# 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). Nous aimerions étudier la similarité entre les données textuelles des tickets issus de leur logiciel Assist. 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 à un environnement professionnel.

# 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 dans cette première partie les principales étapes de ce prétraitement. Commençons par 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 via un dictionnaire python : {"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 groupes de mots 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 étudiée dans le cadre de nos cours, l'idée étant d'utiliser un découpage à longueur variable pour représenter un mot ou une expression. Un exemple notable de cette approche est le Byte Pair Encoding (BPE) [6]. 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" [5]. 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 :

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 [11]. Notons $\mathcal{C}$ le corpus de référence connu par le modèle pour tester les scores d'information mutuelle et ainsi constituer le dictionnaire final. Notons $\mathcal{D}_n = \{T_1, T_2, \ldots, T_n\}$

**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.

le dictionnaire contenant l'ensemble des $n$ caractères unicode constituant $\mathcal{C}$ listés comme tokens initiaux. Pour chaque étape $i \geq 0$ et $k \in [\![1, \ldots, n+i]\!]$, il est possible de définir la fréquence $P_k$ du token $T_k$ dans le corpus $\mathcal{C}$. Pour $k, l \in [\![1, \ldots, n+i]\!]$, nous noterons $P_{kl}$ la fréquence dans le corpus $\mathcal{C}$ du bigramme $T_k T_l$. 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 :

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

Enfin, pour trouver le token à ajouter pour constituer $\mathcal{D}_{n+i+1}$, 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, il suffit de trouver le couple "$T_k T_l$" avec le meilleur score comme ci dessous :

$$T_{n+i+1} = \underset{\text{"}T_k T_l\text{"}}{Argmax}\left(\frac{P_{kl}}{P_k \times P_l}\right)$$

**Remarque 1.** *Comme pour 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 donner la tokenisation d'une phrase comme sur l'exemple suivant :*

