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 supervisée 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 sous la direction de Tomas Mikolov, sont parmi les plus célèbres et sont ceux sur lesquels se concentreront 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* avec l'architecture Skip-Gram. Danns un second temps, l'objectif est de mettre en application ce modèle sur une base de données créée à partir de plusieurs millions de tweets écrits en France sur la période 2013-2017 et de mettre en oeuvre à partir de cela des techniques d'analyse de sentiments, afin d'arriver à comparer nos résultats avec certains indicateurs de l'Insee par exemple sur l'opinion des ménages.

3 Travail effectué

3.1 Compréhension du modèle

Nous avons débuté le projet en nous imprégnant du champ des méthodes de NLP et même de Deep-learning qui nous était jusqu'alors inconnu. Pour cela nous avons lu les articles présentés dans la bibliographies ainsi que d'autres articles de blogs, en particulier concernant l'implémentation sur python de ces modèles. Nous avons consacré une grande partie de notre temps dans la compréhension du modèle.

Afin de représenter les mots d'un corpus sous forme d'un vecteur, nous entraînons un réseau factiCe, consistant à prévoir le lien entre des mots *focus* et leur contexte, et de récupérer les poids issus de cet entraînement. Ce sont ces poids qui seront utilisés comme représentation vectorielle des mots.

Différentes architectures sont possibles avec le modèle, nous nous intéresserons ici à l'approche *Skip-gram*. Le principe est de partir d'un mot dit focus dans une phrase est de prédire, pour chaque mot du vocabulaire, la probabilité d'être dans le contexte de ce mot focus.

Nous pouvons résumer l'algorithme en 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 de notre vocabulaire et la dimension des vecteurs que nous désirons en sortie.
- L'agorithme parcourt ensuite chaque phrase du corpus de texte. Pour chaque phrase, un mot focus est sélectionné au hasard. Pour ce mot focus, un mot est alors tiré dans le contexte, c'est-à-dire que l'on selectionne un mot dans une fenêtre autour de notre mot focus. Ce couple sera alors l'objet de notre étude : on sélectionne la ligne de la matrice de poids en entrée correspondant à notre mot en entrée puis calculons un score pour les mots du vocabulaire grâce à la matrice en sortie. Ce score mesure le fait d'être dans le conxte et est transformé en probabilité, nous avons testé deux approches, via une fonction softmax et une fonction sigmoid. La fonction de coût est alors mise à jour en regardant la probabilité du mot contexte sur lequel on travaille, puis les matrices de poids sont mises à jour par descente de gradient.
- L'opération est répétée un certain nombre de fois fixé au préalable, on parle de nombre d'*epochs*.

En sortie, on a ainsi à disposition deux matrices de poids utilisée pour former nos vecteurs. Nous avons décidé d'utiliser la moyenne de ces deux matrices pour créer nos vecteurs finaux.

3.2 Implémentation du modèle

Après la compréhension du modèle et afin de nous approprier au maximum ces nouvelles méthodes, nous avons décidé d'implémenter tous les trois individuellement le modèle sur Python en utilisant la librairie Pytorch avant de mettre en commun nos codes. De cette manière, nous avons pu chacun approfondir les points qui nous semblaient les plus flous et arriver à une mouture de modèle qui compile.

L'implémentation du modèle s'est déroulée en deux étapes. En effet, nous avons commencé par implémenter une version plus simple du *word2vec* en utilisant une fonction *softmax*. Cela induit de calculer à chaque fois des informations sur tous les mots de notre corpus et donc, algorithmiquement, cette solution n'est pas viable sur les corpus trops gros. Les résultats étaient concluants sur de petites bases donc donc nous sommes passés à une version plus élaborée du modèle et plus optimisée. En effet, il n'est pas nécessaire de mettre à jour à chaque étape de l'algorithme toutes les lignes des matrices de poids, on peut seulement s'intéresser au mot contexte sur lequel on travail. Cependant, en faisant ça on sur-estime l'effet de ce mot, il faut donc également s'intéresser à des mots qui ne sont pas dans le contexte, qu'on appelle *negative sampling* pour lesquels on va chercher à minimiser l'impact dans notre fonction de coût. En partant de la version simplifiée du modèle que nous avons implémenté et en intégrant ces *negative samplings* dans notre fonction de coût, nous avons réussi à implémenter une version quasi-définitive du modèle qui est conforme au papier de Mikolov.

