CMSC 476 Information Retrieval: Phase 4 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 search query**

# 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, 2, and 3. I extended my function “extract\_data” that extracts the html texts from the html files to also return doc\_freq\_per\_word, which is a document frequency hash map. In other words, the variable stores the token as a key and the corresponding html files that are associated with that token as a value. I then used .get() to extract the appropriate data per search query word.

I could not implement the actual algorithm correctly because I realized after that my postings.txt file was done incorrectly and was not able to fix it, but if my postings.txt was correctly implemented, then all I would have to do is to find the matching word from my query to my dictionary.txt file. Once I matched the word to the token in dictionary, the second line would give me how many times that word occurred and the third line would give me the first instance of that occurrence in my postings file. My postings file would have the html file names and the weight of that token per html file name. For example, my dictionary.txt file has the format for the token diet as:

Diet

13

160634

This means that my token diet should be first found on line 160634 in my postings file, and because it occurs 13 times, lines 160634 to 160647 should all be html to weight per html for the token “diet”. You would then take all 13 html id’s and weights, and return only the 10 highest weights + html id per weight.

The command format was python phase4.py <input directory> <output directory>

1. *Issues*

My previous homework 3 had tokenized all the words per space, so words like “international affairs” were not existent, since my tokenizer would see it as two tokens in one. And I could not figure out how to solve this without breaking my code.

1. *Complexity*

O(n+N) where n is length of the file\_lists and N is the number of all documents in the corpus.

# II. RESULTS

diet

{'252.html', '353.html', '263.html', '009.html', '018.html', '050.html', '152.html'}

\*\*\*\*\*\*\*\*

international

{'436.html', '001.html', '040.html', '375.html', '161.html', '461.html', '240.html', '434.html', '236.html', '239.html', '143.html', '333.html', '022.html', '152.html', '365.html', '296.html', '104.html', '433.html', '289.html', '181.html', '247.html', '492.html', '286.html', '117.html', '002.html', '043.html', '282.html', '308.html', '346.html', '351.html', '368.html', '298.html', '244.html', '377.html', '364.html', '246.html', '092.html', '243.html', '115.html', '287.html', '031.html', '242.html', '332.html', '189.html', '229.html', '044.html', '345.html', '058.html', '232.html', '334.html', '355.html', '221.html', '179.html', '331.html', '226.html', '184.html', '160.html', '125.html', '336.html', '025.html', '353.html', '108.html', '124.html', '235.html', '360.html', '249.html', '159.html', '437.html', '277.html', '426.html', '197.html', '222.html', '076.html', '350.html', '376.html', '136.html', '010.html', '339.html', '133.html', '347.html', '303.html', '280.html', '358.html', '007.html', '185.html', '370.html', '198.html', '215.html', '087.html', '205.html', '238.html', '269.html', '276.html', '419.html', '237.html', '366.html', '348.html', '361.html', '192.html', '211.html', '362.html', '446.html', '119.html', '279.html', '019.html', '138.html', '338.html', '335.html', '188.html', '435.html', '340.html', '156.html', '384.html', '174.html'}

\*\*\*\*\*\*\*\*

affairs

{'336.html', '351.html', '353.html', '233.html', '419.html', '235.html', '364.html', '219.html', '348.html', '426.html', '240.html', '129.html', '311.html', '242.html', '359.html', '284.html', '279.html', '019.html', '229.html', '295.html', '349.html', '133.html', '289.html', '181.html', '345.html', '232.html', '388.html', '286.html', '369.html', '340.html', '344.html', '331.html', '281.html', '226.html', '002.html', '389.html', '361.html', '282.html'}

\*\*\*\*\*\*\*\*

computer

{'269.html', '380.html', '124.html', '037.html', '159.html', '249.html', '437.html', '064.html', '223.html', '243.html', '434.html', '146.html', '076.html', '275.html', '267.html', '060.html', '030.html', '119.html', '022.html', '038.html', '027.html', '152.html', '393.html', '047.html', '433.html', '280.html', '317.html', '382.html', '388.html', '258.html', '078.html', '164.html', '247.html', '177.html', '145.html', '435.html', '272.html', '290.html', '156.html', '499.html', '226.html', '205.html', '002.html', '502.html', '204.html', '042.html'}

\*\*\*\*\*\*\*\*

network

{'436.html', '368.html', '501.html', '040.html', '437.html', '064.html', '161.html', '016.html', '315.html', '223.html', '197.html', '434.html', '140.html', '446.html', '022.html', '027.html', '376.html', '075.html', '019.html', '133.html', '047.html', '433.html', '128.html', '392.html', '135.html', '181.html', '388.html', '396.html', '164.html', '007.html', '145.html', '395.html', '435.html', '355.html', '221.html', '156.html', '292.html', '087.html', '043.html', '166.html', '290.html', '282.html'}

\*\*\*\*\*\*\*\*

hydrotherapy

{'273.html'}

\*\*\*\*\*\*\*\*

identity

{'245.html', '308.html', '298.html', '309.html', '235.html', '360.html', '348.html', '301.html', '243.html', '328.html', '332.html', '027.html', '304.html', '019.html', '303.html', '397.html', '391.html', '383.html', '307.html', '272.html', '043.html'}

\*\*\*\*\*\*\*\*

theft

{'292.html', '379.html', '380.html'}

\*\*\*\*\*\*\*\*

cats

{'436.html', '434.html', '433.html', '435.html', '437.html'}

\*\*\*\*\*\*\*\*

“Shell Code”:

A screenshot of a cell phone

Description automatically generated