Microsoft en France
Ecole IA

SIMPLON.CO

Etat de l’art

La représentation avancée des mots dans le traitement du langage naturel : le plongement de mots

Rédacteur : Quentin Angelot

Introduction

L'IA en particulier et l'informatique en général sont des domaines en perpétuelle évolution : nouveaux frameworks, nouvelles méthodes de travail, nouvelles technologies du cloud, nouveaux modèles de NLP, de computer vision...

La veille technologique répond donc à un besoin de formation continue, c'est une condition sinequanone pour être un professionnel accompli dans ces métiers. A chaque nouvelle mission, il faut s'adapter aux technologies et algorithmes les plus à même de répondre aux besoins. D'où l'intérêt de développer la bonne méthodologie en la matière.

Dans le domaine du machine learning, l'état de l'art peut remplir un objectif double :

- d'un côté, il peut se concentrer sur les dernières avancées en matière d'IA
- de l'autre, sur les évolutions technologiques qui permettent d'actionner cette IA sur un cas d'usage précis (méthodes agiles, devops, cloud…)

Dans le cadre d'un projet de fin d'études sur l'IA appliquée à la prédiction de la fréquentation des restaurants scolaires nantais, nous avons à disposition un jeu de données qui reprend les différents menus des 10 dernières années dans ces restaurants.

Afin de valoriser ces données et d'en extraire de l'information, j'ai appliqué des techniques assez classiques de la NLP :

- Le sac de mots : il permet d'obtenir les mots et les n-grams, issus des menus, qui sont associés à chaque jour. Ensuite, nous pouvons dériver les mots ou les n-grams les plus discriminants quant à la fréquentation.
- La similarité de document : c'est le processus d'utilisation d'une métrique basée sur la distance ou la similarité qui peut être utilisée pour identifier à quel point un document textuel est similaire à tout autre document en fonction de caractéristiques extraites comme le sac de mots.
- L'allocation de Dirichlet latente : elle permet d'extraire des thèmes à partir des différents menus. On fixe un nombre k de thèmes et on cherche à apprendre la proportion de chaque thème représenté dans chaque document et les mots associés à ces thèmes. Ensuite, nous pouvons dériver des caractéristiques sur les thèmes les plus discriminants quant à la fréquentation.

Ces techniques n'ont pas eu un impact significatif sur l'amélioration de la capacité prédictive du modèle. A partir de là, deux conclusions sont possibles :

1. Les features textuelles dérivées du menu sont redondantes par rapport aux autres features des différents jeux de données à disposition. Ainsi, si tous les lundis, on sert un menu végétarien, alors l'information sur le menu sera déjà contenue dans la date.

2. Les techniques utilisées ne sont pas les bonnes au regard de ce jeu de données précis. En effet, le sac de mots par exemple consiste en un simple comptage sans aucune information sur l'ordre ou la signification des mots.

Dans le second cas, il faut donc s'intéresser à une autre manière de dériver des features textuelles pour le machine learning : trouver une méthode plus sophistiquée par exemple, qui tienne davantage compte du contexte.

C'est ainsi qu'émerge une problématique claire : **comment dériver des features textuelles plus informatives d'un point de vue contextuel afin d'améliorer la puissance prédictive du modèle ?**

Après une brève recherche, il apparaît que le word embedding pourrait répondre à cette problématique. C'est un sujet complexe qui nécessite donc un état de l'art rigoureux. Ainsi, il ne sera pas nécessaire de reproduire le processus de recherche dans son intégralité, mais au contraire, nous pourrons valoriser le travail et les conclusions ayant déjà été validées par des pairs.

Le word embedding (ou plongement de mots) est une représentation de texte où les mots qui ont le même sens ont une représentation similaire. En d'autres termes, cette technique représente des mots dans un système de coordonnées où les mots liés sémantiquement, sont placés plus près les uns des autres. Ce processus d'embedding repose sur des techniques de le deep learning.

I. Recherche bibliographique

Pour créer des plongements de mots, il est nécessaire d'avoir à disposition un corpus de texte et une méthode d'embedding.

Le contexte d'un mot indique quel type de mots ont tendance à se produire à proximité de ce mot. Le contexte est important car c'est ce qui donnera un sens à chaque embedding. Les techniques d'embedding populaires ont émergé au début des années 2010, avec l'émergence du deep learning à la même époque.

