Assignment 2

Scoring and Evaluation

[Deadline: 05.11.2023, 11:59 PM]

This assignment is on building a tf-idf based ranked retrieval system to answer free text queries. 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-ItAbeYCj3VgIdl_KGAQRkQD4qeHpm?usp=sharing

You will require three 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 .
```

- ***cranqrel:*** This is the gold-standard relevance file. Each line in this file contains a relevance score between a particular query and document. Each line contains three numbers, separated by spaces
    - The first number is the query ID
    - The second number is the document ID
    - The third number is the relevance judgment
    - All pairs not present in the file can be assumed to have 0 relevance score
    - Some records contain negative scores, treat them as 0.

      1 184 2 →Score between query 1 and doc 184 is 2

Task 2A (TF-IDF Vectorization)

For this assignment, you have to use the Inverted Index built in Assignment 1, i.e. ***model_queries_<ROLL_NO>.bin*** (pickle) file saved in the main code directory. To build a ranked retrieval model, you have to vectorize each query and each document available in the corpus.

➔ Consider all the terms (keys) in the inverted index to be your vocabulary *V.* Obtain the **Document Frequency** for each term ***DF(t)*** as the size of the corresponding posting list in the inverted index.

➔ The **Term Frequency** *TF(t, d)* of term *t* in document *d* is defined as the number of times *t* occurs in *d.* **TF-IDF weight** W(t, d) of each term *t* is thus obtained as W(t, d) = TF(t, d) × IDF(t).

➔ Obtain the query and document texts as previously done in Assignment 1. Our goal now is to obtain |*V*|-dimensional TF-IDF vectors for each query and each document in the corpus.

➔ Represent each query *q* as [q] = [W(t, q) ∀ t in *V*]. Similarly, represent each document *d* in the corpus as [d] = [W(t, d) ∀ t in *V*], where *V* is the vocabulary defined above.

➔ Refer slide #41 of Lecture 5 and write codes for implementing the following three ddd.qqq schemes for **weighting and normalizing** the |V|-dimensional TF-IDF vectors:
    - ➢ **lnc.ltc**
    - ➢ **lnc.Ltc**
    - ➢ **anc.apc**

➔ Rank all the documents in the corpus corresponding to each query using the **cosine similarity metric** as described in slide #36 of Lecture 5.

➔ For each of the schemes, store the query ids and their corresponding top 50 document names/ids in ranked order in a txt file, in the following format.

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

This means, there will be at one entry per <query id>

Save the following three files in your main code directory:

Assignment2_<ROLL_NO>_ranked_list_A.txt for **"lnc.ltc"**
Assignment2_<ROLL_NO>_ranked_list_B.txt for **"Lnc.Lpc"**
Assignment2_<ROLL_NO>_ranked_list_C.txt for **"anc.apc"**

➔ Name your code file as: ***Assignment2_<ROLL_NO>_ranker.py***

➔ Running the file : Your code should take the path to the dataset and inverted index file, i.e. ***model_queries_<ROLL_NO>.bin*** (obtained in Assignment 1) as input and it should run in the following manner:
$>> python Assignment2_<ROLL_NO>_ranker.py <path to data folder>
<path_to_*model_queries_<ROLL_NO>.bin*>

Task 2B (Evaluation)

1. For each query, consider the top 20 ranked documents from the list obtained in the previous step.

2. For each query, calculate and report the following metrics with respect to the gold-standard ranked list of documents provided in *"cranqrel":*
    a. Average Precision (AP) @10.
    b. Average Precision (AP) @20.
    c. Normalized Discounted Cumulative Gain (NDCG) @10.
    d. Normalized Discounted Cumulative Gain (NDCG) @20.

    ***"cranqrel"* has 3 fields - *query_id*, *document_id*, and *score*.**
    For binary relevance, consider non-zero score values to be relevant.
    Assume the relevance of any pair of (*query_id*, *document_id*) not present in *"cranqrel"* to be 0.
    Also assume that pairs having score -1 can be considered to be irrelevant (treat them the same way as score 0).

3. Finally, calculate and report the Mean Average Precision (**mAP@10** and **mAP@20**) and average NDCG (**averNDCG@10** and **averNDCG@20**) by averaging over all the queries.

4. For each of the three ranked lists ***Assignment2_<ROLL_NO>_ranked_list_<K>.txt*** (K in A,B,C) obtained in the previous step, create a separate file in the main code directory with name ***Assignment2_<ROLL_NO>_metrics_<K>.txt*** and systematically save the values of the above mentioned evaluation metrics (both query-wise and average).
    .
5. Name your code file as: ***Assignment2_<ROLL_NO>_evaluator.py***

6. Running the file : For each value of K (A/B/C), your code should take the path to the obtained ranked list and gold-standard ranked list as input and it should run in the following manner:
    $>> python Assignment2_<ROLL_NO>_evaluator.py
    <path_to_*gold_standard_ranked_list.txt*>
    <path_to_*Assignment2_<ROLL_NO>_ranked_list_<K>.txt*>

Submission Instructions:

Submit the files:

Assignment2_<ROLL_NO>_ranker.py
Assignment2_<ROLL_NO>_evaluator.py
Assignment2_<ROLL_NO>_metrics_A.txt
Assignment2_<ROLL_NO>_metrics_B.txt
Assignment2_<ROLL_NO>_metrics_C.txt

README.txt
in a zipped file named: **Assignment2_<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.