Orkhan Abilov

Advanced Java
Project Specification

This project is a command-line tool for indexing a collection of documents and running boolean retrieval queries on the indexed data. The concept of boolean retrieval was introduced to me during the "Searching the Web" course. I became passionate about retrieval models, especially the boolean retrieval model, and decided to implement this project in Java as a practical application of what I learned.

What is Boolean Retrieval

Boolean retrieval is a simple and classic information retrieval model that works based on the Boolean logic operators: AND, OR, and NOT. It is primarily used in the context of searching and indexing textual data. Key Concepts are:

Inverted Index:

An inverted index is a data structure used to map words (or terms) to the documents that contain them. It allows for efficient querying.

Boolean Operators:

AND: The AND operator is used to retrieve documents that contain all the specified terms. For example, the query "word1 AND word2" returns documents that contain both "word1" and "word2".

OR: The OR operator is used to retrieve documents that contain at least one of the specified terms. For example, "word1 OR word2" returns documents that contain either "word1" or "word2".

NOT: The NOT operator is used to exclude documents that contain the specified term. For example, "word1 NOT word2" returns documents that contain "word1" but not "word2".

Stopwords:

Stopwords are common words that are typically filtered out during the indexing and querying process because they have no value in information retrieval.

Examples of stopwords include: "and", "the", "is", "in", "at", "of", "on".

Removing stopwords helps in reducing the size of the index and improves the efficiency of the retrieval.

The tool is implemented in Java and has two main functionalities:

Indexing: Processing and indexing documents.

Querying: Searching the indexed data with boolean queries.

Indexing

The indexing process involves:

Reading Documents: The program reads text documents from a specified directory.

Preprocessing Text: The text is normalized to lower case, non-word characters are removed, and common stop words (e.g., "and", "the", "in") are filtered out.

Building an Inverted Index: An inverted index is created, which maps each unique word to a set of document filenames where the word appears.

Storing Indexed Data: The inverted index and the dictionary of words are stored in specified files for later retrieval.

Querying

The querying process involves:

Loading the Inverted Index: The program loads the previously stored inverted index from a file.

Preprocessing the Query: The query string is normalized to lower case, non-word characters are removed, and stop words are filtered out.

Executing the Query: The program finds the intersection of sets of documents for each word in the query. This results in a set of document filenames that match all the words in the query.

Outputting the Results: The program then prints the result, which is a set of document filenames that satisfy the query.

Data Sources

Documents Directory: Contains the text documents to be indexed.

Data Directory: Used to store the indexed data, including the dictionary and the inverted index.

How will it run:

Indexing:

The command line will take the path to the documents directory and the path to the data directory.

Querying:

And querying will take the script and the string to look for