Rapport projet TLNL : Plongements de mots statiques

Leader : Paul Peyssard, Follower : Idir Saidi

24 septembre 2023

Ce document décrit un des deux projet réalisé par Idir Saidi et Paul Peyssard pour l'UE Traitement du Langage Naturel et Linguistique du Master 2 Intelligence Artificielle & Apprentissage Automatique à Aix-Marseille Université.

1 Introduction

Dans le vaste domaine du Traitement du Langage Naturel et Linguistique (TLNL), la représentation sémantique des mots joue un rôle crucial pour de nombreuses applications, allant de la recherche d'information à la traduction automatique, en passant par l'analyse de sentiments. Les plongements de mots, également connus sous le nom de word embeddings, offrent une méthode puissante pour capturer les relations sémantiques et syntaxiques entre les mots, en les représentant comme des vecteurs dans un espace continu.

Ce projet, réalisé dans le cadre de l'UE Traitement du Langage Naturel et Linguistique du Master 2 Intelligence Artificielle & Apprentissage Automatique (IAAA) à Aix-Marseille Université, se penche sur l'étude et l'implémentation des plongements de mots statiques à travers le modèle Word2Vec et sa variante Skip-gram. L'objectif principal est de comprendre en profondeur les mécanismes sous-jacents de ce modèle, ainsi que d'évaluer sa performance et ses limites.

En se basant sur l'idée "you shall know a word by the company it keeps"[1] (vous connaîtrez un mot par la compagnie qu'il garde), popularisée par J.R. Firth en 1957, ce projet explore comment les mots qui apparaissent dans des contextes similaires tendent à avoir des significations similaires. Cette idée est au cœur des plongements de mots et guide notre exploration ainsi que notre évaluation du modèle Word2Vec.

Les pistes explorées dans ce projet incluent l'évaluation d'analogies calculées par le biais des distances entre embeddings permettant de sonder la capacité du modèle à capturer des relations sémantiques et syntaxiques spécifiques entre les mots. Ces tâches d'analogie, variées et issues de différents domaines, sont utilisées comme un prisme à travers lequel nous pouvons mieux comprendre et interpréter les représentations apprises par le modèle.

2 Modèle Initial

Nous avons donc implémenté le modèle Word2Vec[2] afin de générer les embeddings à partir d'un corpus textuel : "Le Comte de Monte-Cristo" d'Alexandre Dumas, un texte riche et volumineux, offrant une diversité lexicale et contextuelle propice à l'apprentissage d'embeddings de qualité.

2.1 Préparation des Données et Tokenization

La première étape du processus a consisté à préparer le texte pour l'apprentissage. Nous avons commencé par convertir le texte en minuscule pour assurer une certaine uniformité. La seconde étape a été la tokenization, c'est-à-dire la transformation du texte en une liste de mots (ou tokens). Nous avons assigné à chaque mot unique dans le corpus un identifiant numérique unique, afin de pouvoir optimiser les performances des divers algorithmes que nous allions implémenter par la suite (manipuler des nombres entiers est beaucoup plus efficace que de manipuler de nombreux mots avec des longueurs variables). Cette étape est cruciale car elle transforme le texte brut en une forme qui peut être traitée par le modèle.

2.2 Initialisation et Apprentissage des Plongements

Les plongements de mots ont été initialisés de manière aléatoire avant d'être évolués à travers le processus d'apprentissage. Le modèle a été entraîné en utilisant la méthode skipgram with negative sampling[3], une variante du modèle Word2Vec, où le modèle apprend non seulement à reconnaître les paires de mots qui apparaissent fréquemment ensemble, mais aussi à différencier ces paires des paires de mots qui n'apparaissent pas ensemble. Nous avons donc créé un contexte positif ($2L$ mots qui apparaissent dans un intervalle de L mots avant et après le mot) ainsi qu'un contexte négatif ($2 * K * L$ mots qui n'apparaissent jamais dans un intervalle de L mots avant et après le mot). On utilise un contexte négatif plus grand pour garantir que le modèle apprenne efficacement à distinguer les paires de mots cohérentes de celles qui ne le sont pas, augmentant ainsi la robustesse et la précision des plongements obtenus.

