Assignment 3 Report - Group 9

Mark Amirthanathan - 217266677

Mohammad Asif Khan - 216498453

Yaotian Huang - 217470873

Basic Description

In this program we are given a collection of Web documents. We will be implementing an indexer that can create an index for the provided Web documents, searching these web documents from the created index for 20 topics/queries and outputting our search result in a file.

Here is a List of Methods We Used

1. Unzip GZ files in HTML
2. Preprocess XML into title dictionary { docno: docoldno } and word dictionary {docno: [word1, word2, …..]}
3. Built inverted indexer using word dictionary
4. Group by initial and save each file into a json file to save RAM usage when applying searching

How to Build Our Index

The files need to be unzipped first, and then the doc tags are extracted from the XML. The text from the <html> tag inside each doc tag is retrieved, and if there is no <html> tag, a new one is created, and all HTML elements are appended to it. The NLTK library is used for text preprocessing, which involves tokenization, stemming, keeping only English words, and encoding into UTF-8. Multithreading is used to speed up the processing for each document. Finally, an inverted index is built by splitting the word dictionary into chunks by initial character and processing each chunk in parallel using multithreading, and the results are saved into JSON files based on the initials of the words.

Design of Our Program

1) In the preprocess stage, we could calculate the TF-IDF value to mark importance of each token (word)
2) Added multithreading during the preprocessing stage to speed up the process of reading html content and converting it to tokens
3) For weighting, we used the vector space function
    a) In the vector space model, we used the Dot product for calculating the similarity

Analysis of Results

According to experimentation, the runtime of the program takes approximately 2 hours. Prior to the second step of the design phase, it would take approximately 3 hours to complete. After adding multithreading during the preprocessing stage, it sped up the reading process of html content greatly. The second half of the program which is the querying takes only 5 seconds to complete.