Information Retrieval course - Mini Project 1 documentation

Kimia Esmaili - 610398193

To make our system, we will follow these steps:

1. **Reading and Preprocessing Text Documents:**
    - Read the collection of text documents.
    - Preprocess the documents by removing any special characters, punctuation, and stop words.
    - Tokenize the documents into individual words.
    - Perform stemming or lemmatization to reduce words to their base form.
    - Create a list of preprocessed documents.

2. **Creating an Inverted Index:**
    - An inverted index is a data structure that maps terms to the documents in which they appear.
    - Initialize an empty inverted index dictionary.
    - Iterate over each preprocessed document:
        A. Tokenize the document into individual words.
        B. For each word, update the inverted index dictionary:
            I. If the word is not already a key in the dictionary, add it and initialize an empty list as the value.
            II. Append the current document ID to the list of documents associated with the word.
    - The inverted index will have terms as keys and associated documents as values.

3. Handling Standard Boolean Queries:

- Parse the Boolean query input from the user.
- Identify the query terms and operators (AND, OR, NOT).
- Retrieve the list of documents corresponding to each query term from the inverted index.
- Combine the retrieved document lists based on the Boolean operators:
    A. For AND operator, take the intersection of document lists.
    B. For OR operator, take the union of document lists.
    C. For NOT operator, remove the documents in the second list from the first list.
- Return the final list of relevant documents.

4. Handling Proximity Queries:

- Parse the proximity query input from the user.
- Identify the query terms and the maximum distance between them.
- Retrieve the list of documents corresponding to each query term from the inverted index.
- Iterate over each document in the first query term's document list.
- For each document, check if any other query term appears within the maximum distance.
- Return the final list of relevant documents.

Index Optimization:

- One way to optimize the inverted index is by using a technique called "Posting Compression."

  Posting Compression reduces the size of the inverted index by storing only the differences between consecutive postings. Instead of storing the complete list of document IDs for each term, we store the differences between consecutive document IDs.

  For example, if a term appears in documents [1, 5, 8, 10], we store it as [1, 4, 3, 2].
  To retrieve the original list of document IDs, we can recover it by adding the differences to the previous document ID.

  This optimization reduces the memory footprint of the inverted index, making it more efficient for large collections.

Code Explanation:

- The code consists of several functions, each performing a specific task such as preprocessing, creating the inverted index, handling Boolean queries, and proximity queries.
- The main function acts as a driver, providing an interface for the user to interact with the system.
- The preprocessing function takes the collection of text documents as input and returns a list of preprocessed documents.
- The inverted index function takes the preprocessed documents as input and returns the inverted index data structure.
- The Boolean query function takes the query input as well as the inverted index and returns the list of relevant documents based on the Boolean operators.
- The proximity query function takes the query input, the inverted index, and the maximum distance as input and returns the list of relevant documents based on proximity.
- The code is modular and well-documented for easy understanding and maintainability. There are comments in the code on what every part does, so it is redundant to show how every part and function of the code works again in this document.
- Note that the code uses the NLTK library for stopwords and Porter stemming. Make sure to install NLTK (pip install nltk) and download the necessary resources by running nltk.download('stopwords') before executing the code.