Analyse statistique et empirique des modèles de Word Embedding sur Twitter

Kim Antunez, Romain Lesauvage, Alain Quartier-la-Tente
sous l'encadrement de Benjamin Muller (Inria)

1 Contexte

Grâce à l'évolution des méthodes d'apprentissage profond (*Deep Learning*), l'appréhension du langage naturel est aujourd'hui devenu une discipline à part entière (*Natural Language Processing*). Ce succès s'explique en partie grâce à l'émergence de techniques non supervisées d'apprentissage de représentation de structures linguistiques. Les méthodes de *word embedding* (« plongement lexical » en français) permettent de représenter chaque mot d'un dictionnaire par un vecteur de nombres réels afin que les mots qui apparaissant dans des contextes similaires possèdent des vecteurs correspondants qui sont relativement proches. Les modèles **word2vec**, développés par une équipe de recherche chez Google (Mikolov, T., *et al* (2013)), sont parmi les plus célèbres et sont ceux sur lesquels se concentrera notre projet.

2 Objectif du projet

Dans ce projet de statistiques appliquées, nous étudierons dans un premier temps en détail et implémenterons le modèle *word2vec*. Dans un second temps, nous appliquerons ce modèle sur une base de données composée de plusieurs millions de tweets publiés en France entre 2013 et 2017. Pour ce faire, nous mobiliserons des techniques d'analyse de sentiments afin de créer des indicateurs qui pourront être comparés aux indicateurs produits dans la statistique publique, en particulier concernant l'opinion des ménages.

3 Travail effectué jusqu'alors

3.1 Appropriation du modèle

Pour débuter ce projet, nous nous sommes documentés sur les méthodes de NLP et de Deep-learning en lisant la bibliographie ainsi que des articles de blogs, en particulier concernant l'implémentation sur Python du modèle. Ces différentes lectures nous ont permis de bien comprendre les différentes étapes de l'algorithme dont l'objectif est, pour rappel, de représenter les mots d'un corpus sous la forme d'un vecteur de nombre réels. Différentes architectures du modèle word2vec existent ; nous nous intéresserons ici à l'approche appelée *Skip-gram*. À l'inverse de l'approche *Continuous bags of words*, le principe de *Skip-gram* est de prédire, pour chaque mot du vocabulaire, sa probabilité d'être proche d'un mot appelé *focus*.

Nous entraînons le modèle grâce à un algorithme qui peut se résumer à ces quelques étapes :

— Nous partons de deux matrices de poids : une en entrée et une en sortie. La taille de ces matrices est fixée par le nombre de mots différents (vocabulaire) présents dans le corpus (lignes) et la dimension souhaitée des vecteurs en sortie (colonnes).
— L'algorithme parcourt ensuite chaque phrase du corpus (une phrase correspondant à un tweet). Pour chaque phrase, un mot appelé *focus* est sélectionné au hasard. Puis, on associe à chaque mot focus un mot *contexte* tiré lui aussi au hasard dans une fenêtre autour du mot focus choisi [1].
— Nous sélectionnons ensuite la ligne de la matrice en entrée correspondant au mot focus. En multipliant matriciellement cette ligne avec la matrice de sortie, nous obtenons un score (produit scalaire) qui mesure pour chaque mot du vocabulaire la probabilité d'appartenir au contexte du mot focus. Ce score est ensuite transformé en probabilité d'appartenance grâce à la fonction *softmax*. Nous mettons ensuite à jour la fonction de perte (*loss*) qui dépend de ce score.
— Les matrices d'entrée et de sortie sont mises à jour successivement par descente de gradient $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L(\theta^{(t)})$ avec θ la matrice (entrée ou sortie), $L(\theta)$ la fonction de perte et η le taux d'apprentissage (constante).
— Ces étapes sont répétées pour l'ensemble des couples [focus, contexte]. Puis, l'ensemble de l'opération est répétée un certain nombre d'*epochs* fixé au préalable .

Au fur et à mesure de l'apprentissage, les deux matrices d'entrée et sortie sont actualisées, et de plus en plus pertinentes. À la fin de l'algorithme, nous faisons la moyenne de ces matrices. C'est cette moyenne qui représentera les vecteurs finaux de représentation des mots du corpus.

