Project 1: Document Retrieval

Luca Engel
luca.engel@epfl.ch

Team Name: Gigubyte
Final notebook: [1]

Damian Kopp
damian.kopp@epfl.ch

1 Introduction

Document retrieval using a large corpus comes with numerous challenges. Finding a robust system that, given an input query, retrieves the 10 most relevant documents out of a large document corpus was the goal of this project. This report compares and discusses various retrieval methods for this task. It evaluates TF-IDF, BM25, and dense document retrieval on a multilingual dataset. We find that BM25 is efficient, robust, and effective at handling long documents, while dense retrieval performs worse due to memory and computational constraints. TF-IDF performs the worst. These results suggest that BM25 offers a suitable solution for retrieving long-form documents with limited resources, while dense retrieval, despite its strengths for capturing semantic similarity, cannot match this performance under the given constraints.

2 Approach

2.1 TF-IDF

One of the simplest methods for document ranking is TF-IDF. It highlights words that appear more frequently within the input query but are rare across the corpus. TF-IDF ranks documents based on TF (term frequency) and IDF (inverse document frequency):

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

- $\text{TF}(t, d) = \frac{\text{Number of appearances of } t \text{ in } d}{\text{Total number of terms in } d}$
- $\text{IDF}(t, N) = \log(\frac{N}{\text{Number of documents containing } t})$

t denotes a term, d a document, and N the number of documents in the corpus.

The IDF values of all terms in the corpus are computed independently of the queries. During inference, the input query's TF values are calculated, while the corresponding IDF values have been precomputed over the corpus. By using cosine similarity, the TF-IDF scores of the query are compared to those of the corpus documents, providing a similarity score between 0 and 1. The corpus documents are then ranked based on these scores to identify the most relevant matches for the query [2]. One notebook was used for precomputations [3] while another notebook [4] was used for the inference using the precomputations.

2.2 BM25

BM25 is another bag-of-words retrieval method that ranks documents based on query relevance. It balances document length normalization, term frequency, and inverse document frequency. The core concept of using TF and IDF is similar to the TF-IDF system. However, other than TF-IDF, which simply multiplies term frequency with inverse document frequency, BM25 introduces additional parameters, $k1$ and b, to adjust the contribution of term frequency. The BM25 score for a term t in a document d is computed as:

$$\text{BM25}(t, d, N) = \frac{\text{TF}(t, d) \cdot (k_1 + 1) \cdot \text{IDF}(t, N)}{\text{TF}(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}$$

where

$$\text{IDF}(t, N) = \ln\left(\frac{N - n(t) + 0.5}{n(t) + 0.5} + 1\right)$$

N is the total number of documents, $n(t)$ is the number of documents containing t, k_1 is a term frequency parameter (often between 1.2–2.0), b is the document length normalization parameter (between 0 and 1), $|d|$ is the length of d, and avgdl is the average document length.

BM25 uses the parameter k_1 to ensure that the term frequency's impact is capped. This prevents frequently occurring terms from excessively affecting the score. Additionally, parameter b is used to apply document length normalization. This helps to prevent longer documents from being overly favored [5].

We chose BM25 due to its robustness in handling large-scale datasets and its ability to deal

effectively with lengthy documents while keeping computational resource requirements low. Similar to TF-IDF, two notebooks were used for precomputations [6] and inference [1], respectively.

2.3 Dense Vector Retrieval Text Embeddings

Dense retrieval leverages embeddings to capture semantic representations of text. We implemented this approach in a notebook [7] by leveraging Sentence Transformers combined with FAISS, a state-of-the-art vector similarity search library. [8] [9] [10] Given the long document lengths, we split each document into smaller chunks to make them compatible with embedding models' token limit. Also, this may improve retrieval as long documents make finding precise matches when querying the index more difficult. [11]

To optimize retrieval time and accuracy for our multilingual dataset, we divided the database into 7 subsets, one for each language.

