Multilingual Document Retrieval: A BM25+ Approach Across Seven Languages

Kyuhee Kim, Christina Kopidaki, Kamel Charaf
Recommendos
Kaggle notebook
kyuhee.kim@epfl.ch, christina.kopidaki@epfl.ch, kamel.charaf@epfl.ch

I. Introduction

In this report, we describe our solutions to the problem of retrieving the ten most relevant documents for every query, choosing from a vast collection of 268022 documents. Both the queries and the documents are in 7 different languages, including *English (EN), French (FR), German (DE), Italian (IT), Spanish (ES), Arabic (AR)* and *Korean (KO)*. The problem is formulated as a Kaggle competition[1] were the models are evaluated using 2,000 queries on the metric Recall@10.

To tackle the task mentioned above, we explored multiple approaches, which include *Vector Space Retrievals*, *Probabilistic language models* and also *Transformer-based language models*. The proposed solutions are compared based on their simplicity, their required dependencies, their accuracy and their resource needs. Our primary and best-achieving solution is based on fitting a different BM25+ model to the documents in each language.

II. Experiment Setup

A. Dataset

The provided dataset consists of queries and documents in seven different languages, divided into four parts:

- **Train Data**: Contains queries with one relevant and twenty non-relevant documents per query.
- **Dev Data**: Follows the same structure as the Train Data.
- **Test Data**: Contains only queries, which we need to find the top ten most relevant documents to.
- **Corpus**: The whole collection of documents needed for the information retrieval task.

The number of documents and queries in each language is summarized in Table I. The majority of our corpus is composed of English documents, but only 40% of our queries are in English. For word occurrence-based techniques that are highly sensitive to word variations, we preprocessed the queries and documents.

B. Preprocessing

Our methods and models (see section III) contain not only semantic-based approaches but also simple, word occurence-based techniques that are highly sensitive to the variation of the words. We tackled the sensitivity by preprocessing the queries and documents with the following pipeline which includes **lowercasing**, **punctuation removal**, **tokenization**, **stopword elimination** and **stemming**. We used NLTK's tokenizer for tokenization and the Porter stemmer for stemming. We used NLTK's remove stopwords for every language except Korean, for which we used the stopwords found on GitHub).

TABLE I
Dataset Statistics by Language. 'Words' indicates the average number of words per document.

	EN	FR	DE	ES	IT	KO	AR
Documents	207,363	10,676	10,992	11,019	11,250	7,893	8,829
Words	2117.16	5599.21	4337.67	5615.71	5340.02	2650.62	4496.05
Test Queries	800	200	200	200	200	200	200

C. Evaluation Metric

In addition to the leaderboard score, we used the dev dataset to further assess our model's performance.

$$Recall@10 = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\begin{cases}1 & \text{if } Q_{p_i} \in K_{10} \\ 0 & \text{if } Q_{p_i} \notin K_{10}\end{cases}$$

where

- Q_{p_i} is the positive docid related to the i^{th} dev query
- $|Q|$ denotes to the number of queries
- K_{10} denotes to the retrieved top-10 documents by the model

III. Models and Methods

TF-IDF is a vector model that uses both term-frequency (TF) and Inverse Term Frequence (IDF) in order to assess term importance in documents. Each query and each document is represented as a sparse-vector and the query-document matching is determined using cosine similarity. In our approach, we used IndexFlatIP in FAISS [1], which employs indexing structures for efficient ranking. Also, we created TF-IDF models for each language, selecting the top 100 most frequently occurring words per document to form the vocabulary for each language. Since there are too many English documents, in this case, we used the top 10.000 words globally as the vocabulary for English. This approach scored **0.42** on Kaggle, though it was computationally heavy, time-intensive and memory-expensive.

