Assignment 1

Q-1)

Preprocessing using NLTK library

a. Lowercasing

The initial step in the preprocessing pipeline involved converting all text to lowercase. This ensures consistency and eliminates potential discrepancies caused by variations in letter casing. The NLTK library provided a straightforward method for transforming the entire corpus to lowercase.

b. Tokenization

Tokenization is the process of breaking down text into individual words or tokens. NLTK's tokenizer was utilized to achieve this. By segmenting the text into tokens, we establish a foundation for further analysis, allowing us to explore the linguistic structure of the data.

c. Stopword Removal

Stopwords are common words that often do not contribute significant meaning to the text. Removing these stopwords can enhance the efficiency of downstream tasks. NLTK's predefined stopword list was employed to filter out irrelevant words, thus focusing on the more informative content.

d. Punctuation Removal

Punctuation marks do not always carry semantic meaning and can introduce noise in the analysis. To address this, NLTK's punctuation removal functionality was applied, resulting in a cleaner text representation.

e. Blank Space Token Removal

During tokenization, blank space tokens might be generated, leading to potential issues in analysis. These were systematically removed to ensure a more accurate representation of the text.

The primary goal of these steps is to make the corpus of the file usable for further operations. The files have been saved in google drive.

Q-2)

Unigram Inverted Index

Now the above preprocessed files are used for creating the unigram inverted index which i have made from scratch using NLTK. These are following steps used:

Tokenization: Each document in the corpus was tokenized into unigrams, providing a list of terms for each document.

Inverted Index Creation: For each term, an inverted list was created, containing the document IDs in which the term occurred.

Normalization: The inverted lists were sorted and normalized for efficient querying.

Storage using Pickle
I have stored the unigram inverted index file using pickle.

Boolean Operations Support
The implemented system provides support for the following Boolean operations:

a. T1 AND T2
The intersection of the inverted lists for terms T1 and T2 is computed, providing the documents where both terms co-occur.

b. T1 OR T2
The union of the inverted lists for terms T1 and T2 is calculated, yielding the documents containing either term or both.

c. T1 AND NOT T2
The documents containing term T1 but not T2 are identified by subtracting the inverted list for T2 from that of T1.

d. T1 OR NOT T2
The union of the inverted list for T1 and the complement of the inverted list for T2 is determined, representing documents with either T1 or without T2.

Input and Output format

The input format for the code includes the number of queries (N) and N lines containing phrase queries. The output format consists of 2N lines, with each pair representing the results for a specific query:

Number of documents retrieved for the query using the positional index.
Names of documents retrieved for the query using the positional index.

Q-3)

Now I have created the positional index using preprocessed files done in Q1.
Creating the positional index
The initial step is to create a positional index using the NLTK library. The provided code demonstrates a function create_positional_index(folder_path) to achieve this. This function

iterates through each text document in a specified folder, tokenizes the text, and builds a positional index for each term.

Here's an overview of the key steps:

a) Iterate through each text file in the specified folder.
b) Tokenize the content of each file using the NLTK library.
c) Build a positional index for each term, storing document names and corresponding positions.

Input and Output format

The input format for the code includes the number of queries (N) and N lines containing phrase queries. The output format consists of 2N lines, with each pair representing the results for a specific query:

Number of documents retrieved for the query using the positional index.
Names of documents retrieved for the query using the positional index.