Université de Montréal

IFT6285 – Traitement automatique des langues naturelles

Devoir 4

---

par:

Eugénie Yockell
(20071932)
Bassirou Ndao
(0803389)

Date de remise: 23 octobre 2020

Nous avons entraîné un modèle `Word2Vec` avec `gensim`. Un tel modèle permet de créer des plongements de mots, soit des vecteurs pour représenter un mot. Nous entraînons le modèle sur l'ensemble de texte composé de 18 819 bloggers qui ont écrit environs 657 000 publications. Une fois le modèle entraîné, nous pouvons lui passer un mot pour qu'il créer un plongement de mots.

Une des importantes difficultés est de déterminer une mesure de qualité des plongements de mots. En effet, `Word2vec` est une tâche d'entraînement non supervisée. Ainsi, de manière objective, il n'y a pas un bon moyen d'évaluer sa qualité dans l'absolu. La méthode doit être adaptée à notre besoin. Comme mentionné dans le papier de `Alshargi, Shekarpour, Soru et Sheth`, nous utilisons une mesure de qualité à l'aide de la similarité sémantique. La similarité sémantique est une métrique qui définit la similarité des mots selon leur sens et leur contexte. Par exemple, les mots *car* et *bus* sont sémantiquement similaires. Notre mesure de qualité cherche à maximiser la conservation de la similarité sémantique des prolongements des mots.

On utilise la fonction `accuracy` de `gensim` pour mesurer la qualité de modèle. Elle lit un fichier de 19 558 phrases comme on le voit plus bas à la gauche de la flèche et effectue le calcul à la droite de la flèche. Les opérations sont effectuées sur les vecteurs de prolongements de mots. Si la similarité sémantique est conservé alors le modèle devrait retourner les bons résultats.

large larger tough **tougher** ⇒ large - larger + tough = **tougher**
possibly impossibly logical **illogical** ⇒ possibly - impossibly + logical = **illogical**
Athens Greece Berlin **Germany** ⇒ Athens - Greece + Berlin = **Germany**

Le prétraitement effectué consiste à retirer toutes les ponctuations excepté de - et '. Les espaces blancs en excès sont retirés. Tous les mots contenant que des chiffres sont placés sous le label 'NOMBRE'. Nous nous sommes aussi questionné sur l'importance des *stop words* comme *the*, *he*, *she*, etc. Nous avons mesuré la qualité des prolongements de mots avec ces différents prétraitements, les résultats sont présentés à la table 1. On remarque que le retrait des *stop words* diminue grandement la qualité des prolongements de mot. Cet impact peut être due au fait que les *stop words* sont importants pour le contexte sémantique des mots.

	Qualité
Sans prétraitement	24.2 %
Avec prétraitement	33.0 %
Avec prétraitement et retrait des *stop words*	21.1 %

Table 1: Qualité du prolongement de mots sur un entraînement de 50 000 phrases sur le meilleur modèle (défini plus loin).

Un des problème important d'entraîner un modèle `Word2Vec` avec `gensim` est le temps de calcul. Les modèles sont très longs à entraîner lorsqu'on utilise un grand nombre de phrases et beaucoup d'epochs d'entraînement. Nous avons mesurer le temps de calcul nécessaire pour entraîner un modèle `Word2Vec` en fonction du nombre de phrases à la figure 1.

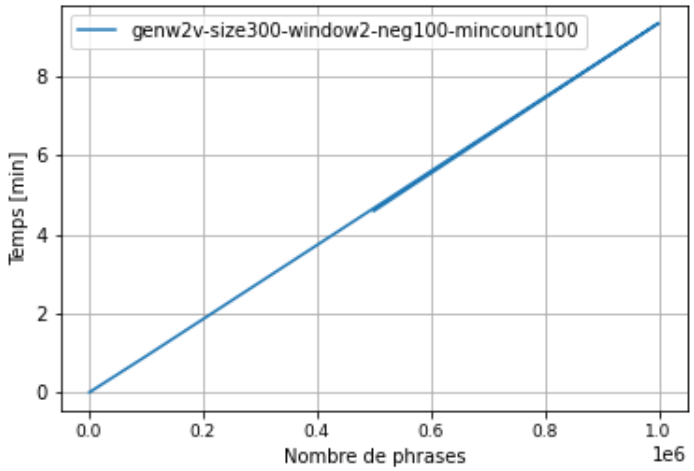

Figure 1: Évolution du temps nécessaire pour entraîner un modèle `Word2Vec` avec `gensim` en fonction du nombre de phrases.

