Ben-Gurion University of the Negev
Faculty of Engineering Science
Department of Information Systems Engineering

Wikipedia Search Engine

Information Retrieval Project

Submitted by

Stav Nimrod
E-mail: stavnim@post.bgu.ac.il

Yuval Gorelik
E-mail: goreliy@post.bgu.ac.il

1 Links

- **GitHub**: Repository Link
- **Google Storage**: Bucket Link

2 Preprocessing

In this section, we will describe the preprocessing we made to be used later in the experiments.

- **Title Inverted Index**: We used pyspark to load all title parts in the english wikipedia stored in parquets. We tokenized the titles- removed symbols, removed english stopwords, removed corpus stopwords, swapped into lower case letters. We saved a binarized posting list for each token. Our title index was composed of postings locations ('index.posting-locs'), document frequency for each token('index.df'), and finally we also calculated and added the tokenized title lengths ('index.doc-len') which for a document id given as a key, returns the tokenized title length of the document.

- **Body Inverted Index**: We used pyspark to load all body parts in the english wikipedia stored in parquets. We tokenized the body texts- removed symbols, removed english stopwords, removed corpus stopwords, swapped into lower case letters. We saved a binarized posting list for each token. Our body index was composed of postings locations('index.posting-locs'), document frequency for each token('index.df'), and finally we also calculated and added the tokenized body lengths('index.doc-len') which for a document id given as a key, returns the tokenized body length of the document.

- **Dictionary (doc-id:title)**: We calculated and pickled a dictionary that for a document id given as a key, will return the document's title(without tokenization). This dictionary is used at the last part of our engine, in order to return the title of the top N documents, sorted by similarity to the query.

- **Dictionary (doc-id:page-rank)**: We calculated and pickled a dictionary that for a document id given as a key, will return the document's PageRank. This dictionary is used in some of the experiments.

3 Experiments

In this section, we will describe the major experiments we conducted, in the evaluation section we will explain how we evaluated them too.

- **Experiment V1:** Our engine started as a simple calculation of term frequency, retrieving the documents that their tokenized title shared the most tokens with the tokenized query.
- **Experiment V2:** Next, we calculated cosine-similarity, between the document titles and the query.
- **Experiment V3:** We calculated BM25, between the document titles and the query. For this calculation we utilized the 'index.doc-len' dictionary described earlier. We also explored various combinations for BM25 hyper-parameters, and used the best among them.
- **Experiment V4:** We combined similarity models, we gave a ratio for each similarity model. We used title index only, 0.7*BM25 +0.3*PageRank combination. We explored different ratios too.
- **Experiment V5:** Calculation of term frequency, retrieving the documents that their tokenized body text shared the most tokens with the tokenized query.
- **Experiment V6:** We calculated cosine-similarity, between the document body texts and the query.
- **Experiment V7:** We calculated BM25, between the document body texts and the query. For this calculation we utilized the 'index.doc-len' dictionary described earlier. We also explored various combinations for BM25 hyper-parameters, and used the best among them.
- **Experiment V8:** We combined similarity models, we gave a ratio for each similarity model. We used body index only, 0.7*BM25 +0.3*PageRank combination. We explored different ratios too.
- **Experiment V9:** We combined indices and similarity models. For tokenized queries with length of more than 3, we used simple term frequency (like in V5) on body index. If the tokenized query length was 3 or below, we used 0.7*BM25 +0.3*PageRank combination on title index (like in V4).

4 Evaluation

In this section we will explain the evaluation of the experiment and key findings.

Table 1: Experiment's Quality and Runtime

Experiment Version	**Quality**	**Runtime (s)**
Version 1	0.206	3.532
Version 2	0.067	1.599
Version 3	0.157	3.022
Version 4	0.229	3.279
Version 5	0.038	4.694
Version 6	0.013	7.251
Version 7	0.112	4.827
Version 8	0.141	4.974
Version 9	**0.272**	**1.734**

- **Engine Retrieval Quality:** We evaluated the quality of the experiments by calculating the harmonic mean of precision@5 and F1@30.

$$\text{Harmonic Mean of } P_5 \text{ and } F1_{30} = \frac{2 \times P_5 \times F1_{30}}{P_5 + F1_{30}}$$

- **Engine Retrieval Time:** We also evaluated our experiments by the mean runtime that took the engine to retrieve results.

- **Qualitative Evaluation and Key Findings:** Initially, our approach involved employing a straightforward Term Frequency (TF) model focused on the title inverted index (Experiment V1). This strategy yielded better results for brief queries such as 'neuroscience' (achieving a score of 0.641) and 'nanotechnology' (with a score of 0.583). Nonetheless, it was less effective for more complex or lengthy queries like 'What is the meaning of the term "Habeas Corpus"?' and 'Who is known for proposing the heliocentric model of the solar system?', both of which received a quality score of 0.0. Motivated by this, we sought to improve the effectiveness of our title inverted index, leading us to undertake experiments V2, V3, V4. As documented in table 1, the average quality scores remained consistent across these experiments, with individual queries receiving comparable scores. Subsequently, our investigation extended to the body inverted index in experiments V5 through V8. As indicated in table 1, relying solely on the body index typically resulted in inferior performance compared to the title index. However, a slight enhancement was observed for more extended queries such as 'What is the meaning of the term "Habeas Corpus"?' (scoring 0.21) and 'Who is known for proposing the heliocentric model of the solar system?' (scoring 0.32). In our final iteration, experiment V9, we integrated both indices, applying the title index to shorter tokenized queries (with three tokens or fewer) and the body index to longer ones (exceeding three tokens). This integration led to an average quality score of 0.27 in V9, as detailed in table 1.