BM25 model ranks documents based on the presence and frequency of query terms in documents, using factors like

[1] https://www.kaggle.com/competitions/dis-project-1-document-retrieval/overview

term frequency (TF), inverse document frequency (IDF), and document length normalization. However, BM25 can penalize long documents disproportionately, which leads to lower relevance scores for these documents even if they contain relevant content. BM25+ adjusts the term frequency component to provide a lower bound on the contribution of each term occurrence. Each of the variations we tried, is further discussed below.

A. Single BM25+ model

The initial implementation of BM25+ involved training a single model on the entire corpus (with all of the languages combined) and subsequently calculating the similarity for each query. The primary advantage of this model is its capability to perform cross-lingual retrieval. This approach achieved a maximum score of **0.63** on Kaggle.

B. BM25+ and BERT Reranking

We used BM25+ to retrieve the top 100 relevant documents to a query, then we used a re-ranker that utilizes the similarity between the queries and retrieved documents to find the top 10 most relevant documents. We generated embeddings using three different pre-trained models: "all-MiniLM-L12-v2" [4], "distiluse-base-multilingual-cased-v2" [3] and "Language-agnostic BERT" [2]. However, the Recall@10 reduced significantly compared to the previous version of the Single BM25+ model.

C. Language-separated BM25+ models

As BM25+ is a model that ranks documents based on the presence and frequency of query terms in documents, fitting a different BM25+ model on the documents of each language and then using the proper model based on the query's language, improved the score on Kaggle up to **0.77**. We experimented with different values of top-K words per document to check how it affects the performance (Table II). Our results show, that the more tokens we use, the more accurate the model becomes. However, retaining more than 2000 tokens/document was producing slight difference, thus not worth using it.

TABLE II
LEADERBOARD SCORES BASED ON THE CHOOSEN AMOUNT OF TOKENS

Top-K tokens	**500**	**1000**	**2000**
Recall@10	0.671	0.701	0.767

IV. FURTHER REFINEMENTS BY LANGUAGE

The evaluation results in Figure 1 show that our model performs better in some languages than others, prompting further improvements in Korean, Arabic, and German. For Korean, using the Open Korean Text tokenizer and removing grammatical morphemes boosted the score from 0.65 to 0.71. For Arabic, however, the ISRI and ARLSTem2 stemmers led to lower scores (0.56 and 0.62) compared to PorterStemmer. Applying the Cistem stemmer for German also reduced performance to 0.695.

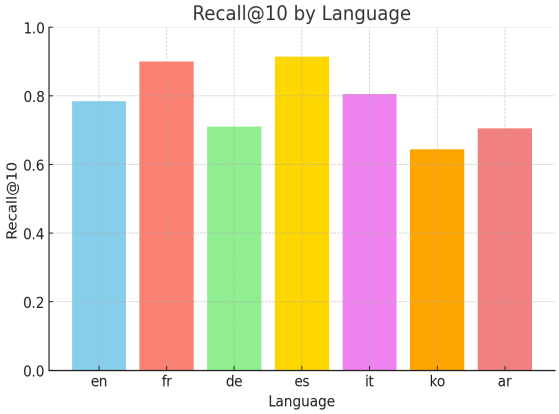

Fig. 1. Recall@10 score separated by languages

With the improved Korean preprocessing, we achieved our final and best score of **0.785** on Kaggle.

V. DISCUSSION

When Simpler Models Prove Robust Our investigation revealed that although pre-trained embeddings are powerful, they do not consistently outperform traditional models like BM25 in certain retrieval tasks. This finding highlights, that simpler, well-established models can often deliver competitive and robust solutions, especially when computational efficiency and interpretability are key concerns.

Tokenizer Performance Not Corresponding with BM25 Interestingly, the performance of tokenizers and stemmers did not directly correlate with BM25's effectiveness. While advanced tokenizers can enhance understanding, they may introduce overhead that complicates retrieval tasks without necessarily improving relevance.

