Assignment 2
Document Ranking System

Session: 2021 – 2025

Submitted by:

Muhammad Umair Shahid

2021-CS-144

Supervised by:

Dr. Khaldoon Syed Khurshid

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

1 Introduction

The primary goal of this assignment was to develop a basic document ranking system that utilizes advanced information retrieval techniques. The system is designed to process a collection of text files, extract meaningful information using heuristic-based noun extraction, and rank documents based on their relevance to a user-provided query. The ranking process leverages Term Frequency-Inverse Document Frequency (TF-IDF) and Cosine Similarity for accurate and efficient results.

This system focuses on preprocessing, tokenization, and scoring, enabling users to search through documents effectively. Additionally, it demonstrates the practical implementation of core information retrieval concepts in Python.

2 Goal Definition

The project aims to:

- Build a system that ranks documents based on their relevance to user queries using TF-IDF and Cosine Similarity.
- Ensure robust noun extraction optimized for technology-related content.
- Provide a flexible and user-friendly interface for real-time query input.
- Enhance ranking accuracy through the use of mathematical models and heuristics.

By achieving these goals, the system enables efficient document retrieval, making it useful for handling large collections of text files.

3 Document Collection

The system processes input files from a folder containing .txt documents. The preprocessing pipeline performs the following steps:

1. Converts document text to lowercase for uniformity.
2. Tokenizes text to extract individual words.
3. Filters out trivial words (e.g., ”and”, ”the”) and identifies nouns using heuristic rules tailored for technology-related terms.

3.1 Example Terms Extracted:

”automation”, ”robotics”, ”AI”, ”ethics”, ”technology”

The documents used for testing included various themes and keywords, allowing the system to demonstrate its capability to handle diverse content.

4 Implementation Plan

The implementation plan involved the following steps:

1. Data Preprocessing:
    - Convert text to lowercase.
    - Tokenize and clean data to extract meaningful tokens (nouns).
2. Scoring Algorithms:
    - Implement TF (Term Frequency), IDF (Inverse Document Frequency), and TF-IDF for ranking.
    - Use Cosine Similarity to measure the relevance between the query and documents.
3. User Interaction:
    - Design a command-line interface (CLI) for user queries.
    - Allow dynamic folder path updates and query processing.
4. Results Display:
    - Present the scores and rankings in a structured format using tables.

5 Approaches

The following approaches were utilized to achieve the objectives:

1. **Heuristic Noun Extraction:** A function was developed to extract nouns by:
    - Filtering trivial words like “and”, “in”, “of”.
    - Identifying domain-specific nouns such as “AI”, “robotics”.
    - Detecting noun suffixes like “tion”, “logy”, and “ics”.

2. **TF-IDF Scoring:**

   - **TF (Term Frequency):** Measures the frequency of a term in a document.
     Formula:
     $$TF = \frac{\text{Frequency of the term}}{\text{Total number of terms in the document}}$$
   - **IDF (Inverse Document Frequency):** Measures the uniqueness of a term across all documents.
     Formula:
     $$IDF = \log_{10}\left(\frac{N}{n+1}\right)$$
     where N is the total number of documents and n is the number of documents containing the term.
   - **TF-IDF:** Combines TF and IDF to determine term importance.
     Formula:
     $$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

3. **Cosine Similarity:**

   - Measures the similarity between the query vector and document vectors.
     Formula:
     $$\text{Cosine Similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

6 Backend Concepts

The backend system was implemented in Python, leveraging several custom-built modules to achieve the desired functionality. The major components include:

6.1 Tokenization and Noun Extraction

- The `tokenize_nouns()` function processes the document text, extracting meaningful nouns based on heuristics.
- Filters trivial terms and identifies domain-specific nouns to create a refined token set.