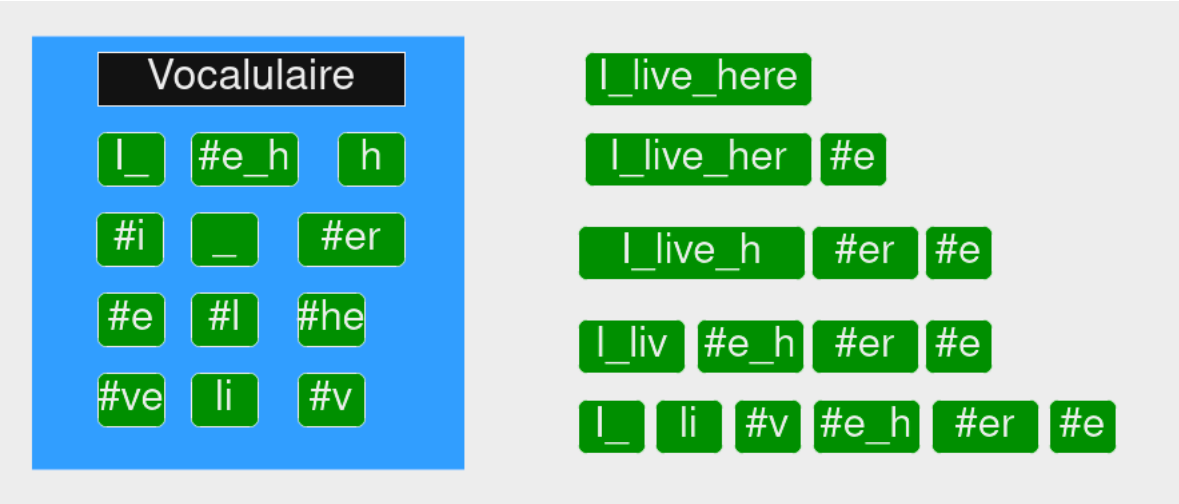


FIGURE 1 – Étapes pour la tokenization

**Remarque 2.** *Quand on demande à ChatGPT de nous donner le tokenizer qu'il utilise, il nous répond :*

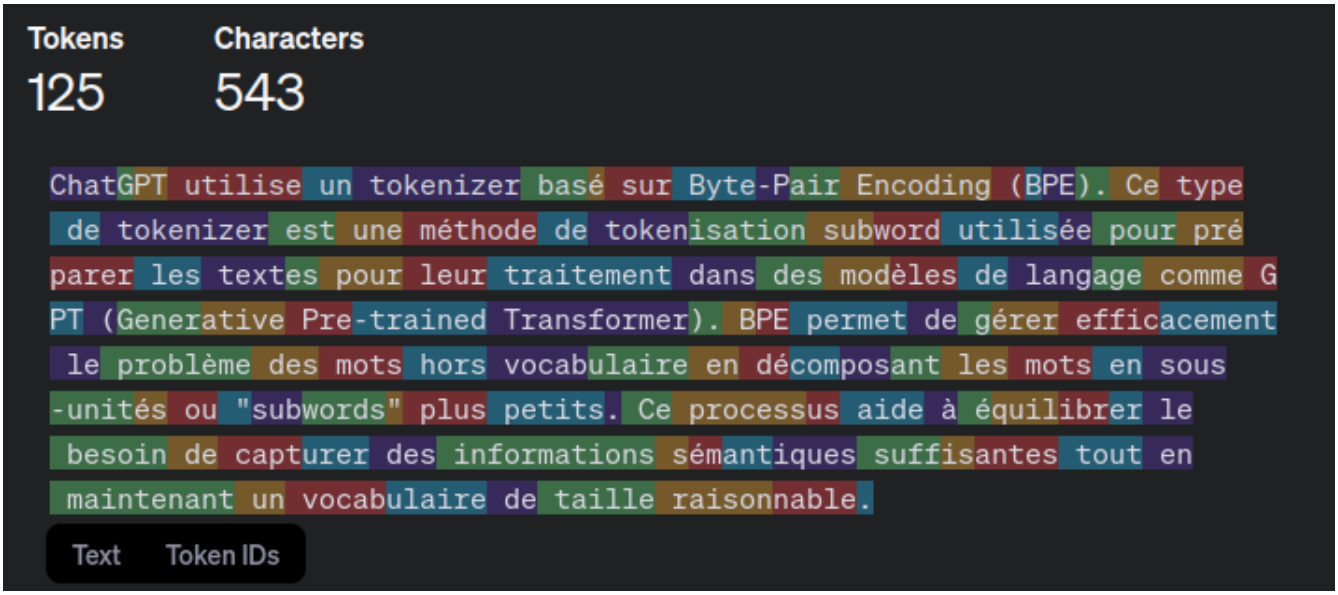


FIGURE 2 – Tokenization de la réponse de ChatGPT [7]

## 2.3 Embedding pour les tokens

Maintenant nous savons comment segmenté un corpus en une liste de tokens, 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 de nombreux mécanismes pour obtenir une vectorisation de chaque token, nous présenterons dans cette partie les algorithmes du type "Word2Vec" afin d'avoir une idée plus intuitive 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 [9]. 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 |

FIGURE 3 – Tableau des embeddings en 2d

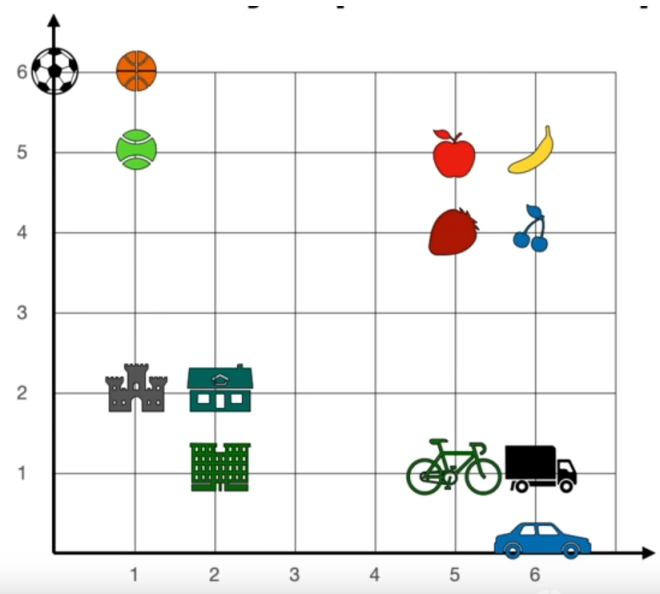


FIGURE 4 – Projection 2d des embeddings

### 2.3.1 Word2vec

Les méthodes de type Word2Vec [10] 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 la matrice des poids $W_1$ de la première couche d'un réseau de neurones de type MLP entraîné sur des tâches de classification. Le réseaux de neurones est entraîné à prédire un ou plusieurs mots selon la tâche de classification. La fonction de perte d'entropie croisée sert à évaluer les prédictions à travers un coût et permet d'ajuster les poids du modèle via la rétro-propagation 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 des modèles Word2Vec, les tâches d'entraînement pour le réseaux de neurones 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 un mot en entrée et une fenêtre de prédiction donné (ex : 4 mots), on construit un modèle capable de prédire les mots voisins.

