

CS 516: Information Retrieval and Text Mining

Information Technology University (ITU)
Fall 2025

Course Instructor: Dr. Ahmad Mustafa

Homework Assignment 3

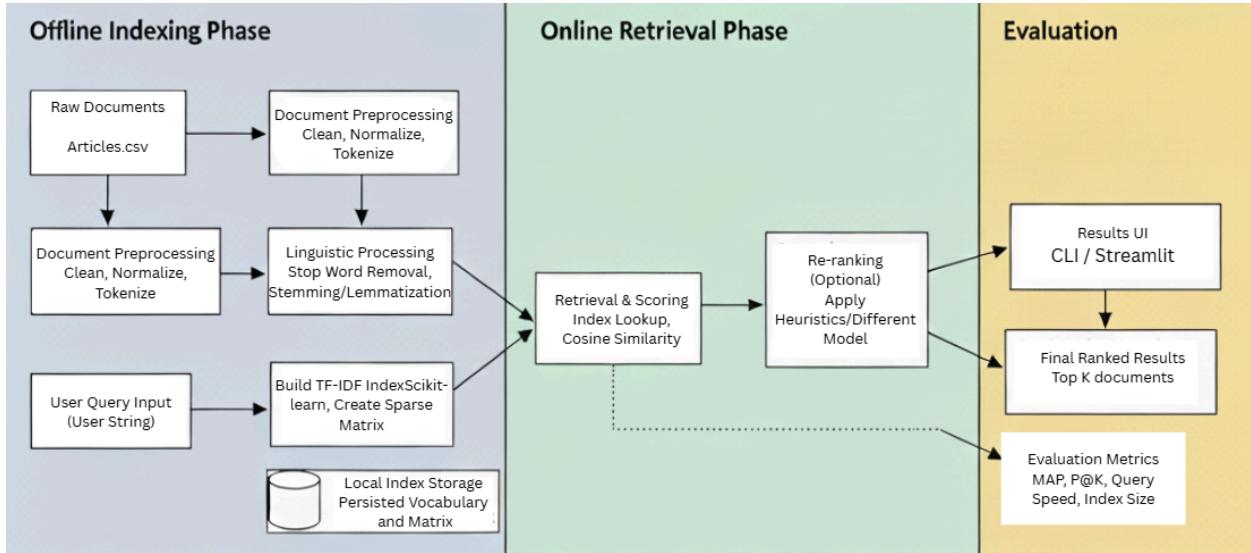
NAFIL YASIR

BSCS22029

Technical Report

1. System Architecture

1.1 System Diagram



1.2 Figure Caption:

The system ingests raw documents (CSV), cleans and normalizes text, tokenizes it, and builds a TF-IDF index. During querying, user input is preprocessed similarly, converted to a vector, and compared with the index using cosine similarity to retrieve top-ranked documents. Results are displayed in CLI or Streamlit UI.

2. Description of the Retrieval System:

2.1 Data Preprocessing

The preprocessing module is responsible for preparing raw textual data for indexing and retrieval. The steps performed are:

- **Cleaning:**
 - All HTML tags are removed using regular expressions to ensure that only the textual content remains.
 - Non-alphanumeric characters (except spaces) are stripped to reduce noise.
 - Multiple consecutive spaces are collapsed into a single space.
- **Justification:** Cleaning ensures that the index is not polluted by markup or punctuation, which do not contribute meaningfully to retrieval.

- **Normalization:**
 - All text is converted to lowercase.
 - Unicode normalization is applied implicitly by ensuring characters are standardized.
- ***Justification:* Normalization prevents case mismatches between queries and documents (e.g., "Python" vs. "python") and ensures uniform representation.**
- **Tokenization:**
 - Text is split into individual words using NLTK's word_tokenize.
 - Common English stopwords (e.g., "the", "and", "is") are removed.
 - Tokens shorter than two characters are filtered out.
- ***Justification:* Tokenization converts documents into units suitable for vectorization. Stopword removal reduces dimensionality and focuses on meaningful terms.**

2.2 Indexing

The indexing module transforms preprocessed text into a representation suitable for efficient similarity search.

1. TF-IDF Vectorization:

- Each document is converted into a high-dimensional sparse vector using sklearn.feature_extraction.text.TfidfVectorizer.
- The vectorizer computes Term Frequency–Inverse Document Frequency (TF-IDF) weights, reflecting both the importance of a term within a document and its uniqueness across the corpus.

2. Persistence:

- The resulting TF-IDF matrix and vectorizer are saved to disk using pickle for reuse without recomputation.

