Similarité textuelle pour le ticketing

Ivanhoé Botcazou
Université des sciences d'Angers
i.botcazou@gmx.fr

Table des matières

1 Introduction

La gestion quotidienne de plus de 500 tickets issus de divers services représente un défi majeur en termes d'efficience opérationnelle pour les employés du Centre Logistique de Pièces de Rechange international (CLPR). L'accès rapide à des informations pertinentes et déjà traitées est crucial pour les équipes métier afin d'optimiser leur réactivité et leur efficacité. Ce projet de fin d'études vise à explorer et développer des méthodes avancées de traitement du langage naturel (NLP) pour créer un indice de similarité entre les données textuelles des tickets. Nous aimerions créer un outil pivot pour faciliter la recherche d'informations pertinentes pour les équipes du support technique. Soit un tableau de bord interactif pour que les utilisateurs puissent consulter efficacement la base de données des tickets Assist et ainsi identifier rapidement les solutions aux problèmes rencontrés.

Pour répondre à cet enjeu, il est essentiel de disposer d'une base de données exhaustive des tickets, comprenant la questions et la réponse associées à chaque interaction. Des informations spécifiques telles que les détails techniques des machines concernées, les pièces demandées, ou le service responsable du ticket nous servira de fondement à l'application de filtres avancés sur le tableau de bord.

Notre travail se concentrera autour de plusieurs questions de recherche clés, notamment :

- Comment traiter et interpréter des données textuelles complexes pour qu'elles soient intelligibles par des modèles informatiques ?
- De quelle manière peut-on identifier les correspondances les plus pertinentes entre les demandes actuelles et les tickets passés ?
- Comment adapter un modèle de Large Language Model (LLM) déjà existant pour répondre aux besoins des équipes ?
- Pourrions-nous faciliter les réponses des équipes à l'aide d'une IA générative ?

Cette recherche s'inscrit dans un contexte où l'efficacité de la gestion des tickets est directement liée à la qualité du service offert par le CLPR. En développant un outil capable de réduire significativement le temps nécessaire à la recherche d'informations, ce projet entreprend la transformation des opérations quotidiennes en apportant une réponse concrète aux défis posés par le volume et la complexité des tickets traités. Nous souhaiterions ainsi améliorer des performances opérationnelles du CLPR mais aussi développer les connaissances dans le domaine du traitement automatique du langage naturel appliqué aux environnements professionnels.

2 Tokenization, embeddings et similarité

Les données textuelles dans leur forme brute, ne sont pas directement exploitables par les modèles de machine learning. Pour permettre la classification de textes, la génération de résumé, ou

encore l'analyse de similarité entre deux documents, il est essentiel de prétraiter ces données afin de les convertir en vecteurs numériques. Nous décrirons ici les principales étapes de ce prétraitement. La première étape consiste en la normalisation des données, cette étape est cruciale car elle permet de nettoyer les données. Comment pouvons nous obtenir un texte sous une forme canonique ? Vient ensuite la segmentation (Tokenization) qui consiste à structurer les données pour créer un dictionnaire de valeurs textuelles. Cette étape est particulièrement complexe et peut varier selon les méthodes employées. Nous examinerons certains aspects sans rentrer explicitement dans les détails de toutes les méthodes existantes. La dernière partie consiste en la vectorisation, l'encodage des données (embedding) pour obtenir une représentation vectorielle de chaque Token. Nous finirons par présenter la similarité cosinus qui est une métrique permettant d'identifier la proximité entre deux vecteurs.

2.1 Normalisation

La normalisation des données joue un rôle essentiel dans la réduction de la taille du vocabulaire. La conversion de tout le texte en minuscules, par exemple, permet d'unifier les variantes d'un même mot (comme "Hello" et "hello"). Cette opération peut cependant masquer certains sens implicites ou nuances. Prenons l'exemple de : "I am waiting for my parts URGENTLY". Le passage en minuscules dilue le sentiment d'urgence et l'impatience du message envoyé par notre client. Par conséquent, une connaissance métier des données est importante pour la normalisation des données textuelles.

