Information Retrieval Final Project Report

Search Engine for Personal Documents

Harshal Gajjar
Section: 601.666
hgajjar1@jhu.edu

May 16, 2025

Code Repository

The complete codebase for this project is available at: https://github.com/harshal301002/Information-Retrieval/tree/master

---

Contents

1 Project Summary **2**

2 User Guide **2**

3 Libraries and Tools Used **2**

4 System Architecture **3**

5 Achievements **3**

6 Limitations **4**

7 Future Work **4**

8 Evaluation Results **5**

9 Conclusion **6**

---

1 Project Summary

This report details the design, implementation, and evaluation of a local information retrieval (IR) system. The system is engineered to index and search personal '.txt' and '.md' document collections utilizing TF-IDF, BM25, and dense semantic retrieval methodologies. Key functionalities encompass query expansion, snippet generation, and a comprehensive evaluation suite for comparing the efficacy of various retrieval strategies.

2 User Guide

- Execution of the command-line search engine is performed using:

```
python run_search.py
```

- Evaluation of retrieval methods based on ground-truth data is initiated via:

```
python eval.py
```

- Users can select from TF-IDF, BM25, and Dense retrieval methods, with an option to enable query expansion.

3 Libraries and Tools Used

This project makes use of a range of open-source Python libraries to implement key components of the information retrieval pipeline. All tools used are clearly documented below:

- `nltk` (Natural Language Toolkit): Used for text preprocessing tasks including tokenization, stopword removal, and lemmatization. Also supports query expansion through integration with the WordNet lexical database.
- `scikit-learn`: Provides vectorization and similarity computation tools. Specifically used for:
  - `TfidfVectorizer` for TF-IDF matrix construction
  - `cosine_similarity` to compute query-document similarity
- `rank-bm25`: A Python implementation of the BM25 algorithm, used for traditional probabilistic document ranking.
- `sentence-transformers`: Used to generate dense vector embeddings for both queries and documents via a pre-trained MiniLM model. Enables semantic search based on cosine similarity in embedding space.
- `matplotlib` and `pandas`: Utilized for result analysis and visualization. Supports creation of evaluation charts comparing retrieval methods with and without query expansion.
- `numpy`: Used for numerical operations, especially in computing ranking scores and similarity measures.

- `os`, `glob`, `re`, `pathlib`: Standard Python libraries used for file I/O, recursive directory traversal, text matching, and file path management.

All external libraries used are open-source and were integrated into the system with clear attribution and appropriate API usage. No proprietary or licensed third-party code was included.

4 System Architecture

The system architecture is founded upon a modular and extensible IR pipeline, which delineates document ingestion, preprocessing, indexing, retrieval, and evaluation into distinct, well-defined components. A detailed description of each core module follows:

- **FileLoader:** This component recursively scans a user-specified directory for valid ‘.txt‘ and ‘.md‘ files. It ingests the content, utilizing relative file paths as document identifiers. This approach facilitates the management of hierarchically organized notes and demonstrates scalability for corpora comprising several hundred documents.
- **TextProcessor:** Each document undergoes tokenization, lowercasing, and lemmatization. Standard stopwords are eliminated using NLTK’s corpus. Lemmatization promotes consistent term representation (e.g., “earning” and “earned” map to “earn”), thereby enhancing indexing and matching precision. This preprocessing pipeline is applied symmetrically to user queries to ensure consistency.
- **RetrievalEngine:** This unified engine accommodates three distinct retrieval models: (1) **TF-IDF + Cosine Similarity**, (2) **BM25 Scoring**, and (3) **Dense Embedding Similarity** using MiniLM. It abstracts the underlying variations in scoring mechanisms through a common API, enabling experimentation and comparative analysis without necessitating modifications to the interface code.
- **QueryExpander:** This module facilitates optional query expansion using WordNet. For each query term, a configurable number of synonyms are appended. This process aims to enhance recall by encompassing documents that employ alternative yet semantically related terminology, proving particularly beneficial for sparse term-based models such as BM25 and TF-IDF.
- **SnippetGenerator:** For each top-ranked result, a snippet is extracted by identifying the first occurrence of any query term and returning a constrained window of surrounding text. In instances where no match is found, a fallback mechanism presents the initial segment of the document. This feature enhances usability by providing users with an immediate indication of document relevance.
- **Evaluator:** A dedicated module facilitates evaluation against a ground-truth dataset of manually labeled relevant documents. It calculates standard IR metrics, including Precision@K, Mean Reciprocal Rank (MRR), and nDCG@K, across multiple queries. This component underpins the quantitative comparison of different retrieval configurations and supports rigorous IR experimentation.

5 Achievements

- **Development of a Modular and Extensible Retrieval Engine:** A significant achievement was the design and implementation of a scalable retrieval engine supporting multiple ranking models: TF-IDF with cosine similarity, BM25 scoring, and dense vector similarity using MiniLM sentence embeddings. The engine's modularity permits the integration of new retrieval methods or configurations with minimal codebase modifications.

- **Incorporation of WordNet-Based Query Expansion:** The system integrates query expansion functionality leveraging WordNet synonyms. This feature allows for the semantic broadening of user queries through the addition of related terms, which demonstrably improves recall in numerous instances, particularly for keyword-centric ranking models like BM25. This capability is optional and can be controlled on a per-query basis via the interface.
- **Generation of Snippets with Context Highlighting:** To enhance the interpretability of search results, the system extracts a concise snippet from each returned document that contains the query terms or their expanded variants. This provides contextual evidence of relevance, analogous to the user experience offered by established search engines.
- **Establishment of an Empirical Evaluation Framework and Subsequent Analysis:** A benchmark dataset comprising 15 realistic queries with manually annotated relevant documents (ground truth) was curated. Utilizing this benchmark, the system performs automated evaluations employing Precision@5, Mean Reciprocal Rank (MRR), and nDCG@5. These evaluations were conducted both with and without query expansion to facilitate a comparative analysis of retrieval effectiveness across different methods.

