Université de Rennes

Valentin Regnault
Rémi Loison

Grand plongeon dans la vectorisation de mots

2022-2023 – Licence 1

Table des matières

1 Introduction

1. **“Au fond, je crois que la Terre est ronde”**
2. **“Après tout, je pense que la Planète bleue est sphérique”**
3. **“Au fond, je vois que la mer reste onde”.**

Voici trois phrases. La première et la deuxième ont le même sens, elles signifient la même chose. Pourtant, elles ont peu de caractères en commun. Il faudrait ajouter, supprimer ou remplacer 30 caractères pour passer de l’une à l’autre. À l’inverse, la première et la dernière sont très proches en termes de caractère : il suffirait de remplacer/ajouter/supprimer seulement 8 caractères pour passer de l’une à l’autre. Pourtant, elles ne veulent pas du tout dire la même chose.

Des textes peuvent être similaires de deux façons : en termes de caractère, et en termes de sens. On peut mesurer assez facilement des différences de caractères avec des algorithmes comme la distance de Hamming où la distance de Levenshtein. Il est par contre plus difficile de mesurer les différences de sens, ce qu’on appelle la différence sémantique.

On peut définir la différence sémantique comme une valeur numérique qui dit si deux mots ou des textes ont plus ou moins le même sens. Par exemple, la différence sémantique entre la phrase 1 et 2 est un petit nombre, tandis que la différence sémantique entre la phrase 1 et 3 est un grand nombre. Cela permettrait par exemple, dans le cas d’une barre de recherche, de classer les résultats de la recherche en fonction de leur différence sémantique avec les mots-clefs recherchés.

Mais on peut aller encore plus loin. Plutôt que de mesurer à quel point deux textes ou deux mots diffèrent, on peut mesurer en quoi ils diffèrent. Au lieu d’une valeur numérique, on pourrait avoir une multitude de critère qui nous dirait en quoi les deux textes diffèrent. On pourrait établir une liste de concepts, de propriétés des mots, et pour chaque mots mesurer à quel point il correspond à chaque propriété. Par exemple, le mot « chien » aurait une grande valeur pour la propriété « est un animal » mais une toute petite valeur pour la propriété « est une action ». On aurait alors un ensemble de valeur qui représente de manière unique chaque mot, chaque mot aurait sa propre décomposition en propriétés. Deux mots qui ont une signification proche aurait une décomposition proche, par exemple « chien » et « chienne » ne différerait que d’une propriété : le genre.

D’un point de vue mathématique, un ensemble de valeurs qui correspondent à des propriétés, on appelle cela un vecteur. Décomposer un mot en propriété, c’est le faire correspondre à un vecteur qui a autant de coordonnées qu’il y a de propriétés, et

placer ce vecteur dans un espace qui a autant de dimensions qu'il y a de propriétés. Deux mots au sens proches seraient des points proches dans cet « espace lexical » (qui a des centaines de dimensions). On place les mots dans l'espace, les uns par rapport aux autres.

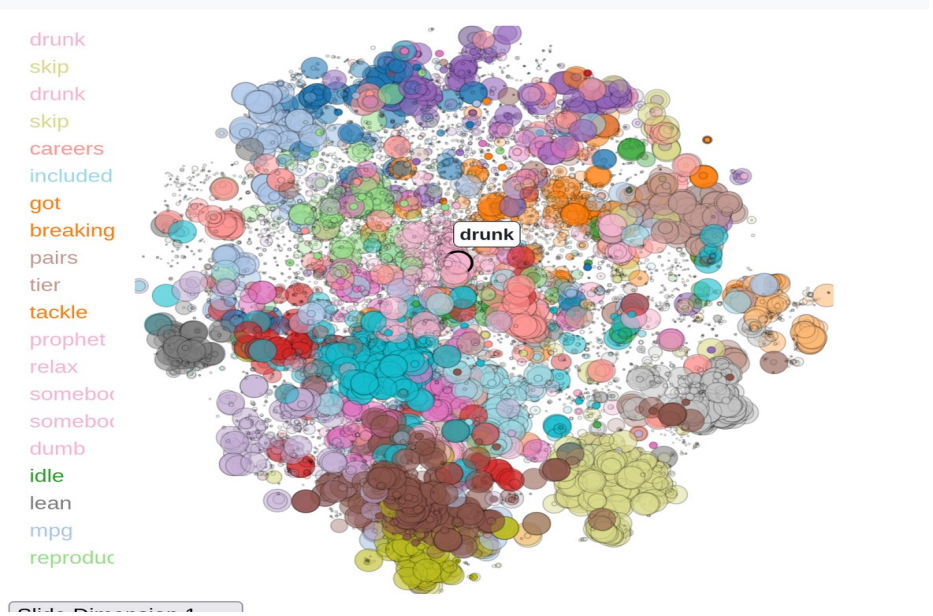


Figure 1: Représentation artistique d'un espace vectoriel lexical, Edwin Chen, (voir en dynamique : http://blog.echen.me/embedding-explorer/#/)

