Cours 5: Natural Language Processing

Introduction

Le Natural Language Processing (NLP) ou Traitement automatisé de littérature (TAL) est comme son nom l'indique l'utilisation d'algorithme pour traité de la donnée textuelle. Il y a différent type d'algorithme et d'objectif mais les plus connus sont:

- **Sentiment Analysis**: à partir de texte, souvent des reviews, commentaires, tweet, il est possible de donner une valence positive ou négative en fonction des mots utilisés
- **Résumé de texte**: pour ce faire l'algorithme va se baser sur les mots qu'il juge les plus important et supprimer les autres, tout en recomposant les mots gardés pour en faire des phrases intelligibles. Ou il va tout simplement supprimés des phrases qu'il juge moins importantes.
- **Traduction:** Très certainement le champ le plus utilisé du NLP. Il y a maintenant des algorithmes qui sont capables de traduire des textes dans la quasi totalité des langues vivantes et mortes avec une très bonne qualité.
- **Génération de texte:** plus récemment certains algorithme comme GPT-3 se sont lancés dans la générations de texte à partir de simple début de phrase.

Comme tout types d'algorithmes que nous avons vu jusqu'à présent, il est indispensable d'avoir des données numériques. Il faut donc transformer le text en valeur numérique.

Tokenisation

La tokénisation est simplement le fait de simplifier le texte. Il existe énormément de techniques différentes et le niveau de simplification dépend du type des tâches et de la langue utilisée. Les plus courantes sont:

- Enlever les majuscules (cependant en allemand cela posera problème)
- Enlever la ponctuation (il est possible que dans certaine tâche de haut niveau ce soit aussi problématique)
- Supprimer les "stop words" (petits mots) comme "le", "une" etc qui n'apporte pas grand chose au texte (sauf en cas de traduction)
- Changer les mots proche comme "train" et "training" pour simplifier encore le vocabulaire

Bag of Words

Prenons 3 phrases avec des mots qui se répètent:

"Il était une fois un jeune garçon"
"Il était une fois un jeune développeur"
"Il était une fois un jeune pays"

On a donc comme vocabulaire total de 9 mots:
"Il", "était", "une", "fois", "un", "jeune", "garçon", "développeur", "pays".

On peut donc utiliser une séquence de 9 représentations dans un vecteur pour obtenir nos phrases sous forme chiffrés. Il y aura un 1 si le mot apparait dans la phrase et 0 sinon, tout en suivant l'ordre du vocabulaire:

"Il était une fois un jeune garçon" ==> [1, 1, 1, 1, 1, 1, 1, 0, 0]
"Il était une fois un jeune développeur" ==> [1, 1, 1, 1, 1, 1, 0, 1, 0]
"Il était une fois un jeune pays" ==> [1, 1, 1, 1, 1, 1, 0, 0, 1]

Il est aussi possible de faire des bi-gramme c'est à dire d'utiliser des paires de mots que l'on représente. Cela permet d'avoir plus de sens par exemple:

"Il était", "était une", "une fois", …. Il est aussi possible de faire des tri-gramme et ainsi de suite. Cependant plus on augmente le nombre de séquence (bi/tri/tetra…) plus le vocabulaire va augmenter et plus on va avoir des problèmes de mémoires.

Une fois que le vocabulaire a été créer, il est possible de le tester en comptant le nombre de mots, ou en regardant la fréquence (nombre d'utilisation/nombre de mots). Bien que cette méthode soit très simple, elle permet d'analyser et de classifier des documents en groupe.