Afin d'utiliser le meilleur modèle possible, il est nécessaire de faire une étude des meta-paramètres. Nous avons alors étudier toutes les possibilités pour les paramètres suivants: **sg** [0,1] pour l'algorithme d'entrainement skip-gram ou CBOW, **hs** [0,1] si on utilise le *hierarchical softmax* pour l'entraînement du modèle ou le *negative sampling*, **cbow_mean** [0,1] pour l'algorithme de CBOW à savoir si on utilise la somme des contextes ou la moyenne, **negative** [0,5,10] la quantité de bruits pour le *negative sampling* et **ns_exponent** [0, 0.75, 1] l'exposant utilisé pour formé la distribution du *negative sampling*. Nous avons aussi testé les paramètres de **window** [2,5,9] qui représente la quantité de contexte à prendre en compte et **min_count** [10,20,30,40] le modèle ignore tous les mots ayant une fréquence plus petite que *min_count*. CBOW signifie Continuous Bag of Words. Dans une telle architecture, le modèle prédit un mot en utilisant son contexte (le *window*). L'ordre des mots dans le contexte n'influence pas l'apprentissage. Tandis qu'avec skip-gram, le modèle utilise le mot courant pour prédire le contexte. Il donne un poids plus important au mots près et un poids plus faible au mots lointain.

Nous avons créé des modèles différents pour chaque possibilité de ces paramètres sur 50 000 phrases. Nous avons jugé qu'il était inutile d'énumérer cette grande quantité de résultats. Cependant, nous en avons résumé les effets ci-dessous. On remarque qu'en général, le CBOW perfome mieux que skip-gram et qu'utiliser la moyenne des contextes au lieu de la somme pour l'algorithme de CBOW fonctionne mieux. De plus, les modèles qui n'utilisent pas de *negative sampling* performent très mal. Ce résultat peut sembler logique puisque le *negative sampling* performe bien pour les mots qui sont fréquents. Pour le paramètres *min_count*, la qualité est à son maximum pour une valeur de 30, mais, on pourrait supposer que cette valeur pourrait être augmenté pour un grand ensemble d'entraînement. En ce qui concerne le paramètres *window*, on remarque que prendre un contexte trop grand fait diminuer la qualité. Nous entraînons le modèle final sur 30 epochs. Notre meilleur modèle était avec l'algorithme d'entraînement CBOW et le *negative sampling* sur 14 505 438 phrases. Le temps de calcul à été de 10 heures sur 3 coeurs.

$$
\begin{aligned}
\text{Word2Vec}\ (\ & size = 100 \\
& alpha = 0.025 \\
& window = 5 \\
& min\_count = 40 \\
& min\_alpha = 0.0001 \\
& sg = 0 \\
& hs = 0 \\
& negative = 10 \\
& ns\_exponent = 0.75 \\
& cbow\_mean = 1\ )
\end{aligned}
$$

Epochs d'entraînement	Qualité
10 epochs	61.8 %
30 epochs	48.6 %

Table 2: Qualité du prolongement de mots sur un entraînement de 14 millions de phrases sur le meilleur modèle.

Nous avons testé le meilleur modèle sur un entraînement fait sur 30 epochs et sur 10 epochs à la table 2. L'entraînement sur 10 epochs est beaucoup plus précis, car entraîner sur 30 epochs génère un sur-apprentissage du modèle. Ça peut être due au fait qu'il n'y a pas assez de données ou que le modèle est trop complexe.