Transformer des mots en vecteurs s'appelle le plongement lexical (on dit qu'on « plonge » un mot dans un espace lexical). Plonger les mots, et en faire des objets mathématiques, est extrêmement puissant. Cela permet, entre autres, de faire des calculs avec des mots : Roi – Homme + Femme = Reine !

Dans ce rapport, nous allons donc nous demander comment plonger des mots. Nous allons étudier principalement deux méthodes : la première s'appelle l'analyse sémantique latente et est un procédé mathématique appliqué à des textes. La deuxième, le Word2Vec, est un regroupement de deux algorithmes de machine learning : le modèle sac de mots continus (Continuous Bag-of-word, aussi appelé CBOW) et le modèle skip-gram. Nos recherches ont été faites à partir d'article scientifiques, de vidéos et articles de vulgarisation, que vous pouvez trouver dans la section bibliographie. Pour commencer, nous devons étudier quelques notions fondatrices du champ d'étude qu'est le traitement des langues naturelles.

2 Prérequis : vecteurs et hypothèse distributionnelle

2.1 Les vecteurs à grande dimensions et les distances

Un vecteur est un ensemble de coordonnées. Chaque coordonnée est un nombre.

Pour mesurer la différence entre deux vecteurs, mesurer à quel point ils sont différents, il existe plusieurs méthodes. Nous allons étudier deux d'entre elles.

a) La distance euclidienne

La distance euclidienne tient son nom du mathématicien Euclide. Elle consiste à soustraire les deux vecteurs, coordonné par coordonné, à mettre aux carrés toutes les différences obtenues, à les additionner et à prendre la racine. Autrement dit :

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

Où x et y sont deux vecteurs à n dimensions et x_i est la $i^{ème}$ coordonnée de x.

b) La similarité cosinus

La similarité cosinus est définie par

$$cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

C'est donc le produit scalaire des deux vecteurs diviser par le produit de leurs normes. La similarité cosinus de deux vecteurs est liée à leur distance angulaire par :

$$Ang.dist. = \frac{cos^{-1}(cos(\theta))}{\pi}$$

c) Quelle mesure faut il utiliser ?

La similarité cosinus est souvent préférée dans le cas du traitement des langues naturelles. Elle produit, dans ce domaine, de meilleurs résultats, en partie car elle est agnostique des normes des vecteurs. Comme deux vecteurs avec des normes très différentes peuvent correspondre à des mots de sens proches après un plongement, la similarité cosinus à de meilleurs résultats pour capturer les similarités de sens.

Cependant il n'existe pas d'étude démontrant que la similarité cosinus à de meilleurs résultats. Tout dépends du contexte d'utilisation et de l'objectif recherché.

2.2 L' hypothèse distributionnelle

En 1954, un linguiste, Zellig Harris, publie un article scientifique fondateur[1] dans le traitement des langues naturelles. Dans cet article, il explore une idée géniale : l'hypothèse distributionnelle. Nous allons essayer de comprendre ce qu'est cette hypothèse, en quoi elle permet de mesurer le sens des mots, et nous verrons comment transformer les mots en objets mathématiques avec le plongement lexical.

a) Le contexte d'un mot

Le point de départ de Harris est l'étude du contexte des mots. Le contexte d'un mot M, c'est l'ensemble des fréquences d'apparitions à côté de M de tous les autres mots du dictionnaire.

« Sous le soleil, la jolie tomate est mûre et sucrée »

« Sous le soleil, la jolie tomate est mûre et sucrée »

« Sous le soleil, la jolie tomate est mûre et sucrée »

Par exemple, dans le contexte du mot « tomate », le mot « mûre » est associé à une grande fréquence d'apparition, car, si l'on étudie une grande quantité de texte, on trouve souvent le mot « mûre » à côté du mot « tomate » (c'est-à-dire dans un certain rayon autour de « tomate », comme dans l'exemple ci-dessus). À l'inverse, le mot « guitare » n'est pas associé à une très grande fréquence dans le contexte de « tomate », car ils sont rarement à coté.

