Assignment 3

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

This assignment implements three innovative document retrieval models for Information Retrieval (IR):

1. Probabilistic Model (BIM)
2. Non-Overlapped List Model
3. Proximal Nodes Model

These models enable efficient retrieval by focusing on term relevance, query adaptability, and document relationships. The goal is to retrieve documents that are most relevant to a user's query while providing ranked results using advanced computational methods.

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

Objectives

- Implement three distinct models to retrieve relevant documents.
- Enhance query relevance using term and graph-based techniques.
- Provide a comprehensive comparison of different retrieval approaches.

3 Probabilistic Model (BIM)

Key Concept

The Binary Independence Model (BIM) uses binary vectors to represent documents and queries. Relevance is determined using the **Dice Coefficient**.

Methodology

1. Represent documents and queries as binary vectors:

$$\text{Vector entry} = \begin{cases} 1 & \text{if term exists in document/query} \\ 0 & \text{otherwise} \end{cases}$$

2. Compute term probabilities:

$$P(t) = \frac{\text{Number of documents containing } t}{\text{Total documents}}$$

3. Calculate the **Dice Coefficient** for scoring:

$$\text{Dice Coefficient} = \frac{2 \cdot |\text{Intersection of query and document terms}|}{|\text{Query terms}| + |\text{Document terms}|}$$

4. Rank documents based on Dice scores.

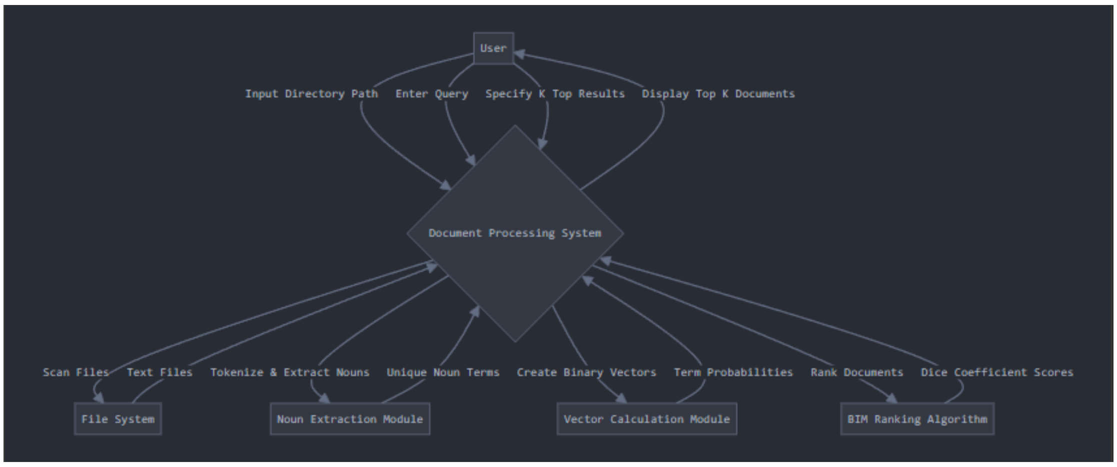


FIGURE 1: Probabilistic Model (BIM)

4 Non-Overlapped List Model

Key Concept

This model ensures comprehensive document coverage by combining lists of documents for all query terms, while avoiding duplicates.

Methodology

1. Construct an **inverted index** mapping terms to documents:

$$\text{Index: } \{\text{term: [document1, document2, ...]}\}$$

2. For each query term:

   - Retrieve the document list.
   - Perform a union operation on the lists to merge them without duplication.

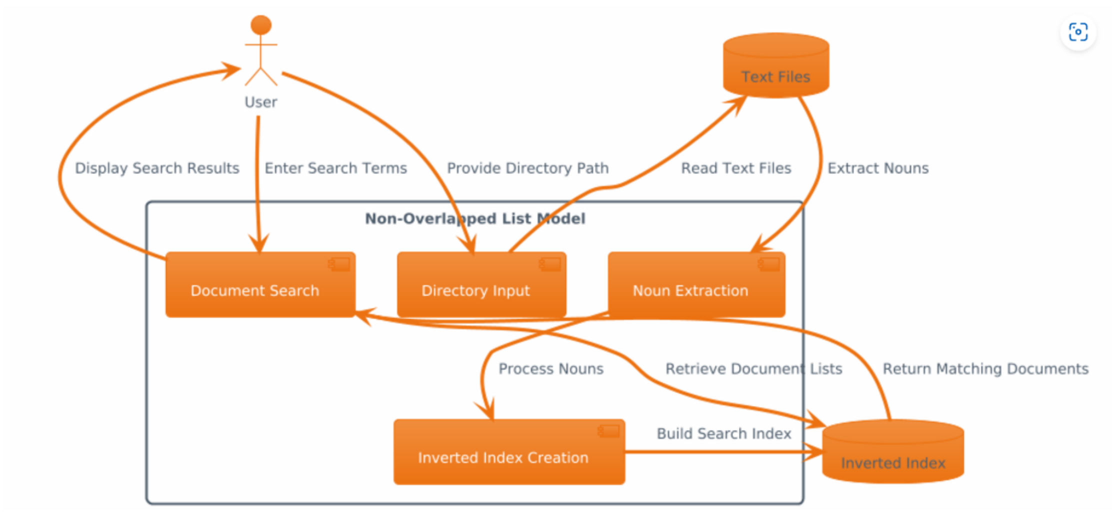


FIGURE 2: Non-Overlapped List Model

5 Proximal Nodes Model

Key Concept

This graph-based model represents documents and terms as nodes, with edges denoting term-document relationships.