Le concept de deep learning prend forme dans les années 2010, avec la convergence de plusieurs facteurs : les réseaux de neurones artificiels multicouches, le big data et la puissance de calcul exponentielle des machines.

a. Word2vec

Ainsi, en 2013 fut publié l'article "Efficient Estimation of Word Representations in Vector Space" par Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean (Google). Dans ce papier, ils décrivent le modèle Word2vec, qui a initialement popularisé l'utilisation de l'apprentissage automatique, pour générer des plongements de mots. Word2vec utilise un réseau de neurones pour apprendre les embeddings. Il propose deux architectures de modèles :

- Continuous Bag of Words, qui est un modèle simple mais efficace, dont l'objectif est d'apprendre à prédire un mot manquant en fonction des mots environnants.
- Continuous skip-gram, plus complexe mais plus puissant qui, à l'opposé de CBOW, apprend à prédire le mot qui entoure un mot d'entrée donné.

b. Glove

L'année suivante, face au retentissement de ce premier papier et à l'engouement pour le deep learning en général, l'article "GloVe: Global Vectors for Word Representation" est publié par les chercheurs à Stanford Jeffrey Pennington, Richard Socher et Christopher D. Manning. Glove implique la factorisation du logarithme de la matrice de cooccurrence des mots du corpus. Cela permet à GloVe de ne se baser uniquement sur des statistiques locales comme Word2vec, mais également sur des statistiques globales (cooccurrence des mots) pour créer l'embedding. De ce fait, Glove est une nette avancée par rapport à Word2vec qui ne prenait en considération que le contexte local.

c. fastText

En 2016, un autre géant du numérique s'intéresse de près au traitement du langage naturel et au plongement de mots. En effet, Facebook publie l'article "Enriching Word Vectors with Subword Information" par Piotr Bojanowski, Edouard Grave, Armand Joulin et une fois encore Tomas Mikolov (à l'origine de Word2vec). fastText est basé sur le modèle skip-gram et prend en compte la structure des mots en les représentant comme un n-gram de caractères. Cela permet au modèle de prendre en charge des mots inédits en déduisant leur embedding à partir de la séquence de caractères dont ils sont composés. Un autre avantage de fastText est que les vecteurs d'intégration des mots peuvent être moyennés ensemble pour obtenir des représentations vectorielles de phrases et d'expressions.

Concernant la fiabilité de ces articles, ils ont été rédigés respectivement par des chercheurs de Google AI, Stanford et Facebook Research. De plus, ils ont été cités respectivement 22325, 21393 et 5984 fois dans d'autres travaux de recherche.

II. Synthèse des découvertes

a. Les techniques de base de l'extraction de features textuelles

La façon la plus simple d'encoder des mots sous forme de nombre est, pour un vocabulaire donné, d'attribuer un entier unique à chaque mot. C'est simple mais cela n'a aucun sens sémantique.

Un autre moyen est la représentation vectorielle "one-hot". Quand le document (i.e, le texte) possède le mot en question, on signale sa présence par un 1 sinon 0. Le sac de mots, que nous avons évoqué précédemment, ne se contente pas de marquer l'occurrence mais fonctionne avec le compte, ainsi si le mot est répété deux fois dans le document, on signale sa présence par un 2.

Cela reste une technique assez simple mais ça a déjà plus de sens que l'encodage ordinal. De plus, cette technique et son extension TF-IDF (qui permet d'évaluer l'importance d'un terme contenu dans un document, relativement au corpus) sont souvent très efficaces pour solutionner de nombreux problèmes de traitement du langage naturel. Cependant ces techniques ne considèrent pas la sémantique, la structure et le contexte des mots voisins dans chaque document. D'autre part, ces modèles conduisent à des matrices creuses surdimensionnées majoritairement remplies de 0 qui peuvent parfois poser des problèmes de convergence à un algorithme de machine learning.