**Remarque 3.** *Ces techniques d'apprentissage supervisé utilisent 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. Via une entrée vectorielle en one-hot encodding, le modèle prédit des probabilités de voisinages.*

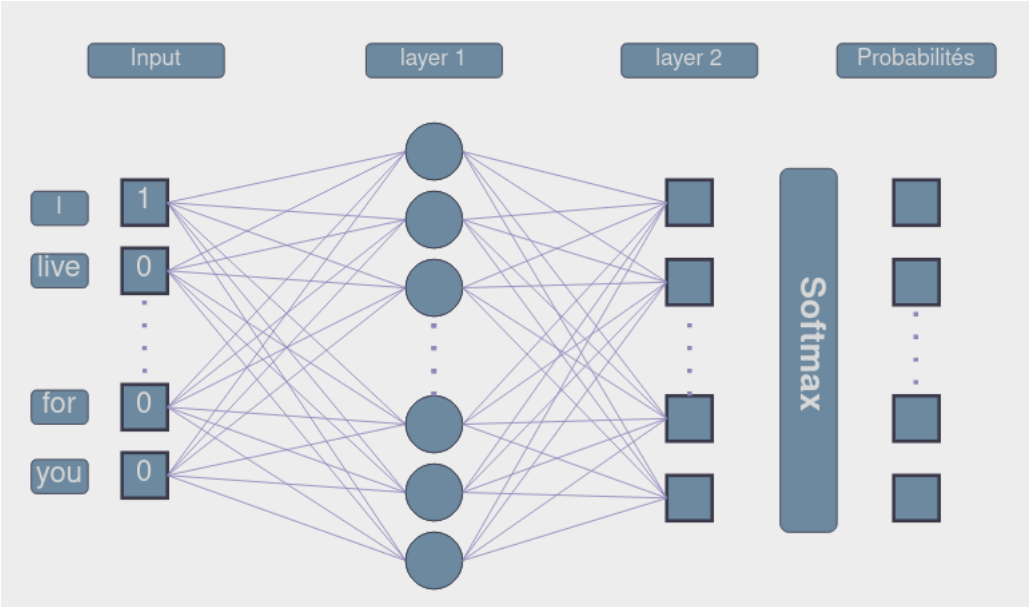


FIGURE 5 – Word2vec du type Skip-Gram

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 première couche du réseau, notons $W_1 \in \mathcal{M}_{N,p}(\mathbb{R})$ la matrice des poids associés à la couche 1. Après entraînement du modèle, l'embeddind de chaque mot se retrouve via le produit matriciel : $V_k W_1 \in \mathcal{M}_{1,p}(\mathbb{R})$ ($V_k \in \mathcal{M}_{1,N}(\mathbb{R})$ 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 par ajustement itératif des poids $(W_1, W_2, \ldots, W_n)$ est la suivante :

$$\mathcal{L}oss(W_1, W_2, \ldots, W_n) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \ln(p(w_{t+j}|w_t))$$

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

$$p(w_{t+j}|w_t) = p(T_k|T_l) = \frac{\exp\left(Layer_n(Layer_{n-1}(\ldots Layer_1(V_l)))_k\right)}{\sum_{i=1}^{N} \exp\left(Layer_n(Layer_{n-1}(\ldots Layer_1(V_l)))_i\right)}$$

**Remarque 4.** *Les méthodes Word2vec ont perdu en popularité depuis l'arrivée des réseaux de neurones type Transformers et l'emploi de tables d'embeddings qui se forgent au moment de l'entraînement du modèle. Les tables d'embeddings sont ainsi des poids aléatoirement initialisés, ils évoluent ensuite au cours de la phase d'apprentissage. Ces nouveaux modèles récupèrent mieux le sens sémantique des mots aux travers des embeddings construits.*

