Ensemble Document Retrieval in a Multilingual Corpus

Team MAM : Aryan Ahadinia and **Matin Ansaripour** and **Madeleine Hueber**
{aryan.ahadinia, matin.ansaripour, madeleine.hueber}@epfl.ch

Abstract

In this study, we present an information retrieval system capable of efficiently retrieving from a large multilingual corpus. We leverage a hybrid approach for combining the strengths of both TF-IDF and BM25 models into a single ensemble model. Furthermore, we prove the efficiency of our method using empirical results. Our code is mainly based on GitHub and our pre-computed tokens and models are available on Kaggle which are used for our **Submission Notebook on Kaggle**. (details in app. A).

1 Introduction

With the rapid growth of online content, there is a growing need for effective information retrieval (IR) systems. In this report, we present a high-recall multilingual retrieval system within a corpus of French (FR), English (EN), German (DE), Spanish (ES), Italian (IT), Arabic (AR), and Korean (KO) documents. We first explored an initial approach using TF-IDF and BM25. Then, we developed a hybrid approach for combining the strengths of both models. We also implemented some other methods such as Dense Passage Retrieval and discussed their unsatisfactory performances in our problem. Finally, we evaluated the performance of our model in terms of $recall@k$ and compared it to the baseline methods.

2 Proposed Method

2.1 Data Preparation

The first step in deploying any retrieval system is *tokenization*. Our process begins with *text cleaning* to remove non-textual data such as URLs, tables, and images, using regular expressions. Then, we remove *stopwords* for each language using spaCy (Honnibal et al., 2020), except for Arabic, where we implemented our cleaner based on the NLTK (Bird and Loper, 2004). Finally, we apply *lemmatization* to reduce tokens to their root forms, using spaCy's pre-trained models (and CAMeL Tools (Obeid et al., 2020) for Arabic). This step reduces vocabulary size and improves query-documents terms matching, which will lead to an increase in recall. Due to the time-consuming nature of the tokenization pipeline, especially the lemmatization step, our implementation is tailored for distributed computation. This tokenization step ensures that data fed into the IR models is relevant and semantically rich.

2.2 TF-IDF

We used our modification of TF-IDF (Sparck Jones, 1988) as a part of our IR system. For a documents corpus $\mathcal{D}$ and a vocabulary set $\mathcal{V}$, the term frequency vector and inverse document frequency are being calculated using (1) and the TF-IDF representation is given by (2) in which $n_{t,d}$ is the frequency of term t in document d and n_t is the number of documents containing term t.

$$\text{TF}(d) = [n_{t,d}]_{t\in\mathcal{V}}, \quad \text{IDF}(t) = \ln\left(\frac{|\mathcal{D}|}{n_t}\right) \tag{1}$$

$$\text{TF-IDF}(d) = \frac{[\text{TF}(d,t) \times \text{IDF}(t)]_{t\in\mathcal{V}}}{||[\text{TF}(d,t) \times \text{IDF}(t)]_{t\in\mathcal{V}}||_2}. \tag{2}$$

To account for document length variability, in (2) we used a normalized version of TF vectors of both queries and documents, TF$'$ instead of TF. In TF$'$ for documents, we use a pivot based on the median of TFs' norms for each language (details provided in Appendix C). During inference, the relevance score is calculated as the inner dot product between the TF-IDF representations of the query and documents. To optimize computation, we precomputed and stored the TF-IDF matrix of the documents for each language as sparse matrices. For each language, we then computed the query representations in sparse matrix form and used sparse matrix multiplication for efficient inference with minimal time and memory usage.

2.3 BM-25

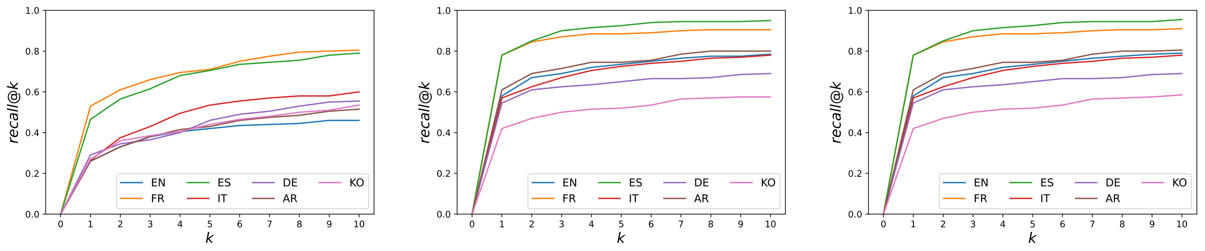
To optimize inference time through pre-computation, we decomposed the BM25 (Robertson and Zaragoza, 2009) formulation into two parts: the fitting phase as (4) and the inference phase as (5) in which L is the average document length in the corpus and $s(q,d)$ denotes the relevance score of document d to query q. Other notations are same as TF-IDF.

