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 Évaluation du modèle implémenté

1.1 Comment évaluer le modèle ?

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. Nous allons décrire dans cette partie 1.1 quelques méthodes que nous avons retenues pour évaluer la qualité des vecteurs-mots obtenus.

1.1.1 Distance entre deux mots

L'un des enjeux principaux du modèle étant de pouvoir estimer la proximité entre deux vecteurs-mots, nous pouvons tout d'abord mesurer cette dernière par des calculs de distance.

Il existe différents types de distances. Chacune d'elles possède des propriétés intéressantes et s'adaptent plus ou moins bien au problème traité. Nous avons ici retenu deux distances classiquement utilisées :

— **la distance euclidienne** $d_e(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\|_2$
La longueur du vecteur mot, captée dans le cas de la distance euclidienne, est positivement corrélée à la fréquence d'apparition du mot (Bortoli, C., *et al* (2015)). Cette information peut s'avérer utile dans l'analyse de la signification des mots, notamment lorsque l'on effectue des opérations sur les vecteurs (comme l'exemple de $\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italie} = \overrightarrow{Rome}$ dans Mikolov, T., *et al* (2013))
Toutefois, cette dépendance à la fréquence d'apparition peut également fausser l'analyse. C'est pourquoi nous avons choisi, par la suite, de normaliser les vecteurs.
$d_e(\vec{u}, \vec{v}) = \left\| \frac{\vec{u}}{\|\vec{u}\|_2} - \frac{\vec{v}}{\|\vec{v}\|_2} \right\|_2$

— **la similarité cosinus** $d_c(\vec{u}, \vec{v}) = \frac{\vec{u}.\vec{v}}{\|\vec{u}\|_2 \|\vec{v}\|_2}$.
La similarité cosinus correspond au produit scalaire entre les deux vecteurs normalisés. Elle mesure ainsi l'angle formé entre deux vecteurs-mots.
C'est la distance que de nombreux papiers fondateurs de la méthode *Word2Vec* (comme Mikolov, T., *et al* (2013) ou Levy, O., Golberg, Y. (2015)) utilisent avec l'argument selon lequel les mots apparaissant dans des contextes similaires sont groupés dans la même direction durant l'entraînement. Une similarité est proche de +1 si deux mots sont positivement reliés (proches), de -1 s'ils sont négativements reliés (éloignés) et de 0 s'ils ne sont pas « reliés ».

Il est toutefois délicat d'interpréter une similarité proche de -1. On pourrait intuitivement penser à des antonymes, comme « grand »et « petit », mais en pratique, les antonymes sont susceptibles d'apparaître dans des contextes semblables et sont donc bien souvent positivement corrélés.

1.1.2 Analyse en Composantes Principales

Une fois le modèle *Word2Vec* entraîné, nous obtenons des *word-embeddings* pour chacun de nos mots, représentés par des vecteurs de grandes dimensions (20, 50 ou même supérieures à 100).

Dès lors, il devient complexe de bien observer la proximité entre deux mots. C'est pourquoi il devient utile de mobiliser des méthodes de réduction de dimensions comme l'analyse en composantes principales (ACP). L'objectif premier de cette méthode est en effet de projeter un nuage de points sur un espace de dimension inférieure afin de rendre l'information moins redondante et plus visuelle, tout en étant le plus proche possible de la réalité.

Considérons le cas où nous disposons de n individus (dans notre cas les mots) et de p variables (dans notre cas, leurs composantes ou dimensions issues du modèle *Word2Vec*). On note $X = (x_{ij})$ la matrice de taille (n, p) des données brutes, où x_{ij} représente la valeur de la j-ème variable pour le i-ème individu. Afin de donner à chaque individu le même poids, nous centrons et réduisons les colonnes de notre matrice de données. On notera par la suite $Z = (z_{ij})$ la matrice des données centrées et réduites.

La construction des axes de l'ACP est faite par projection orthogonale. Nous utilisons ici le produit scalaire $< x, y >_N = x\,{}^tNy$ avec la métrique $N = diag(\frac{1}{n}, ..., \frac{1}{n})$. Ainsi, la projection orthogonale d'un individu i (vecteur ligne) z_i sur une droite de vecteur directeur v vaut ${}^tz_iv$ et les coordonnées de projection des n individus valent Zv.

Les vecteurs directeurs des axes sont définis de manière à maximiser la dispersion du nuage (son inertie) des individus projetés et conserver ainsi au mieux les distances entre les individus. L'inertie se définit comme

$$I(Z) = \frac{1}{n}\sum_{i=1}^{n} d_e^2(z_i, \bar{z}) = \sum_{i=1}^{n} var(z^j) = p$$

avec $d_e(z_i, z_{i'})$ la distance euclidienne entre deux individus z_i et $z_{i'}$: $d_e(z_i, z_{i'}) = \sum_{j=1}^{p} (z_{ij} - z_{i'j})^2$ [1].

