Assignment 2

Ranked Retrieval for Free Text Queries (using tf-idf)

10th Sept 2019

This assignment is on building tf-idf based ranked retrieval system to answer free text queries. It is highly recommended that you use python for this assignment as libraries like *nltk* will make many things easier (stop word removal and lemmatization). However, if you use any other language, you most probably have to design these modules yourselves which might not perform as good as *nltk* library in python.

Dataset:

You can find the dataset in the following link
https://drive.google.com/file/d/1dzWCULwkgqlkQIaBaCLfdr3RwhwC1-Gz/view?usp=sharing
The dataset contains 3253 text files each containing different English news articles on politics and religion.

Task 1 (Building Intermediate Inverted Index)

1. Remove stop words, punctuation marks, make everything to lowercase and perform lemmatization to generate tokens from the document (use nltk library in python).
2. Build Inverted Positional Index (Dictionary with <**term**, **idf>** as keys, and <**file_name,tf**> as postings)
3. "**idf**" of a token is the number of documents in which the corresponding token appears (which is basically the length of its postings list).

   i.e., $idf(t)$ = number of documents in which term **t** appears
4. "**tf**" of a token in a document or file is the number of times the corresponding term appears in that document.

   i.e., $tf(t, d)$ = frequency of the term **t** in document **d**
5. This intermediate index will help us in creating the final TF-IDF vectors for each document.

Task 2 (Creating TF-IDF vectors)

1. The intermediate inverted index created above will be used in creating the TF-IDF vectors. We will create such vectors for each of the documents.
2. First we will create the TFIDF weights for each term-document pair.
3. For a document **d** the TFIDF vector will contain all the weights corresponding to all the vocabulary terms.
4. $TF(t, d) = 1 + log_{10}(tf(t, d))$

5. $IDF(t, d) = log_{10}(\frac{N}{idf(t)})$; where N = total number of documents
6. $W(t, d) = TF(t, d) \times IDF(t)$: This is the TFIDF weight.
7. Vector of document d: $[d] = [W(t, d) \; \forall \; t \; in \; V]$: where *V* is the vocabulary. Do length normalization (or **unit normalization**) on this vector so that each of the document vectors $[d]$ become **unit vectors**.

Task 3 (Answering free text queries)

The queries to be answered are free text queries. Use the same stop-word removal and lemmatization on the queries and find the **top-10 relevant documents** using the following steps.

1. Remove stop words, punctuation marks, make all lowercase and then apply lemmatization on the query text.
2. Let the resulting query after the first step be **Q**.
3. Now find the scores for each of the document using the formula below

$$SCORE(Q, d) = \sum_{t \; in \; Q} W(t, d)$$

Use these scores and find only the **top-10 scoring documents** for the results to the query.

Important Instructions on How to write the code and How to submit

1. Reading the dataset: Your code should read the dataset to create the intermediate index and then the vectors for each document. Assume the dataset to be in the path "./IR_Assignment_Dataset", i.e., the dataset folder is in the same path as your python code.
2. Naming the code file: The name of the code file should be in uppercase letters as below.
   ASSIGNMENT2_<ROLLNO>.py
   e.g. :- For a student with roll no 17CS92R02, the code file name should be **"ASSIGNMENT2_17CS92R02.py"**
   You need to send you final code through email to:
   patrogourab@gmail.com or paramita2000@gmail.com .
3. Reading the queries: Write code which can take "query.txt" file as an argument as below.
   $>> python code.py query.txt
   (query.txt file will contain many queries in free text format for example-- "religion

good or bad". There will be one query in each line. This file will remain unknown to you. Your program will be evaluated based on the results it produces for the queries in the above file.)

4. Saving the search results: Your program should read the queries one by one and get the search results. At the end it should create a text file with results where each line should contain the results of corresponding line's query in the query file. The name of the results file should follow the below convention.
   RESULTS2_<ROLLNO>.txt
   e.g. :- **"RESULTS2_17CS92R02.txt"**
5. Python library restrictions: You can use python libraries like nltk, numpy, os, sys, collections, etc. However, you can't use libraries like lucene, elasticsearch, or any other search api. If your code is found to use any of such libraries, you will be awarded with zero marks for this assignment without any evaluation.
6. Plagiarism Rules: If your code matches (more than 50%) with another student's code, all those students whose codes match will be awarded with zero marks without any evaluation. Therefore, it is your responsibility to ensure you neither copy anyone's code nor anyone is able to copy your code.
7. Code error: If your code doesn't run or gives error while running, you will be awarded with zero mark.