Part A - Task 1

Building an inverted index and Boolean retrieval

[**Deadline:** 20th October 2021 11:59 PM]

You are required to build a simple inverted index and use it to retrieve documents.You have to use *python* as libraries like *nltk* will make many things easier (stop word removal and lemmatization).

Dataset: https://drive.google.com/drive/folders/1PBdkyftTA86SM9R47rztTlib2ZlYdVVW?usp=sharing

Download the zipped corpus file “en_BDNews24.tgz", and the query file “raw_query.txt”. Unzip the corpus file and place the extracted folder named “en_BDNews24” in a *Data* folder within your main code directory. Place the query file in the Data folder as well. You directory structure should look like:

```
[Main Code Directory]
|
|--------Data
|         |
|         |---------- en_BDNews24/ (Ensure that inside this folder, there are folders named 1, 2, 3 …)
|         |
|         |---------- raw_query.txt
```

Task 1A (Building the Index)

1. Read the corpus stored in the *Data* folder. [Hint: Use *glob* or *os* in *python* to access files in the tree like directory structure, then parse each file]

2. Extract the text between the <TEXT> and </TEXT> tags of each document and consider it as your document content (Leave the title field). Use the document name as the doc id.

3. Remove stop words (*nltk* has a list of stopwords), punctuation marks and perform lemmatization (without POS tags) to generate tokens from the corpus. (use *nltk* library in python)

4. Build Inverted Index (A dictionary with tokens as keys, and document name as postings)

5. Save your Inverted Index in the main code directory as ***model_queries_<GROUP_NO>.pth*** using Pickle (binary)

6. Naming the code file: The name of your code file should be exactly as follows:

   PAT1_<GROUP_NO>_indexer.py

7. Running the file : Your code should take the path to the dataset as input and it should run in the following manner:

   ```
   $>> python PAT1_<GROUP_NO>_indexer.py <path to the en_BDNews24 folder>
   ```

   For example, for group 11 your code should run for the following command:

   ```
   $>> python PAT1_11_indexer.py ./Data/en_BDNews24
   ```

Task 1B (Query Preprocessing)

1. Read the raw query file stored in the Data folder

2. Parse the query file and extract the query id between the **<num>..</num>** tags and the query text from between the **<title>...</title>** tags. Ignore the other tags.

   .

3. Remove stop words (*nltk* has a list of stopwords), punctuation marks and perform lemmatization (without POS tags) to generate tokens from the query text. (use *nltk* library in python)

4. Save the list of queries as:

   <query id>, <query text>

   in a .txt file in your main code directory. Name the .txt file as : ***queries_<GROUP_NO>.txt***

5. Naming the code file: The name of your code file should be exactly as follows:

   PAT1_<GROUP_NO>_parser.py

6. Running the file : Your code should take the path to the query file as input and it should run as:

   ```
   $>> python PAT1_<GROUP_NO>_parser.py <path to the query file>
   ```

   For example, for group 11 your code should run for the following command:

   ```
   $>> python PAT1_11_parser.py ./Data/raw_query.txt
   ```

Task 1C (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: ***PAT1_<GROUP_NO>_results.txt*** as a list of:

   <query id> : <space separated list of document names>

   Store this file in your main code directory

5. Naming the code file: The name of your code file should be exactly as follows:

   PAT1_<GROUP_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 PAT1_<GROUP_NO>_bool.py <path to model> <path to query file>
   ```

   For example, for group 11 your code should run for the following command:

   ```
   $>> python PAT1_11_bool.py ./model_queries_11.pth ./queries_11.txt
   ```

Submission Instructions:

Submit the files:

PAT1_<GROUP_NO>_indexer.py
PAT1_<GROUP_NO>_parser.py
PAT1_<GROUP_NO>_bool.py
PAT1_<GROUP_NO>_results.txt
README.txt

in a zipped file named: **PAT1_<GROUP_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. Also, mention your group 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 may use parsing libraries for parsing the corpus and query files.

Plagiarism Rules: 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.