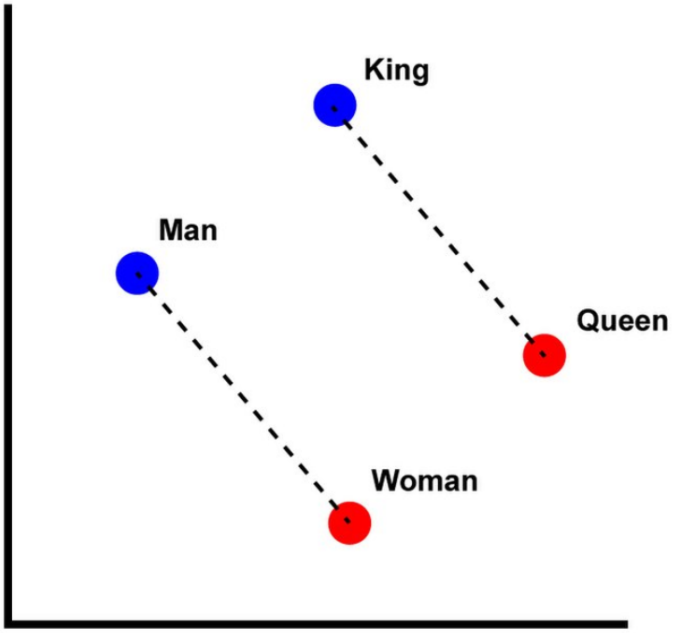


FIGURE 6 – Obtenir un concept de royauté avec les embeddings issus des modèles type Word2vec : King + Woman - Man = Queen [2]

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

Le premier objectif de ce projet serait de construire un moteur de recherche basé sur les questions clients. Nous aimerions être en mesure de comparer chaque nouvelle question client avec les questions clients du passé. Pour cela il faut trouver une méthodologie permettant de plonger l'information d'une phrase entière dans un $\mathbb{R}$ espace vectoriel. 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 ces dernières années, 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 $Q := \{q_1, q_2, \ldots, q_N\}$ englobe toutes les questions passées. Le choix d'un tokenizer nous donnera pour l'ensemble des questions $Q$ un dictionnaire de référence, celui-ci inclut tous les mots ou expressions utilisés dans le corpus, il correspond à l'ensemble des Tokens que nous noterons $T := \{T_1, T_2, \ldots, T_K\}$.

Avec nos notations un document sera assimilé à un vecteur de $\mathbb{R}^K$. Dans ce cadre, l'information contenue dans chaque interrogation 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 [8] 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 [4] peut vous donner plus de renseignements.

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 log-fréquence inverse du document (IDF). Nous noterons dans la suite $P \in \mathcal{M}_{N,K}(\mathbb{R})$ la matrice des poids.

Soient $q_i \in Q$ 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 $i$ donné. La fréquence est calculée selon la formule suivante :

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

La fréquence d'un token dans la question permet d'attribuer un poids plus élevé aux tokens qui apparaissent fréquemment dans celle-ci, reflétant ainsi leur importance relative dans ce contexte spécifique.

**Remarque 5.** *L'appartenance d'un token $t$ à une question $q$ sous entend une tokenisation de la question en amont. Pour simplifier la lecture nous avons fait le choix de noter : $"t \in q"$ ci-dessus au lieux de : $"t \in Tokenisation(q)"$.*

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 :

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

Ainsi, un terme rare dans l'ensemble du corpus aura un poids IDF plus élevé, tandis que les termes communs à de nombreux documents auront un poids IDF réduit.

**Remarque 6.** *Le terme $\frac{N}{\sum_{i=1}^{N} \mathbb{1}_{\{T_j \in q_i\}}}$ 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 Lien avec la théorie de l'information

La méthode TF-IDF est souvent présentée comme une méthode empirique avec de nombreuses variantes dans sa mise en pratique. En s'inspirant de l'article *"An information-theoretic perspective of Tf-Idf"* [3] et par nos connaissances de la théorie de l'information, nous allons proposer une justification mathématiques pour le calcul des poids vu précédemment. Commençons par définir certains objets mathématiques iconiques de la théorie de l'information :

