

Information Retrieval (CMSC 476/676)

Homework 2 – Report

Done by – Anushka Dhekne (CMSC VD19739)

The objective of this assignment is to remove the stopwords and calculate term weights of the tokens that appear in each of the file in the set of documents. The input directory contains the set of tokenized files (from Homework 1) and the output directory contains 1 output file per input file and includes the tokens and their respective weights calculated using tf-idf and bm25. I have used Python programming language to complete this assignment and Visual Studio Code as the code editor. This report also discusses the improvements in implementation while processing the documents from Homework1 to Homework2. It also includes examples of 2 input documents, their respective output files showing the term weights, the graph showing time required to process each set of documents and the efficiency of the overall program.

The first function – ‘preprocessing_text’ is used to convert all text from the input file to lowercase and then remove stopwords from it. The function removes all words given in the list of stopwords, which is hardcoded in the program. Apart from this, it removes all words which are of length 1 and words which appear only once in the entire corpus of documents. The next function calculates the token weights using the TF-IDF and BM25 algorithms. Here, the time calculation for the entire process begins. The system time is started using the ‘time’ Python library. The function iterates over each file in the set of documents provided in the input directory. It extracts the contents of the file which is in the .txt format and tokenizes it using the ‘preprocessing_text’ function . It then computes the document frequencies and calculates document lengths for further calculations related to TF-IDF and BM25.

The TF-IDF scores which are computed for each token are based on the token frequency in the document. The TF (term frequency) is then calculated in each document by dividing the frequency of the token in the document by the document length (number of tokens in the document), and it also calculates Inverse Document Frequency (IDF) values for each token from the corpus. Finally, the TF-IDF scores are calculated by multiplying the value of TF with the value of IDF for that token in the document.

The BM25 algorithm works in a similar way, making use of the IDF values. It also uses term frequency, document length, average document length in the corpus and constants like b and k1. I have taken the values of constant b=0.75 and constant k1=1.5. Thus, BM25 focuses on the document length and term frequency normalization for giving accurate scores on token weights, whereas TF-IDF focuses on frequency of terms which appear frequently in each document, but not in the entire corpus of documents.

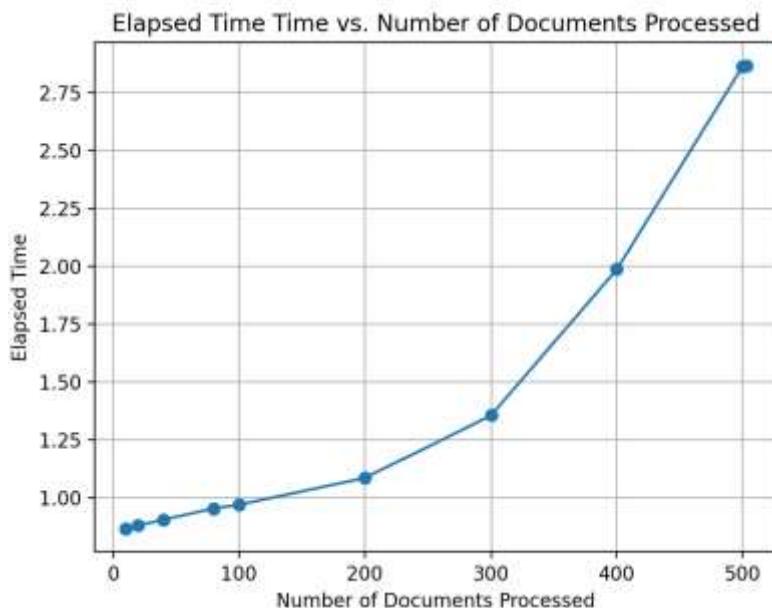
The scores calculated using both the algorithms are then stored in the output files along with the token name. These scores or token weights are written to its respective output files, which are saved in the ‘.wts’ format. The file reading and writing operations take place using the ‘os’ library, which allows performing the file operations seamlessly.

Next, we check if the given number of documents have been correctly processed and if the time elapsed to calculate the processing time for each document has been correctly recorded. The time required to process each file (partial time) is recorded and then subsequently, it is appended to the total elapsed time to calculate the time required to process each set of documents. The set of documents contains – 10, 20, 40, 80, 100, 200, 300, 400 and 500 documents. This output is also shown by plotting a graph of ‘Number of Documents Processed’ versus ‘Elapsed Time’. Finally, after all 503 documents’ process time has been appended, the total time documents are recorded to give the output of the code as - ‘Total time for processing all 503 documents.

