# < *RetrievalRangers* >: Wiki-Search Progress Report

Berk Güler
berk.guler@metu.edu.tr

Mustafa Barış Emektar
baris.emektar@metu.edu.tr

## Abstract

In this project, it is aimed to construct an ad-hoc information retrieval system using the $\text{ir}_{datasets}$ : $DPRWiki100$ dataset by utilizing PyTerrier framework. Also, NLTK libraries are employed since they include functions such as tokenization and stemming. Initially, BM25 ranking scheme will be employed the for retrieval. However, its performance will be enhanced by incorporating optimizations. Additionally, the system will feature a user interface, providing an intuitive experience for users interacting with the retrieved information.

## 1 Introduction

The information retrieval system will be designed to take a user query as input and return a set of relevant documents from a given collection. It aims to efficiently retrieve relevant information based on the user's search intent.

- Indexing: Creating an index of your document collection (in this case, the DPR Wiki100 dataset).
- Query Processing: Implementing query parsing, tokenization, stemming, and removing stop words.
- Term Frequency-Inverse Document Frequency (TF-IDF): Calculating TF-IDF scores for terms in documents and queries.
- Vector Space Models: Using cosine similarity or other similarity measures to rank documents based on query relevance.
- Document Preprocessing: Applying standard preprocessing steps (e.g., stop-word removal, stemming) to improve retrieval quality.

The dataset consists of 6.5K queries, 21M docs, and 980K query relevance judgments.

**The project has significance in terms of the benefits it will provide because it will significantly reduce the time and energy costs associated with locating the desired result among millions of documents. Also, it is engaging because being able to find the results typed in the query from millions of documents is at the heart of modern search engines.**

## 2 Related Work

No additional papers are used. However, the sample projects Pyterrier shared in the documentation were useful for the development of the system.

## 3 Methodology/System Architecture

The development of the project is done in Python. Pyterrier and NLTK libraries are used. Pyterrier library is used in the index, rank and performance evaluation phases while NLTK library is used for other operations like query transformation. **It was discovered that the query expansion method can be used on top of the methods described in the Proposal Report. This method can take the system one step further.**

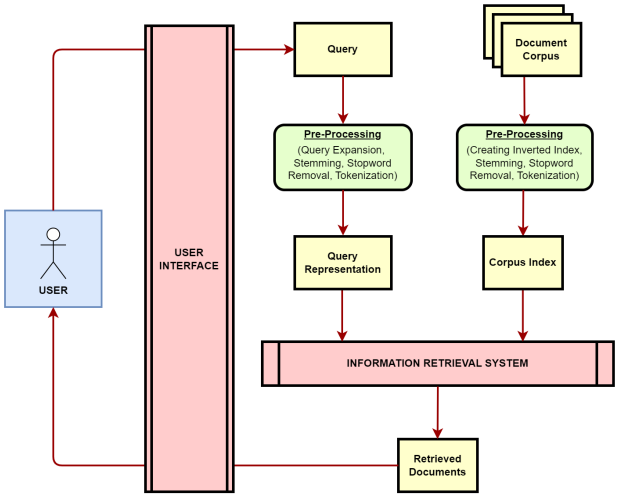


**Figure 1.** System Architecture and Data Flow of The System

Here are the design solutions for the specific steps:

- Data: The source of the is a wikipedia dump from 20 December, 2018, split into passages of 100 words. That amounts to almost 5 GB of data. It consists of 6.5K queries, 21M docs, and 980K query relevance judgments
- Index: Tokenization, stopword removal, stemming are the methods that are applied to corpus while creating the inverted index. The present version of the system employs the widely utilized positional index approach.
- Query Processing: Keyword queries are allowed currently. However; exact match queries, wildcard queries and range queries aren't implemented yet. Results are ranked with *transform* method of the Pyterrier library. Enhanced BM25 model is implemented. This model is improved using tokenization, stopword removal and stemming.

- User Interface: There will be a search box and a search button. Search results will be shown below the search box. Phrase queries, wildcard queries, and proximity operators will be allowed as far as Pyterrier architecture supports them. Providing snippets is not a priority. On the other hand, work will be done on this ability according to the busyness of the calendar. Results are ranked with *transform* method and the performance of the system is calculated using *Experiment* method of the Pyterrier library. The performance evaluation will be done utilizing the MAP (Mean Average Precision), nDCG (Normalized Discounted Cumulative Gain) metrics. The system will be developed using query expansion.

### 3.1 Milestones and Schedule

The Pyterrier library does not have many usage examples on the internet, so it took a lot of effort to use its functions. On the other hand, we are in track with the proposed calendar. Non-priority features like exact match queries, wildcard queries and range queries may not be implemented due to time constraints.

## 4 Experiments and Evaluation

### 4.1 Data set

Yes we acquired the dataset that we need by using *get_dataset* function from the Pyterrier library. Then created index and prepared corpus by using *IterDictIndexer* and *get_corpus_iter* functions.

The numerical details of the dataset are shown below:

- Number of documents: 21015324
- Number of terms: 4023297
- Number of postings: 1000890089
- Number of fields: 2
- Number of tokens: 1286954378

These values are obtained using the *getCollectionStatistics()* method of Pyterrier.

### 4.2 Inverted index

Pyterrier and NLTK libraries for Python are utilized. The inverted indexes with original BM25 and modified BM25 are obtained.

| | name | AP | nDCG@20 |
|---|---|---|---|
| 0 | BR(BM25) | 0.208434 | 0.269578 |

**Figure 2.** Performance Result of the Modified BM25

We created the inverted index using BM25 and applying the stemming, tokenization and stopword removal. Also, queries are preprocessed with query transformations. MAP and nDCG metrics are used.

Since the DPR-W100 dataset is huge, we encountered some problems while evaluating the original BM25 performance. When we tried the same implementation with small dataset, we observed both original and modified BM25 performances. The performance of the modified BM25 performance is better than the original BM25 performance.

### 4.3 Things to do

- Original BM25 performance will be evaluated and showed the difference between the modified one.
- User interface will be designed and implemented.
- Phrase queries, wildcard queries, and proximity operators will be examined and applied if possible.
- Snippet of document will be shown to user if possible.

## References

PyTerrier Github