Methodology

1. Construct a document-term graph:

- Nodes represent documents and terms.
- Edges connect terms to documents where they appear.

2. Extract **proximal nodes** from the query:
   - Tokenize the query into terms.
   - Identify terms (nodes) present in the graph.
3. Perform a **Breadth-First Search (BFS)** to find all connected documents.

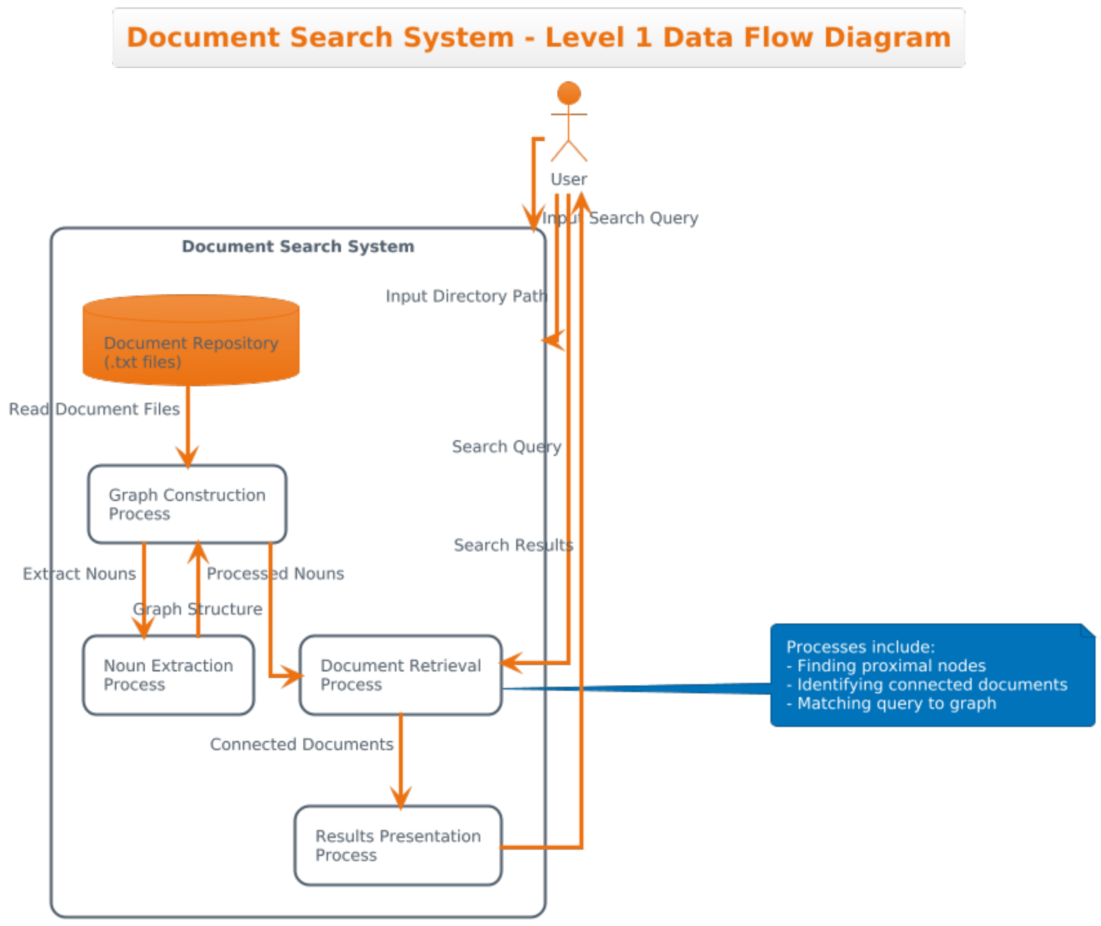


FIGURE 3: Proximal Nodes Model

6 Comparative Analysis

Model	Strengths	Characteristics
Probabilistic Model (BIM)	Accurate term matching	Uses Dice Coefficient
Non-Overlapped List Model	Comprehensive coverage	Combines document lists, avoids duplicates
Proximal Nodes Model	Conceptual relevance	Graph-based approach with BFS

TABLE 1: Comparison of Retrieval Models

7 Backend Concepts

Tokenization and Noun Extraction

- Documents are tokenized, and meaningful nouns are extracted using heuristic-based rules.
- Trivial words such as “and”, “of”, and “the” are filtered out.
- Domain-specific nouns (e.g., “AI”, “robotics”) are identified for relevance.

