Université de Strasbourg

Recherche d'information

Modèles épars

Enzo Doyen

2025 - LGC6KM43 - M2

Méthodes de récupération de documents

Plusieurs méthodes peuvent être utilisées pour récupérer des documents pertinents à partir d'une requête formulée en langage naturel :

- **recherche par mots-clés** : correspondance exacte (ou quasi exacte) entre les termes de la requête et ceux présents dans les documents. La pertinence peut être estimée grâce à des **modèles épars** comme TF-IDF ou BM25, qui pondèrent l'importance des mots en fonction de leur fréquence dans le document et dans la collection.
- **recherche sémantique/dense** : elle vise à interpréter le sens de la requête afin d'identifier des documents pertinents, même si les termes employés diffèrent. Cette approche s'appuie sur des **modèles denses** qui exploitent des plongements vectoriels (au niveau des mots ou de la phrase) pour mesurer la similarité sémantique entre la requête et les documents.

Méthodes de récupération de documents

Plusieurs méthodes peuvent être utilisées pour récupérer des documents pertinents à partir d'une requête formulée en langage naturel :

- **recherche par mots-clés** : correspondance exacte (ou quasi exacte) entre les termes de la requête et ceux présents dans les documents. La pertinence peut être estimée grâce à des **modèles épars** comme TF-IDF ou BM25, qui pondèrent l'importance des mots en fonction de leur fréquence dans le document et dans la collection.
- **recherche sémantique/dense** : elle vise à interpréter le sens de la requête afin d'identifier des documents pertinents, même si les termes employés diffèrent. Cette approche s'appuie sur des **modèles denses** qui exploitent des plongements vectoriels (au niveau des mots ou de la phrase) pour mesurer la similarité sémantique entre la requête et les documents.

📖 Plan

I. Modèle TF-IDF

Recherche par mots-clés : TF-IDF

TF-IDF : *term frequency-inverse document frequency* **(Sparck Jones, 1972)**

Méthode de pondération des termes dans un document, qui permet de mesurer l'importance d'un mot dans un document par rapport à un corpus.

Repose sur deux mesures : TF (*term frequency*) et IDF (*inverse document frequency*).

Recherche par mots-clés : TF-IDF

TF-IDF : *term frequency-inverse document frequency* **(Sparck Jones, 1972)**

Méthode de pondération des termes dans un document, qui permet de mesurer l'importance d'un mot dans un document par rapport à un corpus.

Repose sur deux mesures : TF (*term frequency*) et IDF (*inverse document frequency*).

TF correspond au nombre d'occurrences d'un mot dans **un seul document**, tandis que **IDF** mesure l'importance d'un mot dans l'**ensemble du corpus**.

Recherche par mots-clés : TF-IDF

TF correspond au nombre d'occurrences d'un mot dans **un seul document**, tandis que **IDF** mesure l'importance d'un mot dans l'**ensemble du corpus**.

Intuition : si un mot apparait beaucoup dans un document, il est probablement important et pertinent (capturé par **TF**). Cependant, si le même mot apparait dans de nombreux documents du corpus, il est surement moins pertinent pour la recherche (capturé par **IDF**).

Recherche par mots-clés : TF-IDF

Le score TF-IDF d'un terme t dans un document d par rapport à un corpus D se calcule alors comme suit :

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Recherche par mots-clés : TF-IDF | Calcul de TF

$$\mathrm{TF}(t, d) = \frac{\text{occurrences du terme } t \text{ dans le document d}}{\text{nombre total de termes dans le document } d}$$

Ou plus formellement :

$$\mathrm{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Recherche par mots-clés : TF-IDF | Calcul d'IDF

Pour rappel, IDF mesure l'importance d'un mot dans l'ensemble du corpus.

$$\mathrm{IDF}(t, D) = \log\left(\frac{\text{nombre total de documents dans le corpus } D}{\text{nombre de documents contenant le terme } t}\right)$$

Le logarithme est utilisé pour que les valeurs ne soient pas trop grandes (ce qui serait le cas avec un très grand corpus), et ainsi ne pas donner trop de poids aux mots très rares.

Recherche par mots-clés : TF-IDF

Plus le score TF-IDF d'un terme est élevé, plus ce terme est jugé important et représentatif du document dans lequel il apparait, en comparaison avec les autres documents du corpus.

Les termes avec un TF-IDF élévé peuvent être utilisés comme « mots-clés » du document et servent au référencement et à la recherche de documents par terme.

