A Text Retrieval Approach Using BM25+ and BERT Reranking

Manos Chatzakis
emmanouil.chatzakis@epfl.ch

Hind El Bouchrifi
hind.elbouchrifi@epfl.ch

Lluka Stojollari
lluka.stojollari@epfl.ch

Kaggle notebook link

ABSTRACT

This report presents a solution to a Kaggle Text Retrieval challenge over a vast document collection. Our approach uses BM25 to fetch a number of relevant documents and BERT Sentence Transformers to re-rank the relevant documents. The evaluation of our approach on Kaggle is graded with a score of 75%, which is one among the top of the competition.

1 INTRODUCTION

Context. In this report, we describe our solution to the problems of retrieving relevant documents for a query (Task 1) as well as reranking collections of retrieved documents (Task 2). The problem is formulated as a Kaggle competition [1], where the models solving the aforementioned tasks are automatically evaluated using Recal@10 for Task 1 and NDCG for Task 2. The competition provides a collection of approximately 1.5M documents, and the models are evaluated using a set of approximately 7.5K queries for both Tasks.
Approach. Our best-scoring approach for Task1 uses BM25 to retrieve the top 100 documents of a query and re-ranks them using BERT Sentence Transformers, while for Task2 we recompute the similarities of the documents only using BERT Sentence Transformers. Our method scored 75.7% on the Kaggle competition, while it managed to fit the BM25 Model in 4 minutes [2] and processed both Tasks in less than 12 minutes.

2 DATASET AND QUERY PREPROCESSING

Preprocessing. We preprocess the document corpus and queries by applying a pipeline of Lowercasing, Punctuation Removal, Tokenization, Stopword Removal, and Stemming.
Query Expansion. As an addition to query processing, we experimented with Query Expansion methods, aiming to increase the Recall.
Synonym-based Expansion.[1]. We generated synonyms to non-stopword query terms according to a given probability, and then we performed stemming to the expanded query.
LLM-based Expansion.[4] We prompted an LLM(FLAN[2]) with the raw query, asking it to answer it and explain the rationale[8] of the generated answer, and we concatenated the answer of the model with our initial query.
Query Expansion Results. Query Expansion yielded slightly worse results for all the models we tried. This was mainly because the generated terms introduced noise, adversely affecting performance. The LLM-based expansion also introduced computational overhead when generating the expansions.

[1] https://www.kaggle.com/competitions/dis-project-1-text-retrieval/overview
[2] We have precomputed the corpus BERT embeddings

3 RETRIEVAL MODELS

3.1 Initial Models

Vector Model (TF-iDF). The TF-iDF Vector Model assesses term importance in documents using both term frequency (TF) and inverse document frequency (iDF). Each document is represented as a vector, and retrieval is determined by cosine similarity between the query vector and document vector. The model scored 55% on Kaggle. The score depicts the inability of the model to capture semantic similarities between queries and vectors, especially when queries lack terms found in relevant documents. Additionally, due to the TF-iDF matrix size (about 1.5M rows), we had to restrict consideration to terms appearing more than 3000 times in the corpus, further impacting the model's performance.
Doc2Vec. Doc2vec [5] extends word2vec [6] by representing whole documents as high-dimensional vectors, capturing their semantic content. To find similar documents, a query vector is created, and retrieval is based on cosine similarity. Doc2Vec did not achieve satisfactory performance, scoring approximately 25% on Kaggle. Due to the time limit constraint and limited resources, we could not train the model for high vector dimensions, which reduced the representation power of the model, and our configuration was restricted to only 50 dimensions and 300 epochs. Encoding documents in 50 dimensions is a bottleneck to capturing the semantic contents.
DSSM. DSSM [3] is a retrieval model using a siamese deep learning architecture that maps text to a shared lower-dimensional space, computes semantic relevance scores, and generates 128 features per sentence using novel letter n-gram-based *Word Hashing*. The fine-tuning of relevance scores occurs through a softmax function, focusing on a candidate document set $\mathcal{D}$. The optimization process is driven by the following loss function: $\mathcal{L} = -\log \prod_{(Q,D^+)} P(D^+|Q)$ *where* $: P(D|Q) = \frac{\exp(R(Q,D))}{\sum_{D' \in D} \exp(R(Q,D'))}$
Inference. After training, the DSSM model was used for inference to produce 128 features per sentence for similarity comparisons. However, the Kaggle results weren't good, highlighting limitations in deep learning architectures. The output features appeared insufficient for capturing the semantic subtleties of our corpus and queries. Additionally, the model may not have had enough diverse training examples (four irrelevant and one relevant) to effectively generalize across various semantic contexts.

3.2 BM25

Okapi BM25: Best Matching 25 is a term-based ranking model to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework. The basic instantiation of the function is as follows:

$$R(D,Q) = \sum_{i=1}^{n} \left[\log\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}\right) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avg\_dl}}\right)} \right]$$

