Modèle de Langage, Plongement statistiques

Leader : Chachura Baptiste, Follower : Bouger Lisa

November 26, 2024

1 Introducion

Ce projet s'appuie sur le travail fondateur de Mikolov *et al.* [1], qui ont développé l'algorithme Word2Vec, permettant de générer des représentations vectorielles denses des mots à partir de corpus textuels. Cet algorithme repose sur des architectures telles que Skip-gram et CBOW (Continuous Bag of Words), qui ont révolutionné le domaine du traitement du langage naturel (NLP) en offrant des moyens efficaces de capturer des relations sémantiques et contextuelles entre les mots. Le but de ce projet est d'approfondir l'étude des modèles de plongements statistiques appliqués à des problèmes variés de traitement du langage. Nous concentrerons nos efforts sur l'architecture Skip-gram, qui apprend à prédire les mots du contexte à partir d'un mot cible. Dans le cadre de ce projet, nous mettrons en œuvre plusieurs méthodes d'évaluation pour analyser et comparer la qualité des plongements générés. Ces évaluations incluront :
Des tests de similarité lexicale, pour vérifier si les mots proches en sens sont correctement représentés par des vecteurs similaires. Des tests analogiques, afin d'explorer si les relations entre mots (exemple : roi - homme + femme $\approx$ reine) sont bien capturées.

2 Modèle Initial

2.1 Description du modèle

Le but de ce modèle est d'apprendre des plongements statistiques sur un corpus de texte donnée.

2.1.1 Apprentissage

Une fenêtre de taille fixée parcours l'ensemble du corpus, nous attribuons le contexte (contenu de cette fenêtre) au mot central de la fenêtre, ce contexte est appelé contexte positif. Pour chaque contexte positif créé nous générons k contextes négatifs. Dans ce premier modèle la

création de ces contexte négatifs se fait grâce à un tirage aléatoire sur l'ensemble du vocabulaire. Pour l'apprentissage nous créons deux matrices : une pour stocker les plongements de nos mots ainsi qu'une matrice qui représente nos contextes. Nous créons ensuite un classifieur binaire qui a pour but de nous prédire si un mot c appartient au contexte d'un mot m. on note :

$$P(+|m, c_{\text{pos}}) = \sigma(m.c) = \frac{1}{1 + e^{-m.c}}$$

où sigma est la fonction sigmoïde.

2.1.2 Fonction de perte

Pour chaque mini-batch de l'ensemble d'apprentissage, on souhaite maximiser la probabilité de l'exemple positif : $P(+|m, c_{\text{pos}})$ et minimiser la probabilité des exemples négatifs : $P(+|m, c_{\text{neg}_i})$. Cela revient à maximiser l'expression suivante :

$$P(+|m, c_{\text{pos}}) \prod_{i=1}^{k} P(-|m, c_{\text{neg}_i})$$

Pour simplifier les calcules nous chercherons à minimiser la fonction de perte suivante :

$$L = -\log \left[P(+|m, c_{\text{pos}}) \prod_{i=1}^{k} P(-|m, c_{\text{neg}_i}) \right]$$

2.1.3 Evaluation

L'évaluation de ce modèle se fait sur un fichier test de la forme :

vélo bicyclette chat
mangue goyave balais
...

Nous effectuons une mesure de similarité entre les mots de chaque ligne et souhaitons obtenir une similarité plus importante entre les mots 1 et 2 que entre les mots 3 et 4. Pour effectuer cette mesure de similarité nous utilisons la similarité cosinus :

$$\text{sim\_cos}(\mathbf{m}_1, \mathbf{m}_2) = \frac{\mathbf{m}_1 \cdot \mathbf{m}_2}{\|\mathbf{m}_1\| \|\mathbf{m}_2\|}$$

2.1.4 Résultats

Dans cette section nous allons étudier l'influence des différents paramètres sur notre modèle. Chaque expérience est répétée 10 fois. Dans un premier temps nous nous intéressons à un modèle où les mots choisis pour générer les exemples négatifs sont tirés de manière totalement alèatoire.

Paramètres	Moyennes	Ecarts types
$L = 100$, $lr = 0.01$, $it = 7$, $w = 2$, $k = 10$, $occ_min = 5$	52.1	3.98
$L = 100$, $lr = 0.01$, $it = 7$, $w = 2$, $k = 4$, $occ_min = 5$	46.6	3.92
$L = 100$, $lr = 0.01$, $it = 7$, $w = 2$, $k = 15$, $occ_min = 5$	54.1	3.26
$L = 100$, $lr = 0.01$, $it = 7$, $w = 5$, $k = 10$, $occ_min = 5$	51.1	5.07
$L = 100$, $lr = 0.01$, $it = 7$, $w = 8$, $k = 10$, $occ_min = 5$	51.1	2.72

Nous savons que tirer des mots de manière aléatoire dans un vocabulaire conduit à une sur-représentation des mots rare ce qui peut avoir un impact sur la performence de notre modèle. Afin de résoudre cela nous utilisons cette fois une distribution basée sur la fréquence d'apparition des mots dans notre ensemble d'entraînement. En repétant les expériences précédentes avec ce nouveau tirage nous n'obtenons pas de changements significatifs dans les performances de notre modèle. En effet un tel tirage conduit cette fois-ci à une sous représentation des mots rares. On respecte bien la fréquence des mots dans le texte d'entraînement néanmoins les mots fréquents tels que "le", "des" ... n'apportent que peu d'information dans le calcul de nos contextes, en effet ce sont les mots plus rares et donc plus spécifiques qui portent l'information sémantique.
Afin d'améliorer nos performances nous testons la solutions proposée par Mikolov *et al.* [2]: Il s'agit encore une fois d'une méthode d'échantillonage mais cette fois ci chaque mot w_i dans l'ensemble d'entraînement est se voit attribué une probabilité calculée par la formule suivante:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

où t est un seuil (d'environ 10^{-5} dans l'article) et $f(w_i)$ est la fréquence du mot w_i.

3 Second Modèle

Cette fois ci les exemples négatifs ne seront plus générer de manière aléatoire mais nous effecturons un tirage où la probabilité de tirer un mot correspond à sa fréquence dans le corpus d'entrainement. Cette méthode de génération des exemples permet d'éviter une sur représentation des mots rares de notre corpus dans les contextes négatifs.

4 Opérations sur les plongements

Le but de cette partie est d'étudier dans quelle mesure est-il possible d'utiliser les vecteurs de plongement afin d'effectuer des opérations sémantiques. Un exemple classique de ce genre

d'opérations est 'Père' - 'Homme' + 'Femme' = 'Mère'. Pour évaluer cette capacité nous créons un jeu de données contenant une centaine d'opérations similaraire sur ces mots.
Pour l'évaluation nous utilisons le modèle 43 (hébergé à http://vectors.nlpl.eu/repository), il contient plus de 2,5 millions de mots et des plogements de dimension 100.

5 Conclusion

References

[1] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc., 2013.