CS 6103D Software Systems Lab
Python Programming

Final Evaluation: 17.12.2021 **Time: 90 Mins** **Maximum Marks: 8**

Q. **Topic Analysis is a technique used in Natural Language Processing to extract certain meaningful insights from textual data. These data are generally retrieved from online sources like film reviews, tweets, article abstracts, etc. In this evaluation, you are given four text documents, viz., Doc1, Doc2, Doc3, and Doc4, that contain film/item reviews. You are expected to find the similarity-degree between these reviews following a sequence of well-defined steps. Marks for each step are annotated along with the details of the corresponding step.**

IMPORTANT NOTE: ***Use only pandas, numpy, and re packages to implement this problem***

The following THREE steps are involved in solving the problem:

1. Read the reviews given in the four different text files into 4 different strings:

Reading a text document into a python string variable:

Suppose doc.txt is a text file. The code given in RED can be used to store the contents of the file into a string variable in Python.

```
file = open('doc.txt')
f = file.read()
file.close()
```

Note: In this case, **f** is the string variable containing the contents of the doc.txt file.

For the sake of explaining the different steps involved in solving this problem, let us take three toy(/example) text documents, *viz.*, Doc1, Doc2, and Doc3, containing the following reviews:

- **Doc1 = 'Peaky Blinders: thrilling tv series!!!'**
- **Doc2 = 'Game of Thrones, & Peaky blinders rated high among the tv series: 2014..'**
- **Doc3 = 'TV series these days are far better than movies!'**

After reading all the text documents into strings, perform the following operations:

- *Tokenization:* From the different strings, remove all punctuation (*i.e.*, only keep the alphabets, numbers, and whitespaces) and convert all alphabets to lowercase as part of text pre-processing. Transform the strings into lists composed of words (words are split from the string using blank space(s)). **(1 Mark)**

  For example, after tokenization, the string corresponding to the three documents, Doc1, Doc2 and Doc3, will be transformed to the following three lists, respectively:
  ['peaky', 'blinders', 'thrilling', 'tv', 'series']
  ['game', 'of', 'thrones', 'peaky', 'blinders', 'rated', 'high', 'among', 'the', 'tv', 'series', '2014']
  ['tv', 'series', 'these', 'days', 'are', 'far', 'better', 'than', 'movies']

- *Vocabulary building:* Combine all tokens(words) from all the different lists and store them into a single vocabulary, preferably a numpy array or a set, since each token is stored only once in the vocabulary. **(1 Mark)**

  For example, the vocabulary for the three toy text documents is given below:

['2014' 'among' 'are' 'better' 'blinders' 'days' 'far' 'game' 'high' 'movies' 'of' 'peaky' 'rated' 'series' 'than' 'the' 'these' 'thrilling' 'thrones' 'tv']

2. *Frequency table:*
   - Create a dictionary corresponding to each of the lists, (associated with a particular review), where the keys are the tokens in the vocabulary and the key's(/token's) value is the frequency of that token(key) in the corresponding review.
     (2 Marks)

     For example, the dictionary corresponding to Doc1 is as follows:
     {'2014': 0, 'among': 0, 'are': 0, 'better': 0, 'blinders': 1, 'days': 0, 'far': 0, 'game': 0, 'high': 0, 'movies': 0, 'of': 0, 'peaky': 1, 'rated': 0, 'series': 1, 'than': 0, 'the': 0, 'these': 0, 'thrilling': 1, 'thrones': 0, 'tv': 1}

   - *Combine all the dictionary information into a single dataframe*: The columns of the dataframe are the tokens in the vocabulary and a particular row corresponds to the number of times these tokens are present in a particular review. **(1 Mark)**

     For example, the dataframe corresponding to the three dictionaries is as given below:

	2014	among	are	better	blinders	days	far	game	high	movies	of	peaky	rated	series	than	the	these	thrilling	thrones	tv
0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1
1	1	1	0	0	1	0	0	1	1	0	1	1	1	1	0	1	0	0	1	1
2	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	0	1	0	0	1

3. *Calculate the similarity between the reviews:*
   - Assume each row of the above-mentioned dataframe as a vector. Now, in order to find the similarity between any two vectors(/reviews) A and B, you can use the cosine distance between them, *i.e.*, similarity(A, B) = A.B / (||A|| * ||B||)

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where,

A.B = dot product of vectors ‘A’ and ‘B’
||A|| = length of the vector ‘A’ = L^2 norm of the vector A
||B|| = length of the vector ‘B’ = L^2 norm of the vector B
- NOTE: L^2 norm of a vector A can be calculated using numpy.linalg.norm(A)
- NOTE: Function to calculate dot product of two vectors is present in numpy

||A|| * ||B|| = cross product of the two vectors ‘A’ and ‘B’

Using the above formula, calculate the similarity between all pairs of reviews(/vectors) and store it in a square matrix. **(3 Marks)**

In the given example, the similarity matrix for the 3 reviews is given below:

```
[[1.         0.51639778 0.2981424 ]
 [0.51639778 1.         0.19245009]
 [0.2981424  0.19245009 1.        ]]
```

Note: Here, the similarity between the same documents is 1, the similarity between Doc1 and Doc2 is 0.51639778, between Doc2 and Doc3, is 0.19245009, and so on.