Pour calculer le contexte du mot « tomate », on prend une grande quantité de texte en français, et on regarde chaque apparition de « tomate ». Chaque fois qu'on trouve le mot « tomate », on regarde dans un certain rayon autour de ce mot. On tient compte du nombre d'apparition de chaque mot du dictionnaire autour de « tomate ». On calcule ensuite la fréquence d'apparition en divisant le nombre d'apparition par le nombre de mots qu'on a étudié.

À la fin, on a une table, avec une ligne par mot du dictionnaire, et une fréquence d'apparition.

Tableau 1: Exemple fictif du contexte du mot tomate

arbre	0.05
abricot	0.12
...	...
zombie	0.001

b) L'hypothèse distributionnelle :

À quoi sert de connaître le contexte d'un mot ? C'est là qu'entre en jeu l'hypothèse distributionnelle.

« On ne peut connaître un mot que par ses fréquentations »[2] disait John R. Firth. C'est une formule simple pour dire que le sens d'un mot est caractérisé par son contexte. Le contexte d'un mot donne énormément d'information sur son sens.

Cela signifie que si des mots ont des contextes proches, alors ils ont des sens proches. À l'inverse, s'ils ont des sens totalement différents, alors les fréquences des mots de leurs contextes sont très différentes.

Mieux que cela : le contexte d'un mot contient des informations sur les propriétés de ce mot. Par exemple, le contexte d'un mot peut donner des informations sur sa nature, sa fonction, s'il est plutôt positif ou négatif, etc. En conséquence, si on peut « calculer » avec le contexte des mots. Si l'on cherche le mot qui a à la fois un contexte proche de celui de « roi » et de « femme », mais qu'en même temps le contexte de ce mot est le plus différent possible du contexte d' « homme », on trouve le mot « reine ». Roi + Femme – Homme = Reine !

3 L'analyse Sémantique Latente[3]

Dans cette partie nous allons rentrer dans le cœur du sujet, et étudier une méthode de plongement lexical : l'analyse sémantique latente[3]. Cette méthode consiste à construire une matrice d'occurrences et à réduire sa dimension avec un calcul mathématique : la décomposition en valeurs singulières (SVD, Singular Value Decomposition).

3.1 Les matrices d'occurrences

Plonger un mot ou un texte nécessite une grande quantité de données textuelles, un grand nombre de textes. On peut utiliser l'ensemble du texte de Wikipédia par exemple, ou scanner les pages internet avec des robots.

Une fois qu'on a un grand corpus, séparé en documents (par exemple Wikipédia est séparé en pages), la manière la plus simple pour plonger du texte est de compter les mots. Dans chaque document, on compte le nombre d'occurrence de chaque mot du dictionnaire.

On construit ensuite une matrice, un tableau à 2 entrées : les mots du dictionnaire en abscisse et les documents en ordonnée. Chaque case contient le nombre de fois que le mot correspondant apparaît dans le document correspondant. Un exemple vaut mille mots, voici une matrice d'occurence, dans laquelle chaque case contient le nombre d'apparition d'un mot dans un document (les documents sont ici de simples phrases) :

Tableau 2: Exemple de matrice d'occurrences, avec 4 documents et 8 mots différents

	Histoire	s'écrit s'écrivent	en	tournant	La les L'	pages	Vie	Mots
L'histoire s'écrit en tournant les pages	1	1	1	1	2	1	0	0
La vie s'écrit en tournant les pages	0	1	1	1	2	1	1	0
Les pages s'écrivent en tournant les mots	0	1	1	1	2	1	0	1
La vie s'écrit en tournant les mots	0	1	1	1	2	0	1	1

Cette matrice suffit à plonger des mots ou des textes. En effet, chaque phrase est associée à un vecteur (les coordonnées de ces vecteurs sont les nombres de la ligne). On peut comparer ces vecteurs, par exemple la similarité cosinus entre la première ligne et la deuxième ligne est d'environ 0.9 tandis que celle entre la première et la dernière ligne est d'environ 0.8 (c'est moins, et en effet la première et la deuxième phrase sont plus proches que la première et la dernière).