Homework1 completed the preprocessing using the regex and NLTK libraries to tokenize the words. Homework2 completed the preprocessing by removing the stopwords and unnecessary words, which do not contribute much to the meaning of the text and then, finding the token weights. Thus, the preprocessing step was enhanced in Homework2, which is now giving a more clear, readable, accurate and efficient result. The code was optimized as well by minimizing redundant operations.

The code efficiency was calculated using the ‘time’ Python library. If we consider ‘m’ to be the number of documents, ‘n’ to be the length of the documents and ‘k’ to be the average number of tokens per documents, the time complexity of tokenization is $O(m*n)$, the complexity of the algorithm calculation – TF-IDF and BM25 is $O(m*k)$ and hence, the overall time complexity of the program becomes $O(m*(n+k))$. Overall, the graph shows that the document processing takes place in linear time and the graph shows a linearly increasing trend.

The graph generated by the program is shown below –



The graph shows the relationship between the Number of Documents Processed against the Elapsed Time to process a set of 10,20,40,80,100,200,300,400,500 and 503 documents. The trend of the graph shows that the algorithm is suitable of handling large files and/or a large corpus of files.

The result of the program which shows the total time required to process 503 documents is given below. Once we close the window of the above graph, we get the result as shown below.

```
PS D:\SEM_4\Project\phase2\final_submission> python Homework2_AnushkaDhekne.py D:\SEM_4\Project\phase2\final_submission\input_files D:\SEM_4\Project\phase2\final_submission\output_files
Total time for processing all 503 documents: 2.87 seconds
PS D:\SEM_4\Project\phase2\final_submission>
```

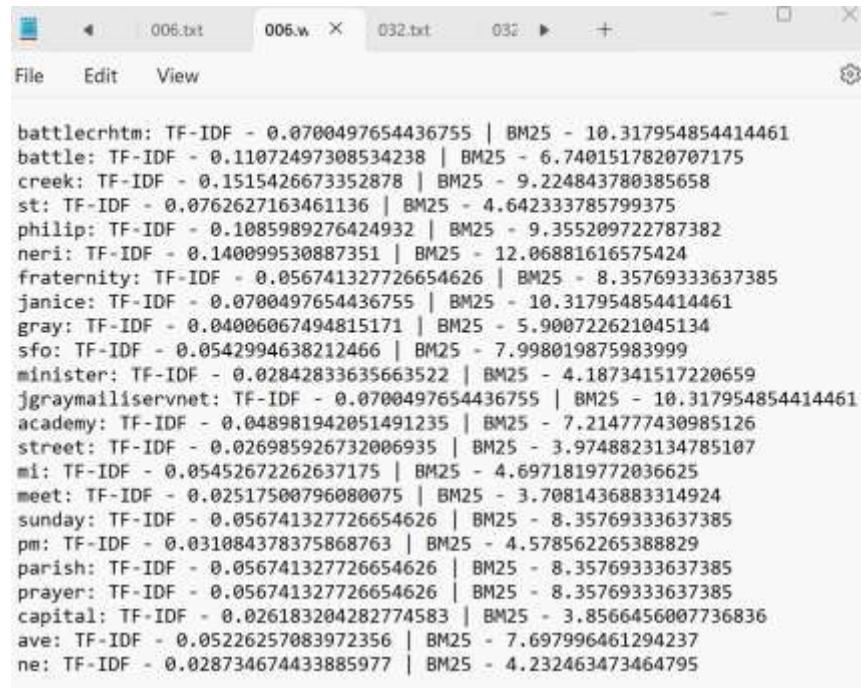
The following images show the input .txt file and the result output file in .wts format.

Input File – 006.txt



```
battlecrhtm battle creek st philip neri fraternity mrs janice
gray sfo minister jgraymailiservnet academy street battle creek
mi meet st sunday at pm st philip neri parish prayer room
capital ave ne battle creek mi
```

Output File – 006.wts



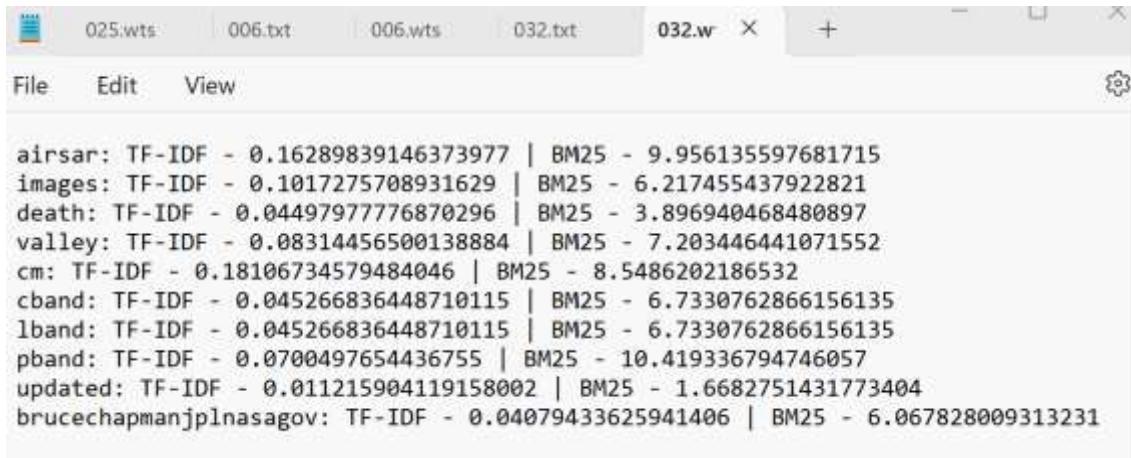
Term	TF-IDF	BM25
battlecrhtm	0.0700497654436755	10.317954854414461
battle	0.11072497308534238	6.7401517820707175
creek	0.1515426673352878	9.224843780385658
st	0.0762627163461136	4.642333785799375
philip	0.1085989276424932	9.355209722787382
neri	0.140099530887351	12.06881616575424
fraternity	0.056741327726654626	8.35769333637385
janice	0.0700497654436755	10.317954854414461
gray	0.04006067494815171	5.900722621845134
sfo	0.0542994638212466	7.998019875983999
minister	0.02842833635663522	4.187341517220659
jgraymailiservnet	0.0700497654436755	10.317954854414461
academy	0.048981942851491235	7.214777430985126
street	0.026985926732006935	3.9748823134785107
mi	0.05452672262637175	4.6971819772036625
meet	0.02517500796080075	3.7081436883314924
sunday	0.056741327726654626	8.35769333637385
pm	0.031084378375868763	4.578562265388829
parish	0.056741327726654626	8.35769333637385
prayer	0.056741327726654626	8.35769333637385
capital	0.026183204282774583	3.8566456007736836
ave	0.05226257083972356	7.697996461294237
ne	0.028734674433885977	4.232463473464795

Input File – 032.txt



```
airsar images of death valley airsar images of death valley airsar cm images
cm cband k cm lband k cm pband k updated brucechapmanjplnasagov
```

Output File – 032.wts



```
airsar: TF-IDF - 0.16289839146373977 | BM25 - 9.956135597681715
images: TF-IDF - 0.1017275708931629 | BM25 - 6.217455437922821
death: TF-IDF - 0.04497977776870296 | BM25 - 3.896940468480897
valley: TF-IDF - 0.08314456500138884 | BM25 - 7.203446441071552
cm: TF-IDF - 0.18106734579484046 | BM25 - 8.5486202186532
cband: TF-IDF - 0.045266836448710115 | BM25 - 6.7330762866156135
lband: TF-IDF - 0.045266836448710115 | BM25 - 6.7330762866156135
pband: TF-IDF - 0.0700497654436755 | BM25 - 10.419336794746057
updated: TF-IDF - 0.011215904119158002 | BM25 - 1.6682751431773404
brucechapmanjplnasagov: TF-IDF - 0.04079433625941406 | BM25 - 6.067828009313231
```

The TF-IDF and BM25 algorithms provide a clean, accurate and reliable solution for data tokenizing, weight calculation and data representation as well. Thus, the preprocessing steps and the efficient algorithm selection has enabled to understand the significance of term weights and has also enhanced the way of document representation.