Plongement de mots

Thach Jean-Pierre **Wong Leo**
Département d'informatique et de recherche opérationnelle
Université de Montréal

1. Introduction

Dans ce ce travail, nous explorons les effets de divers paramètres sur différentes techniques de plongement de mots, notamment Word2Vec et FastText. De plus, notre motif à utiliser FastText est de comparer les différentes bibliothèques disponibles pour le plongement de mots en terme de la taille des modèles sauvegardés sur le disque, le temps d'entraînement, les plus proches voisins pour un mot donné, etc. Finalement, nous comparons également les différents mots voisins générés par chaque modèle.

2. Gensim et spaCy

2.1 Paramètres des modèles

Nous avons décidé d'exécuter plusieurs modèles avec *gensim* et *spaCy* séparément et de comparer certains éléments. L'architecture des modèles *gensim* ont été paramétrés avec la valeur par défaut *cbow*. Le tableau ci-dessous résume nos constatations :

Tableau 1 - Résultats des différents modèles *Gensim* et *spaCy*

Modèle	Taille du modèle binaire (KB)	Temps d'entrainement (mins)	Perte	Taille du vocabulaire	Nombre minimum d'occurence d'un mot	Taille des vecteurs de mots
gensim-1	49400	15.34	75814088[1]	123936	5	100
gensim-2	9597	14.03	73664840[1]	24103	100	100
spacy-1	175900	9.04	71561824	151976	5	100
spacy-2	27000	8.28	70792200	27327	100	100

[1] Note: La perte est obtenue avec la fonction de Gensim *get_latest_training_loss()* qui représente une perte cumulative du modèle. La signification de cette grande valeur n'est pas importante pour déterminer la convergence de la fonction objective.

2.2 Observations

2.2.1 Gensim

Pour ce premier modèle *gensim-1*, nous avons exploré ce que produirait comme sortie si on considère les plongements de mots du vocabulaire au complet incluant le bruit dénoté dans le corpus d'entraînement.

Pour ce faire, nous avons paramétré le modèle avec un seuil de 0.6 après 6 heures de computation afin de filtrer les scores de similarité cosinus pour un mot donné. Nous obtenons un fichier *voisins* trié en termes du nombre de voisins respectant ce seuil de manière décroissante. Par exemple, les jetons *renung, medios, empleados* sont des mots malais et espagnoles possèdent 46188, 45751 et 45639 voisins respectivement avec ce seuil.

En augmentant le nombre minimum d'occurence d'un mot à 100 avec le modèle *gensim-2*, la taille du modèle binaire, le temps d'entraînement, la taille du vocabulaire et la perte diminuent. La diminution de cette perte peut s'expliquer par le fait qu'on retire du vocabulaire le bruit qu'il existe et du fait que l'architecture employée lors de l'entraînement du modèle est *cbow*, qui en général ne performe pas bien pour les mots rares, i.e. cette architecture prédit le mot le plus probable sachant un contexte et donc, favorise moins les mots peu fréquents. De plus, en réduisant la taille du vocabulaire, le temps requis pour obtenir le fichier de sortie *voisins* est réduit substantiellement de 6 heures à 21 minutes.

2.2.2 spaCy

Ensuite, nous avons employé *spaCy* pour déterminer la similarité entre les mots comme nous l'avons fait précédemment avec *gensim*. Comparé à gensim, il semble qu'en convertissant le modèle en format *spaCy*, la perte globale a été moindre. En augmentant le paramètre contrôlant le nombre minimum d'occurrences d'un mot à 100, on observe la même tendance que les modèles précédents, où la taille du modèle, le temps d'entraînement et la taille du vocabulaire diminuent.

2.2.3 Détection de différentes langues

En utilisant le modèle entraîné avec *spaCy*, nous avons constaté que le modèle était capable de saisir des mots voisins dans d'autres langues que l'anglais. Par exemple, le modèle a généré des mots voisins en espagnol pour le mot *como*. Un autre exemple est le mot *ai*, qui signifie "amour" en chinois (mandarin). Les résultats contenaient des mots en pinyin chinois, y compris "aimer" et "cœur".

2.2.4 Élagage vectorielle "Vector Pruning"

En diminuant le nombre de vecteurs que nous conservons dans nos modèles, nous pouvons augmenter la capacité de la mémoire et la vitesse de calcul. Nous observons dans la figure 1 que si nous coupons 10000 vecteurs, les résultats du modèle restent intacts. Si nous éliminons 10 fois cette quantité de vecteurs, nous constatons qu'il y a un changement dans l'ordre des voisins. Cependant, si nous supprimons trop de vecteurs du vocabulaire, le résultat change significativement.