Pour l'apprentissage, nous avons utilisé la Loss suivante :

$$Loss = -\left[\log \sigma(m \cdot C_{pos}) + \sum_{i=1}^{K} \log \sigma(-m \cdot c_{neg_i})\right]$$

Dans cette formule :

- σ est la fonction sigmoid, définie comme $\sigma(x) = \frac{1}{1+e^{-x}}$ qui sert à transformer le produit scalaire en probabilité
- m est l'embedding du mot cible.
- C_{pos} est l'embedding du mot contexte positif.
- c_{neg_i} sont les embeddings des mots contextes négatifs (car pour chaque mot du contexte positif, il y a k mot du contexte négatif)

La première partie de la somme, $\log \sigma(m \cdot C_{pos})$, mesure la similarité entre l'embedding du mot cible et l'embedding du mot contexte positif. La seconde partie, $\sum_{i=1}^{K} \log \sigma(-m \cdot c_{neg_i})$, mesure la dissimilarité entre l'embedding du mot cible et les embeddings des mots contextes négatifs. L'objectif est de maximiser

la similarité avec les contextes positifs tout en minimisant la similarité avec les contextes négatifs.

2.3 Résultats et Limitations

Après avoir mené une série d'expériences pour évaluer la qualité des plongements de mots générés, nous avons utilisé un ensemble de triplets de mots pour mesurer la précision du modèle. Chaque triplet est constitué d'un mot m, d'un mot $m+$ dont le sens est supposé être plus proche de m que ne l'est le mot $m-$.

L'évaluation s'est basée sur la mesure de similarité cosinus entre les vecteurs de mots, visant à confirmer que la similarité entre m et $m+$ est supérieure à celle entre m et $m-$. La similarité cosinus est particulièrement pertinente dans ce contexte car elle est insensible à la magnitude des vecteurs, se concentrant uniquement sur leur direction. Cela signifie que deux vecteurs qui pointent dans la même direction auront une similarité cosinus élevée, indépendamment de leur longueur.

$$cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

À travers dix expériences, le modèle a montré une performance avec un taux de réussite moyen de 51.4% et un écart-type moyen de 4.54%, indiquant une certaine variabilité dans les résultats d'une expérience à l'autre. Bien que le modèle soit légèrement en dessous du niveau de précision obtenu dans les benchmarks fournis par les professeurs encadrants (53.2% de réussite avec un écart-type de 3.9%), les résultats sont néanmoins satisfaisants étant donné la nature aléatoire du processus d'initialisation des embeddings.

Le modèle a donc montré une capacité à apprendre des représentations significatives des mots, comme en témoignent les plongements de mots générés. Cependant, il est important de noter certaines limitations inhérentes au modèle et à l'approche utilisée.

Premièrement, le modèle Word2Vec est un modèle de plongements statiques, ce qui signifie que chaque mot est représenté par un seul vecteur, indépendamment de son contexte d'utilisation. Cela peut être problématique pour les mots polysémiques, qui ont plusieurs significations en fonction du contexte.

Deuxièmement, le modèle est sensible à la fréquence des mots. Les mots rares sont souvent mal représentés, car le modèle n'a pas suffisamment d'exemples pour apprendre une représentation précise. Cela a été atténué en partie par l'utilisation d'un seuil minimal d'occurrence pour inclure un mot dans le vocabulaire, mais le problème persiste pour les mots qui apparaissent juste au-dessus de ce seuil ou si le seuil est trop bas.

2.4 Piste à creuser 1 : Analogies

2.4.1 Description

Dans cette partie du projet, nous explorons les analogies, un aspect fascinant des plongements de mots qui permet de découvrir des relations sémantiques et syntaxiques entre les mots. Une analogie typique dans le contexte des plongements de mots prend la forme : $mot1 : mot2 :: mot3 : mot4$, ce qui signifie que la relation entre $mot1$ et $mot2$ est similaire à celle entre $mot3$ et $mot4$.

L'hypothèse principale que nous cherchons à valider est que les plongements de mots générés par le modèle Skip-gram sont capables de capturer ces relations d'analogie. Nous nous attendons à ce que, en effectuant des opérations vectorielles sur les plongements, nous puissions trouver des résultats qui confirment ces relations. Par exemple, si nous prenons les vecteurs correspondant à *roi*, *homme* et $femme$, et que nous effectuons l'opération $roi - homme + femme$, nous nous attendons à trouver un vecteur proche de *reine* (*"woman is to queen as man is to king"*[1]).

Cependant, comme les textes, interprétations et sémantiques sont souvent subjectifs et sujets à différentes interprétations, nous n'allons pas toujours trouver les bonnes valeurs telles que "reine", mais des mots s'en rapprochant (comme "impératrice" ou "dirigeante"). Nous espérons que cela s'appliquera également à d'autres analogies.

2.4.2 Mise en œuvre

Pour mettre en œuvre cette exploration, nous avons utilisé les plongements de mots générés par le French CoNLL17 corpus du site `http://vectors.nlpl.eu/repository/` contenant un vocabulaire de taille 2 567 698.