Nous avons définit une liste de mots d'intérêt afin de les visualiser. Nous avons définit une opération particulière à étudier, soit mettre un mot au pluriel. Pour ce faire on mesure la similarité sémantique entre *elephant* et *elephants*, que l'on applique à notre liste d'intérêt pour en déduire d'éventuelles correspondances. Ensuite, on projette le tout en deux dimensions par une méthode d'analyse de composante principale. Le résultat est présenté à la figure 2. Les correspondances sont matérialisées par des flèches. Cette représentation nous permet d'avoir une visualisation de notre opération, mais aussi de voir les regroupements par similarité. Par exemple, les êtres vivants se retrouvent à la gauche ensemble, tandis que les objet se retrouvent regroupés en bas à droite. On remarque aussi que les parties du corps humain sont regroupé ensemble. La mise au pluriel est une translation des mots vers le bas.

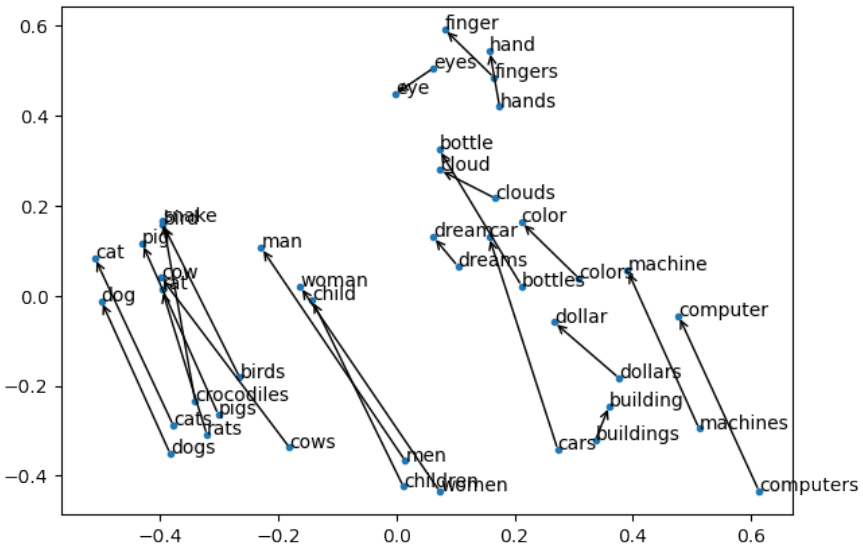

Figure 2: Visualisation de notre modèle dans la tâche d'association des mots singulier/pluriels

Nous avons créer un fichier contenant les 10 premiers voisins d'un mots du vocabulaire. On mesure la distance de prolongement de mots avec la similarité cosinus qui est entre [1,-1]. Le fichier *voisins.txt* contient un mot suivi de ces voisins et leurs valeurs de similarité cosinus. On remarque que les mots voisins on une forte similarité sémantique. Par exemple pour le mot *book* et *strong* nous avons les voisins suivant avec la similarité cosinus en crochet.

book novel[0.86] short_story[0.81] children's_book[0.77] poem[0.72] graphic_novel[0.72] comic[0.72] biography[0.71] screenplay[0.71] tome[0.70]

Bref, il aurait été intéressant d'être en mesure de faire une recherche de paramètres sur un plus grand ensemble de phrases. La grandeur de l'ensemble d'entraînement à un effet important sur les conséquences de chaque paramètres. Malheureusement, les outils à notre dispositions ne permettaient pas de tel calculs. Il aurait été intéressant d'étudier les conséquences de *lemmatisation* ou de *stemming* sur la qualité du modèle. Nous avons tout de même exploré brièvement le modèle `FastText` par curiosité. Celui-ci a été plus précis sur notre liste de test. Ce résultats est cohérent avec le fait qu'il permet d'obtenir une représentation de mots hors de notre vocabulaire.