On trouve tout d'abord le vecteur directeur v_1 qui orientera le premier axe de l'ACP grâce au programme suivant :

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}}\ Var(Zv) = \underset{\|v\|=1}{\operatorname{argmax}}\ v\,{}^tRv$$

où $R = Var(Z) = \frac{1}{n}Z\,{}^tZ$ est la matrice des corrélations entre les p variables. La norme du vecteur v se calcule dans ce nouvel espace comme $\|v\| = \sqrt{< v, v >} = v^tv = \sqrt{\sum_{i=1}^{p} v_i^2}$

1. Nous travaillons ici dans le cadre d'une ACP normée où la matrice X a été centrée puis réduite. La réduction de X a modifié les distances initiales entre individus ($d_e(z_i, z_{i'}) \neq d_e(x_i, x_{i'})$). Cela n'aurait pas été le cas si la matrice Y avait été uniquement centrée (ACP non normée).

Puis, on choisit v_2 orthogonal à v_1 tel que l'inertie soit toujours maximisée

$$v_2 = \underset{\|v\|=1,\, v \perp v_1}{\operatorname{argmax}} \; Var(Zv)$$

En procédant de manière séquentielle, on obtient $q < r$ axes orthogonaux avec $r = rg(Z)$ et q choisi par le statisticien [2].

On peut montrer que $\forall k < q$:

— v_k est un vecteur propre associé à la k^{e} valeur propre λ_k de R
— la composante principale Zv_k est centrée et $V(Zv_k) = \lambda_k$
— Les Zv_k ne sont pas corrélés entre eux

On obtient alors la matrice $F = ZV$ des nouvelles coordonnées factorielles des individus, avec $V = (v_1, \ldots, v_q)$ la matrice des vecteurs propres.

Nous utilisons ici l'ACP en vue d'identifier les individus (ici, nos mots) qui sont proches. Pour ce faire, il suffit de représenter les coordonnées factorielles de la matrice F dans des repères, en général en 2 dimensions pour une question de lisibilité. Deux mots apparaissant dans des contextes similaires seront proches sur ce repère et orientés dans la même direction.

Enfin, pour juger de la qualité de la réduction de dimension, on calcule souvent la proportion de l'inertie totale expliquée par les q premières composantes principales.

$$\frac{V(F)}{I(Z)} = \frac{\sum_{i=1}^{q} \lambda_i}{p}$$

1.1.3 Algorithme t-distributed Stochastic Neighbor Embedding

Bien que l'ACP soit une première manière de résumer l'information contenue dans nos vecteurs, elle présente des limites, notamment dans les vecteurs aux trop grandes dimensions, pour lesquels l'inertie des premiers axes de l'ACP peut se révéler faible.

Pour combler ces lacunes, un autre algorithme de réduction de dimension peut être utilisé, celui dit du t-distributed Stochastic Neighbor Embedding. Contraitement à l'ACP, cet algorithme est stochastique et non-linéaire et il favorise l'apparition de groupes de mots proches. Sa philosophie demeure cependant identique : représenter dans un espace à dimension réduite notre nuage de points de manière à repérer les mots proches.

La première étape de l'algorithme consiste à calculer les similarités entre les n vecteurs-mots $(x_i)_{i=1\ldots n}$. La similarité entre x_i et x_j se mesure comme étant la probabilité conditionnelle $p_{j|i}$ de choisir x_j comme voisin de x_i, si les voisins étaient tirés au sort selon une loi $\mathcal{N}(x_i, \sigma_i)$ [3]

$$p_{j|i} = \frac{\exp(-\frac{(d_e(x_i - x_j))^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{(d_e(x_i - x_k))^2}{2\sigma_i^2})}$$

2. Différentes méthodes existent afin de déterminer le q optimal, comme la règle de Kaiser ou encore celle du coude.

3. σ_i doit être calculé de manière à adapter la loi conditionnelle aux données. Une faible dispersion autour de x_i entraînera un σ_i faible et réciproquement. Il s'agit de trouver le σ_i qui minimise ce qui est appelé en théorie de l'information la « perplexité », c'est-à-dire un indicateur qui décrit à quel point une distribution de probabilité réussit à prédire un échantillon.

