Invert.py

The program takes the input of the cacm collection and then begins to read through the file line by line. For each line read, it is determined whether the information is required to be stored by the program or not. If it is required, it then begins to go through reach word and determine if the word should be stored in the dictionary and posting list. If stop words are being used certain words would be skipped, and it also determines if the user asked for stemming. Once the program has decided what is happening, the term is first placed into the dictionary. If the term doesn't exist in the dictionary, it creates a new occurrence along with the frequency value of one. If it is already in the dictionary, the frequency value is increased by one. For the posting list, it has each term the dictionary has. The set up of the posting list is {term : {doc_num : [frequency, [position_in_text,...]], ...}}By creating the two files this way it creates a system that is similar to the inverted index. While all this is happening each word that is taken into consideration is also stored into a third output which takes each word for each document and places them in the order they occur in the document. It also separates them by title and abstract. Finally, all 3 lists are pickled to compress that data to be transferred to the next file, test.py.

How to Run Invert.py

Make sure the cacm.all file is in the same folder as invert.py, then run the file. The program will ask for user input twice. First one is asking for stemming, if you want to test using stemming type yes and then hit enter. If you don't want stemming, type no and hit enter. The second one is asking for stop words. Do the same as previously mentioned for stemming for the desired output. The program will finish and will output 3 files that will be needed for test.py.

Sarah MacCallum - 500684093 Maria Poveda - 500694400

Test.py

The program uses the 3 files created by invert.py. Each file is open for reading and the object is loaded into a variable that we will use throughout the program. The user is asked what term they want to query, once user input is entered a timer is started. If the query word is not “ZZEND” then we look for the query word in our dictionary. If we do not find the word then a “Not found” statement appears and the timer ends. We print how long the query took and add that value to a list that we later on use to calculate the average query time. If the word is found in the dictionary then we grab the frequency value that is associated with our query word and print it. We also use the query term to get the correct posting index and we iterate through the value associated with the key, in our case the query term. The value, as mentioned in invert.py description is {term : {doc_num : [frequency, [position_in_text,...]], ...}}. We use the posting list to print out the doc id, doc title, frequency of term in doc and position of appearance in doc and a summary of the doc where it shows the terms’ first appearance in the document.We use the third output file that was created in invert.py for the summary and context. If the id is in there we go to the first occurrence of the query word and get the 10 words surrounding it for context. Since the query is over the timer is stopped and we print how long the query took and we add it to our list containing query time. The user is once again asked to query a word and the process starts again. Once “ZZEND” is received as input the search is over, the average query time is calculated and printed.

How to Run Test.py

Make sure that you have already ran invert.py. From terminal make sure you are in the directory containing the files. Run python test.py and program will prompt you for input. From there submit your query word and when you are done type ZZEND.