Model-Specific Logic

Each model incorporates a unique approach:

- **BIM:** Uses binary vectors and the Dice Coefficient to rank documents.
- **Non-Overlapped List:** Combines document lists without duplicates.
- **Proximal Nodes:** Uses a graph structure for query-document relationships.

Dynamic Query Handling

The system allows users to input queries in real-time, with:

- Dynamic folder selection for processing new documents.
- Immediate feedback on query terms and retrieved results.

8 User Interface

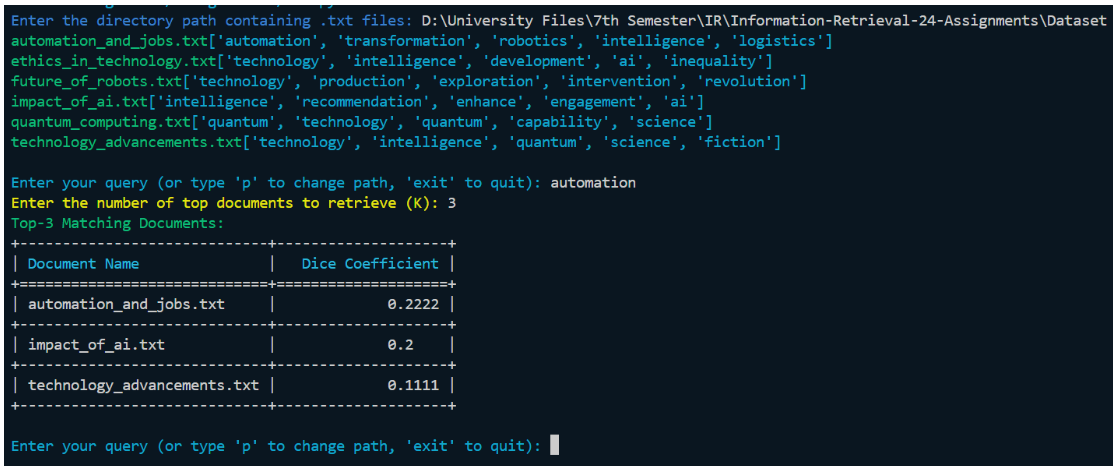
Features

The Command-Line Interface (CLI) is designed for simplicity and usability:

- Allows users to select a directory containing text files.
- Enables input of queries to retrieve relevant documents.
- Provides descriptive error messages for invalid input or missing terms.
- Displays results in color-coded format for readability.

Example Interaction

1. The user enters a query: **”automation”**.
2. The system processes the query and retrieves ranked documents.
3. Results are presented in a tabular format with relevant scores.



```
Enter the directory path containing .txt files: D:\University Files\7th Semester\IR\Information-Retrieval-24-Assignments\Dataset
automation_and_jobs.txt['automation', 'transformation', 'robotics', 'intelligence', 'logistics']
ethics_in_technology.txt['technology', 'intelligence', 'development', 'ai', 'inequality']
future_of_robots.txt['technology', 'production', 'exploration', 'intervention', 'revolution']
impact_of_ai.txt['intelligence', 'recommendation', 'enhance', 'engagement', 'ai']
quantum_computing.txt['quantum', 'technology', 'quantum', 'capability', 'science']
technology_advancements.txt['technology', 'intelligence', 'quantum', 'science', 'fiction']

Enter your query (or type 'p' to change path, 'exit' to quit): automation
Enter the number of top documents to retrieve (K): 3
Top-3 Matching Documents:
+-----------------------------+--------------------+
| Document Name               |   Dice Coefficient |
+=============================+====================+
| automation_and_jobs.txt     |             0.2222 |
+-----------------------------+--------------------+
| impact_of_ai.txt            |             0.2    |
+-----------------------------+--------------------+
| technology_advancements.txt |             0.1111 |
+-----------------------------+--------------------+

Enter your query (or type 'p' to change path, 'exit' to quit):
```

FIGURE 4: User Interface

9 Future Improvements

To enhance the system's capabilities, the following improvements are proposed:

- **Machine Learning Integration:** Use machine learning models for improved noun extraction and relevance ranking.
- **Multilingual Support:** Expand functionality to support documents in multiple languages.
- **Advanced Graph Traversal:** Optimize graph-based retrieval for large datasets using algorithms like Depth-First Search (DFS).
- **Semantic Analysis:** Incorporate word embeddings (e.g., Word2Vec, GloVe) for better contextual understanding.

10 Conclusion

This assignment demonstrates the practical implementation of three distinct models for document retrieval. Each model has unique strengths:

- BIM focuses on precise term matching.
- Non-Overlapped List Model ensures broad document coverage.
- Proximal Nodes Model emphasizes conceptual relationships.

Future improvements can integrate machine learning techniques to enhance relevance ranking and scalability.