3. FastText

3.1 Paramètres des modèles

Nous avons établi comme paramètres constants le nombre d'itérations à 5, le rythme d'apprentissage (*learning rate*) à 0.05 entre chaque modèle. Le tableau ci-dessous dénote nos résultats :

Tableau 2 - Résultats des différents modèles *FastText*

Modèle	Taille du modèle binaire (MB)	Temps d'entraine ment (mins)	Perte	Taille du vocabulaire	Nombre minimum d'occurence d'un mot	Taille des vecteurs de mots
ft-skipgram-1	992	62	0.270724	294857	5	100
ft-skipgram-2	789	48	0.283917	34566	100	100
ft-cbow-1	992	36	0.977428	294857	5	100
ft-cbow-2	2890	74	0.989453	294857	5	300
ft-cbow-3	789	28	1.106871	34566	100	100

3.2 Observations

On peut observer que la perte des modèles *ft-skipgram* est minimisée davantage qu'aux modèles *ft-cbow*. Effectivement, l'architecture du *skipgram* performe mieux pour traiter les mots qui sont rares étant donné que chaque paire du mot source et son contexte parmi une fenêtre de contexte de taille quelconque est fournie comme nouvelle entrée au réseau de neurones, évaluée pour la justesse du mot cible prédit par la propagation avant, ensuite corrigée par la rétropropagation. Ainsi ce modèle permet d'apprendre à comprendre les mots rares. Tandis que le modèle *cbow* qui est conceptualisé pour prédire un mot par son contexte, apportera moins d'attention aux mots rares puisque ce modèle cherche à maximiser la probabilité du mot cible sachant son contexte en prédisant le mot le plus probable. Par conséquent, la probabilité des mots rares seront lissée davantage en présence de mots fréquents.

En employant les même paramètres, le temps d'entraînement du modèle *ft-skipgram-1* (62 mins) comparé à celui du modèle *ft-cbow-1* (36 mins) s'explique par le fait que le modèle *skipgram* effectue autant d'opérations de propagation avant et rétropropagation qu'il y a de paires du mot source et des mots du contexte. Tandis que le modèle *cbow* fait une moyenne des vecteurs des mots parmi la fenêtre de contexte, pour ensuite l'utiliser comme entrée du réseau de neurones.

En augmentant le nombre minimum d'occurence d'un mot pour le modèle *ft-skipgram-2*, on observe une diminution de la taille du vocabulaire, de la taille du modèle binaire et du temps d'entraînement. Ces résultats semblent partager le même phénomène que les modèles *Gensim* et *spaCy* précédemment dénoté, mis à part une augmentation légère de la perte.

L'effet d'augmenter la taille des vecteurs de mots pour le modèle *ft-cbow-2* comparativement au modèle *ft-cbow-1* résulte à une augmentation quasi proportionnelle à la taille du modèle binaire ainsi qu'une hausse du temps d'entraînement.

4. Comparaison des voisins entre chaque modèle

À des fins de comparaisons, nous utilisons un sous-ensemble des plongements de mots présents dans chacun des modèles. Le seuil du score de similarité utilisé pour ces modèles est de 0.5. Nos observations ci-dessous sont basées sur des mots sélectionnés aléatoirement présenté dans le tableau 4 de l'annexe.

4.1 Observations

Les modèles *skipgrams* résultent en des mots plus variés que les modèles *cbow*, ceci confirme que l'architecture *skipgram* est nettement meilleure pour les mots rares. Par exemple, pour le mot "water", le modèle ft-*skipgram-1* a des voisins comme "bathwater", "saltwater", "watertight", "dishwater", "boiling". Tandis que le modèle *ft-cbow-1* possède comme voisins "watery", "waterpipe", "waterway", "waterbugs", etc. Ce modèle *cbow* semble mettre beaucoup d'emphase sur les mots qui contient le mot cible.

Un modèle *cbow* tend à avoir des voisins plus variés lorsque le nombre minimum d'occurrence de mots est élevé. En effet, le modèle *ft-cbow-3* pour le mot "engine", on a les voisins suivant: "engines", "motor", "engineer", "cpu", "vehicle", etc. Cependant le modèle *ft-skipgram-2* performe légèrement mieux avec les voisins suivant: "engines", "vehicle", "fuel", "transmission", "vehicles", "valve", "google", "search". On observe qu'avec le même nombre minimum d'occurrence de mots, à 100 dans ce cas, le modèle *ft-cbow-3* possède encore le même phénomène auquel les mots rares, e.g. "google" et "search", telles que présent pour le modèle *ft-skipgram-2,* n'ont pas le même poids en terme d'importance.

Dans le tableau 3 de l'annexe, on aperçoit que les modèles *Gensim* et *spaCy* produisent des voisins similaires pour les mots sélectionnés aléatoirement. Par ailleurs, on voit qu'en augmentant la taille des vecteurs des mots en passant de 100 à 300 pour le modèle *ft-cbow-2*, les voisins demeurent semblables pour la majorité des voisins. Enfin, nous observons qu'en général, les modèles *Gensim* et *spaCy* ont une plus grande variabilité dans les voisins que les modèles *ft-skipgram* et *ft-cbow* à l'exception du modèle *ft-skipgram-2* dans lequel on augmente le nombre minimum d'occurrence d'un mot.