$$IDF(t) = \log\left(\frac{|\mathcal{D}| - n_t + 0.5}{n_t + 0.5} + 1\right) \tag{3}$$

$$g(t,d;\mathcal{D}) = \frac{\text{IDF}(t)\cdot n_{t,d}\cdot(k_1+1)}{n_{t,d} + k_1\cdot\left(1 - b + b\cdot\frac{|\mathcal{D}|}{L}\right)} \tag{4}$$

Table 1: Recall@10 on development data for the TF-IDF, BM25, and the ensemble model on different languages.

Model	EN	FR	DE	IT	ES	KO	AR	ALL
TF-IDF	46.00	80.50	55.00	60.00	79.00	53.50	51.50	60.85
BM25	78.50	90.50	**69.00**	**78.00**	95.00	57.50	80.00	78.35
Ensemble	**79.00**	**91.00**	**69.00**	**78.00**	**95.50**	**58.50**	**80.50**	**78.78**



(a) TF-IDF (b) BM25 (c) Ensemble

Figure 1: $Recall@k$ vs k (number of retrieved documents) of different models.

$$s(q, d; \mathcal{D}) = \sum_{t \in q} g(t, d; \mathcal{D}) \tag{5}$$

The values of $G = [g(t, d; \mathcal{D})]_{d \in \mathcal{D}, t \in \mathcal{V}}$ are mostly zeros since the term frequencies are mostly zeros, therefore we can pre-compute and save G in a sparse matrix for each language separately. The inference can be done for all documents by adding up the columns of G. So, matrix G is saved as SciPy (Virtanen et al., 2020) *Compressed Sparse Column* matrix for efficient columns summation. As (4) shows, BM25 has two hyper-parameters k_1 and b which we tune on the development dataset for each language, as outlined in 3.1. The IDF is calculated as in (3) for BM25. Although dozens of BM25 extensions have been proposed for better performance such as BM25+ (Lv and Zhai, 2011a) or BM25L (Lv and Zhai, 2011b), we prefer the original version because it can be implemented with sparse matrices and requires the least amount of computations in the inference time.

2.4 Ensemble Model

As illustrated in figure 1, most of the hits of both TF-IDF and BM25 models are on their first predictions based on their scores, therefore, combining their initial guesses can improve performance. In this regard, we include predictions of both of these algorithms in the final top-k prediction in our ensemble model. The portion of contribution of each model in the final prediction is tuned on the development dataset for each language separately. Note that we prioritize the predictions of BM25 over TF-IDF with similar rankings in case of duplication in the first predictions as discussed in appendix D.

2.5 Dense Passage Retrieval (DPR)

We explored the use of a DPR model (Karpukhin et al., 2020), which can be beneficial for our retrieval system for aligning distributions of queries and the corpus. In this regard, we employed `microsoft/mdeberta-v3-base`, a multilingual BERT model. However, during testing, this approach yielded unsatisfactory results, so we decided to set it aside (details in appendix E).

3 Results

3.1 Hyper-parameter Tuning

The hyper-parameters tuning of our model is done in two stages. Firstly, we tune the hyper-parameters of the BM25 model, $k1$ and b, and secondly, we tune the portion of contribution of each model in the final hybrid prediction, both using grid search. Details are available in appendix F. For some languages, the best portion of the contribution of BM25 is 1.00 which indicate that taking TF-IDF predictions into account will not increase the performance.

3.2 Performance Comparison

The performance of TF-IDF, BM25, and the ensemble models are illustrated in table 1. Although BM25 performs significantly better than TF-IDF, taking TF-IDF predictions into account improves the performance of the system up to 1% for some languages and up to 0.43% in overall.

4 Discussion and Conclusion

At a glance, we showed that a well-tuned ensemble model consisting of TF-IDF and BM25 retrieval model can outperform on this dataset. However, despite these positive outcomes, there are notable areas for improvement. One significant idea would be to introduce a more robust supervised learning component, drawing inspiration from the DPR model. Although our initial DPR trials yielded limited results, refining this approach to better fit our data could provide a deeper understanding of document relevance.

References

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Yuanhua Lv and ChengXiang Zhai. 2011a. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 7–16, New York, NY, USA. Association for Computing Machinery.

Yuanhua Lv and ChengXiang Zhai. 2011b. When documents are very long, bm25 fails! In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 1103–1104, New York, NY, USA. Association for Computing Machinery.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. JONES, Nathaniel Smith, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antoinne Bell, Silvester S. Diederichs, Jaime Nunez-Iglesias, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272.