Recherche par mots-clés : TF-IDF

Python

```python
from sklearn.feature_extraction.text import TfidfVectorizer

corpus: list[str] = [] # liste de documents
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus) # shape: (n_docs, n_features)
```

Recherche par mots-clés : TF-IDF

Avec `TfidfVectorizer()`, possibilité de définir plusieurs options :

- `stop_words` : langue de la liste de mots à ignorer
- `ngram_range` : pour utiliser des n-grammes ; prend un tuple (min_x, max_x) où min_x et max_x sont des entiers définissant la taille minimale et maximale des n-grammes à considérer. Par exemple : (1, 2) = unigrammes et bigrammes ; (1, 1) = unigrammes uniquement. Par défaut : (1, 1).

Documentation : **https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html**

Récupération des résultats par requête et classement

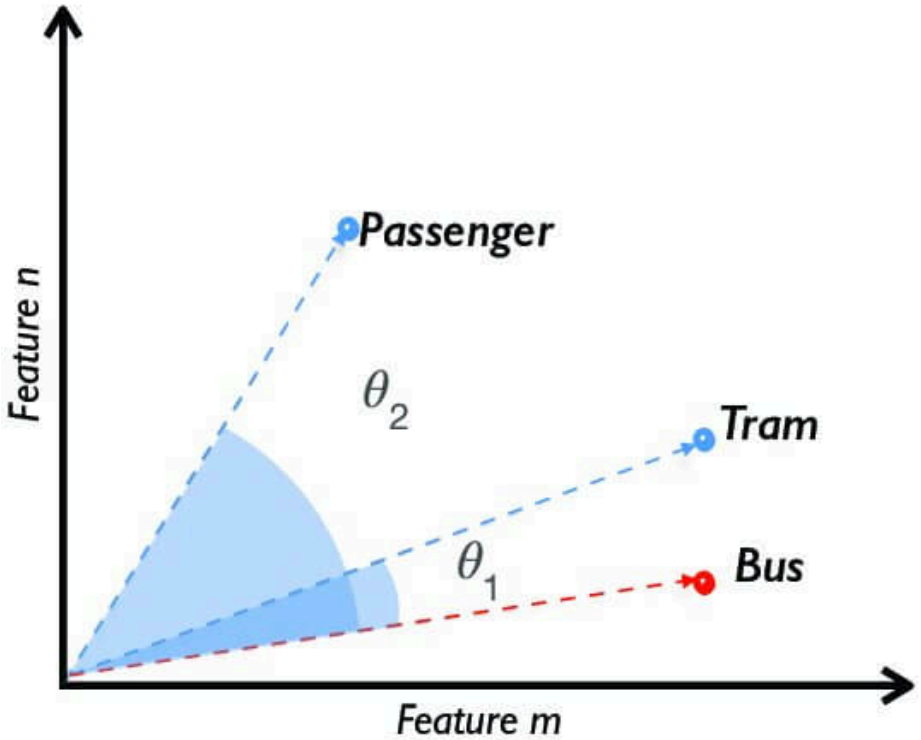
Rappelons le problème de base de la recherche d'information : étant donné une requête de recherche q, on veut trouver les documents les plus pertinents dans un corpus. Ces documents doivent ensuite être **récupérés**, et **classés** par ordre de pertinence.

Similarité cosinus

On peut transformer notre requête q en vecteur TF-IDF de la même manière que pour les documents du corpus, puis calculer la similarité entre ce vecteur de requête et les vecteurs des documents pour obtenir un classement des résultats (documents les plus similaires à q en premier).

Ce calcul de similarité peut être effectué à l'aide de la **similarité cosinus**, qui mesure l'angle entre deux vecteurs dans un espace vectoriel.

Similarité cosinus



Source : Kalwar et al. (2023)

Similarité cosinus

On peut transformer notre requête q en vecteur TF-IDF de la même manière que pour les documents du corpus, puis calculer la similarité entre ce vecteur de requête et les vecteurs des documents pour obtenir un classement des résultats (documents les plus similaires à q en premier).

Ce calcul de similarité peut être effectué à l'aide de la **similarité cosinus**, qui mesure l'angle entre deux vecteurs dans un espace vectoriel.

$$\text{sim}(a, b) = \cos(\theta) = \frac{A \cdot B}{\|A\| \ \|B\|}$$

Où $A \cdot B$ est le produit scalaire des vecteurs A et B, et $||A||$ et $||B||$ sont les normes (ou longueurs) des vecteurs.