6.2 TF-IDF Calculations

- `calculate_tf()` computes the term frequency of a word in a document.
- `calculate_idf()` calculates the inverse document frequency of a term across all documents.
- The TF-IDF scores are then derived by combining the outputs of these functions.

```python
def calculate_tf(word, nouns):
    """
    Calculate Term Frequency (TF).
    """
    word_count = nouns.count(word)
    total_words = len(nouns)
    return word_count, total_words, word_count / total_words if total_words > 0 else 0
```

FIGURE 1: TF

```python
def calculate_idf(word, noun_store):
    """
    Calculate Inverse Document Frequency (IDF).
    """
    N = len(noun_store)
    n = sum(1 for nouns in noun_store.values() if word in nouns)

    # print(f"{N} \t {n}")
    return math.log10(N / (n + 1))
```

FIGURE 2: IDF

6.3 Cosine Similarity

- The `calculate_cosine_similarity()` function creates document and query vectors based on TF-IDF scores.
- Computes similarity scores for each document with respect to the user's query.

```python
def cosine_similarity(vec1, vec2):
    dot_product = sum(a * b for a, b in zip(vec1, vec2))
    norm1 = math.sqrt(sum(a**2 for a in vec1))
    norm2 = math.sqrt(sum(b**2 for b in vec2))
    return dot_product / (norm1 * norm2) if norm1 and norm2 else 0
```

FIGURE 3: Cosine Similarity

6.4 Dynamic Query Handling

- Allows users to input queries through the command-line interface (CLI).
- Supports dynamic folder updates for real-time indexing and retrieval.

7 User Interface

The command-line interface (CLI) is designed for simplicity and usability, with support for dynamic input and interactive options.

7.1 Features of the CLI

- Users can input a query to search for relevant documents.
- Provides options to update the folder path dynamically, enabling real-time document processing.
- Displays results in well-formatted tables, highlighting TF, TF-IDF, and Cosine Similarity scores.