Ceci est un exemple de plongement, avec un vocabulaire (un nombre de mots différents) très petit, et très peu de documents. Dans un cas réel de plongements, nous pourrions avoir des centaines de milliers de documents, et un vocabulaire (c'est-à-dire le nombre de mots différents dans le corpus de texte étudié) énorme, de plusieurs dizaines de milliers de mots. Cela signifie qu'on obtiendrait des centaines de milliers de vecteurs, avec chacun des dizaines de milliers de coordonnées ! Il est impossible de travailler sur autant de données, à grande échelle. Cette limitation à écoper de l'appellation « fléau de la dimension ». Et l'analyse sémantique latente résous ce fléau grâce aux mathématiques.

3.2 Décomposition en valeurs singulières

L'analyse sémantique latente ne s'arrête pas à simplement construire une matrice d'occurrence. L'ASL applique ensuite un procédé mathématique qu'on appelle la décomposition en valeurs singulières (Singular Value Decomposition en anglais, SVD). Cette méthode permet de réduire le nombre de colonnes de la matrice, tout en préservant les relations entre les lignes. Autrement dit, après la SVD, nos documents sont représentés par des vecteurs avec beaucoup moins de coordonnées, mais qui ont toujours les mêmes relations entre eux : si deux vecteurs avaient une similarité cosinus de 0.8 avant la SVD, ils auront (presque) la même similarité après avoir été réduits.

Nous n'allons pas expliquer le détail de la décomposition en valeur singulière, mais simplement survoler l'intuition générale derrière cet outil mathématique.

Imaginons qu'on ait 100 articles qui parlent de politique (cet exemple est tiré des explications limpide de Loana sur Medium[4]). Admettons que ces 100 articles tournent autour de 3 sujets : les élections, la politique extérieure et les réformes. Certains articles parlent d'un sujet exclusivement, certain de deux sujets à la fois et d'autres des 3 sujets.

Pour plonger nos articles et obtenir des vecteurs qui représentent ces documents, on commence par calculer une matrice d'occurrences. L'objectif est de réduire le nombre de colonnes de cette matrice. Pour cela, on va décomposer la matrice en 3 sous matrices.

La première matrice associe les articles aux sujets :

Tableau 3: Exemple des articles politiques, une matrice associant les articles aux sujets (Matrice document-concepts).

	Élections	Politique extérieur	Réformes
Article 1	0.9	0.05	5.2
Article 2	3.62	1.05	8.06
...			

Dans chaque case, plus le nombre est grand plus l'article est en rapport avec le sujet.

La deuxième matrice est une matrice qui associe chaque terme à chaque sujet par un nombre plus ou moins grand, selon si le terme est plus ou moins associé au sujet.

Tableau 4: Exemple des articles politiques, une matrice associant les termes aux sujets (Matrice terme-concepts).

	Élections	Politique extérieur	Réformes
Union Européenne	0.9	8.02	1.02
Facture	0.1	2.02	6.06
...			

Par exemple, « Union Européenne » est un terme plus souvent utilisé pour parler de politique extérieur que pour parler des élections.

Enfin, dans la troisième matrice, on associe chaque sujet à une valeur. Plus cette valeur est grande, plus le sujet « exprime » les articles. Que signifient « exprimer les articles » ? Imaginons que les élections étaient un sujet que la plupart des articles évoquent sans en parler plus que cela, et que la plupart des articles parlent plutôt des politiques extérieures et des réformes. Attention, les élections sont évoquées dans presque tous les articles, ce n'est juste pas le sujet principal. Dans cette situation, on remarquerait dans la seconde matrice qui associe chaque article aux sujets que la plupart des articles ont des valeurs très proches dans la colonne « élections ». En effet, tous les articles parlent plus ou moins autant des élections. On dit alors que le sujet « n'exprime pas beaucoup » l'article. Cela veut dire qu'il ne permet de différencier les articles les uns des autres.

En somme, un sujet évoqué en détail dans peu d'articles donne beaucoup d'information, tandis qu'un sujet survolé dans tous les articles ne permet pas vraiment de différencier les articles les uns des autres. Dans le premier cas, le sujet « exprime » beaucoup les articles, tandis que dans le deuxième cas, le sujet « exprime » peu les articles.

Du point de vue des vecteurs, cela signifie qu'une colonne associée à un sujet qui exprime peu les articles peut être retirée sans modifier substantiellement les

relations entre les vecteurs. Tout les vecteurs sont plus ou moins au même endroit dans cette dimension (ils sont sur le même plan), on peut alors ignorer cet dimension.