Similarité cosinus

Implémentation simple avec *scikit-learn* :

Python

```python
from sklearn.metrics.pairwise import cosine_similarity

query_vec = vectorizer.transform([query])
similarities = cosine_similarity(query_vec, X).flatten()
```

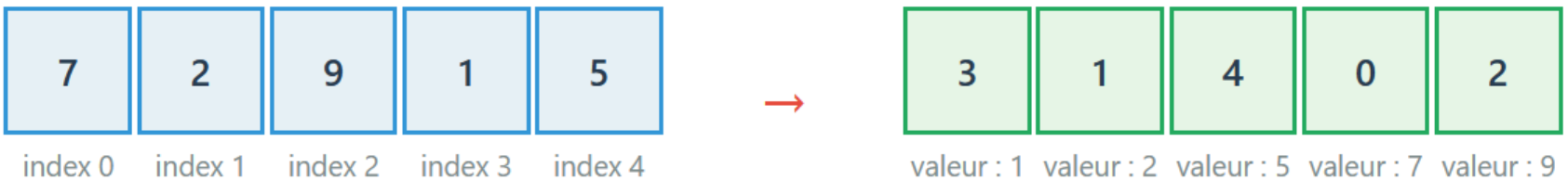
Récupération des documents

Ensuite, on peut récupérer les meilleurs n documents en triant la liste par ordre croissant et en récupérant les indices correspondants avec `np.argsort()` :

```python
import numpy as np
top_n = np.argsort(similarities)[::-1][:n]
```

Exemple : np.argsort()

```
import numpy as np
arr = np.array([7, 2, 9, 1, 5])
indices = np.argsort(arr)
print(indices) # Output: [3 1 4 0 2]
```



II. Modèle Okapi BM25

Recherche par mots-clés : BM25

Le score **Okapi BM25** améliore TF-IDF sur plusieurs points :

- **requête à plusieurs termes** : BM25 gère mieux les requêtes composées de plusieurs termes, en tenant compte de la pertinence de chaque terme dans le document.
- **saturation** : BM25 modélise un effet de saturation de la fréquence des termes. En d'autres mots : à quel point la répétition d'un même terme dans un document augmente-t-il la pertinence de ce dernier ?
- **normalisation de la longueur des documents** : BM25 permet de pénaliser les documents plus longs pour éviter qu'ils ne dominent les résultats simplement en raison de leur taille.

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

Recherche par mots-clés : BM25

La formule de l'**IDF** est légèrement modifiée par rapport à celle de TF-IDF :

$$\mathrm{IDF}(t) = \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5}\right)$$

Où N est le nombre de documents dans le corpus, et $n(t)$ est le nombre de documents contenant le terme t.

BM25 ajoute +0,5 pour empêcher la division par zéro (si le terme t n'apparait jamais dans le corpus) et pour éviter que les termes très rares n'aient un poids trop élevé.