A Statement on Deliverables

Our code is mainly based on GitHub and our trained models are available on Kaggle. The final inference notebook on Kaggle and our repository on GitHub will be publicly available on Nov 9, 2024. On the inference notebook, we have loaded our source code from GitHub as a library. Note that the GitHub development token on the inference notebook can be ignored after Nov 9, 2024, as it is for loading our repository, which was (is) private before the deadline.

B More Detail on the Tokenization

We also employed compound word decomposition using the *pyphen* library for agglutinative languages, however, it resulted in a decrease in the overall performance, so this stage has been discarded in the final model.

C TF-IDF Implementation Details

To improve the robustness of term frequency across documents of varying lengths, we apply a normalization to the vector TF(d). The normalized term frequency vector for a document TF$'(d)$ is defined as (6)

$$\text{TF}'(d) = \frac{\text{TF}(d)}{0.5 \cdot \text{med}_{D_{lan}} + 0.5 \cdot \|\text{TF}(d)\|_2} \tag{6}$$

where lan is the language of the document d and $\text{med}_{D_{lan}}$ represents the median ℓ_2-norm across all documents in the corpus of the language lan. In other words, $D_{lan} : \text{med}_{D_{lan}} = \text{median}(\|\text{TF}(d)\|_2)$ for $d \in D_{lan}$. The normalized TF'(q) vector for a query is defined as (7).

$$\text{TF}'(q) = \frac{\text{TF}(q)}{\|\text{TF}(q)\|_2} \tag{7}$$

The resulting TF-IDF vector for a document d is then given by 8.

$$\text{TF-IDF}(d) = \frac{[\text{TF}'(d,w) \times \text{IDF}(w)]_{w\in\mathcal{V}}}{\|[\text{TF}'(d,w) \times \text{IDF}(w)]_{w\in\mathcal{V}}\|_2} \tag{8}$$

This TF-IDF representation 8 provides a more balanced weighting of terms, especially across documents of different lengths.

D Ensemble Model Details

Needless to say, we ignore the duplicate results of BM25 and TF-IDF while combining their result for ensemble model. If the first predictions of the TF-IDF model are exactly the same to the BM25 so that we reach prediction with lower ranks than the ones we included from BM25, we switch back to BM25 as it performs marginally better as illustrated in details in 1.

E DPR

We explored the use of a DPR model (Karpukhin et al., 2020), which can be beneficial for our retrieval system for aligning distributions of queries and the corpus. We employed `microsoft/mdeberta-v3-base`, a

Algorithm 1 Combining the Models Predictions

```
Require: $p_{BM25}$ as ordered predictions of BM25.
Require: $p_{TF-IDF}$ as ordered predictions of TF-IDF.
Require: $k$ the number of final predictions.
Require: $\eta$ the portion of contribution share.
  $p \leftarrow p_{BM25}[0 : \eta.k]$
  $i \leftarrow 0$
  while $|p| < k$ do
      if $i < \eta.k$ then
          if $p_{TF-IDF}[i] \notin p$ then
              $p \leftarrow p + p_{TF-IDF}[i]$
          end if
      else
          $p \leftarrow p + p_{BM25}[i]$
      end if
      $i \leftarrow i + 1$
  end while
  return $p$
```

multilingual BERT model. We embedded the documents and fine-tuned the embeddings using DPR loss as (9), where d^+ is the relevant document to the query q , $\{d_1^-, ..., d_n^-\}$ are the irrelevant ones and sim is the similarity function in the embedding space. To fit the data within the BERT model context size, we split the documents into overlapping chunks aligned with their sentences. However, during testing, this approach yielded unsatisfactory results, so we decided to set it aside and concentrate our efforts on our ensemble retrieval model. Note that this method requires more computation compared to our method and it requires a lot of computational resources to tune.

$$\mathcal{L} = -\log \frac{e^{sim(q_i,d^+)}}{e^{sim(q_i,d^+)} + \sum_{k=1}^{n} e^{sim(q_i,d_k^-)}} \quad (9)$$

F Hyper-parameter Tuning

In the first stage of hyper-parameter tuning, we tune the BM25 parameter for each language separately in the search space of $\{1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.9, 1.9, 2.0\}$ and $\{.00, .25, .50, .65, .70, .75, .80, .85, .90, .95, 1.00\}$ for k_1 and b, respectively.

In the second stage, we employ a full grid search for finding the best portion of contribution of each models to the ensemble model. The results are demonstrated in table 2.

Table 2: Best portion of contribution of BM25 model in the ensemble model.

Language	BM25 Contribution
EN	0.80
FR	0.90
DE	1.00
IT	1.00
ES	0.80
KO	0.80
AR	0.90