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)

Table des matières

1 Partie 1

2 Évaluation du modèle implémenté

2.1 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 2.1 quelques méthodes que nous avons retenues pour évaluer la qualité des vecteurs-mots obtenus.

2.1.1 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 (Schakel & Wilson (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, Chen, Corrado *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, Chen, Corrado *et al* (2013) ou Levy & Golberg (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 négativement 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.

2.1.2 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 $< x, y >_N = x^t N y$ 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 ${}^t z_i v$ 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 :

$$v_1 = \underset{\|v\|=1}{\text{argmax}}\ Var(Zv) = \underset{\|v\|=1}{\text{argmax}}\ v\,{}^t Rv$$

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

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

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

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. 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).

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

2.1.3 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 $(x_i)_{i=1...n}$. 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]

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

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.

2.1.4 Jugement humain

Les *word-embedding* obtenus par *Word2Vec* sont censés regrouper les mots qui apparaissent dans un contexte similaire. Une dernière façon de vérifier la qualité de nos vecteurs-mots est de les comparer

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.

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.

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

TABLE 1 – Base de données de jugement humain

à un jugement humain. Pour ce faire, nous utilisons la liste de référence RG-65 pour le français [5] (Boumedyen Billami & Gala (2017)). Elle contient 65 paires de noms communs (tableau 1) évaluées sur une échelle de 0 (non liés) à 4 (très liés).

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.

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

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

Pour tester la significativité de ce coefficient, nous utilisons la loi sous (H_0) de la statistique de test $z = \mathrm{arctanh}(r_s) = \frac{1}{2}\ln\frac{1+r}{1-r} \overset{H_0}{\sim} \mathcal{N}(0, \frac{1}{n-3})$ et obtenons l'intervalle de confiance suivant :

$$IC_\alpha(r_s) = \left[\tanh\left(z - \frac{q_{1-\frac{\alpha}{2}}}{\sqrt{n-3}}\right), \tanh\left(z + \frac{q_{1-\frac{\alpha}{2}}}{\sqrt{n-3}}\right)\right]$$

avec $q_{1-\frac{\alpha}{2}}$ le quantile d'ordre $1-\frac{\alpha}{2}$ d'une loi $\mathcal{N}(0,1)$.

2.2 Evaluation sur un corpus fictif

Avant de nous attaquer au jeu de données complet décrit plus bas, nous avons évalué un premier corpus fictif afin de nous assurer de la robustesse du modèle implémenté. 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 ensuite mis en œuvre les différentes techniques d'évaluation [6] présentées en partie 2.1 sur les *word-embeddings* obtenus grâce à ce corpus fictif.

5. Le RG-65 a fait appel à 18 évaluateurs humains. La base, initialement mobilisée dans un article anglophone (Rubenstein & Goodenough (1965)) a été traduite de l'anglais.

6. à l'exception de la méthode par « jugement humain » puisque le corpus est ici créé fictivement par ordinateur sans prêter attention au réel sens des mots.

mot	similarité cosinus
énorme	0,991
taille	0,991
...	...
vanille	0,061
salissures	0,054

TABLE 2 – Mots les plus proches de « grand » par similarité cosinus

Note : Paramètres utilisés : ep = 50 / lr = 0,01 / w = 5 / dim = 10.

