Final Project Report

In this project, we implemented a search engine for the English Wikipedia corpus using classical Information Retrieval techniques.
The system supports free-text queries over the full English Wikipedia dump provided in the course.

Three inverted indexes were constructed over article body, title and anchor text, using tokenization, stopword removal, lowercasing, and Porter stemming.

Additionally, we constructed a map that links page id to title to enable the engine to show the titles of the results and a map that links an id to page views to implement the required get_pageview function.
All index data is stored in Google Cloud Storage (GCP), and no external services are used at query time.

We implemented a tf-idf weighted cosine similarity search over the body index, and binary searches over the title and anchor indexes.
Following empirical evaluation, BM25 over the body index was selected as the final ranking approach for the search function due to its superior retrieval quality and speed.

The system consists of a backend responsible for retrieval and ranking logic, and a Flask-based frontend exposing the main engine from /search API endpoint, and additional /search_body, /search_title, /search_anchor and /get_pageview endpoints.

The system is divided into two main components:

- Backend – responsible for query preprocessing, candidate retrieval from the inverted index, and document ranking.
- Frontend – a Flask-based API enabling external access to the search engine.

a. Noa Segev - 210025631 - senoa@post.bgu.ac.il

Niel Melekh - 215329202 - nielm@post.bgu.ac.il

Shaked Farjun - 212639892 - farjunsh@post.bgu.ac.il

b. Github - https://github.com/nielmelekh/IR-Final-Project
c. Google Storage Bucket – ir_project_2025
d. Shown in the end of the file

e. Experiments and Evaluation:

We conducted a series of controlled experiments to compare different ranking methods for the Wikipedia search engine.
All experiments were performed over the same inverted index, constructed on the article body text only, in order to isolate the effect of the ranking function.
We evaluated two main retrieval models:
Baseline model: TF-IDF weighting with cosine similarity
Final model: BM25 ranking function applied over the same body inverted index
Both systems used:

- The same document collection (English Wikipedia corpus)
- The same preprocessing pipeline (tokenization, stopword removal, and Porter stemming)
- The same query set (queries_train.json)
- The same execution environment, without any form of result caching
- This setup ensured a fair and direct comparison between the ranking approaches.

Evaluation Metrics

System performance was evaluated using the following metrics:
Average Precision@10 (AP@10) - used to measure retrieval quality at the top-ranked results, which is critical for user-facing search engines.
Query processing time - measured per query and summarized using:

- Average query time
- Median query time
- Maximum query time

These metrics allowed us to evaluate both retrieval effectiveness and system efficiency, in accordance with the project requirements.

Results and Key Findings

The experimental results demonstrate a clear advantage of the BM25-based ranking model over the baseline TF-IDF approach.

The baseline TF-IDF system achieved reasonable performance but showed sensitivity to document length and raw term frequency, which negatively affected ranking quality for several queries.

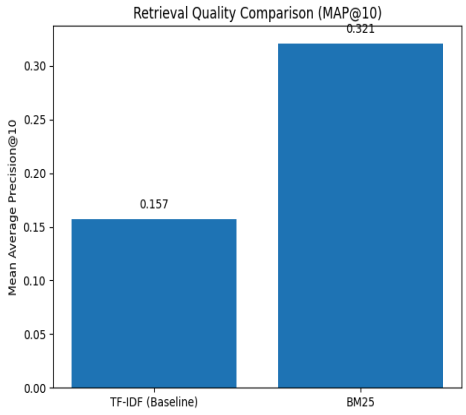
The BM25-based system consistently outperformed the baseline in terms of retrieval quality, achieving more than double the Mean Average Precision@10 on the evaluation query set.

Despite using a more complex scoring function, the BM25 system maintained acceptable query processing times, all well below the required threshold of 35 seconds per query.

Overall, the experiments confirm that BM25's probabilistic weighting and built-in document length normalization provide a more robust and effective ranking mechanism than classical TF-IDF with cosine similarity.

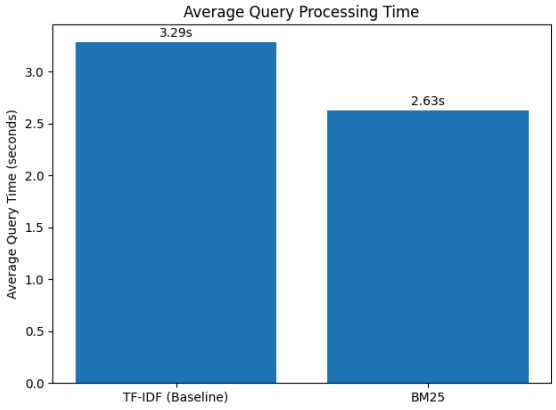
f.

Figure 1 presents the retrieval quality (Mean Average Precision@10) for each major version of the search engine. The BM25-based model significantly outperforms the TF-IDF baseline.



g.

Figure 2 presents the average query processing time for each major version of the search engine. While BM25 introduces slightly more computation, it remains well within the efficiency constraints



h. Successful Query Example

Query: "Printing press invention Gutenberg"

The BM25-based system returned highly relevant articles at the top ranks, including the main article on Johannes Gutenberg and the invention of the printing press. Most of the top 10 results were directly relevant to the query, resulting in a high AP@10 score.

Explanation:

BM25 effectively balances term frequency and document length, prioritizing documents where query terms are frequent but not excessively long, leading to superior ranking quality.

Less Successful Query Example

Query: "Television invention broadcast media"

Observed Issue:

Both retrieval models struggled to consistently rank the most relevant documents at the top positions.

This behavior is reflected in a relatively low AP@10 score for this query.

Root Cause:

The query spans multiple related concepts, while the system relies solely on body text and does not incorporate title information, anchor text, or popularity-based signals.

Potential Improvement:

Future work may include incorporating title-based scoring, anchor text, PageRank, or page view statistics to improve performance for broad or ambiguous queries.

.

Appendix

In the inverted indexes there are too many files to list all of them individually so we will list them by type

(number of files, type, total size)

- Body index in ir_project_2025/body100/

  1 .pkl 125M

  124 .pickle 9.0M

  2732 .bin 5.0G

- Title index in ir_project_2025/title1/

  1 .pkl 22M

  124 .bin 52M

  124 .pickle 13M

- Anchor index in ir_project_2025/anchor4/

  1 .pkl 96M

  124 .pickle 61M

  531 .bin 901M

The other stored data is much smaller in size so we will list all of the files

- Wiki id to title map in ir_project_2025/id_to_title_dict/

  168.88 MiB gs://ir_project_2025/id_to_title_dict/id_to_title.pkl

- Wiki id to page views parquet in ir_project_2025/pageviews_aug_2021/

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00000-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00001-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.24 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00002-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00003-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00004-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.26 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00005-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00006-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.24 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00007-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.24 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00008-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.24 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00009-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.25 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00010-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

3.26 MiB
gs://ir_project_2025/pageviews_aug_2021/part-00011-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

41.13 KiB
gs://ir_project_2025/pageviews_aug_2021/part-00012-3a5ce8cf-6afe-4835-820c-bd4b19074563-c000.snappy.parquet

39.03 MiB total