3.3 Evaluation du modèle implémenté

Malgré leurs utilisations presque généralisées, très peu de travaux théoriques expliquent ce qui est réellement capturé par ces représentations de mots. C'est pourquoi l'évaluation de l'efficacité de ce

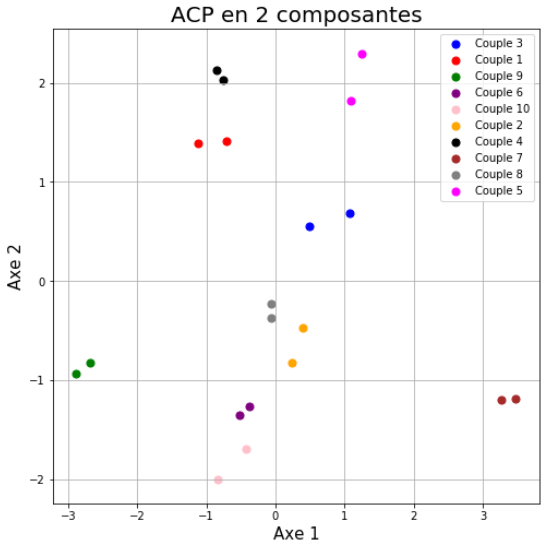

FIGURE 1 – ACP réalisée sur le corpus de données fictives

modèle ne peut se faire qu'à l'aide de méthodes empiriques. Nous avons donc réfléchi aux différentes manières qui s'offraient à nous pour mesurer l'efficacité de notre modèle ainsi implémenté. Avant de s'attaquer à notre jeu de données complet, nous avons effectué une évaluation sur des données fictives.

L'idée est de se dire que si l'on crée un corpus fictif, basé sur dix couples, pour lesquel on maîtrise entièrement le contexte, alors, en construisant des phrases à partir de cela et de mots bruits, nous devrions avoir en sortie des résultats concluants. Nous avons donc créé un corpus à partir de ces règles : nous nous sommes fixés dix couples, pour lequels on a associé dix mots contextes différents et nous avons formé 10 000 phrases de test en tirant aléatoirement, pour chaque phrase, un mot d'un couple, cinq mots du contexte et trois mots bruits. En mélangeant les mots tirés, nous avons ainsi obtenu un corpus fictif sur lequel nous avons pu travailler.

Nous avons retenu principalement trois méthodes d'évaluations pour étudier les vecteurs obtenus en sortie du modèle :

— La *similarité cosinus*, qui consiste à calculer le cosinus de l'angle entre deux vecteurs de dimension n et d'utiliser cette mesure comme similarité, ainsi un cosinus de 1 indiquera deux vecteurs identiques, -1 deux vecteurs opposés et 0 deux vecteurs indépendants.
— L'*ACP* qui consiste à projeter nos vecteurs sur des axes, appelés composantes principales, en maximisant la variance résiduelle du projeté.
— L'algorithme *T-SNE*, ou *t-distributed Stochastic Neighbor Embedding*, qui permis de compléter et même d'améliorer l'ACP pour les grandes dimensions. Il s'agit d'un algorithme stochastique qui permet l'apparition de clusters de points proches.

Nous avons donc mis en oeuvre ces trois techniques sur notre corpus fictif et les résultats ont été très concluants. Nous avons bien vu grâce à la similarité cosinus que les vecteurs de nos couples étaient très proches. L'ACP et l'algorithme t-SNE ont aussi permis de repérer les couples et les mots contextes différents.

