Assignment 1

Inverted Index, Boolean Document Retrieval

[Deadline: 15.09.2023, 11:59 PM]

This assignment is on building a simple inverted index and using it to retrieve documents. You have to use python for this assignment, since python provides many features that will ease up your workload, as compared to other programming languages like C++.

Dataset:

The data for this assignment can be found at this link (CRAN Folder):
https://drive.google.com/drive/folders/19C-ItAbeYCj3VgIdI_KGAQRkQD4qeHpm?usp=sharing

You will require two files for this assignment:

- ***cran.all.1400:*** This is the main document file, containing the information for 1400 documents. To parse each document, you must read, sequentially, the following records from the file.
  - Each document starts with the **.I** field, indicating the ID
  - Followed by the **.T** field, indicating the title
  - Followed by the **.A** field, indicating the author
  - Followed by the **.B** field, indicating the source/location
  - Followed by the **.W** field, which actually contains the text of the document

```
.I 3
.T
the boundary layer in simple shear flow past a flat plate .
.A
m. b. glauert
.B
department of mathematics, university of manchester, manchester,
england
.W
the boundary layer in simple shear flow past a flat plate .
the boundary-layer equations are presented for steady
incompressible flow with no pressure gradient .
```

- ***cran.qry:*** This is the main query file, containing the information for 225 queries. To parse each query, you must read, sequentially, the following records from the file.
  - Each query starts with the **.I** field, indicating the ID
  - Followed by the **.W** field, which actually contains the text of the query

```
.I 001
.W
what similarity laws must be obeyed when constructing aeroelastic models
of heated high speed aircraft .
```

Task A (Building Index)

1. Extract the text in the **.W** field of each document to use as your overall document (Discard the title field). Extract the document ID in the **.I** field.

2. Remove stop words, punctuation marks and perform lemmatization (without POS tags) to generate tokens from the corpus. (you can use **nltk/spaCy** library in python)

3. Build Inverted Index (Dictionary with tokens as keys, and document IDs as postings)

4. Save your Inverted Index in the main code directory as ***model_queries_<ROLL_NO>.bin*** using Pickle (binary)

5. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_indexer.py

6. Running the file : Your code should take the path to the dataset as input and it should run in the following manner:
   $>> python Assignment1_<ROLL_NO>_indexer.py <path to the CRAN folder>

Task B (Query Preprocessing)

1. Extract the text in the **.W** field of each query to use as your overall query. Extract the query ID in the **.I** field.

2. Remove stop words, punctuation marks and perform lemmatization (without POS tags) to generate tokens from the corpus. (you can use **nltk/spaCy** library in python)

3. Save the list of queries as:
   <query id> [TAB] <query text>
   in a .txt file in your main code directory. Name the .txt file as : ***queries_<ROLL_NO>.txt***

4. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_parser.py

5. Running the file : Your code should take the path to the query file as input and it should run as:
   $>> python Assignment1_<ROLL_NO>_parser.py <path to the query file>

Task C (Boolean Retrieval)

1. In this part, you will use the inverted index and the processed query file to retrieve documents

2. Treat each query as an AND of the individual words. For example:
   Australian embassy bombing = Australian AND embassy AND bombing

3. Using this encoding (AND) and the inverted index, retrieve all the documents for that query (write your own merge routine for the lists)

4. Store the results in a file named: ***Assignment1_<ROLL_NO>_results.txt*** as a list of:
   <query id> : <space separated list of document IDs>
   Store this file in your main code directory

5. Naming the code file: The name of your code file should be exactly as follows:
   Assignment1_<ROLL_NO>_bool.py

6. Running the file : Your code should take the path to the saved inverted index and the query file as input and it should run in the following manner:
   $>> python Assignment1_<ROLL_NO>_bool.py <path to model> <path to query file>

Submission Instructions:
Submit the files:

Assignment1_<ROLL_NO>_indexer.py
Assignment1_<ROLL_NO>_parser.py
Assignment1_<ROLL_NO>_bool.py
Assignment1_<ROLL_NO>_results.txt
README.txt

in a zipped file named: **Assignment1_<ROLL_NO>.zip**

Your README should contain any specific library requirements to run your code and the specific Python version you are using. Any other special information about your code or logic that you wish to convey should be in the README file. Further, provide details of your design in the readme, such as the vocabulary length, pre-processing pipeline, etc.
Also, mention your roll number in the first line of your README.

IMPORTANT: PLEASE FOLLOW THE EXACT NAMING CONVENTION OF THE FILES AND THE SPECIFIC INSTRUCTIONS IN THE TASKS CAREFUL. ANY DEVIATION FROM IT WILL RESULT IN DEDUCTION OF MARKS.

Python library restrictions: You can use simple python libraries like nltk, numpy, os, sys, collections, timeit, etc. However, you cannot 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. You also cannot use parsing libraries either for parsing the corpus and query files, do it by writing your own code.

Plagiarism Rules: We will be employing strict plagiarism checking. If your code matches 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.

Code error: If your code doesn't run or gives error while running, marks will be awarded based on the correctness of logic. If required, you might be called to meet the TAs and explain your code.