Pour trouver des analogies avec des embeddings, une approche est d'exploiter la linéarité observée dans l'espace des embeddings, on peut donc modéliser les analogies par la relation vectorielle suivante.

$$\vec{v}_{mot1} - \vec{v}_{mot2} \approx \vec{v}_{mot3} - \vec{v}_{mot4}$$

En prenons donc homme (mot1), roi (mot2), femme (mot3) et reine (mot4), nous obtenons la formule :

$$\vec{v}_{mot2} - \vec{v}_{mot1} + \vec{v}_{mot3} \approx \vec{v}_{mot4}$$

$$\vec{v}_{roi} - \vec{v}_{homme} + \vec{v}_{femme} \approx \vec{v}_{reine}$$

Après avoir testé plusieurs centaines d'analogies, nous nous sommes rendu compte que nous n'obtenions pas pas exactement le même taux de réussite pour chaque "type" d'analogies. Nous avons donc décidé de comparer les différents type d'analogies que nous pouvions imaginer :

- **Relation de conjuguaison :**
  - **Des verbe conjugué au même temps et à la même personne** : dors dormons roules roulons
  - **Des verbes à divers temps suivit de l'infinitif** : volons voler marchent marcher
- **Relation de genre :**
  - **Relation feminin - masculin** : Acteur Actrice Chanteur Chanteuse
  - **Relation familiale** (Similaire mais supposé plus complexe pour un modèle) : mère père fils fille
- **Relation de localisation :**
  - **Pays - Capitale** : Grèce Athènes Italie Rome
  - **Ville - Pays** : Seville Espagne Lisbonne Portugal
  - **Quartier - ville** : Panier Marseille Montmartre paris (Très précis donc ambitieux dans un corpus non axés sur la géographie)
  - **Localisation plus générale** : ville pays pays continent

— **Relation diverse :** tous types de relation qui ne rentre pas forcément dans une catégorie générale ou assez grande, exemple :
  — Lait Vache Miel Abeille
  — Chocolat Sucré Citron Acide
  — Piano Pianiste Guitare Guitariste
  — Glace Froid Feu Chaud

2.5 1ère méthode : le mot le plus proche

Suite à la création de ces analogies, nous avons créé deux algorithmes différents pour les analyser. Pour ces deux algorithmes, nous avons essayé différentes approches pour optimiser le temps de calcul. Le premier algorithme a servi à trouver l'embedding le plus proche de la valeur $mot2 - mot1 + mot3$ grâce à la distance euclidienne,

$$e^* = \arg\min_{e \in V} d(e, x)$$

Nous trouvons le mot le plus proche en comparant chaque embedding. Le mot du vecteur minimisant cette distance sera considéré comme le mot recherché (mot4).

Les résultats obtenus par cette expérience n'étaient pas parfait, mais au vu du caractère très subjectif et de la forte sensibilité aux variations (embedding, corpus, compréhension personnelle des mots...), il étaient tout de même déjà satisfaisants.

Nous avons décidé de modéliser les résultats de la manière suivante :
— **Mot trouvé :** le mot exact a été trouvé
— **Synonyme :** Un synonymé du mot cherché à été trouvé
— **Mot presque trouvé :** Un mot de sens proche ou acceptable a été trouvé (exemple : musicien et chanteur)
— **Mot non trouvé** (ot n'ayant aucun rapport) ou **Erreur de corpus** (Certain corpus, dont celui utilisé, qui était le meilleur que nous avons essayé, ne sont pas totalement juste, et on peut retrouver des mots erroné ou inexistant)

Nous avons donc obtenu les resultats suivants :

	Conjuguaison	**Genre**	**Localisation**	**Divers**
Mot trouvé	49%	48%	62%	22%
Synonyme	6%	4%	0%	0%
Mot presque trouvé	23%	30%	23%	41%
Mot non trouvé ou erreur	22%	18%	15%	37%

TABLE 1 – Résultats analogies trouvées (Unique plus proche voisin)

Nous pouvons voir ici que le modèle performe plutôt bien étant donné la nature du problème, avec une moyenne de mots trouvés d'environ 46% et une moyenne de trouver un mot qui fait plus ou moins sens de 77% (la moyenne des sommes des 3 première lignes). Nous pouvons interpréter les résultats différemment selon les catégories d'analogies :