5. Annotations utilisées

[COHYPO]: les cohyponymes, [RANDOM]: les mots sans lien, [MORPHO]: les variantes morphologiques, [HYPO]: les hyponymes, [RELATED], les mots en lien, [SYNONYM], les synonymes, [ATTR]: les mots apportant une description d'un mot, [ABBREV]: les abréviations, [PARTOF]: les mots qui sont une partie du mot d'intérêt, [HYPERO]: les hyperonymes, [ROOT]: les mots qui forment la base du mot d'intérêt avec un préfixe et/ou suffixe, [ANTONYM]: les mots contraires.

6. Voisins

Pour générer le fichier *voisins*, nous avons paramétré le modèle *spaCy* avec un nombre minimum d'occurrence d'un mot à 5 et le seuil de similarité à 0.5. Nous avons décidé de trouver les 10 voisins les plus proches pour chaque mot. Pour limiter la taille du fichier, nous avons généré aléatoirement du corpus 1000 mots et leurs voisins les plus proches en ordre décroissant.

Annexes

Figure 1 - L'effet de l'élagage sur les plus proches voisins du mot *pencil*

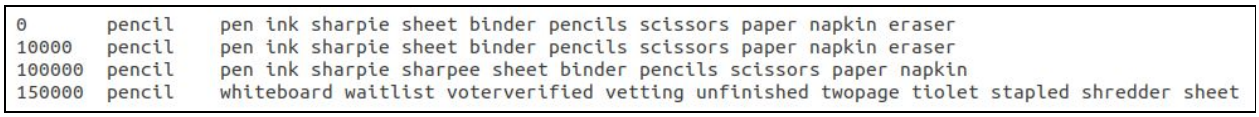

Tableau 3 - Correspondances des voisins entre modèles

Modèles	**Nombre de correspondances du mot “water”**	**Nombre de correspondances du mot “engine”**	**Nombre de correspondances du mot “ministry”**
gensim-1 et spacy-1	6	7	5
gensim-2 et spacy-2	8	7	7

Note: Une correspondance est définie comme étant un mot qui est présent dans les deux modèles qui sont comparés.

Tableau 4 - Voisins de mots basés sur le score de similarité

Modèle	**Mot**	**10 Voisins (en ordre décroissant du score de similarité)**
gensim-1	water	liquid chlorine coals tub faucet tubs buckets droplets helium kerosene
gensim-2	water	chlorine jug buckets coolers sewage liquid coals faucet tub basin
spacy-1	water	chlorine droplets liquid jug coals moisture tub mud helium liquids
spacy-2	water	liquid faucet coolers buckets mud droplets tub jug chlorine coals
ft-skipgram-1	water	bathwater saltwater watertight dishwater waterguns waterpipe waterbottle hotwater rainwater boiling
ft-skipgram-2	water	liquid tub powder mineral boiling bottle soda hose gallons bucket
ft-cbow-1	water	watery waterpipe waterway waterbugs bathwater waterways waterbeds seawater watercooler waterballoon
ft-cbow-2	water	watery waterpipe waterway waterbugs atwater waterbeds bathwater waterways seawater waterboy
ft-cbow-3	water	watery waters watering tub waterfall liquid watered waterfalls underwater heater
gensim-1	engine	engines sensor valve device battery turbo compressor cpu installer motherboard

gensim-2	engine	engines battery valve cpu sensor device kernel turbo accelerator radiator
spacy-1	engine	engines cpu sensor debian valve kernel device battery accelerator motherboard
spacy-2	engine	engines cpu battery kernel sensor device motherboard osx valve partition
ft-skipgram-1	engine	engines fireengine searchengine accelerator tyres speedbump firestarter gearbox turbine vehicles
ft-skipgram-2	engine	engines vehicle fuel transmission vehicles valve google search toolbar compression
ft-cbow-1	engine	engines fireengine engin searchengine accelerator engineer engineers engilsh engish motor
ft-cbow-2	engine	engines fireengine searchengine engineer engilsh engineers engish engineered engineering motor
ft-cbow-3	engine	engines motor engineer cpu vehicle kernel engineers gps engineered electrical
gensim-1	ministry	evangelism community congregation ministries seminary missionary missional liturgy diocese leadership
gensim-2	ministry	congregation evangelism community ministries seminary outreach clergy missionary parish tibetan
spacy-1	ministry	evangelism community congregation seminary ministries priesthood outreach churchs organization parish
spacy-2	ministry	evangelism churchs ministries community seminary outreach congregation clergy presbyterian leadership
ft-skipgram-1	ministry	ministries pastoring ministers pastorate pastoral church ministerial missionary outreach churchwide
ft-skipgram-2	ministry	ministries ministers evangelism youth community pastoral missionary missionaries outreach seminary
ft-cbow-1	ministry	ministries ministerial ministers ministrations ministering mentorship pastorate pastoral churchstate organization
ft-cbow-2	ministry	ministries minister ministers ministerial ministrations ministering ministered minis ministrations westminister
ft-cbow-3	ministry	ministries ministers pastoral organization minister missionary leadership community pastors organizations

Références

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. arXiv:1310.4546, *16 Oct, 2013*.