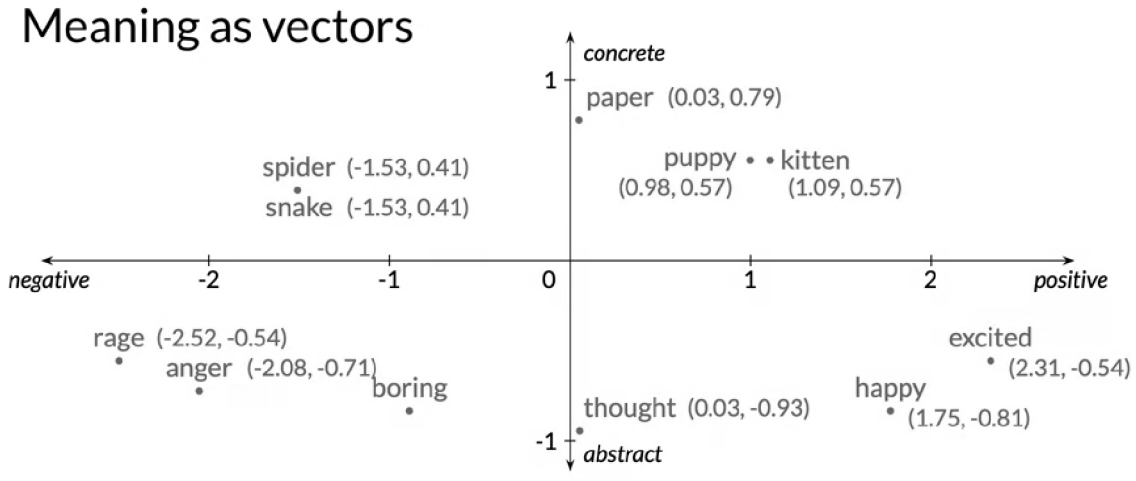
Pour répondre à ces limites, des méthodes statistiques traditionnelles permettent de capturer la sémantique et de réduire la dimension. Ainsi l'analyse sémantique latente permet cela. La LSA apprend les sujets latents en effectuant une décomposition matricielle (SVD) sur la matrice terme-document. L'allocation de Dirichlet latente (LDA) suppose que chaque document est généré par un processus génératif statistique. En d'autres termes, chaque document est un mélange de sujets, et chaque sujet est un mélange de mots. En pratique, la LDA est plus performante que la LSA.

Cependant les techniques les plus utilisées aujourd'hui, car les plus efficaces en termes de ratio performance / temps, sont les techniques basées sur les réseaux neuronaux et notamment le word embedding.

b. Le word embedding ou plongement de mots

Le word embedding est donc une représentation de texte où les mots qui ont le même sens ont une représentation similaire. En d'autres termes, il représente des mots dans un système de coordonnées où les mots liés, basés sur un corpus de relations, sont placés plus près les uns des autres.

Dans le projet de prédiction de la fréquentation des cantines nantaises, nous n'avons pas besoin d'un word embedding mais plutôt d'un document embedding car les menus sont constitués de plusieurs plats et donc de plusieurs mots. Une pratique courante et d'obtenir celui-ci en faisant la moyenne des embeddings des mots qui composent le document. Ici la moyenne des vecteurs des mots du menu donne le vecteur du menu pour un jour donné. Si on se base sur une représentation en 2D, facilement assimilable par le cerveau humain, on peut représenter l'embedding comme sur la figure suivante. Ici, les adjectifs excité et heureux sont proches car ils sont très similaires d'un point de vue de la sémantique. Bien-sûr, un embedding à deux dimensions ne pourra jamais capter toute la sémantique en question et dans la pratique, la dimensionnalité sera bien plus élevée.



Représentation d'un embedding à deux dimensions

Il existe de nombreux modèles de plongement de mots, abordons quelques-uns des plus populaires.

1. Word2Vec

Présentation générale

C’est un modèle basé sur des réseaux de neurones relativement peu profond qui va calculer et générer des représentations vectorielles denses des mots du corpus. Sa force est qu’il capture la similitude sémantique.

Il s'agit d’un modèle non supervisé qui peut intégrer des corpus textuels massifs, créer un vocabulaire de mots possibles et générer des embeddings denses pour chaque mot dans l'espace vectoriel représentant ce vocabulaire. La taille du vecteur est un hyperparamètre du modèle et le nombre de vecteurs correspond à la taille du vocabulaire.