- **Conjuguaison** :
  - Le modèle arrive bien à retrouver le verbe qu'il faut, mais ne trouve pas toujours le bon temps ou la bonne personne. Ce qui fait du sens quand on sait que certains verbes ne sont quasiment jamais employés à certaine personne (nous pleuvons) ou à un temps donné et ne sont pas toujours utilisés dans les mêmes contextes.
  - Le modèle arrive très bien à retrouver l'infinitif des verbes (mangeons, manger, vole, voler) mais est moins performant quand ce sont des temps ou des verbes compliqués ou peu employés, ce qui pourrait montrer une certaine dépendance à la fréquence d'utilisation des mots.
- **Genre**
  - Dans cette catégorie, il y a un très bon taux de réussite, mais également le même problème que pour la conjuguaison. On peut voir que les mots peu employés ne sont pas forcément trouvés (on obtient par exemple "comtesse" au lieu de "vicomtesse") ou les mots qui se font rares au féminin sont également difficiles à trouver, par exemple on trouve "manager" à la place d'"administratrice" car ce mot ne doit pas être beaucoup présent dans la littérature. C'est très révélateur du problème de la sur-utilisation de noms masculins par rapport aux noms féminins.
- **Localisation**
  - Cette catégorie est celle qui a le mieux marché étonnamment. Le modèle arrive très bien à trouver les liens entre les lieux, que ce soit de simples pays à des villes (Barcelone Espagne Milan Italie) ou à retrouver les capitales (Syrie Damas Jordanie Amman), et même des analogies qu'on croirait très difficiles, telles que Ouganda Kampala Rwanda Kigali.
  - Concernant les localisations plus générale, le modèle à plus de mal (ville - pays + pays = industrialisation (Attendu : continent))
  - Les synonymes sont à 0 car il n'existe pas de synonyme de ville ou de pays
  - Ce qui nous a le plus impressionnés est le résultat de cette analogie : Paris - Montmartre + Marseille = Mazargues (Attendu : Vieux-Port). Même si nous attendions un autre résultat (preuve de la subjectivité du problème), le modèle nous en a donné un encore plus précis.
- **Divers** :
  - Ici, les analogies sont plus compliquées et plus subjectives, donc il y a beaucoup plus d'erreurs et de cas dans la catégorie "Presque trouvé" car on peut interpréter les analogies différemment. Cependant, il y a beaucoup plus d'erreurs car les analogies sont plus compliquées et parfois les sens sont plus abstraits, comme :
    - Bleu Couleur Mille Chiffre
    - Chocolat Sucré Citron Acide
    - Feu Brule Eau Mouille
  - Le modèle est donc efficace surtout pour deviner des analogies basiques avec des liens clairs et simples.

Bien que les résultats était satisfaisants, nous avons tout de même décidé d'aller plus loin en essayant une nouvelle approche, trouver les k plus proches embeddings.

2.6 Partie 2 : Analogies avec k mots les plus proches

Dans cette partie, nous avons élargi notre analyse en cherchant les k mots les plus proches pour chaque analogie. Cela nous a permis de voir si le mot attendu apparaissait dans les premières positions, même s'il n'était pas le premier. Nous avons procédé de la même manière que pour la première méthode, mais en gardant en mémoire les k plus proches voisins de l'embedding calculé (Cette méthode peut donc également servir à construire le précédent algorithme si on fixe k à 1).

Voici les résultats obtenus avec les 10 plus proches voisins :

	Conjuguaison	**Genre**	**Localisation**	**Divers**
Mot trouvé (5 premiers voisins)	52%	63%	70%	29%
Mot trouvé (10 premiers voisins)	61%	68%	74%	33%
Mot proche (10 premiers voisins)	25%	18%	13%	38%
Mot non trouvé ou erreur	14%	14%	13%	29%

TABLE 2 – Résultats analogies trouvées (10 plus proche voisins)

Remarque : La deuxieme ligne du tableau comprend également la premiere ligne (interval de 1 à 10)

On peut voir que grâce à ce modèle, la probabilité de trouver le mot exact a augmenté (+20% pour le genre). Il y a tout de même des analogies sans réponses correctes, mais nous avons désormais une moyenne de 82.5% de réponses acceptables (les 3 premières lignes), ce qui est une bonne avancée. On peut voir que si le mot n'a pas été trouvé parmi les cinq premiers voisins, il a peu de chances (en moyenne 5%) d'être trouvé parmi les cinq suivants, ce qui montre que plus la distance augmente, plus le sens des embeddings change rapidement ce qui implique que le modèle n'a pas sut capter le sens du mot.

