Name: Amirali Najafizadeh
Course: CPS 842
Student ID: 500945716

Assignment 1 Report

Data Structures:

Documents List: We organized our documents as a list of dictionaries. Each dictionary represented a single document and contained the content of the document, such as document ID, title, abstract, authors, and date. This structure enabled us to manage and access individual documents effectively.

Dictionary: We employed a dictionary data structure to store our terms (words) as keys and associated values. These values included the document frequency (DF) of each term, indicating how many documents contained the term. This dictionary allowed for fast look-up of term information.

Postings List: To associate terms with specific documents and their positions, we used a list of tuples, known as the postings list. Each tuple represented a posting for a term and contained the document ID, term frequency (TF), and a list of positions where the term appeared in the document. This structure facilitated the retrieval of term-document relationships.

Algorithms:

Tokenization and Preprocessing: We employed tokenization and preprocessing algorithms to transform the text in our documents into manageable terms. Tokenization broke the text into words, while preprocessing removed stop words and, optionally, applied stemming to reduce terms to their root forms.

Term Frequency Counting: For each document, we counted the frequency of each term and maintained a record of the positions where the term occurred. This algorithm allowed us to understand the distribution of terms within documents.

Dictionary Construction: As we processed documents, we constructed a dictionary by iterating through the terms. For each term, we updated its document frequency (DF) and created an entry in the dictionary if it did not already exist. The dictionary stored essential information about the terms in our collection.

Postings List Construction: We created the postings list by associating each term with its document ID, term frequency (TF), and positions where it appeared. This enabled efficient retrieval of information regarding term-document relationships.

Stemming (Optional): We incorporated the Porter stemming algorithm as an optional step to reduce terms to their root forms. This algorithm can enhance retrieval accuracy by treating different forms of a term as equivalent.

Instructions:

1. The program executables can be found within the "assignment1" folder.
2. Ensure that you retain the "cacm" folder as the files contained within are necessary for the main program located in the "assignment1" folder.
3. The main program is named "test.py," which eventually generates three text files: "dictionary.txt," "postings.txt," and "reduced.txt."
4. The "reduced.txt" file will be located inside the "cacm" folder, while the other two text files will be created within the "assignment1" folder.
5. Prior to executing "test.py," verify that these files do not already exist to prevent overwriting.
6. During program execution, you will be presented with options to enable or disable specific features.
7. To enable a feature, press '1'; otherwise, press '2'.
8. Once the files are created, you will be prompted to input a term or enter 'ZZEND' to terminate the program.