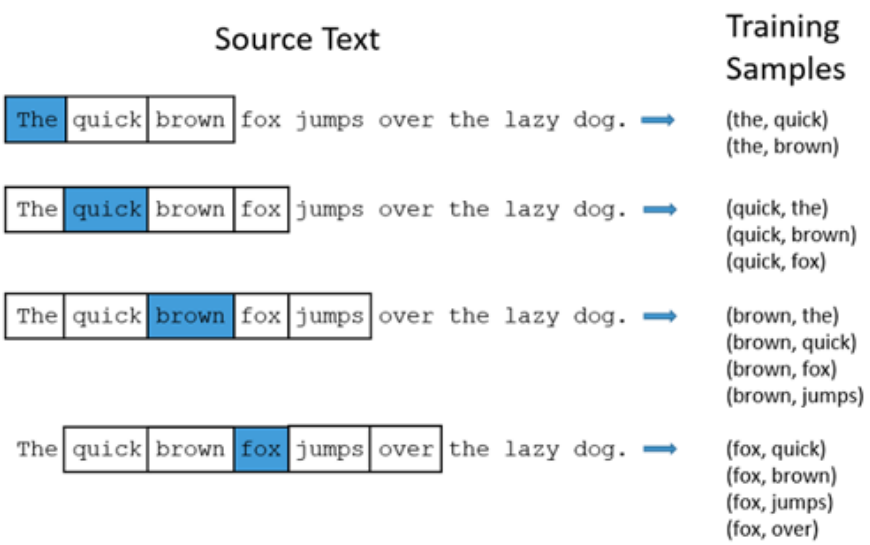
Il y a de grosses limites à cette méthode:

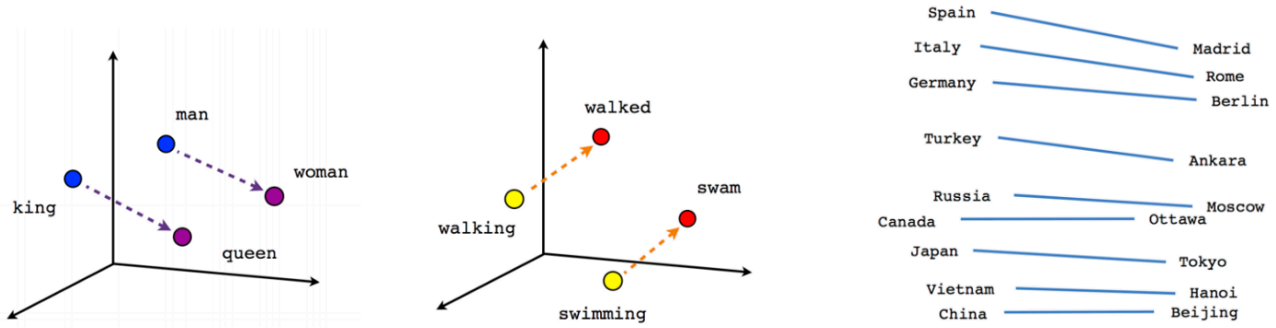
- Le vocabulaire est une limite parce qu'il nécessite de comparer des vecteurs parfois immense.
- Sparsité: la representation entre les vecteurs n'ont pas de liens, connections
- La sémantique n'est pas bien capturé, il n'y a pas de différence entre "tu trouves ça intéressant" et "trouves tu ça intéressant". De même il y a une différence entre "un vieux vélo" et "un vélo ancien"

Word Embeddings

Word Embeddings est un moyen de représenté chaque mot en tant que vecteur. Ceci permet pour des mots de sens relativement similaires d'avoir également des représentations vectorielles similaires. Les vecteurs se basent donc sur la sémantique et le context des mots. Il existe plusieurs techniques mais voici les 2 plus utilisées:

(1) **Word2Vec** (2013): est un algorithme qui s'entraine en prédisant ou les mots aux alentours d'un mot donné, ou alors le prochain mot d'un contexte donné.
(2) **GloVe** (2014): Il se base sur le Word2Vec mais à la place de vecteur il utilise des techniques de factorisation matricielle de la Latent Semantic Analysis qui calcule la probabilité que deux mots apparaissent en même temps





Male-Female | Verb tense | Country-Capital

Il est possible de créer ce word embedding de façon isolé, c'est à dire uniquement pour ce propos ou alors en même temps que faire une tâche (classification ou autre). Il y a aussi déjà beaucoup de ressources de Word Embeddings disponible et entrainés sur des milliards de données qui sont généralement de très bonne qualité.

Long-Short-Term-Memory (LSTM)

Le Long-Short-Term-Memory network est un réseau de neurones séquentiel, c'est à dire qu'il est capable de traiter des informations en séquence, prenant plusieurs valeurs et d'intégrer les valeurs d'intérêt.
Ce réseau à la capacité d'enlever ou de rajouter de l'information qu'il passe au neurones suivants grâce à des **gates** (portes). Pour ce faire il est composé du **cell state**: qui représente la mémoire à long terme.