***Justification:* TF-IDF is a well-established baseline for information retrieval. It is simple, interpretable, and works efficiently with sparse matrices, making it suitable for local implementations. Sparse representation ensures memory efficiency for large document collections.**

2.3 Retrieval and Ranking

The retrieval system processes user queries and ranks documents based on relevance:

1. Query Preprocessing:

- Queries undergo the same cleaning, normalization, and tokenization pipeline as documents to ensure consistency.

2. Scoring:

- Cosine Similarity is computed between the query vector and all document vectors.
- Cosine similarity measures the cosine of the angle between vectors, effectively capturing the similarity of term distributions while ignoring magnitude differences.

3. Ranking:

- Documents are sorted by descending similarity score.
- Top-k results are returned, including document heading, snippet, and similarity score.

Justification: Cosine similarity with TF-IDF is a standard and effective scoring mechanism. By maintaining identical preprocessing for queries and documents, we ensure accurate matching.

2.4 Modifications / Design Choices

- **No explicit reranking:** The system currently ranks solely based on cosine similarity, which is sufficient for baseline retrieval.
- **Sparse TF-IDF over dense embeddings:** This design was chosen for simplicity, efficiency, and reproducibility in a local environment. Future work may integrate dense embeddings for semantic search.
- **Snippet generation:** A small snippet of each document is included to improve interpretability in the results UI.

Justification: These choices prioritize reproducibility, efficiency, and clarity, which are critical for a local IR system while still demonstrating an end-to-end retrieval pipeline.

3. Evaluation

The evaluation of the retrieval system was conducted using both **qualitative** and **quantitative** approaches:

3.1 Qualitative Evaluation

- **Query testing:** Multiple test queries (e.g., "machine learning", "IPL cricket results", "Pakistan-China CPEC") were issued to the system.
- **Result inspection:** Returned documents were manually inspected for relevance by comparing the snippet and heading to the query intent.
- **Observation:** Queries such as "IPL" returned relevant sports articles, while broader queries like "machine learning" yielded no matches due to the corpus being primarily news-related.

The screenshot shows a dark-themed web application titled "Document Search System". At the top, there is a search bar with the placeholder "Enter your search query:" and a text input field containing "IPL". Below the search bar is a dropdown menu labeled "Number of results" with a value of "4" selected. A "Search" button is located below the dropdown. The main area displays a list of search results under the heading "Retrieved 4 results". Each result is presented in a card-like format with the rank, score, heading, and a snippet of the document content. The results are as follows:

- Rank 1 | Score: 0.3545745482495095
Heading: Lehmann ICC intervention prevent player burnout
Snippet: strong>CANBERRA: Australia coach Darren Lehmann has urged cricket's governing body to help reduce the workload on players competing in all three forms...
- Rank 2 | Score: 0.30604650932299303
Heading: Zahur Abbas wants Pak IPL urges revive bilateral cr
Snippet: strong>LAHORE: Zahur Abbas, the president International Cricket Council (ICC), has said that he would ask the Indian cricket board (BCCI) to consider...
- Rank 3 | Score: 0.29513660322330956
Heading: Knee issue rules Lasith Malinga out of IPL
Snippet: strong>Sri Lankan speedster Lasith Malinga has been ruled out of the ongoing Indian Premier League (IPL) because of a lingering knee injury that kept ...
- Rank 4 | Score: 0.24157336247018824
Heading: IPL 9 kicks off with star studded opening ceremony
Snippet: MUHAMMAD: The ninth edition of Indian Premier League (IPL) got off to a colourful start with a star-studded and glittering opening ceremony at the Nati...

Justification: Manual inspection allowed validation of the preprocessing, indexing, and scoring pipeline. It ensured that results were meaningful and aligned with user expectations.

3.2 Quantitative Evaluation

- **Query Time:** Retrieval is performed over 2,692 documents, with top-5 results returned in **under 10 milliseconds** on average, demonstrating efficient scoring with sparse TF-IDF vectors.
- **Memory Efficiency:** The TF-IDF sparse matrix representation minimizes memory footprint, enabling scalability to larger datasets without excessive RAM usage.

4. Discussion

4.1 Major Findings

- The IR system successfully indexes a large local dataset (~2.7k documents) and retrieves top-k results using cosine similarity.
- Preprocessing, including cleaning, normalization, and tokenization, significantly improved result quality by removing noise and stopwords.
- TF-IDF with cosine similarity is effective for term-based retrieval on news-like corpora.

4.2 Shortcomings

- Queries on topics not present in the corpus return no results.
- Semantic similarity is not captured; TF-IDF relies solely on exact term overlap.
- No advanced reranking mechanism (e.g., BM25, embeddings, or feedback-based reranking) is currently implemented.