Lors de la normalisation, il peut être judicieux d'éliminer les URL, les adresses e-mail, et les caractères non reconnus. Il convient également de supprimer les sauts de ligne et les espaces en trop. Nous pouvons aussi gérer les répétitions et effets de style ("coooool" ou "Hiiiii" ou encore "$alut"). Il est crucial de trouver un juste équilibre entre une normalisation excessive, qui pourrait occulter certains sens et une normalisation insuffisante qui serait susceptible d'augmenter inutilement la taille du dictionnaire de référence.

2.2 Dictionnaire ou tokenization

L'étape de tokenisation que nous avons explorée en début de recherche, s'est révélée bien plus variée en termes de méthodologies que nous ne l'avions imaginée. La tokenisation consiste à créer un vocabulaire de référence pour notre corpus en associant à chaque élément textuel (mot, lettre, séquence de caractères, ...) un identifiant numérique. Par exemple : {"hello" :243, "engine" :412, "mlt" :515, ..., " ?" :715}

2.2.1 Tokenisation basée sur les espaces et la ponctuation

Une méthode élémentaire de tokenisation repose sur l'utilisation des espaces et des caractères de ponctuation pour segmenter le texte en tokens. Cette technique présente des limites en revanche,

notamment son incapacité à reconnaître correctement des éléments comme "don't" ou "U.S.A", ces chaînes de caractères spécifiques ne pouvant pas être capturées par cette approche trop simpliste.

2.2.2 Tokenisation en sous-mots

La tokenisation en sous-mots attribue un identifiant numérique à chaque mot fréquent tout en décomposant les mots les plus rares en segments plus petits. Par exemple, le mot "hugely" serait découpé en deux tokens "huge" et le sufixe "#ly". Cette méthode est inspirée de la théorie de l'information et nous rappelle le codage de Huffman étudié dans le cadre de nos cours. Un exemple notable de cette approche est le Byte Pair Encoding (BPE) [3]. Voici une explication de l'algorithme :

Algorithm 1 : Byte Pair Encoding (BPE)

Initialisation : Tous les caractères Unicode sont identifiés et listés comme tokens initiaux du vocabulaire.
Itération : Identification de la paire de tokens (bigramme) la plus fréquente dans le corpus. Ajouter la fusion des deux tokens pour créer un nouveau token et l'ajouter au dictionnaire.
Finalisation : Le processus se termine lorsque la taille du vocabulaire atteint sa limite prédéfinie, incluant une diversité de mots entiers et de sous-mots.

Enfin, il faut distinguer l'utilisation des tokens représentant des entités autonomes et les tokens ayant un rôle en tant que composant de mots complexes. Le token "cat" ne sera potentiellement pas le même que le token "#cat" qui compose le mot "mecatronica". L'ajout de suffixes permet d'améliorer la finesse du vocabulaire et facilite également la compréhension des nuances linguistiques présentes dans le corpus. Cette approche raffine la représentation vectorielle des mots, permettant aux modèles de machine learning de mieux saisir les subtilités contextuelles et sémantiques du texte traité.

Pour finir, nous aimerions présenter une des méthodes de tokenization les plus populaires : "Wordpiece" [2]. Ce tokenizer est devenu célèbre par son utilisation dans le modèle Bert de Google. L'approche de Wordpiece reste relativement semblable à celle de BPE, voici une explication de l'algorithme :

Algorithm 2 : Wordpiece

Initialisation : Tous les caractères Unicode sont identifiés et listés comme tokens initiaux du vocabulaire.
Itération : Identification de la paire de tokens (bigramme) qui a la plus grande information mutuelle. Ajouter la fusion des deux tokens pour créer un nouveau token et l'ajouter au dictionnaire.
Finalisation : Le processus se termine lorsque la taille du vocabulaire atteint sa limite prédéfinie, incluant une diversité de mots entiers et de sous-mots.