3.2 Implémentation du modèle

Après nous être approprié le fonctionnement du modèle **word2vec**, nous avons tous les trois implémenté individuellement le modèle sur Python en utilisant la librairie Pytorch. Ce travail individuel a permis à chacun d'approfondir la compréhension du modèle et d'obtenir une version actuelle conforme aux recommandations de l'article de Mikolov.

En première étape, nous avons implémenté une version simplifiée du modèle *word2vec* en utilisant la fonction *softmax* décrite plus haut. Cette méthode est gourmande computationnellement puisqu'elle nécessite d'actualiser à chaque étape l'intégralité des matrices.

C'est pourquoi nous avons en seconde étape implémenté une version plus élaborée du modèle grâce à la méthode du *negative sampling*. Cette méthode permet d'une part de corriger la sur-estimation des poids des mots contextes tirés au hasard puisqu'elle implique de tirer également au hasard des mots qui ne sont pas dans le contexte du mot focus (*negatives samples*), mots pour lesquels on va chercher à minimiser leur proximité au mot focus dans la fonction de *loss*. Elle est d'autre part moins coûteuse en temps de calcul puisqu'elle permet d'actualiser uniquement une partie de la matrice de poids (celle qui concerne le mot contexte et les *negatives samples*).

3.3 Évaluation du modèle implémenté

Malgré l'utilisation généralisée des *word embedding*, très peu de travaux théoriques expliquent ce qui est réellement capturé par ces représentations de mots. C'est pourquoi ce modèle est principalement évalué à l'aide de méthodes empiriques.

1. Par exemple, dans la phrase « Le chat de la voisine est roux », si le mot focus est « voisine » et que la fenêtre est de 2, le mot contexte sera tiré au hasard parmi les mots « de », « la », « est » et « roux ».

Avant de nous attaquer au jeu de données complet, nous avons évalué un premier corpus fictif. Nous avons associé dix couples (du type [voiture, camion]), à dix mots contexte différents ([véhicule, moto, ...]). Le corpus fictif est formé de 10 000 phrases composées chacune d'un mot d'un couple, de cinq mots du contexte et de trois mots bruits, tous tirés aléatoirement.

Nous avons retenu à ce stade trois méthodes pour évaluer les vecteurs obtenus en sortie du modèle :

— La *similarité cosinus*, qui consiste à calculer le cosinus de l'angle entre deux vecteurs. Le cosinus vaudra alors 1 pour deux vecteurs identiques, -1 pour deux vecteurs opposés et 0 pour deux vecteurs indépendants.
— Une *analyse en composantes principales* (*ACP*) qui consiste à projeter les vecteurs de mots sur un plus faible nombre d'axes afin de rendre l'information moins redondante et plus visuelle.
— L'algorithme *t-SNE*, ou *t-distributed Stochastic Neighbor Embedding*, une autre technique de réduction de dimension, plus efficace que l'ACP dans le cas d'un nombre important de données. L'algorithme se fonde cette fois-ci sur une interprétation probabiliste des proximités et est non-linéaire.

Nous avons donc mis en oeuvre ces trois techniques sur notre corpus fictif. Les résultats semblent concluants : la similarité cosinus montre bien une forte corrélation entre les mots focus et contexte du corpus initial et l'ACP et l'algorithme t-SNE permettent également de montrer graphiquement cette proximité (figure 1).

```
mot_plus_proche("grand", n = 50)
#[('énorme', 0.9914256427481623),
# ('taille', 0.9905528713008166),
# [...]
# ('vanille', 0.06068283530950071),
# ('salissures', 0.0539210063101789)]
```

4 Perspectives

4.1 Implémentation et évaluation

Nous travaillons actuellement à la mise en commun de nos implémentations individuelles afin d'aboutir à une version finalisée du modèle. Nous devrons ensuite faire tourner notre modèle sur les tweets mis à notre disposition. Cette étape sera également l'occasion d'optimiser les étapes chronophages du code si nécessaire.

Les vecteurs obtenus à l'issue de l'algorithme nous permettront ensuite de procéder à l'évaluation du modèle sur données réelles. Pour ce faire, en plus d'utiliser les données d'évaluation décrites ci-dessus, nous mobiliserons une base de données qui demande à des personnes d'indiquer si selon eux certains couples de mots sont proches ou éloignés (*Human Judgement Agreement*). Il faudra comparer cette similarité subjective aux résultats de notre modèle. Cette comparaison s'effectue grâce au calcul d'une *corrélation de Spearman*.