**Définition 1.**

1. Soit $\mathcal{X}, \mathcal{Y}$ deux ensembles discrets, tels que $\mathcal{X} \times \mathcal{Y}$ soit muni d'une mesure de probabilité marginale $P(x_i, y_j)$ qui nous donne respectivement des mesures de probabilités sur $\mathcal{X}$ et $\mathcal{Y}$ telles que :
$$P(x_i) = \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ , \ P(y_j) = \sum_{x_i \in \mathcal{X}} P(x_i, y_j)$$
2. Pour tout élément $x_i \in \mathcal{X}$, la *quantité d'information* est définie par : $\ln\left(\frac{1}{P(x_i)}\right) = -\ln\left(P(x_i)\right)$.
3. Soit $X, Y$ deux variables aléatoires définies sur un espace probabilisé $(\Omega, \mathscr{A}, \mathbb{P})$ à valeur respectivement dans $\mathcal{X}$ et dans $\mathcal{Y}$, dont la loi du couple $(X, Y)$ s'identifie à la mesure de probabilité jointe sur $\mathcal{X} \times \mathcal{Y}$ vue précédemment :
$$\mathbb{P}(X = x_i, Y = y_j) = P(x_i, y_j) \ , \ \mathbb{P}(X = x_i) = P(x_i) \ , \ \mathbb{P}(Y = y_j) = P(y_j)$$
   — On définit l'*entropie* (self-entropie) de la variable aléatoire $X$ par la formule suivante :
$$H(X) = -\sum_{x_i \in \mathcal{X}} \mathbb{P}(X = x_i) \ln(\mathbb{P}(X = x_i)) = -\sum_{x_i \in \mathcal{X}} P(x_i) \ln(P(x_i))$$
   — On définit l'*entropie croisée* (cross-entropie) du couple de variables aléatoires $(X, Y)$ par la formule suivante :
$$H(X, Y) = -\sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln(P(x_i, y_j))$$
   — On définit l'*entropie conditionnelle* de la variable $X$ sachant la variable $Y$ par la formule suivante :
$$H(X|Y) = -\sum_{y_j \in \mathcal{Y}} P(y_j) \sum_{x_i \in \mathcal{X}} P(x_i|y_j) \ln(P(x_i|y_j))$$

**Remarque 7.** *L'entropie reflète le degré d'incertitude (le désordre) d'une variable aléatoire face à ses réalisations possibles. Pour une variable aléatoire constante, l'entropie est nulle.*

*Pour une variable aléatoire $Z$ discrète avec $N$ états équidistribués (loi uniforme) on a :*

$$H(Z) = -\sum_{i=1}^{N} \mathbb{P}(Z = z_i) \ln(\mathbb{P}(Z = z_i)) = -\sum_{i=1}^{N} \frac{1}{N} \ln\left(\frac{1}{N}\right) = \ln(N)$$

4. L'*information mutuelle mutuelle entre deux états* $x_i, y_j$ est donnée par la formule :

$$\ln\left(\frac{P(x_i, y_j)}{P(x_i)P(y_j)}\right)$$

**Remarque 8.** *En cas d'indépendance des évènements* $\{X = x_i\}$ *et* $\{Y = y_j\}$*, l'information mutuelle entre les deux états est nulle.*

5. L'*information mutuelle mutuelle entre deux variables aléatoires* $X$ et $Y$ est donnée par la formule :

$$\begin{aligned}
I(X,Y) &= \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(\frac{P(x_i, y_j)}{P(x_i)P(y_j)}\right) \\
&= \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(P(x_i, y_j)\right) - \sum_{x_i \in \mathcal{X}} \ln\left(P(x_i)\right) \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) - \sum_{y_j \in \mathcal{Y}} \ln\left(P(y_j)\right) \sum_{x_i \in \mathcal{X}} P(x_i, y_j) \\
&= -\sum_{x_i \in \mathcal{X}} \ln\left(P(x_i)\right) P(x_i) - \sum_{y_j \in \mathcal{Y}} \ln\left(P(y_j)\right) P(y_j) + \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(P(x_i, y_j)\right) \\
&= H(X) + H(Y) - H(X,Y)
\end{aligned}$$