2.4 Challenges

One of the main challenges for dense vector retrieval was handling long documents. Sentence Transformers, which are used to generate dense embeddings, have a token limit. This made it necessary to split documents into smaller chunks. While chunking significantly improved the performance, memory and computational constraints limited our ability to fully exploit the model's capacity, and the retrieval score remained below 0.5. For these document lengths, finetuning large pretrained models would likely have resulted in a better performance. However, due to the available GPU's memory and computational constraints, this approach could not be tested.

Additionally, while BM25 can efficiently handle large document lengths, dense retrieval's reliance on embeddings introduces overheads that can only be mitigated with more computational resources.

3 Results

TF-IDF	BM25	FAISS
0.37376	0.77351	0.47153

Table 1: Best scores per method

BM25 achieved a retrieval score of 0.77, significantly outperforming the other two approaches, which both remained below 0.5. For all approaches, performance was evaluated across the seven languages in our dataset, but the overall trend remained consistent: BM25 provided a more accurate and computationally feasible solution. The highest performance was with $k_1 = 1.5$ and $b = 0.45$.

	b=0.35	**b=0.45**	**b=0.55**
k_1=1.4	**0.77351**	0.76980	0.77227
k_1=1.5	**0.77351**	**0.77351**	0.76980
k_1=1.6	0.76980	0.77103	0.76856

Table 2: BM25 Recall@10 Scores for Different b and k_1 Values on the Validation Set

4 Analysis

The superior performance of BM25 in this case can be attributed to its ability to handle long documents effectively without requiring complex embeddings and to still capture the information detailed enough. In contrast, dense retrieval struggles with long documents because embedding models are constrained by token limits, and chunking strategies may not always capture the document's context effectively. Additionally, the computation time and memory requirements for dense retrieval were significantly higher, further complicating its implementation. TF-IDF is less computationally costly than BM25, but that clearly harms the number of correct document rankings as TF-IDF is more sensitive to document length variations.

Splitting the database by language slightly improved the retrieval speed for all approaches, but it did not substantially impact the final score for dense retrieval. The choice of model (Sentence Transformer vs. BM25 vs. TF-IDF) was crucial. In this case, BM25 proved a suitable trade-off between being lightweight and capturing information accurately for our task.

5 Conclusion

In scenarios where documents are particularly long, BM25 remains a highly efficient and effective method for document retrieval. Dense retrieval, while powerful in theory due to its semantic understanding, requires significant computational resources to outperform simpler statistical methods like BM25. Future work could explore more scalable dense retrieval models or hybrid approaches that combine the strengths of both methods. Also, the use of more sophisticated tokenizers could further improve the performance.

6 Appendix

The notebooks described and referenced in the report can be found here:

- Dense Document Retrieval with Sentence Transformer and Faiss: [7]
- BM25
  - Precomputations: [6]
  - Inference: [1]
- TF-IDF:
  - Precomputations: [3]
  - Inference: [4]

The algorithms for BM25 and TF-IDF are divided into two separate notebooks: The "Precomputations" notebook computes the needed elements such as the vocabulary and necessary matrices, while the "Inference" notebook generates the final predictions.

References

[1] ThymianTheHerb. Submission bm25 inference. https://www.kaggle.com/code/thymiantheherb/submission-bm25-inference, 2024. Accessed: November 1, 2024.

[2] Wikipedia contributors. tf-idf, 2024. Accessed: 2024-10-01.

[3] ThymianTheHerb. Submission tf-idf precomputations, 2024. Included in submission ZIP file.

[4] ThymianTheHerb. Submission tf-idf inference, 2024. Included in submission ZIP file.

[5] Wikipedia contributors. Okapi bm25, 2024. Accessed: 2024-10-01.

[6] ThymianTheHerb. Submission bm25 precomputations, 2024. Included in submission ZIP file.

[7] Luca Engel. Submission sentence transformer embedding approach, 2024. Included in submission ZIP file.

[8] Faiss: A library for efficient similarity search, March 2017.

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024.

[10] SentenceTransformers Documentation — Sentence Transformers documentation.

[11] Roie Schwaber-Cohen. Chunking Strategies for LLM Applications | Pinecone.