En résumé, La troisième matrice est donc une matrice qui associe chaque sujet à un nombre, qui dit à quel point le sujet « exprime » les sujets.

De manière générale, lorsqu'on traite du texte on peut construire 4 matrices :

- M, La matrice d'occurrences originale, qui à m **lignes et** n **colonnes (**m **documents et** n **termes différents dans le corpus. Dans notre** m **= 100 articles et** n **= le nombre de termes différents dans les articles.).**

- U, la matrice document-concepts. U est une matrice (m, r**) où** r **est le nombre de concepts (sujets).**

- Σ**, est une matrice diagonale (**r, r**), c'est la matrice qui associe chaque concept (sujet) à une valeur qui dit à quel point il exprime les documents (articles). Les valeurs sont dans la diagonale de la matrice et le reste de la matrice est composé de 0.**

- V est une matrice (n, r**) qui associe les** n **termes aux** r **concepts**

La décomposition en valeur singulière dit que pour toute matrice m **et n'importe quel** r**, il existe des matrices** U**,** Σ **et** V **tel que :**

$$M = U * \Sigma * V^T$$

(V^T **est la matrice transposée de** V**, les colonnes et les lignes sont inversées).**

Il existe différent algorithmes pour calculer cette décomposition, que nous n'étudierons pas ici. L'intuition derrière ces algorithmes est toujours plus ou moins la même : on suppose que l'hypothèse distributionnelle est vraie, donc que les mots avec le même sens apparaissent dans les mêmes contextes. Grâce à cette hypothèse, on peut regrouper les mots qui apparaissent dans des contextes similaires, car ils partagent un même concept commun, ils sont liés sémantiquement. Chaque groupe de terme est un concept. On trouve les r **meilleurs concepts, c'est-à-dire les concepts qui rassemblent le plus de termes. Cela nous permet de créer la matrice V, qui associe les mots à des concepts. On calcule ensuite la matrice** Σ **qui nous dit à quel point chaque contexte exprime les documents. On en déduit la matrice U, qui associe les documents aux concepts. La matrice U est alors le résultat de notre plongement : chaque document est associé à un ensemble de valeurs qui disent a quel point il est proche de chacun des** r **concepts, on a en fait créer des vecteurs de** r **coordonnées qui représentent chaque document.**

r est un nombre arbitraire, qui correspond au nombre de coordonnées des vecteurs que l'on souhaite construire. Plus r est grand, plus les vecteurs qui représentent nos documents sont grands. Plus les vecteurs sont grands mieux les relations entre les vecteurs sont préservés par rapport à avant, quand on travaillait avec une matrice d'occurrences. Mais plus r est grands, plus nos vecteurs prennent de la place en mémoire et du temps de calcul pour êtres traités.

Enfin, si l'on souhaite plonger juste un mot et pas un document complet, on peut :

$$\text{vecteur}(w^t) = w^t * U$$

Où w est la colonne de la matrice d'occurrence M qui correspond au mot qu'on souhaite plongé, et w^t est la matrice transposée (colonne et lignes inversées) de w.

3.3 Conclusions sur l'analyse sémantique latente

L'analyse sémantique latente est une méthode qui permet de plonger des documents ou des mots. Il est pertinent de l'utiliser pour catégoriser des documents, par exemple pour grouper des articles qui ont les mêmes sujets.

Il existe de nombreuses variantes et amélioration de l'analyse sémantique latente. On peut améliorer la matrice d'occurrences en utilisant le TF-IDF, une mesure qui permet de donner plus d'importance aux mots dit « spécifique », c'est-à-dire des mots qui ne sont pas génériques et utilisés partout (comme les déterminants et les pronoms) mais qui sont spécifique à certains documents (comme les termes techniques). Cela permet d'avoir dès le départ une matrice d'occurrences qui représente mieux les documents.

On peut aussi filtrer certains mots qu'on sait qu'il ne donne pas beaucoup d'informations (comme les déterminants et les pronoms), et on peut enfin ramener les mots à leurs racines (« mangeons » devient « manger » par exemple). Toutes ces techniques améliorent les performances du plongement.

On évalue les performances d'un plongement par analyse sémantique latente en comparant les relations entre les vecteurs « bruts » de la matrice d'occurrences avec les relations entre les vecteurs issus du plongement. On souhaite que ces relations soient le mieux préservées possible par le plongement. Comme dit plus haut, si deux vecteurs étaient similaires avant d'être réduit par la décomposition en valeurs singulières, leurs versions réduites doivent être tout aussi proches.

