**Introduction to Computers (CS 150)**

**Final project details and guidelines**

Total marks 60

This project is mainly related to all the concepts we learnt and practiced in class. While Python language is well known to be used in data science related problems so I choose something called “**Topic modeling**” in data science. Topic modeling is part of “**Natural Language Processing**”. We will work with data problem and use read concepts to solve it.

Here is further study about Topic modeling and NLP: <https://monkeylearn.com/blog/introduction-to-topic-modeling/>

Please follow the instructions given to complete the final project.

1. You need to write the function **read\_DataFiles()** which will ask use to provide file paths to be read. User can choose to read as many **data files** to read as many he wants.

\* You can write a loop and save the file objects in a list to manipulate later

\* Data files will contain words or textual data to be explored

1. After reading data files you need to write a function **read\_StopWords()** which will ask user to provide file path to stop words file.

\* Stop words are usually part of speech like is, a, the. They are not meaningful in topic modeling yet are part of speech. So, we remove them from textual data for topic modeling.

\* Before removing stop words from words lists you will make sure that words are punctuation free for example Python, Python: and Python# are same.

\*You can use following lost of stop words for test purposes <https://gist.github.com/larsyencken/1440509>

1. Next you need to write a function **tfidf\_Caulculator(dataFiles[], stowordsFile)** the frequency of each word excluding stop words in every data file. This calculation is referred as TF-IDF or Term frequency- inverse document frequency (Though its bit complex we will stick with basic one).

For example, you are given 3 data files and 1 stop word file:

***Data file 1:*** Hello this is Python

***Data file 2:*** This is Python

***Data file 3:*** Introduction to Python

***Stop words file:*** is to this

TF-IDF will look like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Word | Files 1 | File 2 | File 3 |
| Hello | 1 | 0 | 0 |
| Python | 1 | 1 | 1 |
| Introduction | 0 | 0 | 1 |

You need to think how you will maintain word and their frequency in a suitable data structure e.g., List, Dictionary, Tuple etc.

1. Now you need to write a function **get\_Topics()** which will ask user to provide number of **topics** he wants to get out of TF-IDF. For example, from above matrix if user wants only 1 topic. You will calculate over all appearance of every word in all documents like this:

|  |  |
| --- | --- |
| Word | Overall |
| Hello | 1 |
| Python | 3 |
| Introduction | 1 |

For above resultant matrix “**Python**” is the topic.

\* You need to think of situation where two or more topics have same highest frequency? You will define some strategy to pick one based on criteria etc. For example, choose word which is discussed in most documents over the one discussed only in one document.

\*You can see that how topic modeling was effective to determine topics that were being discussed in text files regardless of their size. Because files can contain millions of words which obviously cannot be looked in manually.

\* Make sure user enters # of topics likely to be extracted correctly. It should be less than total # of topics available.

1. You need to write topic in a file “Topics.txt” at user defined location using **resultant\_Topics()** function

**Some general instructions:**

You will do all of your work in functions and will write main function to go through process step by step. Or main snippet of code which will call other functions in sequence.

Your functions will have all possible exception handling. For example, what is user provides wrong file path? What if someone provides a file that end up on no topics? Empty file? Etc.

You will provide comments everywhere in your code to explain the logic

\* There will be extra credits for those who will filter words based on context like word Play and Playing is same, likewise word Program and Programs is same. However, words from text file will be extracted separately like Play, Playing, Program, Programs

\* Likewise, there will be extra credits for Test oriented development using assertions.