Comme nous l’avons vu les auteurs de Word2vec proposent deux architectures de modèle :

- le Continuous Bag of Words (CBOW) qui prédit le mot manquant en donnant juste les mots environnants
- le continuous skip-gram qui lui fait l'inverse de la méthode CBOW, il apprend à prédire les mots entourants un mot d'entrée donné

La mécanique détaillée de Word2vec avec CBOW

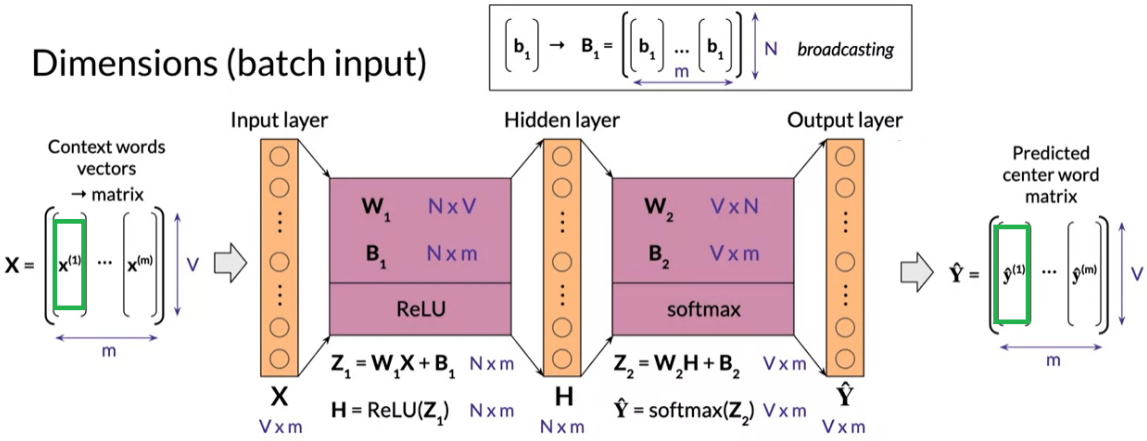
L’idée derrière le modèle est très simple : si deux mots sont fréquemment entourés par des ensembles de mots similaires alors ces deux mots ont tendance à être liés sémantiquement. Le jeu de données d'entraînement est donc constitué des mots environnants (X) et du mot central (y).

Ensuite, comme un réseau de neurones ne peut pas travailler directement avec des mots, le dataset est encodé de façon simple :

Context words	Context words vector	*Center word*	Center word vector
I am because I	[0.25; 0.25; 0; 0.5; 0]	*happy*	[0; 0; 1; 0; 0]
am happy I am	[0.5; 0; 0.25; 0.25; 0]	*because*	[0; 1; 0; 0; 0]
happy because am learning	[0.25; 0.25; 0.25; 0; 0.25]	*I*	[0; 0; 0; 1; 0]

Encodage du dataset dans le cadre de CBOW

Le modèle CBOW est basé sur un réseau de neurones dense et peu profond avec une couche d'entrée, une seule couche cachée et une couche de sortie :



Architecture du modèle CBOW

Où V est la taille du corpus, N est la taille de l'embedding et m correspond à la taille du batch (la longueur du dataset X).

Ainsi x(1) est le contexte du mot y(1) et le réseau va prédire ŷ(1). De même pour les m-1 autres mots de X et y.

Le reste est un réseau de neurones classique :

- Forward propagation : on propage l'entrée X (les mots alentours) dans le réseau de neurones jusqu'à obtenir la sortie ŷ (les mots centraux). Puis, on observe une erreur (y - ŷ) qu'il faut maintenant diminuer.
- Backward propagation : cette étape calcule les dérivées partielles du coût par rapport aux poids et aux biais. Ces poids et biais sont mis à jour lors de la descente de gradient grâce aux dérivées partielles précédemment calculées et à une vitesse définie par le learning rate.
- Une itération après l'autre, l'algorithme converge en ajustant les poids et les biais vers l'erreur minimale.

Finalement les paramètres appris par le modèle lors de son entraînement donnent à chaque mot du corpus son embedding. Chaque mot grâce à ce processus de fenêtres contextuelles glissantes tient compte du contexte et de la sémantique.

La logique est la même pour le continuous skip-gram, la différence tient à l'inversion de X et y : le modèle apprend le contexte à partir d'un mot donné. En principe, le continuous skip-gram est plus long à entraîner mais également plus précis dans la qualité de son embedding.