Voici quelques éléments possibles pour expliquer la partie itérative de la méthode Wordpiece, le code de cet algorithme n'étant pas opensource, il s'agit ici d'une interprétation proposé sur le

site d'Huggingface [7]. Notons $\mathcal{C}$ un corpus de référence connu par le modèle pour tester les scores d'information mutuelle. Soit $\mathcal{D}_n$ le dictionnaire à l'étape n et $T := \{T_1, T_2, \ldots, T_n\}$ l'ensemble des tokens associés. Pour chaque étape n, il est possible de définir la fréquence P_k^n du token k dans le corpus $\mathcal{C}$. La formule de l'information mutuelle entre deux tokens consécutifs T_k, T_l dans le corpus $\mathcal{C}$ au temps n est donnée par :

$$I_n(T_k, T_l) = \ln(P_{kl}^n) - \ln(P_k^n) - \ln(P_l^n) = \ln\left(\frac{P_{kl}^n}{P_k^n \times P_l^n}\right)$$

Enfin, pour trouver le token au temps $n+1$, on maximise sur tous les couples de tokens consécutifs possibles à l'étape n en lien avec le corpus $\mathcal{C}$. Par croissance de la fonction logarithme, cela revient à trouver le couple "T_kT_l" avec le meilleur score comme ci dessous :

$$T_{n+1} = \underset{"T_kT_l"}{Argmax}\left(\frac{P_{kl}^n}{P_k^n \times P_l^n}\right)$$

Comme la méthode BPE la méthode Wordpiece est itérative, elle incrémente d'un élément le vocalulaire à chaque étape. Enfin pour un dictionnaire donné, il est possible de tokenizer une phrase comme sur l'exemple suivant :