or

$$\begin{aligned}
H(X) - H(X,Y) &= -\sum_{x_i \in \mathcal{X}} P(x_i) \ln\left(P(x_i)\right) + \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(P(x_i, y_j)\right) \\
&= -\sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(P(x_i)\right) + \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(P(x_i, y_j)\right) \\
&= \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(x_i, y_j) \ln\left(\frac{P(x_i, y_j)}{P(x_i)}\right) = \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} P(y_j|x_i) P(x_i) \ln\left(\frac{P(y_j|x_i)P(x_i)}{P(x_i)}\right) \\
&= \sum_{x_i \in \mathcal{X}} P(x_i) \sum_{y_j \in \mathcal{Y}} P(y_j|x_i) \ln\left(P(y_j|x_i)\right) \\
&= -\sum_{x_i \in \mathcal{X}} P(x_i) H(Y|X = x_i) \\
&= -H(Y|X)
\end{aligned}$$

de même on trouver que

$$H(Y) - H(X,Y) = -H(X|Y)$$

Enfin

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

**Remarque 9.** *L'information mutuelle entre deux variables aléatoires est une valeur symétrique :* $I(X,Y) = I(Y,X)$

Cherchons maintenant à trouver un lien entre le calcul des poids dans la méthode TF-IDF et les concepts de la théorie de l'information vus précédemment. Nous reprenons dans cette partie la démonstration proposée dans l'article *"An information-theoretic perspective of Tf-Idf"* [3]. Soit $\mathcal{Q} = \{q_1, \ldots, q_N\}$ un ensemble de questions client et $\mathcal{T} = \{t_1, \ldots, t_K\}$ le dictionnaire de référence reprenant tous les mots présents dans l'ensemble des questions $\mathcal{Q}$. Soit $D$ et $T$ deux variables aléatoires à valeurs dans $\mathcal{Q}$ et dans $\mathcal{T}$, modélisant le tirage respectivement d'une question et d'un mot selon les lois empiriques suivantes :

$$P(D = q_i) = \frac{1}{N} \ , \ P(T = t_j) = \frac{1}{N} \sum_{i=1}^{N} f_{i,j}$$

On rappelle que $f_{i,j}$ est la fréquence du mot $j$ dans la question $i$ donnée par la formule de coefficient $TF_{i,j}$.

Considérons l'information mutuelle entre $D$ et $T$, on trouve ainsi :

$$I(D,T) = H(D) + H(T) - H(D,T) = H(D) - H(D|T)$$

Or :

$$H(D) = -\sum_{i=1}^{N} P(D = q_i) \ln(P(D = q_i)) = -\sum_{i=1}^{N} \frac{1}{N} \ln\left(\frac{1}{N}\right) = \ln(N)$$

Intéressons nous à la quantité :

$$H(D|T) = -\sum_{j=1}^{K} P(T = t_j) H(D|T = t_j)$$

On a :

$$H(D|T = t_j) = \sum_{i=1}^{N} P(D = q_i|T = t_j) \ln(P(D = q_i|T = t_j))$$

Soit $t_j \in \mathcal{T}$, notons $N_j$ le cardinal du sous-ensemble $\mathcal{Q}_j$ des documents contenant le mot $t_j$.

Or :

$$\begin{cases} P(D = q_i|T = t_j) = \dfrac{1}{N_j} & \text{si} \quad q_i \in \mathcal{Q}_j \\ P(D = q_i|T = t_j) = 0 & \text{si} \quad q_i \notin \mathcal{Q}_j \end{cases}$$

Donc :

$$H(D|T=t_j) = \sum_{q_i \in \mathcal{Q}_j} P(D=q_i|T=t_j)\ln(P(D=q_i|T=t_j)) + \sum_{q_i \notin \mathcal{Q}_j} P(D=q_i|T=t_j)\ln(P(D=q_i|T=t_j))$$
$$= \sum_{q_i \in \mathcal{Q}_j} \frac{1}{N_j}\ln\left(\frac{1}{N_j}\right) + 0 = N_j \times \frac{1}{N_j}\ln\left(\frac{1}{N_j}\right) = \ln\left(\frac{1}{N_j}\right)$$

On trouve donc :

$$I(D,T) = H(D) - H(D|T) = \ln(N) + \sum_{j=1}^{K} P(T=t_j)\ln\left(\frac{1}{N_j}\right)$$
$$= \sum_{j=1}^{K} P(T=t_j)\ln(N) + \sum_{j=1}^{K} P(T=t_j)\ln\left(\frac{1}{N_j}\right)$$
$$= \sum_{j=1}^{K} P(T=t_j)\ln\left(\frac{N}{N_j}\right) = \sum_{j=1}^{K} P(T=t_j) \times IDF_j$$
$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} f_{i,j} \times IDF_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} TF_{i,j} \times IDF_j$$

### 2.4.2 Indice de similarité entre deux documents

Nous avons exploré plusieurs méthodes pour convertir des données textuelles en vecteurs. L'objectif de cette partie est de proposer un moyen de comparer la proximité sémantique entre les mots ou les questions. Quelle métrique pourrions-nous employer pour obtenir un indice de similarité entre des mots ou des documents ? Nous exploiterons les propriétés géométriques des espaces euclidiens pour introduire la similarité cosinus.

