l'encodeur et une autre pour le decodeur, 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 modèle de deep learning de Tenserflow 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.

LIMITES DE LA REPRESENTATION VECTORIELLE BASEE 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, 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 morphologique à l'intérieur du mot. Pour une tâche de NER, cette technique s'est même montré plus performante pour des langues morphologiquement riches telles que l'espagnol et le portugais.

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 lexicale, mais aussi son importance dans la tâche de NLP à effectuer. Le choix de la méthode d'embedding est donc cruciale pour les performances de l'analyse réalisée.

Suite à cette veille technologique, mon choix de technique d'embedding s'est porté sur XXXX. En effet, effectuant une analyse de sentiment la méthode me permet XXXX.

BIBLIOGRAPHIE

<https://nlpoverview.com/>

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*.

~~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).~~

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).

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014, June). Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)* (pp. 1555-1565).

Upadhyay, S., Chang, K. W., Taddy, M., Kalai, A., & Zou, J. (2017). Beyond bilingual: Multi-sense word embeddings using multilingual context. *arXiv preprint arXiv:1706.08160*.

*SITES DE DOCUMENTATION*

*tenserflow*

*word2vec*

*huggingface*