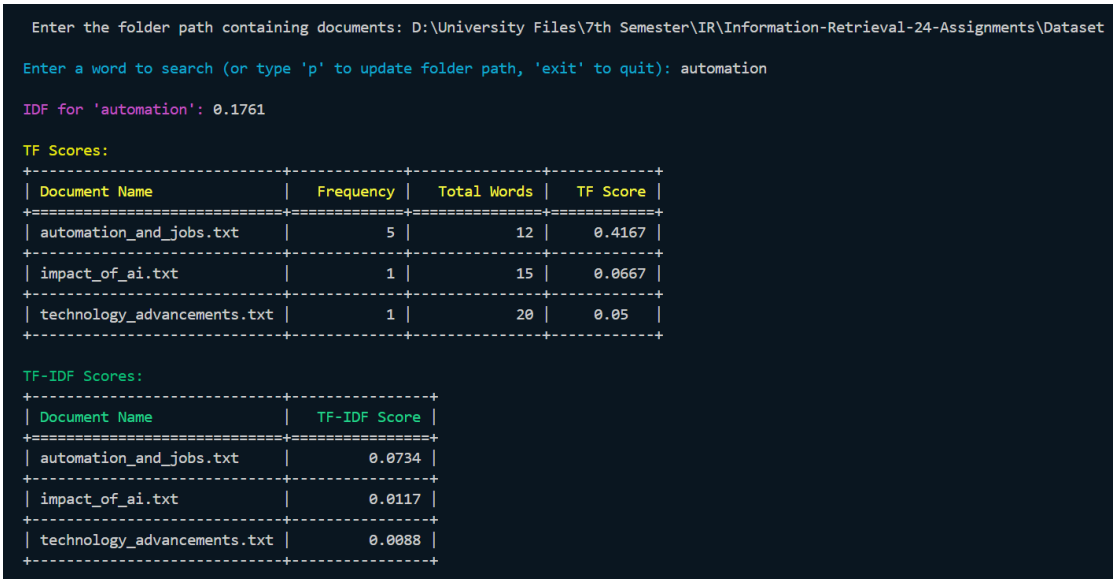

FIGURE 4: TF-IDF

7.2 Color-Coded Results

- Results are displayed with enhanced readability using the `colorama` library.
- Relevant terms are highlighted, and documents are ranked based on their relevance to the query.

```
Cosine Similarity Rankings:
+-----------------------------+---------------------------+
| Document Name               |   Cosine Similarity Score |
+=============================+===========================+
| automation_and_jobs.txt     |                    0.9806 |
+-----------------------------+---------------------------+
| impact_of_ai.txt            |                    0.9806 |
+-----------------------------+---------------------------+
| technology_advancements.txt |                    0.9806 |
+-----------------------------+---------------------------+
| ethics_in_technology.txt    |                    0      |
+-----------------------------+---------------------------+
| future_of_robots.txt        |                    0      |
+-----------------------------+---------------------------+
| quantum_computing.txt       |                    0      |
+-----------------------------+---------------------------+

Enter a word to search (or type 'p' to update folder path, 'exit' to quit):
```