2. GloVe

L'avantage de GloVe, développé par Stanford, par rapport à Word2vec, est qu'il ne s'appuie pas uniquement sur des statistiques locales (informations sur le contexte local des mots), mais intègre des statistiques globales (cooccurrence des mots) pour obtenir des vecteurs de mots.

Il s'avère que chaque type de statistique a son propre avantage. Par exemple, Word2vec, qui capture des statistiques locales, obtient d'excellents résultats dans les tâches d'analogie. Mais contrairement aux méthodes de factorisation matricielles (comme LSA, par exemple), Word2vec est basée sur des fenêtres peu profondes qui présentent l'inconvénient de ne pas opérer directement sur les statistiques de cooccurrence du corpus. Ce qui ne permet pas de tirer parti de la répétition dans les données. Glove essaye de combiner ces deux aspects.

Glove se base donc sur la matrice de cooccurrence pour dériver des relations sémantiques. L'idée de cette matrice est de décrire le nombre de fois qu'un mot donné apparaît en présence d'un autre et ce pour tous les mots du corpus.

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

Étant donné un corpus ayant V mots, la matrice de cooccurrence X sera une matrice V x V.
Ici par exemple, le mot "sat" apparaît en présence du mot "cat" une fois.
A partir de cette matrice, il est donc possible de calculer les probabilités de cooccurrence.

Soit P(k|w) la probabilité que le mot k apparaisse dans le contexte du mot w.

Considérons un mot fortement lié à la "glace", mais pas à la "vapeur", comme "solide". P(solide|glace) sera relativement élevé et P(solide|vapeur) sera relativement faible.

Ainsi, le rapport de P(solide|glace) / P (solide|vapeur) sera grand. Si nous prenons un mot tel que gaz qui est lié à la vapeur mais pas à la glace, alors le rapport P(gaz|glace) / P(gaz|vapeur) sera plutôt petit.

Pour un mot lié aux deux (glace et vapeur), comme l' eau, nous nous attendons à ce que le rapport soit proche de un. On s'attend également à un ratio proche de un pour les mots qui ne sont liés ni à la glace ni à la vapeur, comme la mode .

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Exemple de rapports de probabilités de cooccurrence

L'intuition principale qui sous-tend le modèle est la simple observation que les rapports des probabilités de cooccurrence des mots ont le potentiel de coder une certaine forme de signification.

Ainsi, l'objectif d'entraînement de GloVe est d'apprendre des vecteurs de mots tels que leur produit scalaire soit égal au logarithme de la probabilité de cooccurrence des mots. En raison du fait que le logarithme d'un rapport est égal à la différence des logarithmes, cet objectif associe les rapports des probabilités de cooccurrence aux différences de vecteurs dans l'espace vectoriel des mots.

Comme ces rapports peuvent coder une certaine forme de signification, cette information est également codée sous forme de différences vectorielles. Pour cette raison, les vecteurs de mots résultants sont très performants dans les tâches d'analogie de mots, telles que celles examinées dans le package Word2vec.

3. fastText

fastText est basé sur le modèle skip-gram et prend en compte la structure des mots en représentant les mots comme un n-gram de caractères.

Cela permet au modèle de prendre en charge des mots jamais vus auparavant, appelés Out-of-Vocabulary (OOV) words. Cela se fait en déduisant leur embeddings à partir de la séquence de caractères dont ils sont constitués et des séquences correspondantes sur lesquelles l'algorithme a été initialement formé. Plus simplement, si l'algorithme a déjà été entrainé sur le mot "chat" mais qu'il n'a jamais été confronté au mot "chaton", alors il pourra malgré tout encodé ce mot par le simple fait que "chat" et "chaton" sont très similaires du point de vue des lettres qui les composent.

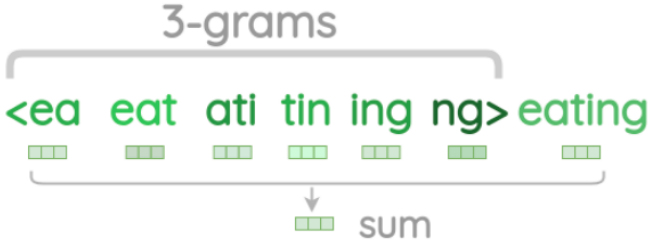
Concrètement l'algorithme, qui se base sur celui du modèle skip-gram, se déroule comme suit :