Time and Memory Management Furthermore, managing time and memory resources emerged as critical factors, since the task was to be done in 10 minutes and the given RAM was 28GB. To accelerate the retrieval time, we pre-saved the calculated BM25 word scores for each document. Converting them into vectors would allow faster score retrieval; however, due to their sparsity and high memory requirements, we opted to create a dictionary and index words directly instead.

VI. CONCLUSION

While our findings indicate that simpler models can deliver competitive results, substantial room for improvement remains. More complex approaches, such as ranking initially with BM25+ and re-ranking with Cross-encoders, or Hybrid Search combining their scores (see further details in A), could yield significant gains. Although we initially attempted to train a bi-encoder model from scratch, high computational costs limited a full exploration of its potential. Further tuning of existing models and hybrid approaches that blend traditional and neural retrieval methods could enhance system performance.

REFERENCES

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

[2] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding, 2022.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[4] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

APPENDIX

A. BM25+ and Cross-encoder Reranking

To improve Recall@10, we tested re-ranking with cross-encoders after the initial BM25+ ranking. We first analyzed potential performance gains by increasing the number of documents retrieved from our best BM25+ model (see Table III) for the English portion. We observed that increasing the number of documents retrieved by our BM25+ model led to a higher Recall value. This suggests that the pipeline could perform effectively if the re-ranker accurately identifies the most relevant documents. Given that the most notable improvement occurred with the top 30 results, our goal was to rerank the top 30 documents retrieved by this model.

TABLE III
RECALL@K VALUES OF OUR BEST BM25+ MODEL FOR ENGLISH DEV QUERIES (K = 10, 20, 30, 40, AND 50)

Recall@10	**Recall@20**	**Recall@30**	**Recall@40**	**Recall@50**
78.5	81	85.5	86	86.5

Keeping all other experimental settings the same (see Appendix B for details), we configured the training data for the Cross-encoder used in English as follows:

- *Base*: The original training data
- *Top*: The top 10 documents retrieved from our best BM25 model for each training query
- *Base-neg 4*: The Base data with 4 negative documents sampled per query
- *Top-neg 4*: The Top data with 4 negative documents sampled per query

None of the reranking models, across all configurations, surpassed the best Recall@10 score of 0.785 achieved by our top-performing BM25+ model on the development queries (see Table IV), with each configuration exhibiting a decline in performance.

TABLE IV
RERANKING RESULTS: RECALL@10 ON ENGLISH DEV QUERIES

	Our best BM25+	***Base***	***Top***	***Base-neg4***	***Top-neg4***
recall@10	0.785	0.685	0.65	0.68	0.615

To address this, we combined the original BM25 score with the reranked document scores by applying min-max normalization to each and summed the normalized scores to re-rank the top 10 documents based on the highest combined score.

$$\text{Normalized score} = \frac{\text{score} - \min(\text{score})}{\max(\text{score}) - \min(\text{score})}$$

As shown in Table V, this method improved Recall@10 across all models, with the unsampled *Top* configuration achieving the best results. However, since it didn't improve the scores on the test dataset, we did not incorporate it in our final submission.

TABLE V
HYBRID SEARCH RESULTS: RECALL@10 ON ENGLISH DEV QUERIES

	Our best BM25+	***Base***	***Top***	***Base-neg4***	***Top-neg4***
Recall@10	0.785	0.805	0.815	0.805	0.795

B. Cross-Encoder Training Details

We used the Sentence BERT library to train a cross-encoder with default hyperparameters. For selecting the best checkpoint, we used the top 30 documents retrieved by our best BM25 model from each dev query and employed the CERerankingEvaluator.

- Base Model: microsoft/deberta-v3-base
- Training Epoch: 1
- Warm-up steps: 500
- Evaluation step: 100 (for non-sampled data 1000)
- Evalator: mrr@10 (CERerankingEvaluator)
- Max Token Length: 512
- Dev Dataset: The top 30 documents retrieved from our best BM25 for each dev query