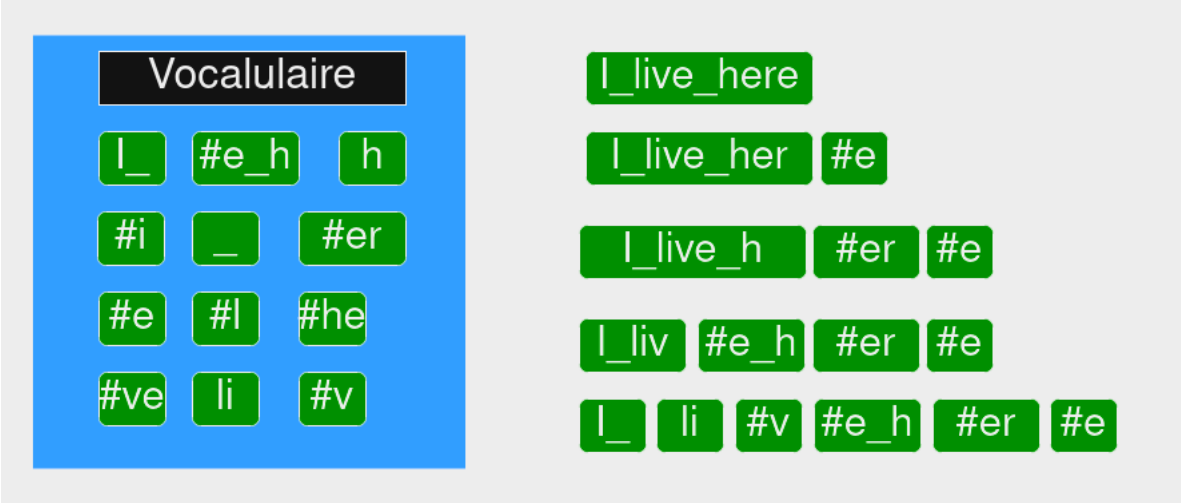


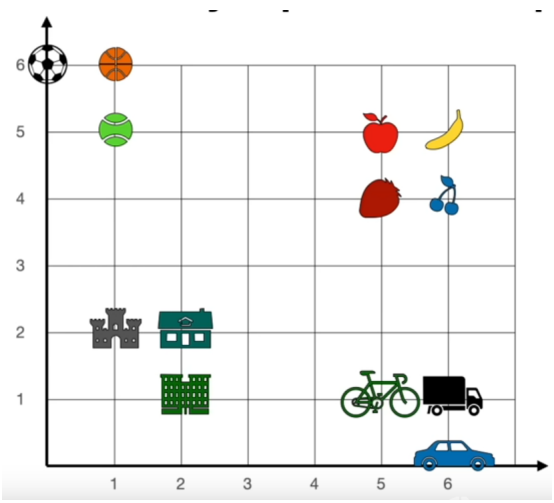
FIGURE 1 – Étapes pour la tokenization

2.3 Embedding pour les tokens

Une fois le corpus segmenté en tokens nous obtenons une suite d'identifiants. En revanche, nous n'avons intégré aucun sens sémantique pour le moment, l'objet est de convertir chaque token en une donnée vectorielle qui intégrerait le sens sémantique des mots. Il existe plusieurs méthodes pour obtenir une vectorisation de chaque token, nous présenterons dans cette partie les algorithmes du type "Word2Vec" afin d'avoir une idée globale du fonctionnement de l'étape d'embedding. Cependant, avant de donner une approche profonde sur la méthodologie, nous proposons une expérience

fictive pour donner l'intuition sur la vectorisation des tokens [5]. Cette expérience fictive, inspirée des neurosciences proposerait de placer des électrodes sur le crâne de plusieurs patients afin de mesurer l'activité des différentes zones du cerveau suite à une stimulation auditive. À l'écoute de certains mots, nous pourrions mesurer l'activité cérébrale moyenne en chacune des zones du cerveau. Les réponses électriques issues de ces stimulations auditives pourraient nous renvoyer des vecteurs dans un espace vectoriel multidimensionnelle. Une représentation 2d via une projection, pourrait nous donner une visualisation des mots avec leur sens sémantique.

Word	Numbers	
Apple	5	5
Banana	6	5
Strawberry	5	4
Cherry	6	4
Soccer	0	6
Basketball	1	6
Tennis	1	5
Castle	1	2
House	2	2
Building	2	1
Bicycle	5	1
Truck	6	1



2.3.1 Word2vec

Les méthodes de type Word2Vec [6] reposent sur le principe suivant : les mots apparaissant régulièrement à proximité les uns des autres dans un corpus partagent un sens sémantique similaire. Ainsi, ils doivent être représentés par des vecteurs proches les uns des autres selon une certaine métrique (ex : la similarité cosinus). Pour obtenir des représentations vectorielles pour chaque token, nous allons utiliser les poids de la première couche d'un réseau de neurones de type perceptron multicouche entraîné sur des tâches de classification. Le réseaux de neurones est entraîné à prédire une ou plusieurs classes de mots. Nous pouvons utiliser la fonction de perte d'entropie croisée pour ajuster les poids du modèle via la rétropropagation du gradient. Les embeddings obtenus sont généralement de quelques centaines de dimensions et le réseau de neurones entraîné comporte souvent deux à trois couches. Dans le cadre de Word2Vec, les réseaux de neurones utilisés pour l'entraînement des poids sont de deux types :

— Bag of Words : Pour une fenêtre donnée (ex : 9 mots), on construit un modèle capable de prédire le token central (ex : 5eme mot).

— Skip-gram : Pour mot en entrée et une fenêtre de prédiction donné (ex : 4 mots), on construit un modèle capable de prédire les mots environnants du mot initialement donné.

Cette technique d'apprentissage supervisé utilise le corpus de référence pour ajuster les poids à chaque étape. Le schéma ci-dessous représente un réseau de neurone MLP simple de type Skip-Gram.

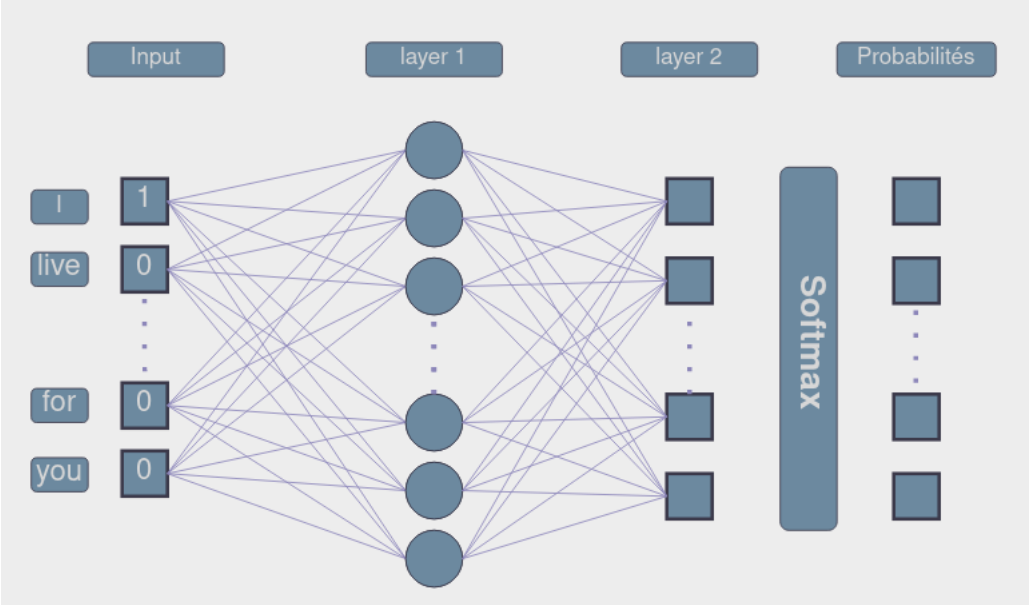


FIGURE 2 – Word2vec du type Skip-Gram

En entrée du modèle, nous utilisons une représentation one hot encoding des mots. En sortie du modèle, nous obtenons des probabilités via l'utilisation de la fonction Softmax. Pour une dictionnaire de taille N et $p \geq 100$ neurones dans la couche 1, notons $W_1 \in \mathcal{M}_{pN}$ la matrice des poids associés à la couche 1. Nous retrouvons après entraînement du modèle l'embeddind de chaque mot via le produit matriciel : $W_1 V_k$ (V_k est la représentation one hot encoding du mot k).

Pour l'entraînement du modèle de type Skip Gram ci dessus, sur un corpus de référence (ex : $(W_i)_i$ une suite de mots) avec un Batch Size $T \geq 1$ et une fenêtre de référence $c \geq 2$, la fonction de coût à minimiser avec l'ajustement des poids est la suivante :

$$\mathcal{L}oss = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \ln(p(w_{t+j}|w_t))$$

Avec les mots w_{t+j}, w_t associés respectivement aux tokens T_k, T_l on a :

$$p(w_{t+j}|w_t) = p(T_k|T_l) = \frac{\exp(Output\_layer2(V_l)_k)}{\sum_{i=1}^{N} \exp(Output\_layer2(V_l)_i)}$$

Remarque 1. *Les méthodes Word2vec ont perdu en popularité depuis l'arrivée des réseaux de neurone avec des mécanismes d'attention type Transformers. Ces nouveaux modèles récupèrent mieux le sens sémantique des mots, ainsi mous privilégierons nos recherches sur ce type de technologie.*

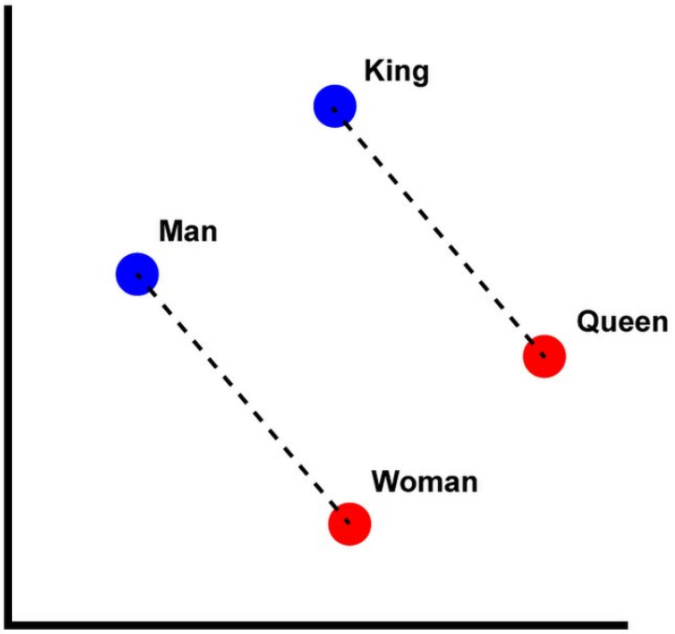


FIGURE 3 – Obtenir un concept de royauté avec Word2vec : King - Man + Woman = Queen

2.4 Embedding de document basé sur une approche fréquentielle : la méthode TF-IDF

La méthode TF-IDF, acronyme de Term Frequency-Inverse Document Frequency, constitue une technique fondamentale en recherche d'information appliquée aux données textuelles. Utilisée principalement par les moteurs de recherche, cette méthode évalue la pertinence d'un document en fonction de l'occurrence d'un terme, d'une expression, ou d'un ensemble de termes. Initialement conçue pour améliorer la pertinence des résultats de recherche internet en lien avec une requête spécifique, nous adopterons dans un premier temps cette méthode afin de classer les questions antérieures au regard des nouvelles problématiques de nos clients.

Le principe de la méthode TF-IDF repose sur une mesure statistique attribuant à chaque terme d'un document (ou "token") un poids donné par le produit de sa fréquence dans le document donné et le logarithme de son importance inverse au sein du corpus d'apprentissage. Ainsi, pour chaque question antérieure, l'importance d'un terme est calculée selon sa fréquence dans cette question et sa distribution dans l'ensemble du corpus initial. Le corpus de référence noté $D := \{d_1, d_2, \ldots, d_N\}$ englobe tous des documents, soit l'ensemble des questions antérieures. Le dictionnaire de référence inclut tous les mots ou expressions utilisés dans le corpus, celui-ci correspond à l'ensemble des Tokens que nous noterons $T := \{T_1, T_2, \ldots, T_K\}$.

Cette approche nous permet de représenter un document sous la forme d'un vecteur dans un

espace de grande dimension. Avec nos notations un document est assimilé à un vecteur de $\mathbb{R}^K$. Dans ce cadre, l'information contenue dans chaque interrogation de client (un document) est convertie en vecteur, ceci en mettant l'accent sur les termes et leur pertinence, indépendamment de leur position dans le texte. Par la suite, notre étude s'attachera également à explorer d'autres méthodes de vectorisation de documents plus sophistiquées.

Avant de détailler le calcul des coefficients de la matrice des poids de référence, il convient de mentionner une étude de 2015 [4] qui estimait qu'un grand nombre des systèmes de recommandation basés sur le texte s'appuyaient sur la méthode TF-IDF. Cette méthode offre également des perspectives intéressantes pour la synthèse de textes. Cela se fait par la sélection des phrases les plus significatives d'un corpus, cette application spécifique ne sera pas abordée dans le cadre de notre projet mais cet article [1] peut vous donner plus de renseignements.

Nous abordons maintenant l'explication détaillée de la méthode TF-IDF, en mettant l'accent sur le calcul des poids attribués à chaque terme (ou "token") dans un document. La méthode se divise en deux composantes principales : la fréquence du terme (TF) et la fréquence inverse du document (IDF). Nous noterons dans la suite $P \in \mathcal{M}_{N,K}(\mathbb{R})$ la matrice des poids.

Soient $d_i \in D$ et $T_j \in T$, on a $P_{i,j} = TF_{i,j} \times IDF_j$

$$P = \begin{pmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,K} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1} & P_{N,2} & \cdots & P_{N,K} \end{pmatrix}$$

1. Fréquence du Terme (TF) :

   Cette composante mesure la fréquence d'un terme T_j dans un document donné. La fréquence est calculée selon la formule suivante :

   $$TF_{i,j} = \frac{\#\{t \in d_i \mid t = T_j\}}{\sum_{k=1}^{K} \#\{t \in d_i \mid t = T_k\}}$$

   Cette formule permet d'attribuer un poids plus élevé aux termes qui apparaissent fréquemment dans un document, reflétant ainsi leur importance relative dans ce contexte spécifique.