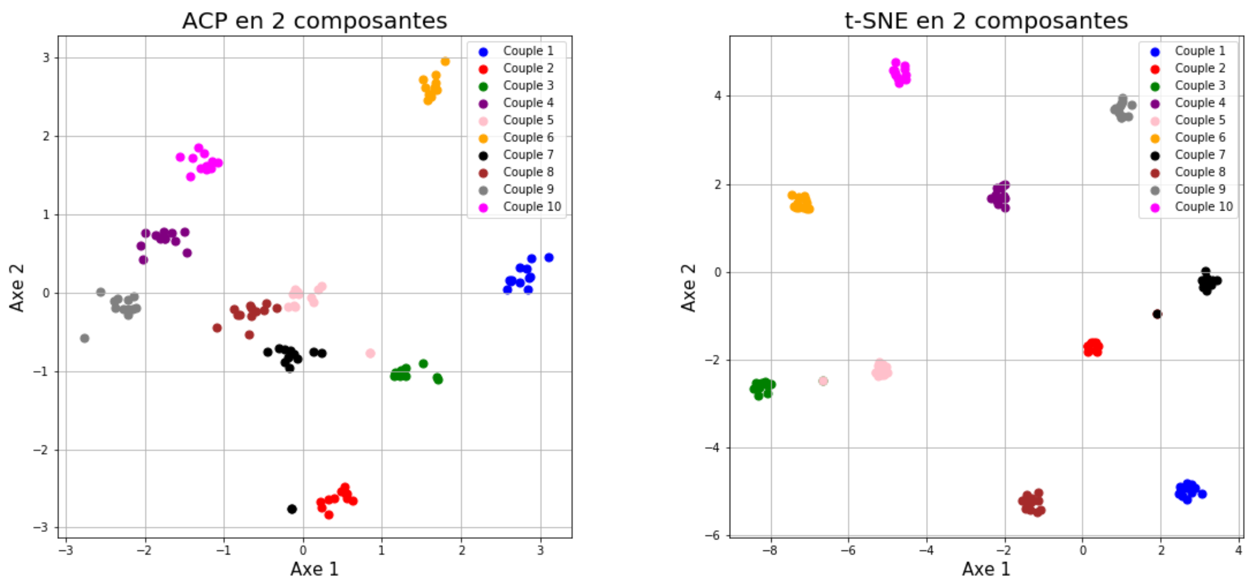

FIGURE 1 – Évaluation du modèle sur données fictives

Note : Paramètres utilisés : ep = 50 / lr = 0,01 / w = 5 / dim = 10.

Les résultats semblent concluants : la similarité cosinus montre bien une forte corrélation entre les mots focus et contexte du corpus initial et une faible corrélation avec les mots bruits (tableau 2). L'ACP et l'algorithme t-SNE permettent également de montrer graphiquement cette proximité (figure 1). Les clusters apparaissent de manière plus évidente avec t-SNE.

2.3 Choix des meilleurs hyperparamètres pour le modèle

Une fois nous être assurés de la bonne implémentation du modèle (partie 2.2) grâce au corpus fictif, nous nous sommes attachés à identifier les hyperparamètres les plus pertinents au regard des données dont nous disposons.

Ces données correspondent à un ensemble de 1,3 million de tweets [7] postés en France entre 2014 et 2017, supposés être représentatifs de l'ensemble de tweets nationaux publiés durant cette période.

Le modèle *Word2Vec* version CBOW, décrit en partie 1, fait en effet intervenir un certain nombre d'hyperparamètres parmi lesquels :

— *ep* : le nombre d'« *epochs* »

7. Ces tweets, achetés à twitter, sont la propriété de l'Inria.

— lr ou α : le « *learning rate* », ou taux d'apprentissage
— w (*window*): la taille de la fenêtre de sélection des mots contextes
— *dim* : la dimension des vecteurs-mots (ou *word-embeddings*)

Or, la performance de nombreuses méthodes de *machine learning*, dont *Word2Vec*, dépend fortement des valeurs choisies pour ces paramètres, ces valeurs étant elles-mêmes très dépendantes des données mobilisées.

Même si les méthodes d'optimisation bayésiennes deviennent de plus en plus performantes pour optimiser la valeur de ces hyperparamètres en tenant compte de leurs interactions (Hutter, Hoos & Leyton-Brown (2014)), ce choix s'effectue régulièrement de manière empirique, en testant différentes valeurs d'hyperparamètres sur les données mobilisées. C'est l'approche que nous retenons ici.

Le package `Gensim` (« Generate Similar »), dans lequel la méthode *Word2Vec* est implémentée, est un des outils actuels les plus robustes et performants [8] pour la modélisation sémantique non supervisée (Řehůřek & Sojka (2010)).

Nous avons choisi de mobiliser `Gensim` dans la suite de ce rapport, en parallèle du modèle que nous avons implémenté, en raison de son temps d'exécution bien plus rapide [9]. Cette rapidité d'exécution nous a permis de réaliser des tests d'hyperparamètres plus nombreux.