Dans les parties précédentes de ce rapport, nous avons proposé plusieurs méthodes pour plonger des données textuelles dans un espace vectoriel de grande dimension. Nous avons présenté l'exemple de la méthode TF-IDF, capable de fournir pour chaque question un embedding de la taille du dictionnaire. Ainsi, chaque question est assimilée à un vecteur de $\mathbb{R}^K$, où $K$ est la taille du dictionnaire. Rappelons que $\mathbb{R}^K$ est un espace vectoriel réel de dimension finie et que nous pouvons le munir du produit scalaire usuel $\langle\cdot,\cdot\rangle$, qui est une forme bilinéaire, symétrique et définie positive sur $\mathbb{R}^K \times \mathbb{R}^K$ à valeurs dans $\mathbb{R}$.

**Propriété 1** (Inégalité de Cauchy-Schwarz)**.**
*Pour tous vecteurs $x, y \in \mathbb{R}^K$, l'inégalité suivante est satisfaite :*

$$\langle x, y\rangle \leq |\langle x, y\rangle| \leq \|x\| \cdot \|y\|,$$

*avec égalité si et seulement si $x$ et $y$ sont colinéaires.*

*Démonstration.* Soit $x, y \in \mathbb{R}^K$,

or :

$$\begin{aligned}\|x+\lambda y\|^2 &= \langle x+\lambda y, x+\lambda y\rangle = \langle x,x\rangle + 2\lambda\langle x,y\rangle + \lambda^2\langle y,y\rangle \\ &= \|x\|^2 + 2\lambda\langle x,y\rangle + \lambda^2\|y\|^2\end{aligned}$$

La fonction $\lambda \mapsto \|x\|^2 + 2\lambda\langle x,y\rangle + \lambda^2\|y\|^2$ est polynomiale de degré deux, elle n'est jamais négative car : $\forall \lambda \in \mathbb{R}, \|x+\lambda y\| \geq 0$. On en déduit que sont discriminant $\Delta \leq 0$.

Or : $\Delta = 4\langle x,y\rangle^2 - 4\|x\|^2 \cdot \|y\|^2$ Donc :

$$\begin{aligned} & \Delta \leq 0 \\ \Leftrightarrow \quad & 4\langle x,y\rangle^2 - 4\|x\|^2 \cdot \|y\|^2 \leq 0 \\ \Leftrightarrow \quad & 4\langle x,y\rangle^2 \leq 4\|x\|^2 \cdot \|y\|^2 \\ \overset{(*)}{\Leftrightarrow} \quad & |\langle x,y\rangle| \leq \|x\| \cdot \|y\| \end{aligned}$$

(*) Par croissance de la fonction carrée sur $\mathbb{R}^+$.

$\Rightarrow$ Dans le cas où $x$ et $y$ sont colinéaires, par bilinéarité du produit scalaire nous donne l'égalité.
$\Leftarrow$ Dans le cas de l'égalité $|\langle x,y\rangle| = \|x\| \cdot \|y\|$, on en déduit que $\Delta = 0$. Ainsi il existe un unique $\lambda_0$ tel que $\|x+\lambda_0 y\| = 0$. Par le caractère définie de la norme on a donc : $x + \lambda_0 y = 0$. Donc $x$ et $y$ sont colinéaires. □

**Propriété-définition 1** (Cosine Similarity)**.**
*Soit $x, y \in \mathbb{R}^K \backslash \{0_K\}$, on trouve que :*

$$-1 \leq \frac{\langle x,y\rangle}{\|x\| \cdot \|y\|} \leq 1$$

*La Cosine Similarity entre les deux vecteurs $x$ et $y$ est donnée par la valeur : $\dfrac{\langle x,y\rangle}{\|x\| \cdot \|y\|}$.*

*Par bijectivité de la fonction $\cos : [0,\pi] \to [-1,1]$, il existe un unique $\theta \in [0,\pi]$ tel que :*

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

*Cette propriété nous permet de retrouver une notion de géométrie, $\theta$ est l'angle entre les vecteurs $x$ et $y$ semblable à l'intuition que nous pouvons avoir si nous étions dans le plan.*

Nous utiliserons ce score de ressemblance dans le moteur de recherche que nous allons créer afin d'indexer les questions par ordre de similarité croissante selon la similarité cosinus. Le choix de cette métrique est motivé par son efficacité avec des données creuses.