- Pour chaque mot du vocabulaire, on génère les n-grams de longueur 3 à 6 présents à l'intérieur de celui-ci. Pour le mot eating, cela donne la table suivante :

Word	Length(n)	Character n-grams
eating	3	<ea, eat, ati, tin, ing, ng>
eating	4	<eat, eati, atin, ting, ing>
eating	5	<eati, eatin, ating, ting>
eating	6	<eatin, eating, ating>

Ensemble des n-grams présents à l'intérieur du mot "eating"

- Évidemment, il peut y avoir un grand nombre de n-grams uniques, et dans le papier, les chercheurs appliquent une fonction de hachage pour limiter les besoins en mémoire du modèle. Au lieu d'apprendre un embedding pour chaque n-gram, le modèle apprend un total de B embeddings, où B désigne la taille du corpus après hachage. Dans le papier, B est d'une taille de 2 millions. Bien que cela puisse entraîner des collisions, cela permet de contrôler la taille du vocabulaire et cette approximation est le prix à payer pour avoir un modèle utilisable en pratique.
- A partir de ce hachage, l'embedding pour le mot central est calculé en faisant la somme des vecteurs pour les n-grams de caractères et le mot entier lui-même :



Pour les mots du contexte, nous prenons simplement leur vecteur brut sans se préoccuper des caractères. Au-delà du mot central et des mots du contexte, le papier collecte des échantillons négatifs au hasard dans le corpus (échantillonnage négatif aléatoire). Ce sont ces échantillons négatifs aléatoires qui vont permettre l'entraînement de fastText.

- La suite est très classique : nous calculons le produit scalaire entre le mot central et les mots du contexte et appliquons la fonction sigmoïde pour obtenir un score de correspondance entre 0 et 1. Nous faisons de même avec les mots issus de l'échantillonnage négatif aléatoire.
- Enfin, nous calculons une erreur à l'issue de ces calculs, et en fonction de celle-ci, nous mettons à jour les vecteurs de mots afin de la minimiser (pour se faire, le papier utilise l'algorithme du gradient stochastique). Par un processus itératif, la vectorisation va être corrigée afin de permettre de rapprocher les mots contextuels du mot central tout en augmentant la distance avec les échantillons négatifs.

fastText permet donc de prendre en compte de nouveaux mots étrangers au corpus initial, mais du fait du processing lié aux n-grams, il requiert un temps d'entraînement plus long.

Conclusion

Dans le contexte du projet de prédictions de la fréquentation des cantines nantaises, le menu contient incontestablement de l'information à même de guider le modèle. Il était néanmoins nécessaire de trouver des méthodes suffisamment puissantes pour dériver des features informatives. Cet état de l'art nous permet de faire un choix éclairé quant à la technique à utiliser. Pour ce cas précis, j'utiliserais un modèle de vectorisation simple à mettre en place et peu gourmand en ressources : Word2vec. En effet, ici je n'aurais ni besoin de gérer du vocabulaire hors corpus (fastText), ni besoin de tenir compte de la cooccurrence sur tout le corpus (Glove).

Enfin, il existe désormais des techniques de vectorisation de mots encore plus sophistiquées. Ces méthodes utilisent des architectures complexes de réseaux neuronaux profonds pour affiner la représentation de la signification des mots en fonction de leurs contextes. Dans les modèles précédents, un mot donné a toujours le même embedding. Dans ces modèles plus avancés, les mots ont différents embeddings en fonction de leur sens. Cela permet de prendre en charge la polysémie, c'est-à-dire les mots ayant plusieurs significations différentes suivant le contexte et leur nature. Ces méthodes reposent sur les transformers et le mécanisme de l'attention notamment. On recense notamment le modèle BERT de Google ou GPT-3 développé par OpenAI.

Ces modèles sont très puissants mais ils sont extrêmement gourmands en ressources de calcul et ne respectent pas les contraintes techniques qui me sont imposées dans le cadre du projet des cantines nantaises.

Références bibliographiques

- Efficient Estimation of Word Representations in Vector Space (2013) par Tomas Mikolov, Kai Chen, Greg Corrado & Jeffrey Dean
- GloVe: Global Vectors for Word Representation (2014) par Jeffrey Pennington, Richard Socher & Christopher Manning
- Enriching Word Vectors with Subword Information (2016) par Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov