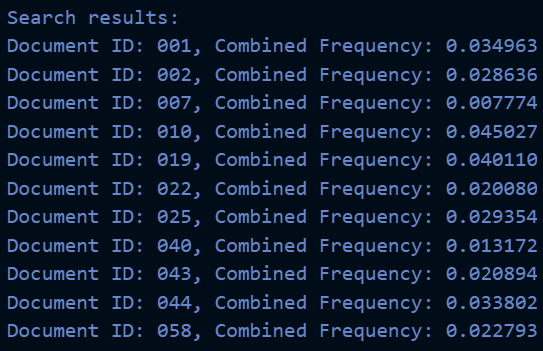
Kurt Drexel

Hw4

4/24/2024

For this homework assignment I decided to start from the point which was left off at in homework 3. I did this instead of creating a new program that would simply search through my output files (dictionary.txt and Postings.txt) because I thought it made the most sense to have to program start from the beginning on every run, in case new files are added to the corpus. While not necessary for this class as our corpus will always have a size of 503, I believed it would be a more crucial part when dealing with real world scenarios. When I began editing the code I simply began to add more code to the bottom as the current state of the code left from homework 3, left the ground work for this current assignment. The first step I began in the assignment was to iterate through my dictionary.txt for the specified *term* as I knew once the *term* was found that the amount of files the term was found in would be the next line, and the line after that would be the line within the postings.txt file that the *term’s* occurrences are noted. This was originally done for a single term before I began to worry about query’s from the user containing two terms such as “international affairs”. To begin searching for a term I first began with a print statement to the user asking for the term they wish to search for and taking input from the user. With this input I then open my dictionary.txt file in read mode. With my dictionary.txt file open I then use the readlines() function to read all lines from my dictionary file. I then use a for loop to iterate through each line, checking if each line is equal to the word the user Is searching for. If this work is found I then can get the number of documents containing the word and its position in the postings file. I do this as num\_docs\_containing\_term = int(lines[i + 1].strip()) which will go to the current line ( i ) and add 1 to it so that we will be at the position of the total number of documents it occurs in. I then get its position within the postings file in the similar manner as line\_in\_postings = int(lines[i + 2].strip()) in a similar manner to the occurrences I can simply add 2 to i to get the *terms* position within the postings file. For debug purposes I then printed out the found *term*, the number of documents it is found in, and the line in the postings file where its occurrences can be found. This was done so that when error checking I could compare the found values printed to the accurate values within my dictionary file. I then have a simple else statement with a print statement for if the term is not found within the dictionary, as debug or error checking tool. Once the *term* I then need to search through my postings file for the found term, using the num\_doc\_containing\_term and line\_in\_postings variable I found when iterating through the dictionary file. The line\_in\_postings variable will be used as that is the line within the postings file the program will need to start at for finding the documents the *term* occurs in. Then using the num\_docs\_containing\_term variable will be used as that will be the number of lines we need to get the information from (document id, weight of term in document). My first step in the postings.txt file is to open it in read mode. I then iterate through my postings file line by line, on each iteration I check if the current line number is greater than or equal to the line in postings variable found previously. Once we are greater than or equal to the line I then get the information on the current line and place it in a variable using postings\_data = line.strip().split('\n'), splitting the current line at the newline variable. I then iterate through my postings\_data variable and print its contents to screen displaying the document the term is found in and its weight for the document. I then do a simple if statement to allow for breaking out of the loop early to avoid un-necessary iterations. The if statement checks if the current line is greater than or equal to the line in the postings file where the word is found plus the number of documents containing the word, if the current line is greater than that the loop will break ending the A computer screen shot of a number

Description automatically generatedsearch. An example with aero is provided. I then altered my program to incorporate more than one word doing this I first split my input form the user at the space using the split function and adding a line before opening my dictionary that would iterate through the words provided by the user. I also created an list to store the found A screenshot of a computer

Description automatically generatedwords. Once the dictionary file was iterate through if the found words were equal to the split of the input from the user I would then iterate through my postings dictionary. Using the same search through the postings as before. This would display each word along with the number of documents it is found in and the line in the postings file it is found in. The program would then print out the documents each word is found in along with the weights for the word in the document. At this point there would be repeats in printouts for the documents for if each word was found in a similar document. An example is shown below with the words “International affairs”. Once I was able to successfully search for more then one word and produce the correct outputs for each word searched for I then altered the program so the weights of the words were added together if the words searched for were found in the same document. To do this I altered the program once more to now have a dictionary to contain the frequencies for the terms in each document stored under each document id. At this point my search through my dictionary file remained the same. My postings file search was altered so that within my if statement ensuring I was within the current lines to iterate through for each word I then needed to check if each document id was already within my dictionary to store the frequencies by document id. If the document id was not already within dictionary it would be added in along with the corresponding frequency. I then iterate through my dictionary combining the frequencies of the words if they both occur within the same dictionary. My final alteration for my program was a simple print statement at the bottom that would iterate through my final version of the dictionary, printing each document id and its respective frequency and to order the responses by the weights returned for the terms occurring in each document. A screenshot of a computer code

Description automatically generatedWhen running the program it will take on average 3 minutes to run to completion. Most of the 3 minute runtime stems from the iteration through all 503 files and the creation of the dictionary and the postings file. The actual finding of the term within the dictionary and returning its occurrences takes minimal time. The runtime of the newly added code can be broken down into several parts. The reading of the dictionary file will be O(n) with n representing the number of lines within the dictionary file. Searching through each dictionary and obtaining the information needed will take O(n\*m) where n is the number of lines in the dictionary file and M is the average number of documents each term is listed in. Iterating through my created dictionary and updating the frequencies for words that occur in the same document will take O(d) time where d is the number of keys (documents) within the dictionary. Finally sorting the words by their frequencies within my created dictionary in descending order will take O(d\* log d) time where d represents the number of documents. The combined runtime for my newly added code will be O(n) + O(n\*m) + O(d) + O(d\*logd). The most dominant part of my newly added code will be the part of iterating through my dictionary file and finding the relevant information for each term or terms being searched for by the user.

Diet

A screenshot of a computer

Description automatically generated

International affairs

A screenshot of a computer screen

Description automatically generated

Zimbabwe

A blue text on a black background

Description automatically generated

computer network

A screenshot of a computer program

Description automatically generated

Hydrotherapy

A screen shot of a computer

Description automatically generated

Identity Theft

A screenshot of a computer

Description automatically generated