CSI4107

Info Retrieval & Internet

Assignment 1
Information Retrieval System

Submitted by:

Anas Taimah # 300228842

Sami Diouf # 300296671

John Ghai Manyang # 300272467

February 15, 2026

University of Ottawa

Introduction

In this project, we built our own information retrieval system using the SciFact dataset. Our goal was to understand how search systems work behind the scenes by implementing the core components ourselves. When a user types a query into the search engine, the system identifies the most relevant documents and ranks them accordingly. To do this, we followed a three-step approach: preprocessing, indexing, and retrieval. First, we cleaned the text by tokenizing it, removing stopwords, filtering out punctuation and numbers, and applying stemming. Then, we built an inverted index to store term frequencies and compute TF-IDF weights. Finally, we ranked documents using cosine similarity between query and document vectors.

Link to GitHub repository: https://github.com/AATaimah/Information-Retrieval

Dataset

We used the SciFact dataset, a scientific fact-checking benchmark containing a document corpus, query claims, and relevance judgments (qrels). The corpus is provided in scifact/corpus.jsonl, where each document has an _id, title, and text. Queries are provided in scifact/queries.jsonl, where each query has an _id and a claim text. Relevance judgments are provided in scifact/qrels/train.tsv and scifact/qrels/test.tsv, indicating which documents are relevant to each query.

For this assignment, we used:

- the full SciFact corpus (5,183 documents),
- all queries from queries.jsonl,
- odd-numbered query IDs for testing (as required by the assignment setup).

This dataset is appropriate for evaluating retrieval because it includes both realistic scientific claims and judged relevant documents, allowing us to measure ranking quality using standard IR metrics such as MAP.

Methods and setup :

Step 1 - Preprocessing

The preprocessing stage was implemented to normalize and clean both the document corpus and the query set before indexing and ranking. This step ensures consistency in text representation and improves the effectiveness of term-based retrieval using TF-IDF. The preprocessing pipeline was implemented in preprocessing.py.

Step 2 - Indexing

In this step, we built an inverted index for the SciFact corpus using the Vector Space Model (VSM), mapping each term to a posting list of (doc_id, term frequency). We computed document frequency (df), inverse document frequency (idf), and TF-IDF document vector lengths required for cosine similarity normalization. All index structures were saved to JSON files so they could be reused directly during retrieval and ranking in Step 3. The final vocabulary size after preprocessing was 18,799 unique terms.

Step 3 - Retrieval and Ranking

In the final step, we loaded the prebuilt index and processed each query using the same preprocessing pipeline as the corpus. We then built a TF-IDF query vector and computed cosine similarity scores against indexed documents. Documents were ranked in descending order of similarity score, and the top results were written to the output run file in TREC format.

Integration & Evaluation

Setup

We evaluated our TF-IDF cosine retrieval system on the SciFact dataset. The index was built over the HEAD + TEXT fields of the corpus (N = 5,183 documents), and queries were taken from scifact/queries.jsonl.
Following our pipeline, we retrieved up to the top 100 documents per test query and saved results in TREC format (query_id Q0 doc_id rank score run_tag).

Test Query Selection

As required in our setup, we used odd-numbered query IDs (1, 3, 5, ...) from queries.jsonl, giving 547 retrieval queries in total.

Evaluation Protocol

We evaluated it with trec_eval using SciFact test qrels. Since scifact/qrels/test.tsv is TSV with a header, we first converted it to standard TREC qrels format before evaluation:

```
awk 'NR>1 {printf "%s 0 %s %s\n", $1, $2, $3}' scifact/qrels/test.tsv > /tmp/scifact_test.qrels
./trec_eval/trec_eval -m num_q -m map /tmp/scifact_test.qrels Results_title_only
./trec_eval/trec_eval -m num_q -m map /tmp/scifact_test.qrels Results_title_plus_text
```

Results

We report MAP (Mean Average Precision), which measures how well relevant documents are ranked near the top across queries.

- num_q = 153 (judged test queries used by trec_eval)
- title_only MAP = 0.2508
- title_plus_text MAP = 0.4116

Interpretation of Results

The comparison between the two required query configurations shows a clear performance difference. Using title_only, the system achieved MAP = 0.2508, while using title_plus_text, it achieved MAP = 0.4116 on the same judged test set (num_q = 153). This indicates that including the full query text provides substantially more useful matching terms than using only short title-style queries, leading to better ranking quality.

The value num_q = 153 means trec_eval evaluated only queries with available relevance judgments in the test qrels. Therefore, the MAP values reflect performance on judged queries only, not all retrieved queries. Overall, the title_plus_text run is the better configuration and is selected as the best run for submission.

Task Distribution

- Sami: preprocessing of data, report
- John: building the inverted index, report
- Anas: retrieval, ranking, and evaluation of results, report

Conclusion

In this assignment, we successfully implemented a complete Information Retrieval system using the SciFact dataset. The system includes preprocessing, inverted index construction, and document ranking using TF-IDF cosine similarity. We indexed 5183 documents and constructed a vocabulary of 18,799 unique terms.

Our retrieval model achieved a MAP score of 0.4116, demonstrating strong ranking performance on the judged test queries. Through this project, we gained practical experience with the core components of classical search systems and vector space models. This work provides a strong foundation for exploring more advanced retrieval methods, such as BM25 and neural information retrieval, in future coursework.