**Exemple 1** (Cosine Similarity with Python [1])**.**

```python
import numpy as np

u = np.array([1, 1, 0, 0])
v = np.array([1, 0, 0, 0])
w = np.array([2, 1, 0, 0])

def cosine_similarity(x, y):
    # Compute the dot product between x and y
    dot_product = np.dot(x, y)
    # Compute the L2 norms (magnitudes) of x and y
    magnitude_x = np.sqrt(np.sum(x**2))
    magnitude_y = np.sqrt(np.sum(y**2))
    # Compute the cosine similarity
    cosine_similarity = dot_product / (magnitude_x * magnitude_y)
    return cosine_similarity

print(cosine_similarity(u, v))
    #-> 0.707
print(np.arccos(cosine_similarity(u, v)))
    #-> 0.013
print(np.linalg.norm(u-v))
    #-> 1.0

print(cosine_similarity(u, v))
    #-> 0.948
print(np.arccos(cosine_similarity(u, v)))
    #-> 0.005
print(np.linalg.norm(u-v))
    #-> 1.0

print(0.013/0.005)
#-> 2.44
```

En effet, les embeddings de documents contiennent souvent de nombreux zéros, la similarité cosinus mesure la proximité angulaire plutôt que l'amplitude des vecteurs.Considérez l'exemple suivant où les vecteurs $v$ et $w$ sont tous deux à une distance euclidienne de 1 du vecteur $u$. Cependant, il existe un facteur de 2 de différence dans leur proximité angulaire. Dans ce cas, nous dirions que le vecteur $w$ est deux fois plus semblable à $u$ selon le score de similarité cosinus que le vecteur $v$.

### 2.4.3 Limites de la méthode

Dans le cadre de ce projet, nous avons développé une première version de l'application en utilisant les embeddings de documents produits par la méthode TF-IDF pour créer le moteur de recherche. Les résultats étaient satisfaisants et meilleurs qu'avec un simple filtre par mots-clés. Cependant, les faiblesses de cette méthode résident principalement dans la non-prise en compte de l'emplacement des mots dans la phrase et la production d'embeddings dont la dimension est égale à la taille du dictionnaire total.

La méthode TF-IDF traite chaque terme indépendamment, ce qui signifie qu'elle ne capture pas les informations syntaxiques ni sémantiques qui peuvent être dérivées de l'ordre des mots ou de leur contexte dans la phrase. Par exemple, "Manitou's team awaits a response from the dealer" et "The dealer awaits a response from Manitou's team " produiront le même vecteur avec TF-IDF malgré leurs significations différentes.

De plus, la dimensionnalité des embeddings créés par TF-IDF est directement liée à la taille du dictionnaire, ce qui peut entraîner des vecteurs très grands, particulièrement dans le cas de grands corpus de texte. Cela peut non seulement augmenter les besoins en stockage et en calcul mais aussi réduire l'efficacité du moteur de recherche en augmentant le temps nécessaire pour comparer les vecteurs.

Pour surmonter ces limites, des techniques plus avancées comme les modèles basés sur les Transformers peuvent être envisagées. Ces méthodes prennent en compte le contexte des mots et génèrent des embeddings de dimension réduite, ce qui améliore la gestion de la sémantique.

# 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] Cosine similarity. https://www.learndatasci.com/glossary/cosine-similarity/. Accédé le [08-05-2024].

[2] Distributed representations of words and phrases and their compositionality. https://arxiv.org/pdf/1310.4546. Accédé le [7-05-2024].

[3] An information-theoretic perspective of tf–idf measures. https://ccc.inaoep.mx/~villasen/index_archivos/cursoTL/articulos/Aizawa-tf-idfMeasures.pdf. Accédé le [08-05-2024].

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

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

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

[7] Openai tokenizer. https://platform.openai.com/tokenizer. Accédé le [7-05-2024].

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

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

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

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