# Modèle de Langage, Plongement statistiques

Leader : Chachura Baptiste, Follower : Bouger Lisa

December 1, 2024

## 1 Introducion

Ce projet s'appuie sur le travail fondateur de Mikolov *et al.* [1], qui a 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 (par exemple : roi - homme + femme $\approx$ reine) sont bien capturées.

## 2 Modèle Initial

### 2.1 Description du modèle

Le modèle utilisé dans ce projet repose sur l'architecture Skip-gram de Word2Vec, développée par Mikolov *et al.* [1]. Son objectif principal est d'apprendre des représentations vectorielles continues des mots (plongements lexicaux) à partir d'un corpus textuel. Le principe fondamental de Skip-gram est de maximiser la probabilité conditionnelle des mots du contexte $c \in C$ donné un mot cible $w$, définie par :

$$P(C|w) = \prod_{c \in C} P(c|w),$$

où $P(c|w)$ est modélisée comme une distribution softmax sur les produits scalaires des vecteurs des mots cibles et de contexte.
Pour améliorer l'efficacité, des approches comme l'échantillonnage négatif (*negative sampling*) sont utilisées, remplaçant la normalisation globale de softmax par une optimisation locale sur un sous-ensemble de mots. Cela permet d'entraîner le modèle de manière scalable sur de grands corpus.
Les plongements obtenus sont des vecteurs dans un espace de faible dimension(ici d=100), où des relations sémantiques et syntaxiques complexes entre mots sont représentées par des propriétés géométriques (proximité ou direction dans l'espace vectoriel). Ce modèle s'appuie sur l'hypothèse distributionnelle, qui stipule que les mots partageant des contextes similaires ont des significations proches.

### 2.1.1 Apprentissage

Une fenêtre de taille fixée parcourt 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 contextes 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 dont le but est de prédire si un mot $c$ appartient au contexte d'un mot $m$. On note :

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

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 aléatoire.

| **Paramètres** | **Moyennes** | **Écarts-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 |

Table 1: Tableau des paramètres, moyennes et écarts-types.

Nous savons que tirer des mots de manière aléatoire dans un vocabulaire conduit à une sur-représentation des mots rares, ce qui peut avoir un impact sur la performance 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 répétant 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", etc., 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 solution proposée par Mikolov *et al.* [2] : il s'agit encore une fois d'une méthode d'échantillonnage, mais cette fois-ci, chaque mot $w_i$ dans l'ensemble d'entraînement se voit attribuer 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 $w_i$.
Une première expérience avec $t = 10^{-5}$ et des paramètres similaires aux expériences précédentes ($L = 100$, $lr = 0.1$, $it = 5$, $w = 2$, $k = 10$) nous donne une précision de 54.1 et un écart-type de 3.2. Nous constatons déjà une amélioration comparée au modèle précédent. Néanmoins, ce n'est toujours pas satisfaisant. Dans l'article de Mikolov *et al.* [2], il n'y a pas de justification de la valeur du paramètre $t$. Cependant, on remarque que dans l'article, $t$ est de l'ordre des fréquences des mots. Dans notre cas, nous avons un vocabulaire d'environ 3000 mots ; si nous ajustons notre $t$ en fonction, $t = \frac{1}{3000}$, nous obtenons cette fois-ci :
53.4 de précision et un écart-type de 5.02.

# 3 Opérations sur les plongements

Le but de cette partie est d'étudier dans quelle mesure il est possible d'utiliser les vecteurs de plongement afin d'effectuer des opérations sémantiques. Un exemple classique de ce genre d'opération est 'Père' - 'Homme' + 'Femme' = 'Mère'. Pour évaluer cette capacité, nous créons un jeu de données contenant une centaine d'opérations similaires sur ces mots.
Pour l'évaluation, nous utilisons le modèle 43 (hébergé à http://vectors.nlpl.eu/repository), qui contient plus de 2,5 millions de mots et des plongements de dimension 100.
Nous créons un dataset de 100 exemples de composition de mots tels que décrits précédemment.
Le but de cette expérience est de sommer ou soustraire les vecteurs de plongement des mots de notre opération, puis de chercher le vecteur le plus proche de notre vecteur résultant.
Pour la mesure de distance, nous utiliserons la similarité cosinus décrite précédemment.
Pour cela, nous utilisons la bibliothèque Gensim fournie avec le modèle, qui permet d'implémenter un arbre kd afin d'effectuer une recherche efficace.
Lors de l'évaluation, nous effectuons l'opération décrite dans le dataset sur nos vecteurs de plongement et nous extrayons les 30 mots les plus proches correspondant au vecteur de plongement calculé. Nous regardons ensuite si le mot cible (le résultat attendu de notre opération) apparaît dans cette liste et, si oui, en quelle position.
Nous obtenons les résultats suivants :
La position $-1$ signifie que le mot attendu n'a pas été trouvé dans les 30 mots extraits. On constate que seulement 3 opérations produisent exactement le résultat attendu. Dix opérations produisent un résultat acceptable : le mot cible apparaît dans les 5 mots les plus proches de notre vecteur calculé.
Nous pouvons néanmoins discuter la précision de cette expérience. En effet, les exemples ayant été écrits manuellement peuvent être subjectifs, et les résultats de ces opérations ne sont pas nécessairement uniques, ce qui explique en partie les faibles performances de ce modèle sur ce test.
Un dataset de test plus grand, ayant été écrit par diverses personnes, serait plus adapté.

| Position | Compte |
|---|---|
| -1 | 79 |
| 1 | 3 |
| 2 | 2 |
| 3 | 3 |
| 4 | 2 |
| 8 | 1 |
| 18 | 1 |
| 28 | 2 |

Table 2: Tableau représentant les positions et le nombre d'occurrences des résultats.

# 4 Conclusion

En conclusion, ce projet a permis d'explorer les fondements théoriques et pratiques des plongements lexicaux à travers l'algorithme Skip-gram et ses variantes. Nous avons pu mettre en œuvre et analyser différentes techniques d'optimisation, telles que l'échantillonnage négatif et la pondération par fréquence, afin d'améliorer la qualité des vecteurs générés.
Les résultats obtenus mettent en évidence l'impact des paramètres du modèle sur la précision des plongements, mais soulignent également les limites des approches actuelles. Par exemple, l'évaluation basée sur la similarité cosinus a montré des performances satisfaisantes pour certaines tâches de similarité lexicale, mais les opérations sémantiques complexes restent un défi, en partie dû à la subjectivité des jeux de données de test.
Ces travaux ouvrent la voie à plusieurs pistes d'amélioration, telles que :

- l'augmentation et la diversification des jeux de données d'évaluation,
- l'exploration de méthodes plus avancées de régularisation ou d'échantillonnage,

Une exploration plus approfondie, notamment avec des modèles plus récents ou des architectures combinées, pourrait permettre de dépasser certaines des limitations identifiées dans ce travail.

# References

[1] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *None*, 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.