Digital Information Retrieval System

Pegah Moradpour

---

1. Introduction

This report documents the development and implementation of a Digital Information Retrieval System. The system is designed to index a collection of HTML documents and efficiently retrieve relevant documents based on user queries. This project is part of the Digital Information Retrieval course and adheres to the specified requirements and additional features.

2. Objectives

The primary objectives of this project are:

1. **Indexing Function**: Develop a procedure that creates an inverted index from a collection of documents and stores it permanently.
2. **Retrieving Function**: Implement a user interface for querying the index and displaying relevant documents.

3. Implementation Overview

Indexing Function

- **Inverted Index Creation**: The system processes HTML documents from the dataset, extracting terms and creating a comprehensive inverted index. Each term in the index maps to a list of documents and their respective positions within those documents.
- **Stemming and Stop Words**: Terms are stemmed using the Porter Stemmer algorithm, and common stop words are removed to improve the quality of the index.
- **Permanent Storage**: The inverted index and document paths are stored in JSON files to allow for quick retrieval and to accommodate the lengthy indexing process.

Retrieving Function

- **User Interface**: A command-line interface prompts the user to enter search queries. The system supports basic term queries as well as more complex conjunctive and disjunctive queries.
- **Search and Display**: The system searches the inverted index for the specified terms and displays the results, including document IDs and paths, in a well-formatted table using the `tabulate` library.

4. Features Implemented

Group A

1. **Conjunctive and Disjunctive Queries**: The retrieval function is extended to handle complex queries, retrieving and combining posting lists as necessary. This allows users to search for documents that contain all specified terms (AND queries) or any of the specified terms (OR queries).

2. **Edit Distance**: To enhance user experience, the system suggests corrections for misspelled terms based on edit distance calculations. If a term is not found in the index, the system suggests the closest matching term.

Group B

1. **Porter Stemmer Algorithm**: The indexing function incorporates the Porter Stemmer to standardize terms by reducing them to their root forms. This improves retrieval accuracy by ensuring that different forms of the same word are treated as the same term.
2. **Positional Index**: The system not only indexes the presence of terms but also records their positions within documents. This enables more precise searches, such as phrase searches where the relative positions of terms matter.

Optional Features

1. **Vector Space Model and Cosine Similarity**:
    - **Vector Space Model (VSM)**: This model represents text documents as vectors of identifiers. Each document is represented as a vector, where each dimension corresponds to a separate term. The value of each dimension is usually the term frequency (TF) of the term in the document.
    - **Cosine Similarity**: This measure calculates the cosine of the angle between two non-zero vectors in an inner product space. In the context of document retrieval, it is used to measure the similarity between the query vector and document vectors. The formula for cosine similarity is:

$$\text{cosine similarity} = \frac{\sum_{i=1}^{n} \text{TF}_{\text{query},i} \times \text{TF}_{\text{document},i}}{\sqrt{\sum_{i=1}^{n} (\text{TF}_{\text{query},i})^2} \times \sqrt{\sum_{i=1}^{n} (\text{TF}_{\text{document},i})^2}}$$

    - **Implementation**: The system computes term frequencies for both the query and the documents, then calculates the cosine similarity scores to rank the documents. This provides more accurate and relevant search results by considering the context and frequency of terms.

5. Dataset Information

The dataset used for this project consists of a collection of HTML documents stored in a directory structure. Each document contains various terms that are indexed and retrieved based on user queries. The dataset is representative of typical web documents and includes a variety of topics, providing a robust testbed for the retrieval system.

6. Implementation Details

Indexing Process

- The indexing function processes all HTML files in the dataset directory, extracting and processing terms from each document.
- Terms are stemmed and filtered using a stop words list.
- Document paths and term positions are recorded, creating a detailed inverted index.
- The inverted index and document paths are saved as JSON files for persistent storage.

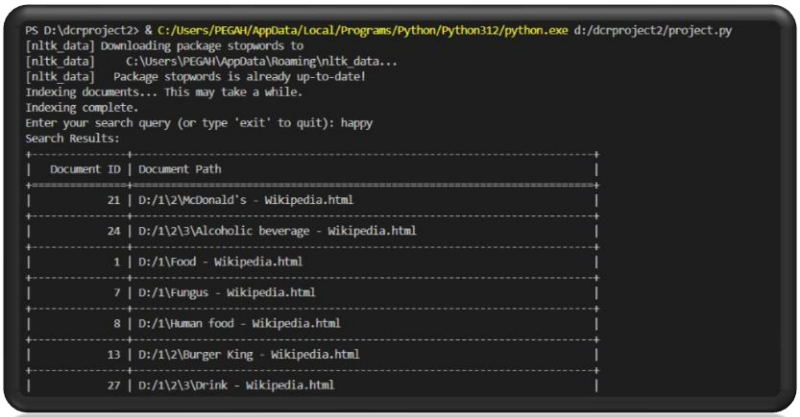
Retrieval Process

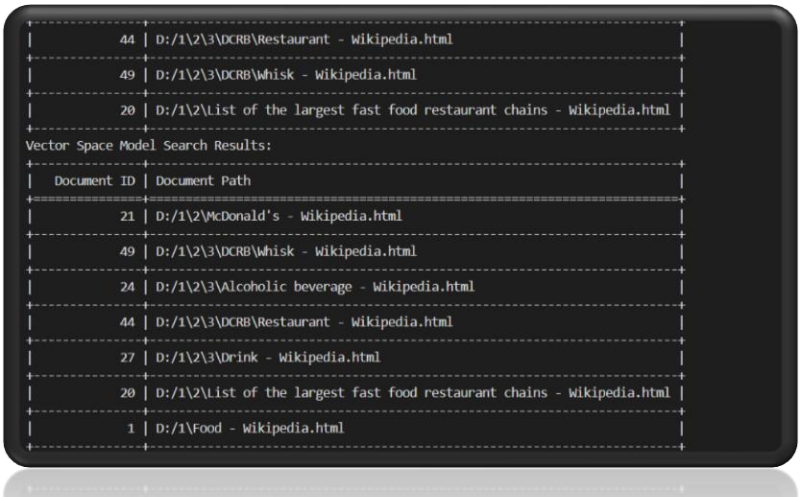
- The retrieval function allows users to input search queries via a command-line interface.
- The system processes queries, including complex conjunctive and disjunctive queries, and suggests corrections for potential misspellings.
- Results are retrieved from the inverted index and displayed in a tabulated format, showing document IDs and paths for clarity.

7. Results and Output

The retrieval system was tested with various queries to ensure its functionality. The results, including document IDs and paths, are displayed in an organized table format. The system also computes document relevance scores using the Vector Space Model for more accurate results.

Search Results Example:





8. Conclusion

The implemented Retrieval System meets the project requirements by efficiently indexing a collection of documents and providing an effective retrieval mechanism. The additional features from Groups A and B, along with the optional feature, enhance the system's functionality and user experience.