Il se trouve que l'analyse sémantique latente à de bonnes performances comparée aux autres techniques sur des petites quantités de texte, sur des petits corpus. Mais

sur de très grandes quantités de données, les techniques utilisant des modèles de prédictions et du machine learning sont plus efficaces[5].

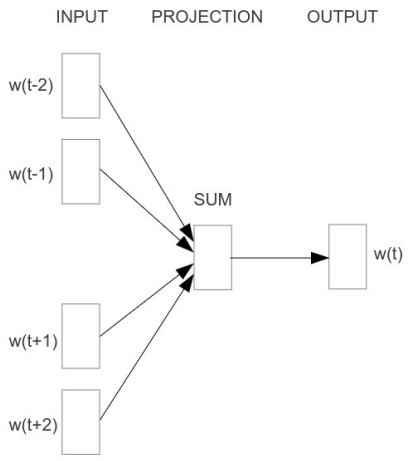
4 Le plongements lexical par machine learning

En 2013, une équipe de Google menée par Tomas Mikolov, publie un article sur une technologie qu'ils nomment Word2Vec[6][7]. C'est en fait le regroupement de deux algorithmes similaires permettant de plonger des mots, tous deux basés sur le machine learning.

Nous considérons dans ce document que les lecteurs à des connaissances préalables du machine learning, notamment qu'il maîtrise la notion de réseau de neurones et sait à quoi sert la rétro-propagation du gradient (backpropagation).

Le premier algorithme se nomme le Continus Bag of Word (sac de mot continu) où CBOW, et le deuxième se nomment le modèle Skip-gram. Nous allons étudier ces deux modèles, et les comparer.

4.1 Le modèle CBOW



CBOW

Soit N le nombre de mot différent dans le corpus de texte utilisé pour entraîner le modèle, soit R le nombre de coordonnées des vecteurs crées.

Le modèle CBOW est un réseau de neurones composé de 3 couches :

- **Une couche d'entrée de N neurones.**
- **Une couche cachée de R neurones.**

- **Une couche de sortie de N neurones.**

Le modèle prend en entrée un bout de texte, de quelques mots (entre 3 et une vingtaine). Il est entraîné à prédire le mot au centre du bout de texte en fonction des mots autour. Par exemple :

« Sous le soleil, la jolie tomate est mûre et sucrée »

Le modèle prend les mots en vert en entrée, et prédit le mot rouge en sortie.

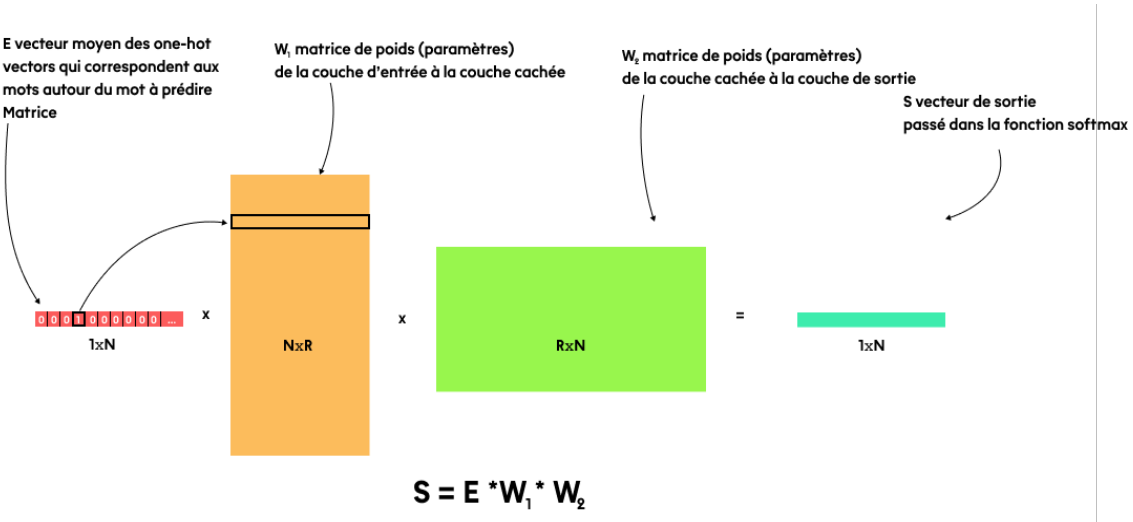
Le modèle prend en entrée des « one-hot vector » de N coordonnées, c'est-à-dire des vecteurs composés de N-1 zéros et de 1 un. Cela permet de représenter un mot. Par exemple, le troisième mot du dictionnaire est représenté par un vecteur dont la troisième coordonnée est un 1, et toutes les autres sont des 0.
Il prédit en sortie le mot central. La sortie est passée dans une fonction softmax, qui va normaliser les coordonnées du vecteur de pour que toutes les coordonnées soient entrées 0 et 1 et que leur somme fasse 1. Autrement dit, après être passée dans la fonction softmax, la $i^{ème}$ coordonnée du vecteur correspond à la probabilité que le mot central soit le $i^{ème}$ mot du dictionnaire.