2. Fréquence Inverse du Document (IDF) :

   Cette composante vise à diminuer le poids des termes qui apparaissent trop fréquemment

dans l'ensemble du corpus, elle est calculée par la formule suivante :

$$IDF_j = \ln\left(\frac{N}{\sum_{i=1}^{N} \mathbb{1}_{\{T_j \in d_i\}}}\right)$$

Ainsi, un terme rare dans l'ensemble du corpus, mais fréquent dans certains documents, se verra attribuer un poids plus élevé, tandis que les termes communs à de nombreux documents auront un poids réduit.

Remarque 2. *Le terme* $\frac{N}{\sum_{i=1}^{N} \mathbb{1}_{\{T_j \in d_i\}}}$ *est compris entre* 1 *et* N. *Donc* $0 \leq IDF_j \leq \ln(N)$.

Si le token est présent dans chaque document on a : $IDF_j = 0$.
Si le token est présent dans un unique document on a : $IDF_j = \ln(N)$

L'efficacité de la méthode TF-IDF repose sur son aptitude à équilibrer l'importance des termes en fonction de leur fréquence dans un document particulier et leur unicité dans l'ensemble du corpus. Les termes rares au niveau du corpus mais fréquents dans un document spécifique reçoivent un poids élevé, ce qui permet de valoriser leur pertinence spécifique. À l'inverse, les mots très communs, tels que les prépositions ("a", "the", "I", ...) ou les formules de politesse ("hello", "hi",...) n'apportent pas d'information significative sur le contenu du document, ils sont pénalisés par un poids réduit. Pour chaque terme dans un document, le poids TF-IDF est calculé comme le produit de TF et IDF. Cette opération donne un poids qui caractérise l'importance relative du terme dans le contexte du document par rapport à l'ensemble du corpus. En fin de compte, nous obtenons une matrice de poids où les lignes représentent les documents du corpus initial et les colonnes les tokens.

2.4.1 Indice de similarité entre deux documents

Nous avons vu plusieurs manières pour convertir des données textuelles en vecteur. L'objet de cette partie est de revenir sur la notion de proximité sémantique entre les mots ou encore les documents. Quelle métrique pourrions nous employer afin d'obtenir un indice de similarité entre des mots, des documents ? Nous profiterons des propriétés géométriques des espaces euclidiens pour présenter la similarité cosinus.

2.4.2 Lien avec la théorie de l'information

2.4.3 limites de la méthode

3 Deep learning et mécanisme d'attention

3.1 Les modèles Transformers : "Attention is all you need"

3.2 Le modèle Bert

4 Conclusion

Références

[1] Intelligence artificielle : Résumer un texte grâce au tf idf. `https://datascientest.com/tf-idf-intelligence-artificielle`. Accédé le [08-02-2024].

[2] Japanese and korean voice search. `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37842.pdf`. Accédé le [14-03-2024].

[3] Neural machine translation of rare words with subword units. `https://aclanthology.org/P16-1162.pdf`. Accédé le [14-03-2024].

[4] Research-paper recommender systems : a literature survey. `https://kops.uni-konstanz.de/entities/publication/861ddd16-fbc6-4ee4-a77f-6bba081041f3`. Accédé le [08-02-2024].

[5] Video (min 18) : What are transformer models and how do they work ? `https://www.youtube.com/watch?v=qaWMOYf4ri8`. Accédé le [14-03-2024].

[6] Word2vec. `https://en.wikipedia.org/wiki/Word2vec`. Accédé le [15-03-2024].

[7] Wordpiece tokenization. `https://huggingface.co/learn/nlp-course/en/chapter6/6`. Accédé le [15-03-2024].