Pour réaliser ces tests, nous avons fait tourner le modèle *Word2Vec* plusieurs fois en modifiant un à un les paramètres. Nous avons ensuite évalué ces différents modèles par la méthode du « jugement humain » (cf. partie 2.1.4) en comparant la mesure de la similarité cosinus [10] entre deux mots obtenue à partir de notre modèle à l'évaluation subjective de cette proximité par des individus. En outre, un même modèle est lancé six fois (six « seeds » différentes) afin de construire des intervalles de confiance de la matière décrite précédemment, en empilant les six échantillons de mesure de proximités correspondant aux six implémentations d'un même modèle [11].

2.3.1 Nombre d'epochs, taille de fenêtre et taux d'apprentissage

Pour cette première série de tests d'hyperparamètres, nous avons fixé la dimension des word-embedding à 50 [12] et évalué l'impact du nombre d'épochs, de la taille de la fenêtre et du taux d'apprentissage (figure 2) .

2.3.1.1 Le nombre d'epochs

Le nombre d'epochs à un effet net. Passer de 10 à 100 epochs fait nettement augmenter le score de corrélation de Spearman entre données subjectives et données en sortie du modèle.

8. Grâce à sa dépendance à `NumPy`, `Gensim` puise dans des bibliothèques de bas niveau. Ainsi, alors que le code de haut niveau est du Python, c'est en fait du Fortran et du C hautement optimisés qui sont utilisés, ce qui rend `Gensim` bien plus performant que `PyTorch` que nous avons utilisé pour implémenter le modèle décrit en partie 1.

9. A titre d'exemple, alors qu'une epoch sur l'ensemble des tweets met une vingtaine d'heures à tourner pour « notre » modèle, elle met 1 minute via `Gensim`.

10. Nous avons également évalué les modèles en utilisant (l'inverse de) la distance euclidienne à la place de la similarité cosinus. L'effet des paramètres devient alors bien moins clair et la performance du modèle est inférieure, ce va dans le sens de l'utilisation plus fréquente de la méthode de la similarité cosinus dans la littérature.

11. Pour chaque modèle, nous calculons les statistiques de rang des 65 paires de mots de la base de jugement humain ainsi que le rang des similarités cosinus des mots obtenus en sortie du modèle. Nous réalisons ces actions pour les six implémentations du même modèle et empilons les résultats obtenus. C'est à partir de cette base empilée de 6x65 lignes moins les données manquantes que nous calculons chaque intervalle de confiance selon la formule décrite en partie 2.1.4.

12. En réalisant les mêmes tests sur uniquement 100 000 tweets, puis en testant une dimension de word-embedding de 20, les effets observés et commentés ici se confirment.

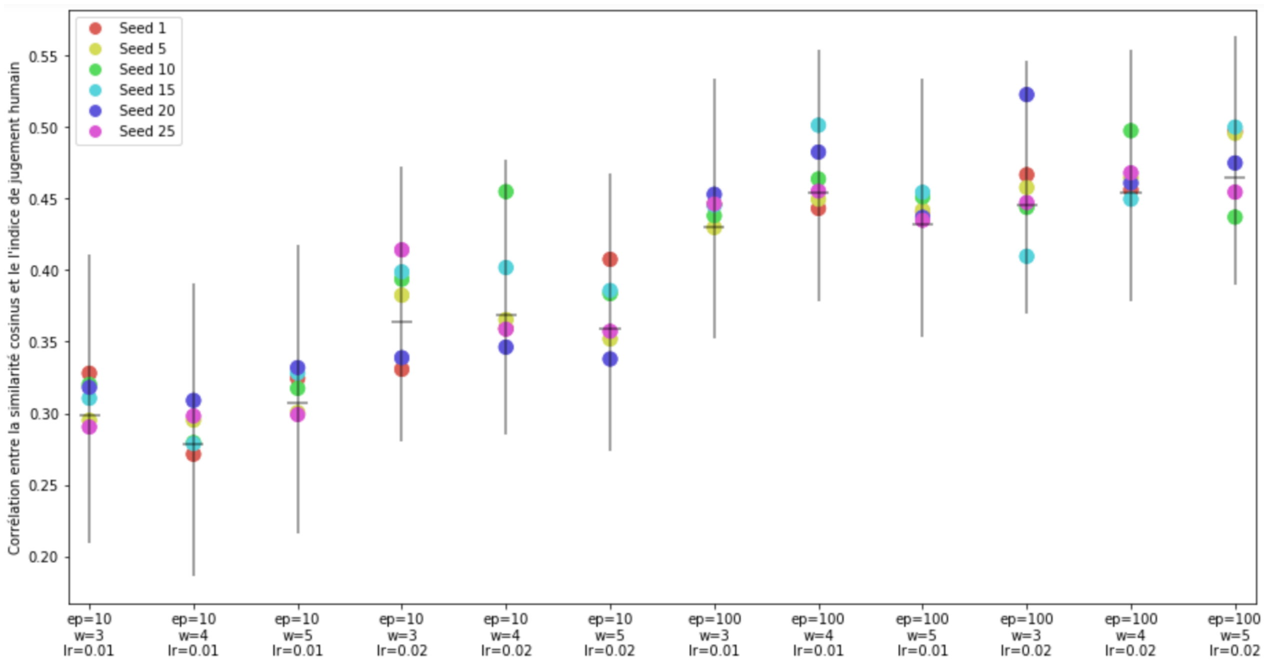

FIGURE 2 – Tests d'hyperparamètres : epochs, fenêtre et taux d'apprentissage

Note : Paramètres utilisés : dim = 50
Le trait horizontal correspond au coefficient de Spearman calculé sur les échantillons empilés des six modèles et la barre verticale à l'intervalle de confiance associé.

➔ Nous retenons alors le paramètre **ep = 100**.

2.3.1.2 Le taux d'apprentissage

La valeur 0,02 semble donner systématiquement de meilleurs résultats que 0,01. En réalisant davantage de tests de taux d'apprentissage en fixant les autres hyperparamètres, les différents taux d'apprentissage présentent des performances similaires[13].

➔ Nous retenons alors le paramètre **lr = 0,2**.

2.3.1.3 La taille de la fenêtre

La taille de la fenêtre ne semble pas jouer un rôle majeur, et dépend beaucoup des autres paramètres choisis.

Certains travaux (Levy & Golberg (2014)) indiquent que, suivant la taille de fenêtre choisie, les informations capturées sont différentes. Cela pourrait expliquer la complexité de choisir la « meilleure » taille de fenêtre. Alors que les « grandes » fenêtres capturent des informations sur le domaine du mot (autres mots de tout type étant utilisés dans des discussions connexes), les « petites » fenêtres saisissent davantage le mot en lui-même (ses extensions, synonyme, lui sont alors proches). La valeur de 4 représente une taille de fenêtre « ni trop grande ni trop petite » et qui présente de bons résultats dans la plupart des tests effectués.

13. En fixant les paramètres dim = 50, ep = 100 et w = 4 (celles du modèle retenu *in fine*), et en testant les taux d'apprentissage 0,005, 0,01, 0,02, 0,03 et 0,04, les valeurs moyennes des corrélations s'échelonnent entre 0,41 et 0,48, soit des valeurs proches.

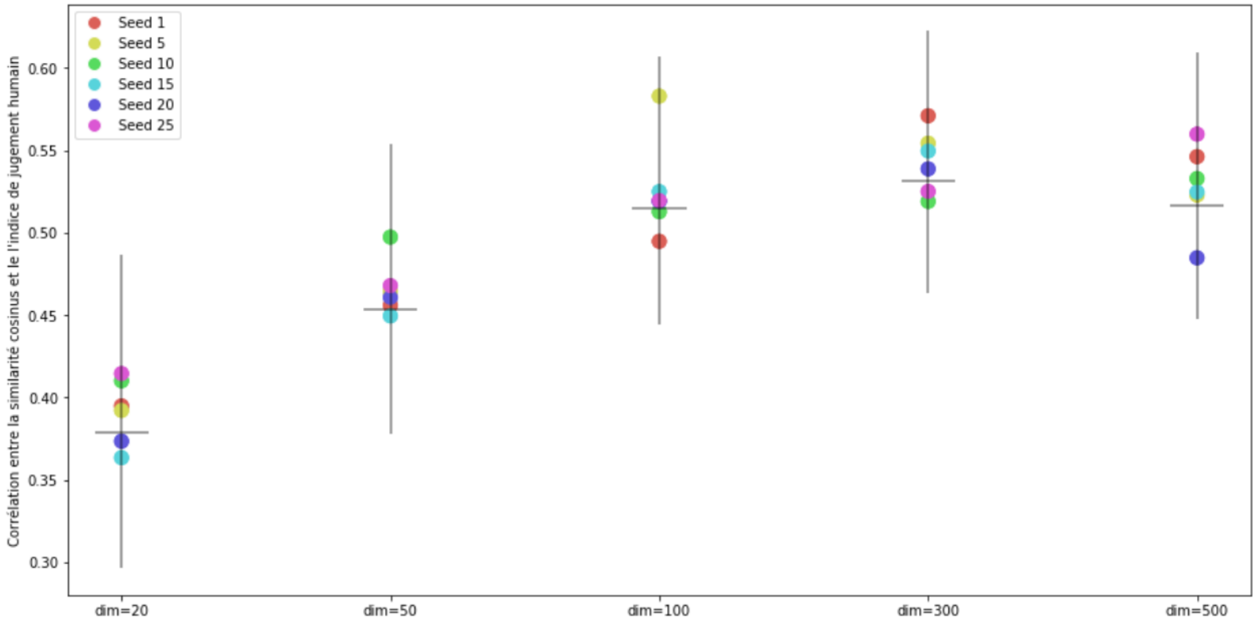

FIGURE 3 – Tests d'hyperparamètres : dimension des word-embedding

Note : Paramètres utilisés : ep = 100 / w = 4 / lr = 0,02.
Le trait horizontal correspond au coefficient de Spearman calculé sur les échantillons empilés des six modèles et la barre verticale à l'intervalle de confiance associé.

➔ Nous retenons alors le paramètre **w = 4**.

2.3.2 Dimension des vecteurs-mots

On cherche cette fois-ci à évaluer l'effet de la dimension des *word-embedding*. Selon certains papiers (comme Pennington, Socher & Manning (2014)), la qualité des représentations vectorielles s'améliore à mesure que l'on augmente la taille du vecteur, mais seulement jusqu'à atteindre 300 dimensions [14]. Après 300 dimensions, la qualité des vecteurs commence à diminuer et le temps de calcul augmente considérablement.

En pratique, en comparant l'effet de la dimension des vecteurs (modèle fixé à ep = 100, w = 4 et lr = 0,02), on observe bien une augmentation de l'efficacité du modèle jusqu'en dimension 300 et une efficacité moindre en dimension 500 (figure 3). Bien que l'efficacité du modèle semble meilleure en dimension 300, la dimension 100 améliore la rapidité de l'algorithme, pour des résultats d'une qualité similaire.

➔ Nous retenons alors le paramètre **dim = 100**.

14. La dimension des vecteurs doit également être adaptée à la taille du vocabulaire. Un des articles fondateurs de word2vec (Mikolov, Chen, Corrado *et al* (2013)) recommande donc d'augmenter à la fois la dimension des vecteurs et la quantité de données d'apprentissage. Par exemple, avec un vocabulaire d'une centaine de mots, il serait inefficace d'utiliser des projections en grande dimension (risque de surapprentissage).

bonjour *(669 apparitions)*	**femme** *(264 apparitions)*	**1** *(765 apparitions)*	**samedi** *(203 apparitions)*
😘 (0,59)	quelle (0,49)	5 (0,55)	soir (0,57)
😊 (0,59)	cette (0,46)	mois (0,51)	vivement (0,51)
merci (0,54)	une (0,44)	10 (0,49)	demain (0,50)
nuit (0,48)	vie (0,44)	2 (0,48)	end (0,48)
bisous (0,47)	grippe (0,44)	top (0,48)	weekend (0,47)
bonne (0,47)	belle (0,43)	depuis (0,47)	matin (0,45)
☺️ (0,46)	ma (0,43)	saison (0,46)	jeudi (0,45)
vous (0,46)	magnifique (0,43)	ans (0,44)	prochain (0,43)
plaisir (0,44)	nouvelle (0,43)	jours (0,43)	week (0,43)
allez (0,43)	vidéo (0,39)	3 (0,43)	🎉 (0,42)

TABLE 3 – 10 plus proches voisins par similarité cosinus avec « notre » modèle

Note : Paramètres utilisés : ep = 80 / w = 4 / lr = 0,02 / dim = 100 / base : 100 000 tweets
La similarité cosinus de chaque paire de mots est renseignée entre les parenthèses.

2.4 Evaluation sur le corpus final

2.4.1 Avec « notre » modèle

Nous avons ensuite fait tourner le modèle que nous avons implémenté en utilisant les paramètres retenus précédemment [15] mais uniquement sur 100 000 tweets et 80 epochs pour des questions de temps de calcul [16].

Les résultats obtenus semblent relativement satisfaisants. La recherche des plus proches voisins par similarité cosinus (dont quelques exemples sont illustrés en tableau 3) donne des résultats proches de l'intuition.

Par ailleurs, le coefficient de Spearman entre la similarité cosinus des mots obtenus et le jugement humain est de 0,571 (p-valeur : 4,1 %). Toutefois, ce bon résultat est à considérer avec précaution puisque seuls 13 des couples de mots de la base RG-65 ont été reconnus dans le corpus de 100 000 tweets que nous utilisons ici.

Enfin, les représentations graphiques des positions des mots via des ACP et les sommes vectorielles sur les mots [17] donnent des résultats bien moins concluants que le modèle `Gensim` entraîné sur l'ensemble des tweets (cf. partie 2.4.2)

2.4.2 Avec le modèle `Gensim`

Le modèle `Gensim` [18] donne des résultats encore plus convaincants que précédemment, ayant été davantage entraîné, et sur un corpus plus fourni (ensemble des tweets). En effet, les vecteurs-mots en sortie du modèle `Gensim` sur l'ensemble des tweets (figure 4) sont davantage répartis dans l'ensemble du plan, alors que les mots en sortie du modèle que nous avons implémenté sur 100 000 tweets sont

15. w = 4, lr = 0,02 et dim = 100
16. près de 18h.
17. comme l'exemple de $\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italie} = \overrightarrow{Rome}$ dans Mikolov, Chen, Corrado *et al* (2013)
18. w = 4, lr = 0,02, dim = 100 et ep = 100.

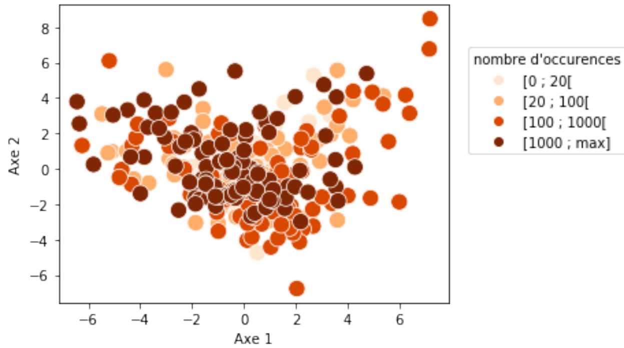

FIGURE 4 – Position des mots en fonction de leur nombre d'occurrences (Modèle `Gensim`)

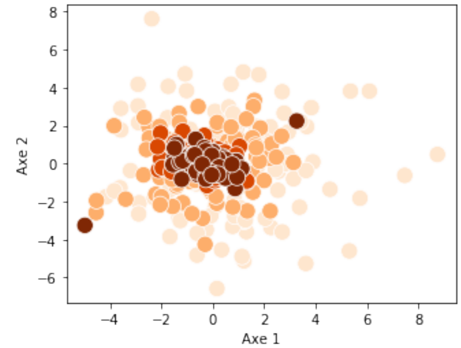

FIGURE 5 – Position des mots en fonction de leur nombre d'occurrences (« notre » modèle)

Note : Paramètres utilisés : ep = 100 (gauche) ou 80 (droite) / w = 4 / lr = 0,02 / dim = 100.
Méthode utilisée : ACP, deux premiers axes.

bonjour *(17 043 apparitions)*	**femme** *(6 177 apparitions)*	**1** *(21 055 apparitions)*	**samedi** *(4 917 apparitions)*
bonsoir (0,85)	fille (0,86)	2 (0,65)	vendredi (0,88)
bjr (0,75)	copine (0,74)	3 (0,64)	jeudi (0,86)
hello (0,71)	meuf (0,71)	6 (0,63)	lundi (0,83)
salut (0,66)	demoiselle (0,66)	4 (0,62)	mercredi (0,83)
coucou (0,55)	nana (0,66)	7 (0,60)	dimanche (0,83)
transmets (0,49)	nièce (0,66)	5 (0,58)	mardi (0,76)
désagrément (0,48)	sœur (0,65)	9 (0,58)	demain (0,72)
avezvous (0,48)	barbe (0,65)	8 (0,56)	barathon (0,56)
bettembourg (0,48)	maman (0,64)	1e (0,55)	22h45 (0,55)
hey (0,47)	princesse (0,64)	34 (0,53)	20h (0,54)

TABLE 4 – 10 plus proches voisins par similarité cosinus avec le modèle `Gensim`

Note : Paramètres utilisés : ep = 100 / w = 4 / lr = 0,02 / dim = 100 / base : ensemble des tweets
La similarité cosinus de chaque paire de mots est renseignée entre les parenthèses.

répartis en fonction de leur nombre d'occurrences, les mots les moins fréquents n'ayant probablement pas (ou peu) été entraînés (figure 5).

Le coefficient de Spearman a une valeur semblable à précédemment : 0,495 mais sa p-valeur est proche de 0 % et, cette fois-ci, 52 des couples de mots de la base RG-65 ont été reconnus dans le corpus de tweets.

Les 10 plus proches voisins calculés par similarité cosinus (tableau 4) semblent encore davantage pertinents. Les plus proches voisins de « 1 » contiennent davantage de chiffres, de « samedi » davantage de jours de la semaine et le tableau contient désormais des synonymes de « femme » et de « bonjour ». Certains mots surprenants subsistent toutefois, comme par exemple « transmets », « désagrément » et « betembourg », voisins de « bonjour ». Toutefois, la fréquence d'apparition de ces mots dans le corpus est faible (moins d'une centaine d'occurrences). La projection de certains vecteurs-mots sur les deux premiers axes d'une ACP (figure 6) nous confirme la qualité de l'entraînement du corpus sur l'ensemble de tweets.

Enfin, nous avons réalisé des opérations sur les mots-vecteurs. Si l'opération $\overrightarrow{Roi} - \overrightarrow{Homme} + \overrightarrow{Femme} =$

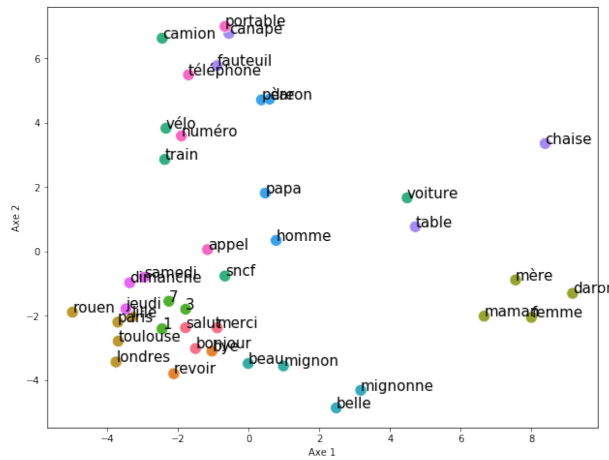

FIGURE 6 – ACP sur un corpus réduit de mots

Note : Paramètres utilisés : ep = 100 / w = 4 / lr = 0,02 / dim = 100 / base : ensemble des tweets