Pour utiliser le modèle CBOW, on fait la moyenne des one-hot vectors des mots autour du mot à prédire, et on utilise le résultat comme entrée du réseau de neurone. Par exemple, si on a le 38^{eme} mot du dictionnaire, le 234^{eme} et le 988^{eme}, alors on donne en entrée du modèle un vecteur dont la 38^{eme}, la 234^{eme} et la 988^{eme} coordonnée valent 1/3.

L'entraînement du réseau de neurones se fait avec une grande quantité de texte, dont on extrait des bouts de phrases, qu'on passe dans le réseau de neurones. On calcule l'erreur entre ce qui était attendu et ce qui est obtenu, et rétro-propage cette erreur pour corriger les paramètres.

Quel est le rapport avec le plongement lexical ?

Regardons les opérations faites par le réseau de neurones d'un point de vue matriciel :



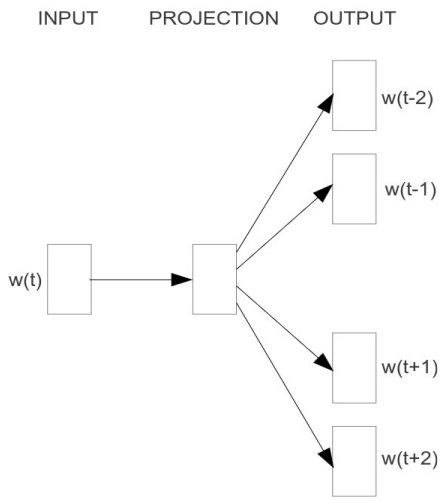
L'idée du modèle CBOW est d'entraîner le modèle à prédire le mot au centre d'un bout de texte, et d'ensuite extraire la matrice de poids.

En effet, la matrice W_1, qui correspond au poids (paramètres) entre la première et la deuxième couche, est une matrice NxR. Si on la multiplie par un one-hot vector qui à un 1 à la $i^{ème}$ coordonnée, alors on obtient la $i^{ème}$ ligne de la matrice W_1. Cette $i^{ème}$ est un vecteur ligne de R coordonnées. En fait, les plongements de chaque mot du dictionnaire sont stockés dans les poids du réseau de neurone ! La matrice W_1 contient les vecteurs de chaque mot du dictionnaire.

Autrement dit, le modèle CBOW prend un vecteur de N coordonnées en entrée, le « compresse » pour le faire passer dans une couche de R neurone de large, et ensuite retourne un vecteur de N coordonnées. La première matrice qui permet de passer de N à R coordonnées a appris à représenter un mot par un vecteur qui contient assez d'information sur le contexte pour pouvoir ensuite déduire le mot central.

Le modèle skip-gram est presque le même modèle que le CBOW,à l'envers.

4.2 Le modèle skip-gram



Skip-gram

Le modèle skip-gram prend en paramètre le mot central et essaye de prédire le contexte dans lequel il se trouvait (les mots autour). Comme le modèle CBOW, il prend un one-hot vector en entrée et applique la fonction softmax sur la dernière couche de manière à obtenir pour chaque mot du dictionnaire la probabilité qu'ils apparaissent autour du mot central.

De la même manière que pour le modèle CBOW, ce qui nous interresse n'est pas temps la capacité à prédire le contexte que la matrice qui transforme un mot sous forme de one-hot vector en un vecteur de R coordonnées dense (par dense on entend que ceux vecteur à des informations dans chacune de ces coordonnées, par opposition aux one-hot vectors qui ont qu'une seule coordonnée utile).

4.3 Comparaison des deux modèles CBOW et skip-gram

Pour comparer les performances des deux modèles, les auteurs de Word2Vec posent des questions aux deux modèles. Des questions du type « Quel est le mot qui ressemble à « Italie » de la même manière que le mot « France » ressemble à «Paris» ? ».