6 Limitations

Although the system demonstrates robust modularity and quantifiable retrieval performance, several limitations persist that may affect precision, adaptability, and scalability in practical applications:

- **Potential for Noise Introduction via Query Expansion:** The WordNet-based query expansion mechanism operates on lexical synonyms without contextual disambiguation. This can occasionally dilute query intent by introducing irrelevant or overly broad terms, particularly affecting dense semantic models where expanded terms may distort vector similarity. A feedback mechanism to filter detrimental expansions is not currently implemented.
- **Suboptimal Performance of Dense Retrieval due to Lack of Domain-Specific Fine-Tuning:** Sentence embeddings employed in dense retrieval are generated using a general-purpose MiniLM model. These embeddings have not been adapted to the specific linguistic characteristics, structure, or terminology of the document corpus (e.g., academic notes, domain-specific jargon). Consequently, semantic similarity scores may be suboptimal for this domain, potentially leading to false positives or missed relevancies.
- **Limited Scope of Supported Query Types:** The system is optimized for short keyword queries. It does not offer structured support for boolean, phrase, proximity, or natural language questions, nor does it attempt to classify or parse diverse query types for the application of adaptive retrieval strategies.

7 Future Work

Several promising avenues for future development could further enhance retrieval quality, usability, and domain adaptability:

- **Integration of Rocchio Relevance Feedback:** Incorporating user-marked relevant documents into query reformulation, via the Rocchio algorithm, presents an opportunity to significantly enhance ranking accuracy, particularly for multi-turn or exploratory search tasks.
- **Implementation of Query Intent Classification:** Automated detection of query intent (e.g., informational versus navigational) would enable the system to dynamically select the most appropriate retrieval model (e.g., BM25 for factual queries, dense retrieval for exploratory queries).
- **Development of Topic Clustering and Labeling Capabilities:** Unsupervised topic modeling or clustering techniques (e.g., KMeans, LDA) could facilitate the grouping of documents by content, enabling topic-specific search, filtering, or personalized organization of notes.

8 Evaluation Results

Dataset and Queries

The evaluation was conducted using a manually annotated set of 15 realistic queries, each associated with known relevant documents (ground truth). The metrics employed for evaluation were Precision@5, MRR, and nDCG@5.

Metrics Summary

Method	QE	P@5	nDCG@5	MRR
TF-IDF	No	0.2533	0.2735	0.3911
TF-IDF	Yes	0.2267	0.2728	0.4856
BM25	No	0.3733	0.4315	0.5889
BM25	Yes	0.4400	0.4990	0.6500
Dense	No	0.3067	0.3602	0.5356
Dense	Yes	0.1867	0.2495	0.4167

Table 1: Comparative performance of retrieval methods with and without Query Expansion (QE).

Visualization

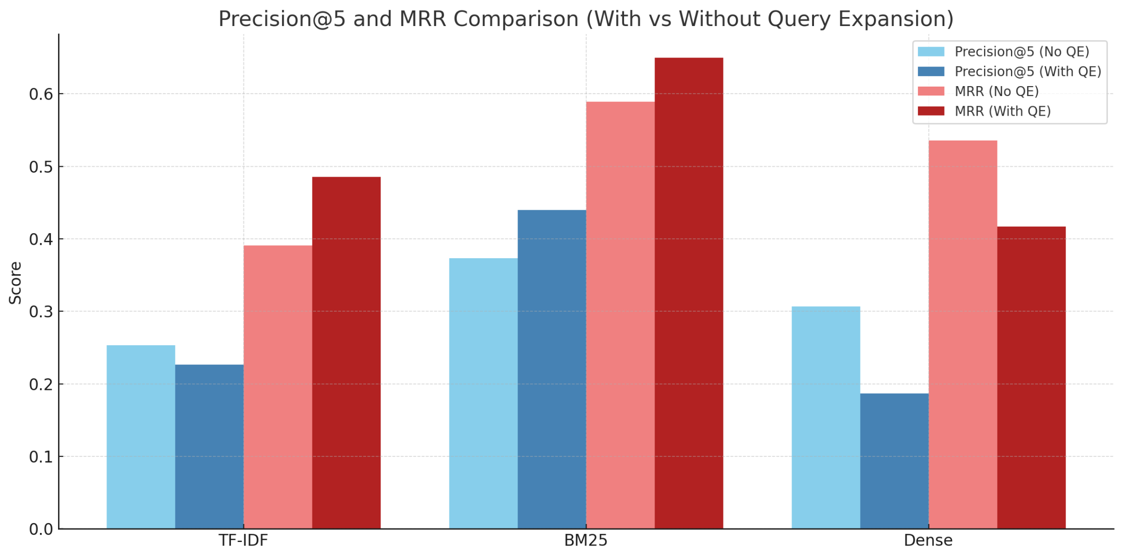

Figure 1: Visual comparison of retrieval method performance (P@5, nDCG@5, MRR).

9 Conclusion

This project presented the implementation and empirical comparison of multiple information retrieval techniques applied to personal text collections. The results indicate that BM25, augmented with query expansion, yielded the most favorable performance overall, thereby confirming its robustness for keyword-based retrieval tasks. Dense retrieval models exhibited sensitivity to noise introduced by query expansion, underscoring the need for future investigations into domain-specific fine-tuning for these approaches.