La seconde étape de l'algorithme consiste à trouver le nouvel espace de projection à faible nombre de dimensions. On appellera g_i les x_i projetés dans cet espace que l'on cherche à déterminer. De la même manière que précédemment, on exprime des probabilité conditionnelles $q_{j|i}$ en fonction des g_i mais qui suivent cette fois-ci une distribution de *Student* - d'où le nom de l'algorithme - plutôt qu'une loi gaussienne [4].

$$q_{j|i} = \frac{(1 + (d_e(g_i - g_j))^2)^{-1}}{\sum_{k \neq i} (1 + (d_e(g_i - g_k))^2)^{-1}}$$

Afin d'obtenir les g_i, on minimise, par descente de gradient, la divergence de Kullback–Leibler entre les distributions de probabilité P et Q :

$$KL(P, Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \qquad \text{avec} \qquad p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

Comme dans l'algorithme de l'ACP, l'algorithme de t-SNE nous permet d'obtenir une nouvelle projection des x_i. Il faut cependant analyser avec précaution ses résultats. L'algorithme n'étant pas linéaire, l'interprétation de la taille des *clusters* obtenus ou de la distance qui les sépare n'est alors pas directe.

1.1.4 Human judgement

Les *word-embedding* obtenus par *Word2Vec* sont censés regrouper les mots qui apparaissent dans un contexte similaire. Une dernière façon de le vérifier peut être de comparer les résultats obtenus au point de vue des êtres humains sur la proximité qu'il peut y avoir entre différentes paires de mots.

Pour ce faire, nous utilisons une base de données d'une soixantaine de paires de mots auxquelles sont associés des scores de similarité allant de 0 (aucun lien entre les mots) à 4 (mots identiques).

mot 1	mot 2	similarité
corde	sourire	0,00
midi	ficelle	0,00
...	...	...
corde	ficelle	3,33
...	...	...
automobile	auto	3,94
coq	coq	4,00

Nous calculons ensuite la corrélation de Spearman entre les similarités cosinus de ces différentes paires issues de notre modèle (notées ici $(X_i)_{i=1..n}$) et les scores proposés ci-dessus par des êtres humains (notés ici $(Y_i)_{i=1..n}$).

La corrélation de Spearman est égale au coefficient de corrélation de Pearson calculé sur les variables de rang.

$$r_s = \text{corr}(\text{rg}_X, \text{rg}_Y) = \frac{\text{cov}(\text{rg}_X, \text{rg}_Y)}{\sigma_{\text{rg}_X} \sigma_{\text{rg}_Y}}$$

4. Dans un espace à faible dimension, la dispersion des vecteurs est réduite. La distribution de Student possède des queues plus épaisses que la loi normale, ce qui permet de mieux différencier les vecteurs distants des vecteurs similaires.

La variable de rang rg_{X_i} est définie telle que $\text{rg}_{X_i} = j \iff X_i = X_{(j)}$ (X_i est la $j$ème plus petite variable).

Pour tester la significativité de ce coefficient, nous réalisons un test de Student avec

$$t = r\sqrt{\frac{n-2}{1-r^2}} \overset{H_0}{\sim} St(n-2)$$

1.2 Choix des « meilleurs » paramètres pour le modèle

Expliquer ici notre cheminement (éventuellement foireux) pour choisir les différents paramètres du modèle avec 100k tweets : en particulier learning rate + window. Parler aussi des autres paramètres qui peuvent jouer mais dont on n'évalue pas l'effet. Expliquer le rôle central joué par la seed et pourquoi et la manière dont nous procédons pour créer des pseudos intervalles de confiance pour répondre à ce problème. Bien indiquer les limites de cette partie en raison du temps de calcul (à préciser) et de la puissance machine et bien dire que l'objectif était de nous initier à la sélection de paramètres sans avoir les capacités techniques de pouvoir réaliser un travail très poussé. Voir éventuellement plus tard si on peut tester paramètres aussi sur corpus fictif

1.3 Evaluation sur un corpus fictif

Kim : Montrer les 3 évaluations du corpus fictif (tous sauf human judgement) dans l'objectif de montrer que notre modèle semble bien implémenté et robuste !

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 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).

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

1.4 Evaluation sur le corpus final [PLUS TARD]

Montrer les 4 évaluation du corpus final de 1 000 000 de tweets. Dire en quoi le modèle semble bon, en quoi il semble pas bon (du style bon pour prédire les jours de la semaine et les chiffres de l'an dernier).

Références

Schakel, A. M., Wilson, B. J. (2015). Measuring Word Significanceusing Distributed Representations of Words. arXiv:1508.02297. https://arxiv.org/pdf/1508.02297v1.pdf.

Levy, O., Golberg, Y. (2015). Neural Word Embedding as Implicit Matrix Factorization. https://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf.

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