Pour répondre à cette question, on peut simplement calculer le vecteur X = vecteur(«Paris») - vecteur(«France») + vecteur(«Italie») **et chercher le vecteur le proche de X, par similarité cosinus, parmi les vecteurs de tout les mots du dictionnaire. Si le modèle est bien entraîné et que les vecteurs de chaque mot sont bien placés les uns par rapport aux autres dans l'espace lexical, alors on devrait trouver « Rome ».**

CBOW surpasse légèrement skip-gram quand il s'agit de répondre à des questions syntaxiques, , par exemple : «Quel est le mot qui ressemble à «vendu » de la même manière que «mangé» ressemble à « manger » ? ». Dans cette question on essaye de mesurer si le modèle est capable de repérer la différence entre un infinitif et un

participe passé. De manière générale, sur les questions de syntaxes, CBOW a de meilleurs résultats (64 % de réussite à trouver le mot, contre 59 % pour skip-gram). Cependant, sur les questions de sémantique comme la question avec les villes et les pays, skip-gram est largement meilleurs (55 % de réussite à trouver le mot, contre 24 % pour CBOW).

4.4 Pistes d'amélioration

Comme pour l'analyse sémantique latente, il existe de très nombreuses améliorations possibles de Word2Vec.

On peut découper le texte en token plutôt qu'en mots. Les tokens sont des mots, des bouts de mot ou des caractères. Découper en token permet de prendre en compte les suffixes et préfixe qui peuvent apporter du sens aux mots.

On peut pendant la phase d'entraînement pondéré les mots du contexte en fonction de leur distance avec le mot central.

Des méthodes plus avancées encore comme GloVe vont prendre en compte la probabilité d'apparition de deux mots à côté.

5 Conclusion

Nous avons étudié l'hypothèse distributionnelle, qui dit que le contexte d'un mot donne une indication sur son sens. Nous avons parlé de deux méthodes pour vectoriser des mots et des documents, parmi de nombreuses autres.

La première, l'analyse sémantique latente, est une méthode mathématique qui permet de réduire les dimensions de la matrice d'occurrences en regroupant les mots qui apparaissent dans les mêmes contextes, et qui ont donc un sens commun.

La deuxième, Word2Vec, est un regroupement de deux algorithmes de machine learning, CBOW et skip-gram. Dans les deux cas, on va entraîner des réseaux de neurones à des taches qui ne nous intéressent pas, puis extraire de leurs paramètres les vectorisations des mots.

Pour comparer les performances de ces deux méthodes, on peut utiliser la méthode décrite dans la 4.3. Cependant, il faut faire attention au fait que les modèles de Word2Vec auront tel quel beaucoup de mal à plonger des documents, contrairement à l'analyse sémantique latente. Il existe un technique, nommé doc2Vec, qui résoud ce problème.

L'analyse sémantique latente à de meilleurs résultats sur des petits corpus de textes et peut très vite devenir gourmande en temps de calcul, tandis que Word2Vec a de

meilleurs résultats sur des grandes quantités de texte et est plus facile à entraîner à grande échelle[5]

Des extensions de ces méthodes, comme GloVe qui est basé sur Word2Vec, améliorent significativement les performances. Des méthodes innovantes comme ELMo surpasse aujourd'hui Word2Vec et l'analyse sémantique latente, d'assez loin.

Enfin, le plongement est une notion qui ne s'applique pas qu'au texte, mais aussi aux images, aux sons, et à toutes formes de données de manière générale. Des start-up de bases de données vectorielles, qui stockent les données sous leurs formes vectorielles, ont une croissance impressionnante et un avenir prometteur, tant elles permettent de rapidement rechercher des données similaires. Le plongement est un champ de recherche au fort potentiel, qui risque de devenir incontournable dans l'informatique des prochaines années.

Bibliographie

1: Zellig S. Harris, Distributional Structure, 1954
2: John Rupert Firth, Selected Papers of J. R. Firth, 1952-59, 1968
3: Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K., Indexing by latent semantic, 1990
4: Ioana, Latent Semantic Analysis: intuition, math, implementation, , https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8
5: Edgar Altszyler, Mariano Sigman, Sidarta Ribeiro and Diego Fernández Slezak, Comparative study of LSA vs Word2vec embeddings,
5: Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, 2013
6: Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, Distributed Representations of Words and Phrases, 2013