Information Retrieval Phase 2 Report

PROGRAM STRUCTURE

1. Open and read through all the html files
2. Using BeautifulSoup library to parse through all the .html files and extract only the text from it
3. Tokenise the extracted words
4. Handles cases such as special characters, numerical inside a word, and also decimal numbers
5. Converts all the words to lower case
6. Reads the text file stopwords.txt and stores the given specific stop words in a list
7. The extracted words are now passed into a counter to get the word and its frequency value
8. Checks the given conditions that is
    a. Words of length 1
    b. Words with frequency 1
    c. Words that match the stop words
9. Updates the counter filtering out the words from those three conditions
10. Computes the term frequency and stores the value in a global dictionary with file name as key and another dictionary as value, containing word and tf
11. Computes the value that gives the occurrence count of any given word in number of files
12. Computes the inverse document frequency and stores the value in a global dictionary with file name as key and another dictionary containing word and idf as value
13. With these values the tf-idf is calculated for each word in the global dictionary

EXPLANATION

The program is built on top of the previous submitted logic with few minor changes and with better efficiency.

The developed program successfully pareses through all of the html files with the help of beautiful soup library and tokenises all of the words. The main basis of tokenisation is spacing. The inbuilt function '.split' is used to split the words extracted.

The program handles punctuations and special characters using regular expression. We give a set of predefined symbols that is filtered out of the extracted and tokenised words. Then all the tokenised words are converted to lower case.

The program then reads the given set of stop words from a text files and filters out the words based on the three given conditions. Then the counter is updated accordingly with the new values. The globally declared dictionary has the file name as the key and a counter of word and its frequency as its value.

The **compute_tf** function in the program then takes this dictionary as the parameter and computes the term frequency. It first computes the word count of every file and stores the file name and its word count in a dictionary. Then with the word count of that word and total count of words in that file term frequency is calculated. This is done for each and every word in the dictionary and the dictionary is accordingly updated with the new set of values.

Then the **doc_containing** function takes a word as its parameter and calculates the count of documents that word exists in and returns the count.

The **compute_idf** takes the dictionary as an argument and computes the inverse document frequency of the words. It first checks the number of documents to be checked and stores that length in the variable num_doc. For every word in the dictionary it calculates idf with the formula, log of number of documents divided by the word count(calculated using the doc_containing function). With these values it updates another dictionary with the key as file name and another dictionary as its value with the word and its idf.

The last function **compute_tf_idf** takes the dictionary as its argument and then multiplies the values of tf and idf to compute the tf-idf of every word. It then writes the obtained value to a text file.

The program runs through completion in about 10 seconds.

RESULTS

The result of the program is that it creates a text file with word and its tf-idf value for every html file.

Here is an example of one of the output file.

```
created : 0.027185456426151628
accept : 0.03946363925697836
appropriate : 0.03198371491514852
authority : 0.2392523393030253
authoritys : 0.07617049187877231
companies : 0.02301545133418436
compliance : 0.13812273739942424
computer : 0.039488055876070384
containers : 0.07617049187877231
county : 0.0993481253796504
deliver : 0.04512198358807996
delivered : 0.06768297538211993
delivery : 0.03946363925697836
efficient : 0.03946363925697836
ensure : 0.030495375676014198
facilities : 0.18064090416489273
facility : 0.05968393163490943
farm : 0.03489534711056674
frey : 0.04512198358807996
generated : 0.037642059246250116
haul : 0.07617049187877231
hauler : 0.12695081979795386
haulers : 0.13536595076423985
homes : 0.037642059246250116
information : 0.019418708349429836
```

FORMULAS IMPLEMENTED

1. compute_tf() : number of times a word appears in a document dividing by the total number of words.
2. computer_idf() : log of total number of documents to the number of documents containing word.
3. compute_tf_idf() : product of tf and idf.

EXECUTION

The program was coded in PyCharm. It can be run on any IDE that supports Python.

Imports,

1. Import os
2. Import glob
3. Import math
4. from collections import Counter
5. from bs4 import BeautifulSoup
6. Import time