**Information Retrieval Programming project -2  Report**

**Introduction:**

    The main aim of this project phase two  is to find out the term weight of the tokens in the documents collection. I have used python as programming language for the code. Windows Operating system 10 was utilized for the project. The following is the command to execute the code .

**$python calcwts.py files output**

where,

**calcwts.py – python code**

**files – input directory**

**output-output directory**

In addition to term weight calculation , I have calculated Best Matching 25 (BM25).

**Input :**

The inputs for the project are the 503 input html files which has to go some preprocessing to extract only the content leaving the html tags from those 503 files.  The content of the input files are tokenized and given as input for the term weight and BM25 calculation. The input files are from 001 to 503 html files with .html extension.

**Output:**

The output files are the .wts files which contains the term weights for each and every word along with the BM25 index for every word as each word per line in the corresponding output files. All the output files are directed to the corresponding output\_path directory . This output directory contains the corresponding output files with .wts extension for the input files with .html extension in the input directory.

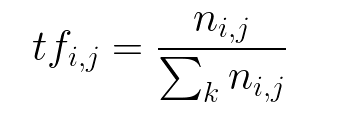
**Implementation work:**

The input files containing the html files are used to extract only the content of those pages by importing and utilizing the html2text python module. This extracted content is tokenized using nltk tokenizer available in python. In this programming project phase two , I have removed the list of stop words from the tokens by iterating it through the list of stop words. This improves the quality of tokens in the documents . The stop words removed tokens in the documents are stored in a nested dictionary containing files names as keys , values are tokens and frequency. This dictionary is  iterated and the length , average length and total length of the documents are calculated. I have implemented two separate classes one for tokenizing the documents and one for calculating the term weights and BM25 so that the program is modularized in a better way. The program is design using object oriented principles.

**Term weights formula:**

The term weight is calculated by the taking a product of term frequency and inverse document frequency.

Term frequency = Frequency of a term in a document /  total frequency of the words in all the documents



The above is the formula for the calculating the term frequency in the documents where

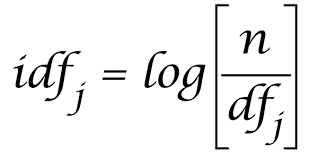
ni,j-number of terms i in a document of j

⅀kni,j -  Total frequency of words in all documents

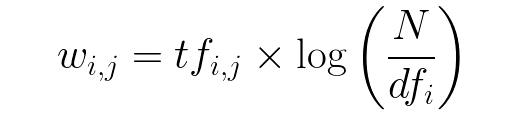
tf i,j- Term frequency

**Inverse Document Frequency :**

Inverse Document Frequency is the logarithmic division of n documents in the whole collection by the ni which is the number of documents in which the term occurs . Idf is computed for the entire corpus whereas tf is computed on a document by document basis.



The Term weighting product is calculated by taking the product of term frequency and the idf. The formula is shown below .



**Best Match 25 (BM25):**

The best match is found by using the following formula

https://lh3.googleusercontent.com/S_Oqx09bohGx3Ah3ovDkQvtoQ9QJO96j4qK4dN5L5bnsfNfp7nTEgIZLwiAdNvX2oXLNR_ySlYUut8iBEX6z4oaVXGm5TFZc3UgNPYOMi_f4TobuBfBetU-5AdVuj3Dza949RaBu

where fi,j - term frequency factor

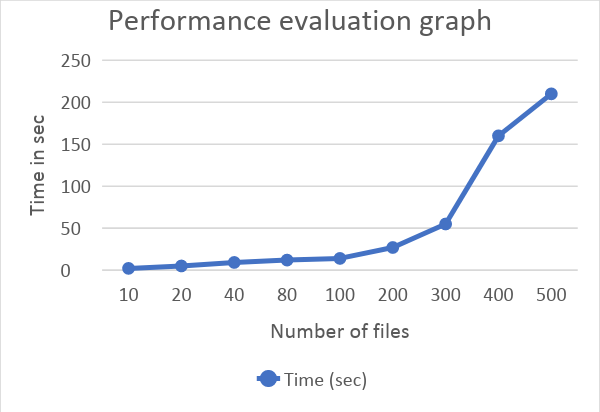
avg\_doclen - is the average length of the documents

