Information Retrieval and Search Engines

Final Project report

Team members:

Omar Ibrahim Ahmed	**2206209**
Marwan Gaber Ramadan	**2206167**
Youssef Tamer Mohamed Ali	**2206208**
Ibrahim Saber Ibrahim	**22011431**
Ahmed Adel Elsayed	**20221453232**

Under supervision of:

Dr Saleh Mesbah

Faculty of Computers and Data Science

Academic Year: 2025–2026

Information Retrieval and Text Analytics: Building a Retrieval Model with Preprocessing and Visualization

---

1. Introduction

1.1 Project Scope and Objectives

This project aims to design and implement a complete **Information Retrieval (IR)** system capable of processing raw text documents, converting them into structured representations, retrieving relevant documents based on user queries, and evaluating the performance of different retrieval models.

The objectives of this project are:

- To collect and prepare a real-world text dataset.
- To perform preprocessing such as tokenization, stopword removal, and lemmatization.
- To represent text numerically using BoW and TF–IDF vectorization.
- To implement and compare two retrieval models:
  - **Vector Space Model (VSM)** using cosine similarity
  - **Boolean Retrieval Model**
- To evaluate retrieval performance using Precision, Recall, and MAP.
- To perform visualization and text analytics through word clouds, frequency distribution, and topic modeling.
- To analyze results and summarize key findings.

This project demonstrates the end-to-end process of building a mini-search engine using classical IR models.

1.2 Overview of Information Retrieval (IR) Systems

Information Retrieval systems allow users to find relevant information from large document collections. IR is used in:

- Search engines (Google, Bing)
- Digital libraries (IEEE Xplore, ACM Digital Library)
- E-commerce search systems
- Question-answering systems
- Recommendation engines

A standard IR pipeline follows:

```
User Query → Preprocessing → Indexing → Retrieval Model → Ranking → Results
```

Classical IR relies on models such as:

- **Boolean Retrieval** — exact term matching
- **Vector Space Model (VSM)** — cosine similarity between document and query vectors
- **BM25** — probabilistic ranking (not implemented in this project per requirement)

This project focuses on **VSM** and **Boolean retrieval**, integrating them into a query-processing mechanism.

---

2. Data Collection

2.1 Dataset Selection

We used the **20 Newsgroups dataset**, a well-known benchmark in IR and NLP.
It contains **18,000 documents** across **20 categories**. For simplicity and efficiency, we selected 5 distinct categories:

1. **alt.atheism**
2. **comp.graphics**
3. **rec.sport.baseball**
4. **sci.med**
5. **talk.politics.misc**

These categories provide strong topic diversity.

2.2 Dataset Loading

Using scikit-learn's built-in `fetch_20newsgroups`, we loaded:

- **4,531 documents**
- Their corresponding labels
- Cleaned text (headers/footers removed)

The first sample document shows typical newsgroup structure: conversational, informal language, technical terms, and multi-topic content.

2.3 Dataset Inspection

Document counts per category:

Category	Count
alt.atheism	799
comp.graphics	973
rec.sport.baseball	994
sci.med	990
talk.politics.misc	775

Document length statistics:

- **Min:** 0 words
- **Max:** 9,196 words
- **Mean:** 196 words
- **Median:** 82 words

This shows high variability typical of newsgroup datasets.

---

3. Text Preprocessing

Preprocessing transforms raw text into a structured, analyzable form.

3.1 Steps Applied

✓ Tokenization

Splitting text into word units.

✓ Lowercasing

Standardizes text ("Apple" → "apple").

✓ Removing URLs, symbols, and punctuation

✓ Stopword Removal

Using scikit-learn's English stopword list.

✓ Lemmatization

Using WordNet (`car`, `cars`, `car's` → `car`).

✓ Stemming (backup method)

Porter Stemmer (used if lemmatizer unavailable).

✓ Vectorization

Two methods implemented:

Bag-of-Words (BoW)

Creates term–frequency matrix.

TF–IDF (Term Frequency–Inverse Document Frequency)

Weights important terms higher.

3.2 Preprocessing Results

Preprocessing all 4,531 documents took **~4.5 seconds**.

Example cleaned document snippet:

```
place temp file current directory differ point batch launch cview dir resides time
crash expert thought better
```

3.3 Vectorization Results

- **TF-IDF vocabulary size:** 16,573 terms
- **Document-term matrix shape:** (4531 × 16573)

This confirms the dataset is large enough for meaningful retrieval experiments.