FIGURE 5: Cosine Similarity

8 Data Flow Diagrams(DFD)

8.1 Level 0

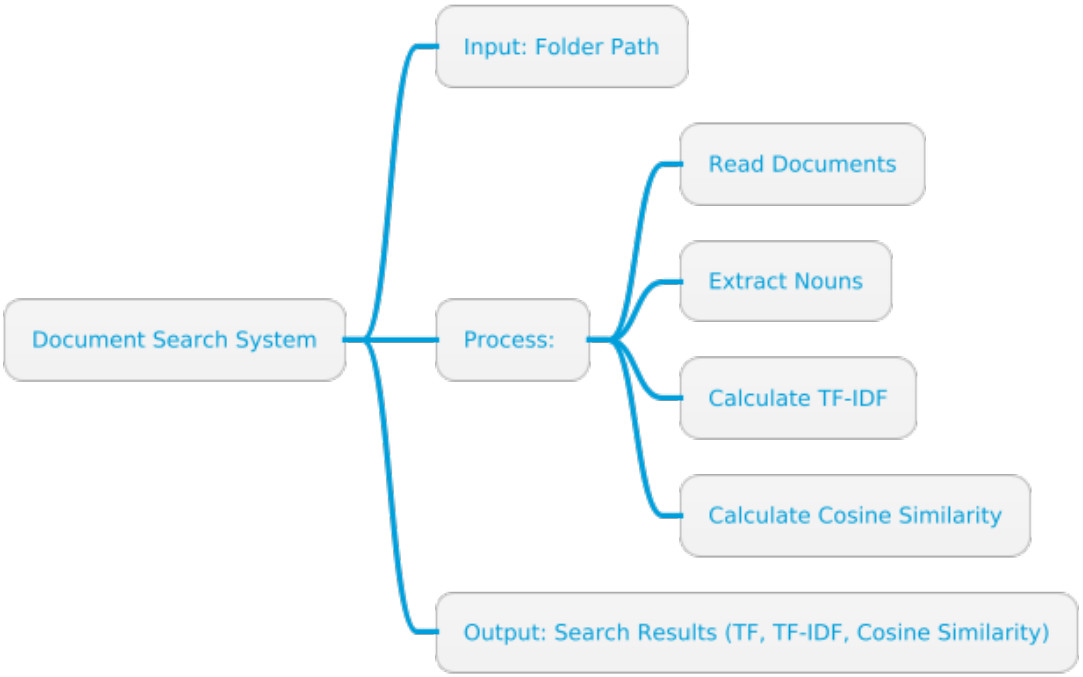

FIGURE 6: Level 0

8.2 Level 1

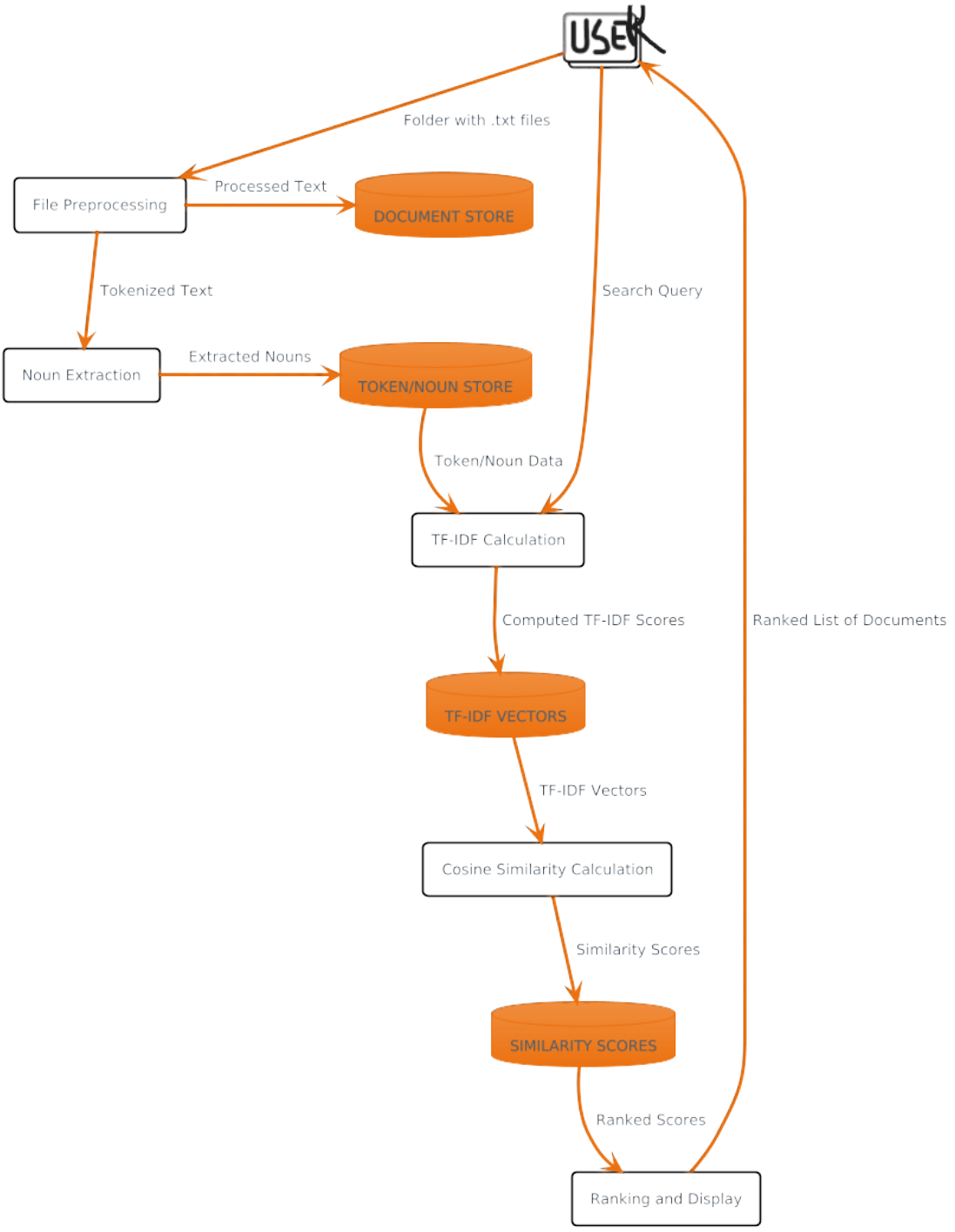

FIGURE 7: Level 1

9 Shortcomings

While the system achieves its primary objectives, several limitations were observed:

- **Static Noun Extraction:** Heuristic-based extraction may miss context-specific nouns.
- **Limited Query Support:** The system does not handle multi-word or phrase queries effectively.
- **Basic Cosine Similarity:** Relies on raw term vectors without leveraging advanced embeddings for semantic analysis.

Suggestions for Improvement:

- Introduce stemming or lemmatization to improve query matching.
- Enhance noun extraction using machine-learning-based techniques or syntactic parsers.
- Incorporate word embeddings like Word2Vec or GloVe for more accurate similarity measures.

10 Conclusion

The document ranking system demonstrates the practical implementation of TF-IDF and Cosine Similarity for information retrieval. By combining heuristic-based preprocessing with mathematical scoring models, the system provides accurate and efficient document ranking. While limitations exist, future enhancements like stemming and phrase matching can further improve its performance.