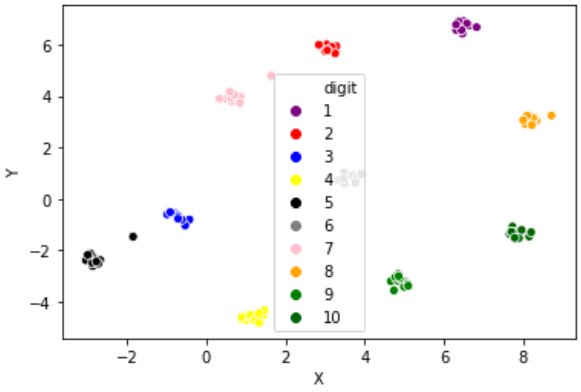

FIGURE 2 – t-SNE réalisée sur le corpus de données fictives

```
[('xl', 0.99155166189624),
 ('taille', 0.9825691349825234),
 ('énorme', 0.9754718206197113),
 ('s', 0.9733300846328123),
 ('m', 0.9729958071788166),
 ('petit', 0.960871955334316),
 ('nain', 0.9601336922743685),
 ('longueur', 0.9592792721624962),
 ('mesure', 0.9558256537073782),
 ('géant', 0.9323142118613931)]
```

FIGURE 3 – Mots les plus proches de “grand” dans le corpus de données fictives

4 Perspectives

4.1 Implémentation et évaluation

Maintenant que nous avons évalué notre modèle sur les données fictives, il nous reste encore quelques étapes à réaliser afin d'avoir un modèle "propre".

Tout d'abord, nous devons finir la mise en commun de nos trois codes afin de ne garder qu'une seule version du modèle que nous devrons ensuite faire tourner sur les données réelles qui sont à notre disposition. Nous allons procéder par étapes, d'abord en faisant tourner le modèle sur 100 000 tweets, puis 1 000 000, puis sur l'ensemble de la base. Une fois que le modèle aura tourné sur l'ensemble des tweets, nous pourrons récupérer nos vecteurs mais nous devrons aussi procéder à l'évaluation du modèle.

En plus de mettre en oeuvre les mêmes techniques que sur le corpus fictif, nous pensons également nous baser sur des résultats issus du *Human Judgment Agreement*. Il s'agit d'une expérience qui a été réalisées auprès de volontaires et qui a permis de mesurer une similarité entre des couples de mots du point de vue humain. L'idée serait alors de comparer les résultats obtenus par cette expériences et les réulstats obtenus par notre modèle.

4.2 Analyse

Tout le travail d'analyse à partir de la base de données exhaustive des tweets reste à effectuer. Une fois que nous aurons nos vecteurs qui représenteront les mots du corpus, l'idée est de pouvoir les utiliser afin d'en tirer quelque chose. Puisque notre groupe est formé de trois administrateurs de l'Insee, nous voulons essayer d'orienter l'analyse vers la statistique publique. Nous avons donc contacté différentes personnes à l'Insee afin de réfléchir à des applications directes de notre travail. L'Insee a déjà publié il y a quelques temps un article qui permet de prévoir le PIB à partir d'articles du Monde grâce à des méthodes de NLP. Bien sûr, notre base de données étant différente, il ne sera pas possible de réaliser exactement la même étude. Cependant, nous avons en tête différentes pistes :

- Nous pouvons regarder si l'analyse des tweets permet de retrouver des résultats proches de ceux réalisées par les enquêtes Insee (Camme, l'enquête de conjoncture auprès des ménages) ou de toute autre enquête de la statistique publique (comme le Baromètre d'opinion de la DREES) et d'essayer de comprendre les différences.
- Nous pouvons également faire des tests autour des questions de sondages électoraux/participations électorales à l'approche des municipales.
- Nous pouvons enfin nous intéresser au taux de chômage et essayer de le prévoir avec les tweets.

A priori, notre idée est plutôt de rester sur la comparaison avec des enquêtes comme l'enquêtes Camme de l'Insee, mais nous gardons bien en tête les limites liées au fait même d'utiliser des tweets et sur la profondeur temporelle de nos données. En effet, beaucoup d'enquêtes sont trimestrielles et nous ne pourrions alors qu'avoir qu'un faible nombre de points à notre disposition.