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

somme des scores pour chaque terme

Recherche par mots-clés : BM25

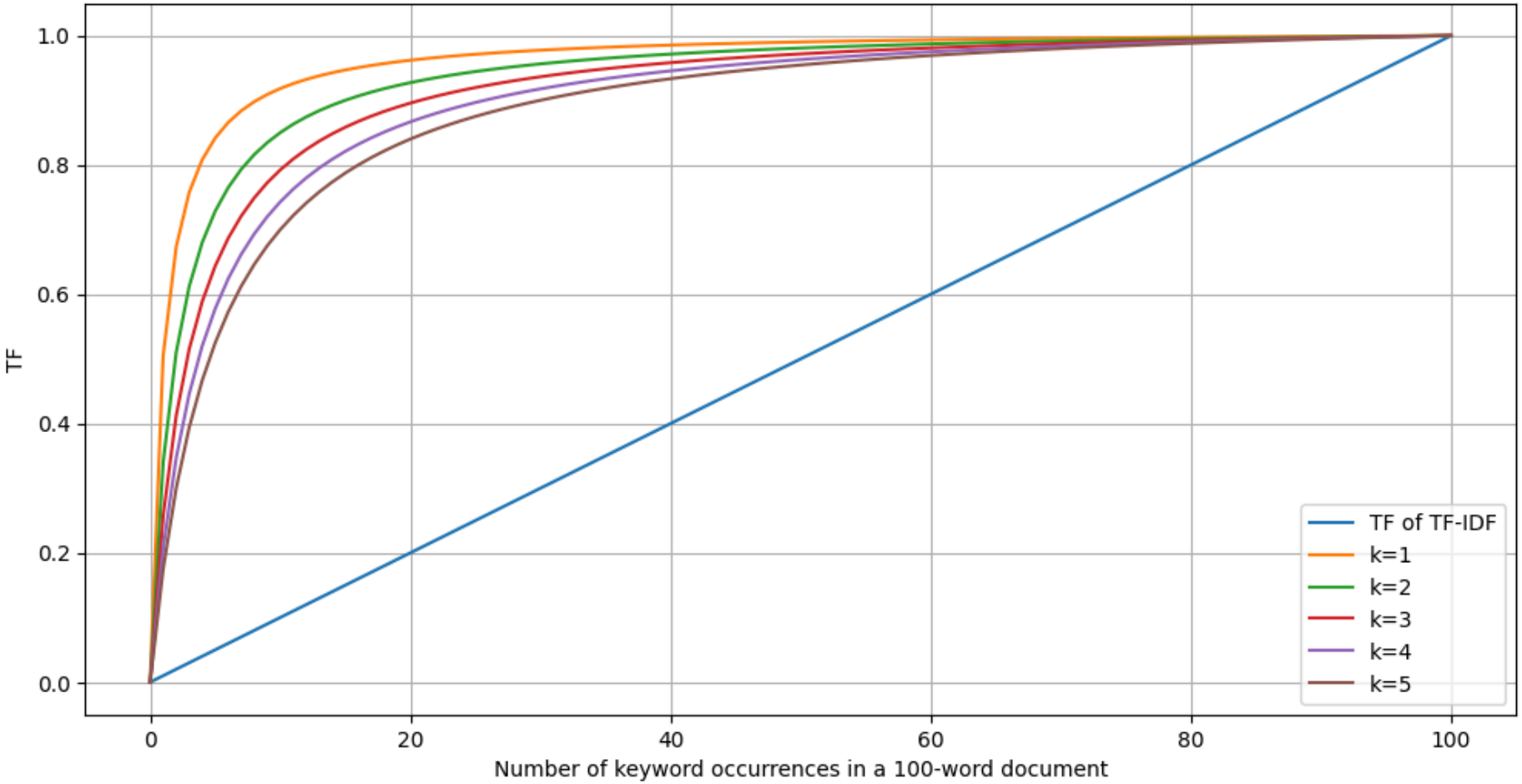
Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

somme des scores pour chaque terme

saturation tf

k_1 contrôle la saturation de la formule TF (*term frequency*) ; limite combien un terme dans la requête change le score d'un document. Un k_1 plus élevé signifie que le score augmente moins rapidement avec la fréquence du terme. Généralement compris entre 1,2 et 2.



Source : https://zilliz.com/learn/mastering-bm25-a-deep-dive-into-the-algorithm-and-application-in-milvus

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

somme des scores pour chaque terme

saturation tf

long. doc.

long. moy. tous docs

Recherche par mots-clés : BM25

Pour un document d et une requête q :

$$\mathrm{BM25}(d, q) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{tf}(t, d)(k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \left(1 - b + b \frac{|d|}{\mathrm{avgdl}}\right)}$$

somme des scores pour chaque terme

saturation tf

norm. longueur

long. doc.

long. moy. tous docs

$b \in [0, 1]$ contrôle la pénalité appliquée aux documents longs.

Recherche par mots-clés : BM25

Plusieurs implémentations Python existent pour BM25 :

- **rank-bm25** : implémente Okapi BM25 (la version de base de BM25), ainsi que des variantes comme BM25+ et BM25L. **https://pypi.org/project/rank-bm25/**
- **bm25s** : une autre implémentation plus rapide comparativement à rank-bm25. **https://bm25s.github.io/**

⊕ Ressources complémentaires

- *Awesome Information Retrieval* : **https://github.com/harpribot/awesome-information-retrieval**

Bibliographie

Kalwar, S., Rossi, M., et Sadeghi, M. (2023). Automated Creation of Mappings Between Data Specifications Through Linguistic and Structural Techniques. *IEEE Access*, *11*, 30324-30339. **10.1109/ACCESS.2023.3259904**

Manning, C. D., Raghavan, P., et Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mitra, B. (2018). *An Introduction to Neural Information Retrieval* (Numéro v.41). Now Publishers.

Sparck Jones, K. (janvier 1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, *28*(1), 11-21. **10.1108/eb026526**

Zhai, C., et Massung, S. (juin 2016). *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool. **10.1145/2915031**