CMSC 476 Information Retrieval: Phase 3 Report

Hafsa Chaudry

Department of Computer Science

and Electrical Engineering

University of Maryland

Baltimore County

***Objective*—This assignment is to update the previous homework and to add a dictionary output file and a postings output file**

# I. METHODS

1. *Testing Environment*

Throughout the experiment, one computer was used to run and test the data in order to ensure the accuracy and consistency of the results. The UMBC GL server was used to execute the algorithms for reasons of convenience and usability. The computer utilized for testing has the following specifications shown in Table 1 below.

*Table 1: Testing Computer Specifications*

|  |  |
| --- | --- |
| Processor | Intel Core i7 CPU 2.80 GHz |
| RAM | 16 GB |
| Operating System | Windows 64 Bit |

1. *Algorithms*

All algorithms were developed in Python and run using Python version 3.7.1. I utilized the python libraries from phase 1 and 2. In order to create the first output file, dictionary.txt, I realized that I had a variable called global\_word\_counter that has the type <collections.Counter>. Because it was a Counter, it had both the key (word), and the value (occurences of word within all documents). Using this knowledge, I had realized that the dictionary.txt file consists of three requirements: the word, the number of occurrences of the word, and the location in the posting file. The Counter solved two out of three requirements, and I realized that the third requirement’s first input would always have the posting position of 1 (counter = 1) and the rest of the inputs were simply the previous values + the counter of the previous key. To create the second output file, I needed to list all the documents that each word had along with it’s weight per document. In order to do this, I first checked if my word existed within the document, and if it did I returned the document and then calculated the weight of the word within the document. And repeated this for every occurrence of the word in every document, as well as the occurrences of every word within the Counter.

The command format was python phase3.py <input directory> <output directory>

1. *Issues*

My previous hw2 took more than an hour to run and would take too long to fix, and because of this I requested another’s students hw2 to continue working and completing hw3. But because the code was originally not my own, it took me a while to understand what each variable held and how to manipulate the code to get the same outcome I was previously getting as well as implementing my original algorithm using the new variables. Once I realized that I had all the necessary steps for dictionary.txt file, it was just a simple algorithm to add. But my postings file took a lot of debugging and kept giving me many errors. In the end, I was able to fix it but because the files are so big, I have to just assume my posting file has the correct output.

# II. RESULTS

1. *Dictionary.txt*

When I tried creating the Term Document Matrix for the first 3 words by hand, I got the first word to be nemzet and the occurrence of the word to be 430 from my Counter variable. Because it is the first word, the position should be 1. The next word in the Counter was “eeeeeee” which occurred 156 times, and so the position should be (430+1 = 431). Third word is “mm”, occurred 101 times, and the position should be (156+431 = 587), and so on and so forth for the rest of the words in the Counter. After running the code, my dictionary.txt outputted the correct values that I had received when doing it by hand.

nemzet  
430  
1

eeeeeee  
156  
431

mm  
101  
587

1. *Postings.txt*

My postings.txt correctly prints out the name of the file and the weight for the word in the file. Because I could not access all the variables, and I could not solve postings through hand, I am assuming my outputs are correct.

1. *Total Size of Output Files*

From my understanding, the size of the output files were larger than the size of the input files*.*

1. *Time vs Number Of Files*

As expected, the more files that were being processed correlated to the amount of time increasing to run.

A close up of text on a white background

Description automatically generated

*Figure 1: Time in milliseconds vs Number files*

# III. “Shell Code” Proof