---

4. Retrieval Models

Two retrieval models were implemented.

4.1 Vector Space Model (VSM)

Uses **cosine similarity** between query vector and document vectors.

```
similarity(q, d) = (q · d) / (||q|| × ||d||)
```

Implementation Steps

1. Vectorize query using same TF–IDF model.
2. Compute cosine similarity
3. Rank documents by similarity score
4. Return top-k results

Example VSM Output

Query: **"tf idf and bm25 ranking"**

Top results included documents related to:

- sports discussions
- political debates

Because the terms are sparse across categories.

4.2 Boolean Retrieval Model

Boolean queries supported:

- `term1 AND term2`
- `term1 OR term2`
- `term1 NOT term2`

Example Results

Query: **"cancer NOT treatment"**

Top hits all from **sci.med**, demonstrating Boolean model's effectiveness in filtering.

---

5. Model Implementation

The project implemented a unified query-processing mechanism:

Step 1 — Preprocess the query

- Lowercase, remove symbols, tokenize, remove stopwords, lemmatize.

Step 2 — Apply retrieval model

- VSM → compute cosine similarity
- Boolean → apply logical rules

Step 3 — Rank results

Sorted by relevance score (VSM) or match count (Boolean).

This satisfies the guideline:

Convert the query into the same vectorized form as documents and retrieve relevant documents.

---

6. Evaluation

We used:

- **Precision**
- **Recall**
- **Average Precision (AP)**
- **Mean Average Precision (MAP)**

We selected three evaluation queries:

- "graphics software"
- "medical treatment"
- "baseball season"

Results Summary Table

k	VSM Precision	VSM Recall	VSM MAP	Boolean Precision	Boolean Recall	Boolean MAP
5	0.620	0.00336	0.2897	0.124	0.000651	0.000666
10	0.638	0.00686	0.2897	0.074	0.000773	0.000666
20	0.639	0.01379	0.2897	0.037	0.000773	0.000666

Interpretation

- VSM greatly outperforms Boolean retrieval.
- Boolean retrieval returns fewer documents → low recall.
- VSM's cosine similarity provides smoother ranking.

---

7. Visualization

7.1 Word Clouds

Generated for each category, showing most frequent words.

7.2 Frequency Distribution

Used CountVectorizer to compute top terms per category:

- `god`, `religion` in *alt.atheism*
- `image`, `jpeg`, `file` in *comp.graphics*
- `game`, `player`, `team` in *rec.sport.baseball*
- `patient`, `disease` in *sci.med*
- `president`, `government` in *talk.politics.misc*

7.3 Document Similarity Charts

Bar charts for top-k similarity scores per query.

7.4 Topic Modeling (LDA)

Used CountVectorizer + Latent Dirichlet Allocation (LDA).
Topics matched expected categories (e.g., medical, politics, graphics).

---

8. Results and Analysis

8.1 Retrieved Documents for Sample Query

Query: **"graphics hardware software image"**

Top VSM results:

- comp.graphics documents discussing **image formats**, **JPEG**, and **rendering**.
  This confirms VSM is effective on well-represented topics.

Top Boolean results:

- Narrow, exact matches; often fewer relevant hits.

8.2 VSM vs Boolean

Criterion	VSM	Boolean
Ranking	Yes	No
Partial matching	Yes	No
Recall	High	Low
Precision	Good	Depends on query
Flexibility	High	Low
Real-world relevance	High	Low

Conclusion: VSM is far superior for practical IR systems.

---

9. Conclusion

9.1 Key Findings

- The 20 Newsgroups dataset works well for IR testing.
- Preprocessing significantly improves document quality.
- TF–IDF vectorization produces strong results with VSM.
- Boolean retrieval is too restrictive for general search tasks.
- VSM demonstrates higher precision, recall, and MAP.

9.2 Areas for Improvement

- Implement BM25 for enhanced probabilistic ranking.
- Use bigrams/trigrams to capture multi-word concepts.
- Improve query expansion using synonyms from WordNet.
- Apply modern embeddings (Word2Vec, BERT) for semantic search.
- Implement a GUI search interface.

---

10. References

- Pedregosa et al., “Scikit-learn: Machine Learning in Python.”
- Bird et al., “Natural Language Toolkit (NLTK).”
- Řehůřek & Sojka, “Gensim: Topic Modeling in Python.”
- 20 Newsgroups Dataset — Scikit-learn datasets.
- Manning, Raghavan & Schütze — *Introduction to Information Retrieval*.