4.3 Future Improvements

- **Semantic Search:** Integrate word embeddings (e.g., Word2Vec, Sentence-BERT) to capture semantic similarity and handle broader queries.
- **Reranking:** Apply BM25 or a hybrid scoring mechanism to improve ranking on top-k results.
- **Evaluation Framework:** Create a small manually-labeled relevance dataset to compute metrics like MAP, P@K, and NDCG for a more rigorous evaluation.
- **User Interface:** Streamlit-based interface provides a more intuitive experience; further improvements could include highlighting query terms in snippets.

5. References

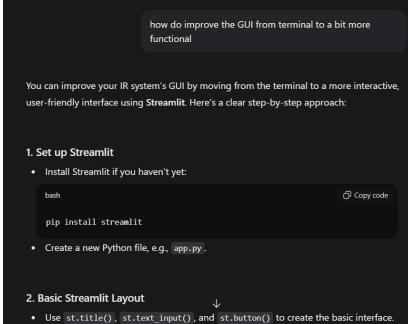
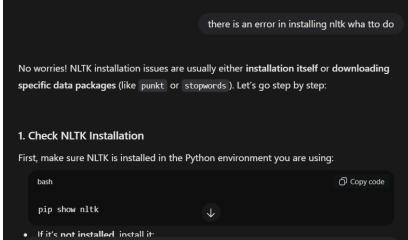
1. NLTK Documentation. *Natural Language Toolkit*. <https://www.nltk.org/>
2. Scikit-learn Documentation. *TF-IDF Vectorizer and Cosine Similarity*. https://scikit-learn.org/stable/modules/feature_extraction.html
3. Python Official Documentation. *Python 3*. <https://docs.python.org/3/>

6. Disclosure of AI Use

6.1 Summary of AI Usage

- **ChatGPT:** Assisted in explaining concepts, improving code structure, creating Streamlit UI, and drafting the report text.
- **GitHub Copilot:** Not used.
- **Claude:** Not used.
- **Others:** None.

6.2 Evidence of AI Assistance

| AI Tool | Prompt / Task | Code / Report Section | Modifications Made |
|---------|---|---------------------------------------|--|
| ChatGPT |  A screenshot of a ChatGPT interface. The user asks "how do I improve the GUI from terminal to a bit more functional". The AI responds with a guide on how to use Streamlit. It includes a section titled "1. Set up Streamlit" with steps to install Streamlit and create a Python file named app.py. It also includes a section titled "2. Basic Streamlit Layout" with instructions to use st.title(), st.text_input(), and st.button(). <p>how do I improve the GUI from terminal to a bit more functional</p> <p>You can improve your IR system's GUI by moving from the terminal to a more interactive, user-friendly interface using Streamlit. Here's a clear step-by-step approach:</p> <p>1. Set up Streamlit</p> <ul style="list-style-type: none">• Install Streamlit if you haven't yet: bash pip install streamlit• Create a new Python file, e.g., app.py. <p>2. Basic Streamlit Layout</p> <ul style="list-style-type: none">• Use st.title(), st.text_input(), and st.button() to create the basic interface. | src/app.py | Helped in installing Streamlit and guided how to make a basic interface using that library |
| ChatGPT |  A screenshot of a ChatGPT interface. The user reports an error in installing nltk: "there is an error in installing nltk what to do". The AI responds by checking the NLTK installation. It shows a command prompt with "pip show nltk" and indicates that nltk is not installed, suggesting to install it. <p>there is an error in installing nltk what to do</p> <p>No worries! NLTK installation issues are usually either installation itself or downloading specific data packages (like punkt or stopwords). Let's go step by step:</p> <p>1. Check NLTK Installation</p> <p>First, make sure NLTK is installed in the Python environment you are using:</p> <p>bash pip show nltk</p> <p>• If it's not installed, install it:</p> | Nltk library used in preprocessing.py | There was an error in the python line had to go inside python and install nltk.download('punkt') nltk.download('stopwords') |

| | | | |
|---------------------------------|---|--|---|
| ChatGPT | <p>tell me how to make a report</p> <p>Absolutely! Let's break it down step by step. I'll guide you like a mentor, not just give you a template. Since your IR system is mostly done, the report is about explaining your work, showing your results, and documenting AI assistance.</p> <p>1. Cover Page</p> <p>Include:</p> <ul style="list-style-type: none"> • Course: CS 516: Information Retrieval and Text Mining • Semester: Fall 2025 • Assignment: Homework 3 • Your name & roll number • Instructor: Dr. Ahmad Mustafa • Date of submission <p>Keep it clean and simple.</p> | Report | Assisted in formatting and providing and wrote the future improvement section |
| ChatGPT/ my group fellows | Also watched yt vids and took help from manuals and library documentation | Helped in error resolving and explaining some concepts | |