In the IDF formula, $IDF(q_i)$ measures term uniqueness in the document collection, N is the total documents, and $n(q_i)$ is term q_i's document count. In the BM25 formula, R(D, Q) computes document D's relevance to query Q. n is the query term count, q_i is each term, $f(q_i, D)$ is term frequency, k_1 and b are tunable parameters influencing term frequency and document length, |D| is D's length, and avg_dl is the average document length.
Since BM25 achieved better results than before mentioned methods, we focused on extensions and optimizations of this model.
BM25L : BM25 retrieval function tends to penalize very long documents. To address this problem, we used BM25L, which "shifts" the term frequency normalization formula to boost scores of very long documents. The score formula is as follows:

$$R(D,Q) = \sum_{i=1}^{n} \left[\log\left(\frac{N+1}{n(q_i)+0.5}\right) \cdot \frac{(c_d q+\lambda)\cdot(k_1+1)}{(c_d q+\lambda)+k_1)} \right]$$

where: $c_{dq} = \frac{f(q_i,D)}{1-b+b\cdot\frac{|D|}{\text{avg\_dl}}}$

BM25+ : BM25+ proposes a general solution to penalizing long documents due to term frequency by setting a lower bound on the contribution of a term occurrence. BM25+ modifies the term frequency component by adding a constant to it before multiplying it by the adjusted IDF term as follows:

$$R(D,Q) = \sum_{i=1}^{n} \left[\log\left(\frac{N+1}{n(q_i)}\right) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avg\_dl}}\right)} + \delta \right]$$

The b parameter controls the impact of document length on the score. The k1 parameter controls the saturation of the term frequency in the BM25 formula. We also replaced negative IDF values with a value ϵ to avoid divisions by zero. After trying different values, k1=1.2, b=0.75, ϵ=0.75, δ = 1 led to the best results.

All variations of the BM25 model managed to outperform other models, scoring 63% on the Kaggle competition.

3.3 SentenceTransformer

SentenceTransformer is a Python library designed for generating general-purpose sentence embeddings using transformer-based models. It simplifies the process of encoding sentences or short texts into fixed-length vectors that capture semantic information effectively. It is mostly based on the BERT model. After having computed the embeddings, we use our implementation of cosine similarity to rank the documents and extract the top 10 relevant ones.

4 METHOD

Here we describe the approach that led to the best results.

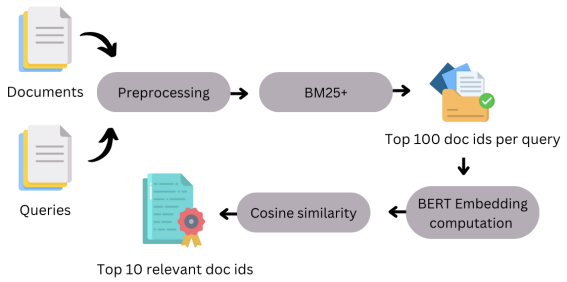

Figure 1: Retrieval methodology for Task 1.

Task1: Text Retrieval and Re-ranking. We employed the BM25+ model to retrieve the N most relevant documents per query and then re-rank them to get the top 10. For re-ranking, we used general-purpose pre-trained embeddings (384) using SentenceTransformer wrapper model 'all-MiniLM-L12-v2'[7]. This approach circumvents the need to compute similarities using the embeddings for the entire corpus, opting instead to focus solely on a subset of the top N documents provided by the BM25+ model. This approach enhances efficiency and resource utilization. Below, we present the observed improvements in scoring across various N selections. The following table presents the Kaggle scores for varying sizes of N, justifying our selection of $N = 100$.

	Top10	Top25 Embs	Top50 Embs	Top100 Embs
Score	0.6515	0.7140	0.74029	0.7577

Task2. Document Re-ranking. In the pursuit of the second task, we initially adhered to the approach applied in the primary task by executing the BM25+ model on the already acquired documents. However, the outcomes proved sub-optimal. Subsequently, we employed a strategy analogous to the secondary phase of task one, where we leveraged general-purpose pre-trained embeddings in conjunction with a customized similarity function to compute scores for each retrieved document. This refined approach returned a more efficient and superior re-ranking process, resulting in an approximate 20% enhancement compared to the results delineated in the preceding table.

Result Summary. Figure 2 presents the final Kaggle scores of all the methods we tried, for both tasks.

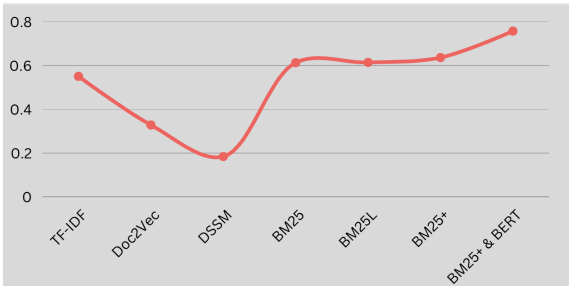

Figure 2: Performance evaluation for different retrieval models

5 CONCLUSION

In conclusion, our findings suggest that the efficacy of information retrieval methods is heavily contingent on the nature of the provided data. Surprisingly, conventional methods often perform better than the sophisticated ones. When a cost-effective classical approach like BM25 is employed for retrieval, it becomes feasible to construct a robust end-to-end system for document retrieval. For enhanced accuracy in obtaining top N documents, a good strategy involves leveraging different models, such as fine-tuned embedding generators or pre-trained embeddings tailored for general purposes. This targeted approach proves more efficient than applying these advanced models across the entire corpus.

REMARKS

We provide all our scripts, evaluation code, models, and datasets in our public GitHub repository. We supplied also the resources used for this project Here.

REFERENCES

[1] Lasmedi Afuan, Ahmad Ashari, and Yohanes Suyanto. 2019. A study: query expansion methods in information retrieval. In *Journal of Physics: Conference Series*, Vol. 1367. IOP Publishing, 012001.

[2] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[3] Jianfeng Deng Jeffrey Acero Heck Huang, Xiaodong. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. (2013).

[4] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query Expansion by Prompting Large Language Models. (2023).

[5] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 1188–1196.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[7] Dong Bao Yang Zhou Wang, Wui. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *arXiv preprint arXiv:2002.10957* (2020).

[8] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]