Assignment 2

CS635

September 2020

INSTRUCTIONS:

1. The assignment2 folder consists of a folder named `documents` along with the current PDF.
2. The `documents` folder consists of 16 documents, named d1, d2 ... d16, where for each dx, x denotes the document ID for that particular document.
3. The two questions mentioned below, refer to these documents, in their respective task descriptions.
4. The root folder to be uploaded on Moodle should be strictly named: **rollno-assignment2.tar.gz** (eg: 12345-assignment2.tar.gz).
5. The below mentioned folder structure should be strictly followed (as it will be auto-graded):
    - `rollno-assignment2` (root folder containing two child folders mentioned below)
        - q1 (child folder containing two files mentioned below)
            - q1.csv
            - q1.py
        - q2 (child folder containing two files mentioned below)
            - q2.csv
            - q2.py

QUESTIONS:

1. There are N document text files contained in the folder (documents/). Create a dictionary of lists, each element of the list is a pair of the form: DocId-Frequency of this word (key of this dictionary element, in this particular Document), and return a csv file for following columns.

   CSV Representation: (to be generated from your code)

   Word, DocumentID-Frequency of the word
   apple, 1-27 2-11 3-17 4-1
   cat, 3-21 8-32
   cats, 4-13 7-29 10-7
   tall, 5-3 7-10 8-19

   The element: **cat, 3-21 8-32** means that the word ***cat*** is present 21 and 32 times in the documents with ID 3 and 8 respectively,

   Tabular representation (just for better understanding and visualization of the task):

Word(in lower case)	Doc-ID(in ascending order) along with their respective frequencies
apple	1-27 4-11 5-17 8-1
cat	3-21 8-32
cats	4-13 7-29 10-7
tall	5-3 7-10 8-19

   Note:

   - ignore the document IDs for which frequency is 0.
   - For this task, the words cat and cats are two different elements of the dictionary and occurrence of ***cats*** shall **not** be considered while considering ***cat***.
   - Words should be in lexicographical order (Ex: apple, cat, catty, tall...)
   - Steps before building the dictonary:
     (a) Fetch the Document
     (b) Remove all the punctuation: full stop, question mark, exclamation point, comma, semicolon, colon, dash, hyphen, parentheses, brackets, braces, apostrophe, quotation marks, ellipsis, etc. (i.e. **consider only words**)
     Help: Use any python library to perform this task

2. There are N document text files contained in the folder (documents/). Create an inverted index using all N documents and return a csv file for following columns.

   CSV Representation:
   ID, Word, Document ID
   1, apple, 1 2 3 4
   2, cat, 3 4 5
   3, mango, 5 14 123

   Tabular representation (just for better understanding and visualization of the task):

ID	Word(in lower case)	Doc-ID(in ascending order)
1	apple	1 2 50
2	cat	3 50 30
3	tall	5 20 120

   Note:

   - ID in lexicographical order (Ex: apple, cat, tall...)
   - Steps before building an inverted index:
     (a) Fetch the Document
     (b) Remove all the punctuation: full stop, question mark, exclamation point, comma, semicolon, colon, dash, hyphen, parentheses, brackets, braces, apostrophe, quotation marks, ellipsis etc. (i.e. Only words)
     Help: Use python library

     (c) Removing of Stop Words:
     Help: Use python library

     ```
     import nltk
     from nltk.corpus import stopwords
     print(stopwords.words('english'))
     ```

     (d) Stemming of Root Word - Use NLTK library.
     Whenever I want to search for "cat", I want to see a document that has information about it. But the word present in the document is called "cats" or "catty" instead of "cat". To relate the both words, I'll chop some part of each and every word I read so that I could get the "root word".

     Help: Use python library

     ```
     from nltk.stem import PorterStemmer
     print (PorterStemmer.stem("cats"))
     ```