Assignment 2: Vector Space Model and Probabilistic Retrieval

Information Retrieval and Web Search (COL764/COL7341)

Naveeta Maheshwari (2024AIZ8309)
Sagar Singh (2025SIY7574)

September 11, 2025

Introduction

The primary objective of this assignment is to extend the functionality of a basic information retrieval system by implementing sophisticated ranked retrieval models. The project encompasses four main tasks: updating the inverted index structure, implementing a boolean phrase search, and building two ranked retrieval models—Vector Space Model (VSM) and Okapi BM25. An optional Task 5 introduces pseudo-relevance feedback for VSM. The entire system is implemented from scratch in Python, adhering to the specified library constraints. This report details the step-by-step implementation, following the logical flow from data preprocessing to the final retrieval models.

Task 0 & 1: Tokenization and Inverted Index Construction

The foundation of any retrieval system is its index. This phase involved generating a vocabulary and then building a comprehensive inverted index with all necessary statistics for the subsequent tasks.

Tokenization and Vocabulary Generation

As per the assignment requirements, tokenization was performed using the `spaCy` library. To ensure efficiency and consistency, a blank English pipeline (`spacy.blank("en")`) was used, which provides a fast tokenizer without the overhead of other NLP components. The implementation in `tokenize_corpus.py` processes each document, tokenizes all text content, collects a unique set of terms, and saves them sorted to `vocab.txt`.

Inverted Index Structure and Statistics

The `build_index.py` script constructs the inverted index. The final index, saved as `index.json`, stores for each term:

- **df**: The document frequency of the term.
- **postings**: A dictionary mapping document IDs to their respective term data, which includes:
  - **tf**: The term frequency within that document.
  - **pos**: A sorted list of 0-indexed positions of the term in the document.

Pre-computation of Statistics for Retrieval Models

To optimize query-time performance, additional statistics for VSM and BM25 are pre-calculated.

VSM : The length (Euclidean norm) of each document vector is pre-calculated using a log-TF-IDF scheme:

$$w_{t,d} = (1 + \log_{10}(\mathrm{tf}_{t,d})) \times \left(\log_{10}\left(\frac{N+1}{\mathrm{df}_t + 1}\right) + 1\right)$$

The resulting vector lengths ($\|\mathbf{d}\| = \sqrt{\sum_{t \in d} w_{t,d}^2}$) are stored in `vsm.json`.

BM25 Statistics: The length of each document $|d|$ and the average document length (`avgdl`) are computed and stored in `bm25.json`.

Task 2: Boolean Phrase Search

The phrase search functionality finds documents containing an exact sequence of query terms. This is a Boolean retrieval task and does not involve scoring. The search proceeds in two steps:

1. Candidate documents that contain all query terms are identified by intersecting postings lists.
2. For each candidate, the positional information is checked to verify that the terms appear consecutively.

Task 3: Vector Space Model (VSM) Retrieval

Query Processing and Vector Construction

The query is tokenized and a query vector $\mathbf{q}$ is constructed using the same log-TF-IDF weighting scheme as the documents.

Scoring and Ranking

An accumulator-based approach calculates the dot product between the query vector $\mathbf{q}$ and the document vector $\mathbf{d}$. The resulting score is then normalized to obtain the final cosine similarity:

$$\cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \, \|\mathbf{d}\|}$$

Evaluation and Results

The VSM model was evaluated using the queries provided and the relevance judgments. The results in Table 1 demonstrate the classic precision-recall trade-off: As more documents are retrieved (increasing k), the recall improves while the precision decreases slightly.

Top-k	Precision	Recall	F1-score
5	0.2840	0.0027	0.0053
10	0.2840	0.0053	0.0105
20	0.2810	0.0105	0.0203
50	0.2464	0.0231	0.0422

Table 1: VSM Performance Metrics at different k values.

Task 4: Okapi BM25 Retrieval

Hyperparameters and Scoring Formula

The standard hyperparameters $k_1 = 1.2$ and $b = 0.75$ are used. The score of a document D for a query Q is:

$$\text{Score}(D, Q) = \sum_{q_i \in Q} \text{IDF}(q_i) \cdot \frac{\text{tf}(q_i, D) \cdot (k_1 + 1)}{\text{tf}(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

The IDF component is calculated as:

$$\text{IDF}(q_i) = \log\left(1 + \frac{N - \text{df}(q_i) + 0.5}{\text{df}(q_i) + 0.5}\right)$$

Evaluation and Results

The BM25 model was evaluated on the same basis as the VSM. The results in Table 2 show that BM25 achieves substantially higher precision than VSM, indicating its superior ranking effectiveness for this dataset.

Top-k	Precision	Recall	F1-score
5	0.7040	0.0066	0.0131
10	0.6540	0.0123	0.0241
20	0.5940	0.0223	0.0429
50	0.5244	0.0492	0.0899

Table 2: BM25 Performance Metrics at different k values.

Task 5: Pseudo-Relevance Feedback for VSM

Objective and Method

Pseudo-relevance feedback (PRF) was implemented on top of the VSM system using the Rocchio algorithm. The top $r = 10$ documents from the initial retrieval were assumed to be relevant, and the query vector was updated as:

$$\vec{q}' = \alpha \cdot \vec{q} + \frac{\beta}{|D_r|} \sum_{\vec{d_r} \in D_r} \vec{d_r}$$

with $\alpha = 1.0$ and $\beta = 0.75$. Then, this new query vector was used for a second retrieval round.

Evaluation and Results

The results of PRF-enhanced retrieval are summarized in Table 3. Compared to baseline VSM, precision shows slight improvements at smaller values of k, but recall remains very low, limiting the overall F1 score.

Top-k	Precision	Recall	F1-score
5	0.3200	0.0030	0.0059
10	0.3020	0.0057	0.0111
20	0.3010	0.0113	0.0218
50	0.2784	0.0261	0.0477

Table 3: Performance of VSM with Pseudo-Relevance Feedback.

Discussion

PRF led to a marginal improvement in precision at small cut-offs (e.g., $k = 5$), but recall gains were negligible, which suppressed the F1-score. While execution time increased moderately, the approach shows promise if combined with better query expansion strategies or adaptive weighting of feedback terms.

Conclusion

This project successfully implements a comprehensive information retrieval system featuring boolean phrase search, VSM, BM25, and an additional extension with pseudo-relevance feedback. By pre-computing and storing document statistics during indexing, both VSM and BM25 were efficiently implemented and evaluated.

The experiments confirmed the known trade-offs: VSM provides a baseline vector space retrieval, while BM25 substantially improves ranking quality and achieves higher precision. With the pseudo-relevance feedback extension, the system was able to further enhance recall and F1-score by expanding the query with terms from the top retrieved documents. This demonstrates how feedback-based methods can compensate for vocabulary mismatch between queries and documents.

Overall, the assignment provided hands-on experience in designing, implementing, and evaluating core IR models, while also exploring query expansion techniques that bridge into modern retrieval approaches. The pseudo-relevance feedback experiment highlights the potential of iterative retrieval strategies for improving system effectiveness beyond static models.