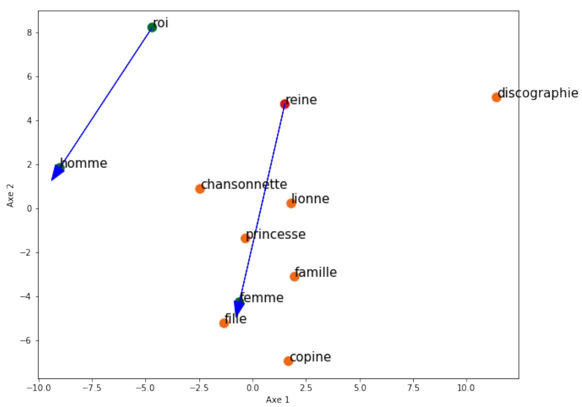

FIGURE 7 – $\overrightarrow{Roi} - \overrightarrow{Homme} + \overrightarrow{Femme} = ?$

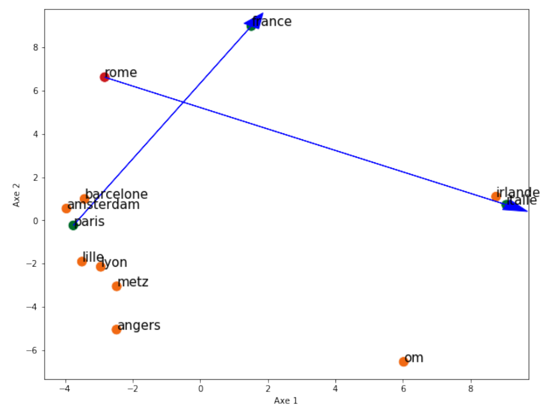

FIGURE 8 – $\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italie} = ?$

Note : Paramètres utilisés : ep = 100 / w = 4 / lr = 0,02 / dim = 100.
Les mots en vert correspondent à ceux présents dans l'opération, le mot en rouge le mot que l'on serait supposé trouver et les mots en orange les 10 mots les plus proches du résultat de l'opération vectorielle.

$\overrightarrow{Reine}$ (figure 7) semble fonctionner [19], l'opération $\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italie}$ (cf. Mikolov, Chen, Corrado *et al* (2013), figure 8) d'identifie pas « Rome » dans les mots les plus proches mais d'autres villes comme « Lyon » (similarité cosinus de 0,62). « Rome » semble effectivement située « trop en haut » dans le plan de l'ACP par rapport aux autres villes. Peut-être ce mot n'a-t-il pas suffisamment été entraîné (246 apparitions dans les tweets contre 46433 pour Lyon par exemple) pour que le vecteur-mot obtenu soit pertinent, ou peut-être que, dans les tweets mobilisés, le mot « Rome » s'utilise dans un contexte différent de l'article de Mikolov.

19. Les similarités cosinus obtenues sont les suivantes : $corr(\overrightarrow{Roi}, \overrightarrow{Homme}) = 0,34$, $corr(\overrightarrow{Homme}, \overrightarrow{Femme}) = 0,35$ et $corr(\overrightarrow{Roi} - \overrightarrow{Homme} + \overrightarrow{Femme}, \overrightarrow{Reine}) = 0,67$. $\overrightarrow{Reine}$ est bien le mot le plus proche de la somme vectorielle calculée.

Références

Boumedyen Billami, M., Gala, N (2017). Création et validation de signatures sémantiques : application à la mesure de similarité sémantique et à la substitution lexicale. TALN 2017. https://hal.archives-ouvertes.fr/hal-01528117/document.

Hutter, F., Hoos, H., Leyton-Brown, K., (2014). An Efficient Approach for Assessing Hyperparameter Importance. PMLR 32(1):754-762. http://proceedings.mlr.press/v32/hutter14.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.

Levy, O., Golberg, Y. (2014). Dependency-based word embeddings. ACL. http://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.

Pennington, J., Socher, R., Manning, C. D., (2014). Glove: global vectors for word representation. Proc. of EMNLP,1532 – 1543. https://www.aclweb.org/anthology/D14-1162.pdf.

Řehůřek, R., Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. p. 46–50, 5 pp. ISBN 2-9517408-6-7. https://is.muni.cz/publication/884893/en.

Rubenstein, H., Goodenough, J. B. (1965). Contextual Correlates of Synonymy. Commun. ACM, 8 (10), 627–633. https://dl.acm.org/doi/10.1145/365628.365657.

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