4.2 Analyse

Notre objectif final est de comparer des données produites par la statistique publique à ce qu'il est possible d'obtenir grâce aux tweets. Différents échanges avec l'Insee nous amènent à nous diriger vers une de ces différentes problématiques :

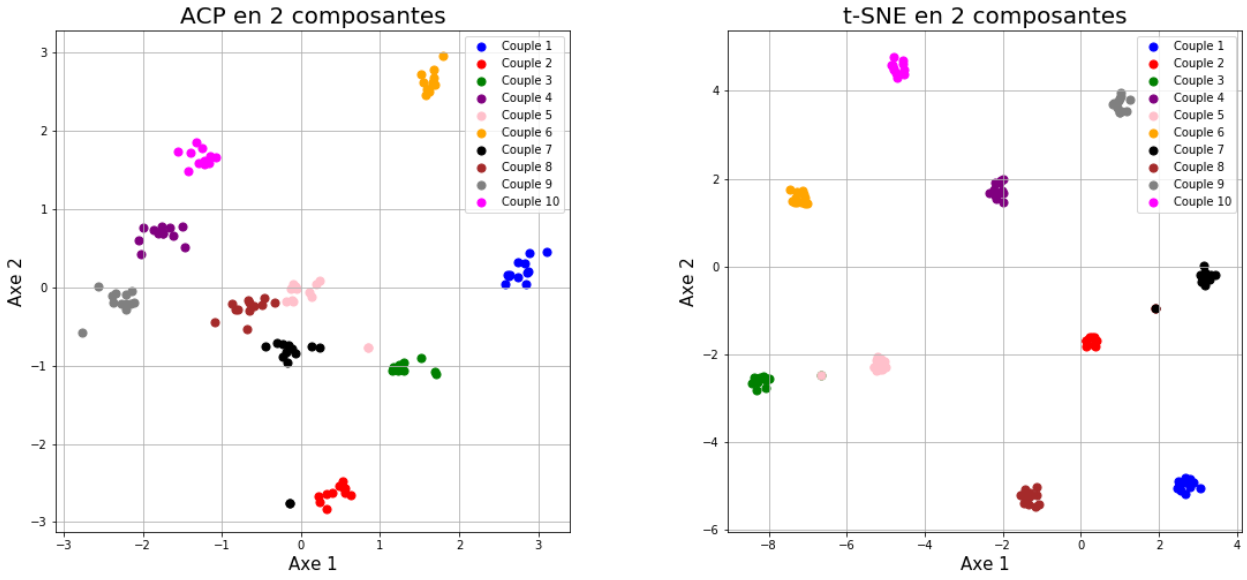

FIGURE 1 – Évaluation du modèle sur données fictives

- Les tweets permettent-ils d'obtenir des indicateurs synthétiques d'opinion des ménages proches de ceux produits dans la statistique publique (comme l'enquête de conjoncture Camme auprès des ménages, Insee) ?
- Est-il possible de prévoir (géographiquement, temporellement) le taux de chômage à partir des tweets ? (Srivastava, M., Nguyen, T. (2018))
- L'utilisation des tweets peut-elle améliorer la prévision du PIB ? (Bortoli, C., *et al* (2018))

Quelle que soit la problématique choisie à terme, nous tenterons de calculer des indicateurs synthétiques à l'aide de techniques d'analyse de sentiments. Nous devons également garder en tête les différentes limites des données à notre disposition : leur faible profondeur temporelle rendant complexe la prévision de données trimestrielles ou encore leur thématique se rapprochant certainement davantage des questions d'opinions des ménages plutôt que des informations liées aux entreprises (habituellement utilisées dans les prévisions macroéconomiques).

Références

Bortoli, C., Combes, S., Renault, T. (2018). Nowcasting GDP Growth by Reading Newspapers. Economie et Statistique / Economics and Statistics, 505-506, 17–33. https://doi.org/10.24187/ecostat.2018.505d.1964.

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space https://arxiv.org/pdf/1301.3781.pdf.

Srivastava, M., Nguyen, T. (2018). "Nowcasting" County Unemployment Using Twitter Data. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2737035.pdf.