K1 = 1

S1= K1+ 1

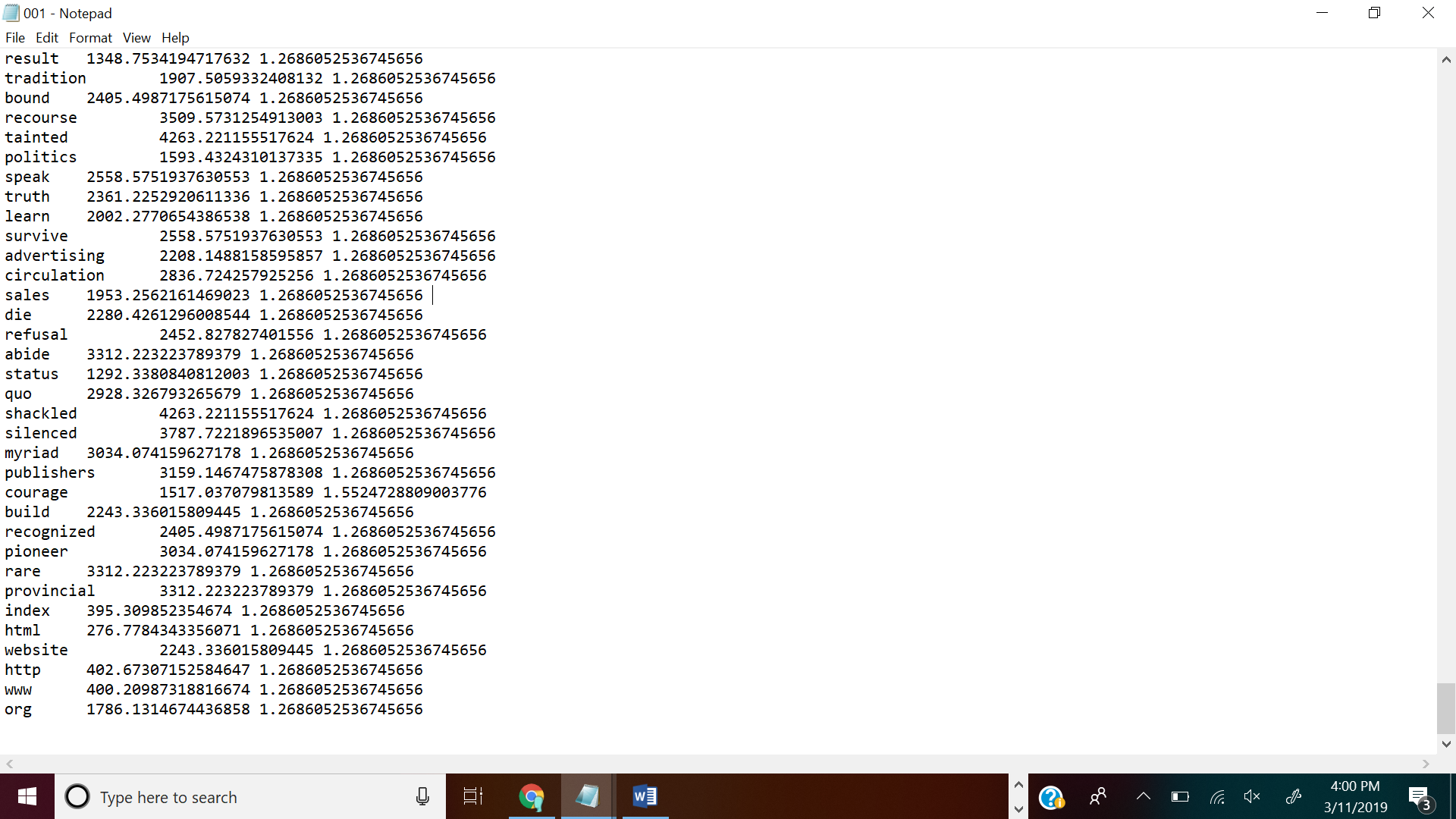
**Performance Evaluation:**

The performance of the code was evaluated by considering the time between the start time and end time. I have considered the time taken by 10 , 20 , 40, 80, 100, 200, 300 , 400 , 500 files . It is clearly seen that the graph grows linearly for the addition of the performance of the files from 001 to 500 files in the output directory .The x axis denotes the number of files and the y axis denotes the execution time in seconds . The graph for the performance evaluation is shown below



**Result:**

This is the result of file 001 . All the output files from 001 to 503 are in the output directory which is in the zip file.



**Conclusion :**

We can see clearly from the graph that the time grows linearly and sub-linearly based on the increase in number of files. Sorting of files are done using merge sort .Hence the total time that must have been taken would be O(nlogn).