- **Conjuguaison** :
  - Nous avons une nette amélioration de la chance de trouver le bon mot, mais il est toujours difficile pour certains verbes, temps ou personnes d'être reconnus pour les mêmes raisons évoquées précédemment.
- **Genre**
  - Comme nous avons beaucoup plus de mots, le problème de la surutilisation du masculin s'est atténué et nous avons plus de chances d'obtenir le mot voulu, même si c'est un mot féminin peu employé.
  - Ici, nous voyons très bien que prendre plus d'un voisin est utile car le mot "vicomtesse" recherché précédemment était juste derrière (à la 2ème position) le mot "comtesse" deviné plus tôt, ce qui montre que des mots proches et similaires ont une distance euclidienne très proche (ici, le résultat de la distance euclidienne était de 1.731 contre 1.741).
- **Localisation**
- Nous avons une bonne évolution du pourcentage de mots trouvés grâce aux nombreux choix que nous avons désormais à notre disposition.
- L'exemple impressionnant cité plus haut s'est encore plus amélioré en donnant d'autres quartiers de Marseille dans ses plus proches voisins : Paris Montmartre Marseille (Mazargues (2.422), Castelet (2.546), Canebière (2.563)...).

— **Divers** :
  — Du fait du nombre de mots prédits, le modèle est plus efficace, nous arrivons à trouver plus de mots similaires, mais trouver le mot exact s'avère toujours difficile.

3 Conclusions et perspectives

Dans ce projet, nous avons rencontré de nombreux défis, dont l'optimisation des algorithmes en raison de la longueur des textes avec lesquels nous travaillons. Pour le calcul des embeddings, nous étions initialement à plus d'une heure d'exécution, mais nous avons fini par réduire considérablement le temps d'exécution à quelques minutes. Pour trouver les analogies, c'était également un problème car nous utilisions un corpus de très grande taille (2 567 698 mots uniques). Nous avons résussi à réduire considérablement le temps d'exécution à moins d'une dizaine de minutes (plusieurs heures pour notre premier algorithme).

Les résultats obtenus dans la partie principale étaient satisfaisants étant donné leur proximité avec ceux indiqués par les professeurs. Cependant, ils pourraient être améliorés par une méthode d'échantillonnage du contexte négatif et d'autres méthodes que nous pouvons trouver dans l'article *Distributed representations of words and phrases and their compositionality.*

Les résultats obtenus dans la partie sur les analogies ont montré que le modèle est plus efficace pour capturer des relations sémantiques (notamment le genre ou les associations de localisation) que poru les relations grammaticales. Bien que le taux de réussite ne soit pas parfait, il est important de souligner la subjectivité inhérente à ce type de tâche, ainsi que la sensibilité aux variations dans les embeddings, le corpus et la compréhension personnelle des mots. Nous obtenons tout de même une moyenne de 82.5% de réponses acceptables et 53.5% de réponses exactes sur les analogies testées (environ 200, dont 50 par catégories).

Initialement, nous étions inquiets d'avoir "seulement" obtenus ces résultats que nous ne considérions pas parfait, mais après avoir relu le papier *Analogies Explained : Towards Understanding Word Embeddings*[1], nous nous sommes rendu compte que c'était normal de ne pas atteindre un score presque parfait car les embeddings reflètent des relations linéaires approximatives entre les mots plutôt que des relations exactes. Les paraphrases et les transformations de mots jouent un rôle crucial dans ces approximations, et la linéarité est influencée par la factorisation de l'information mutuelle ponctuelle (PMI) entre les mots, comme expliqué dans cet article.

Comme cette partie du projet concernait seulement les plongements de mots statiques, nous pensons que pour améliorer ce modèle, il serait intéressant de se pencher sur le sujet des plongements de mots contextuels. Cependant, cela nécessiterait plus de données spécifiques. Une autre idée d'amélioration serait d'utiliser différents corpus plus spécialisés pour chaque type d'analogies.

En conclusion, ce projet nous a permis de mieux comprendre les mécanismes sous-jacents aux plongements de mots statiques (embeddings statiques) et de mettre en évidence certaines de leurs limites, en particulier en ce qui concerne la représentation des mots polysémiques et la sensibilité à la fréquence des mots. Malgré ces limites, les embeddings de mots ont démontré leur capacité à capturer des relations significatives entre les mots, offrant ainsi des perspectives

prometteuses pour de futures recherches et applications dans le domaine du Traitement du Langage Naturel et Linguistique.

Références

[1] Towards Understanding Word Embeddings : *arXiv*. Available at : `https://arxiv.org/pdf/1901.09813.pdf`

[2] Efficient Estimation of Word Representations in Vector Space : *arXiv*. Available at : `https://arxiv.org/pdf/1301.3781.pdf`

[3] Distributed Representations of Words and Phrases and their Compositionality : *arXiv*. Available at : `https://arxiv.org/pdf/1310.4546.pdf`