CS60092: Assignment 2

Scoring and Evaluation

[Deadline: 15.10.2024, 11:59 PM IST]

IMPORTANT INSTRUCTIONS

- **Plagiarism:** 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 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 an 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.
- **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 such library, you will be ***awarded zero marks*** for this assignment without any evaluation. You also cannot use parsing libraries either for parsing the corpus and query files (You must write your own code for the same).

SUBMISSION INSTRUCTIONS

Submit the following 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 folder named: **Assignment2_<ROLL_NO>.zip (or tar.gz)**

Your README.txt file should contain the following information-

1. **[Mandatory]** Mention your **Roll Number** on the first line of your README.
2. **[Mandatory]** Any specific library requirements to run your code and the specific Python version you are using.
3. **[Mandatory]** Provide details of your design, such as the vocabulary length, pre-processing pipeline, etc.
4. **[Optional]** Any other special information about your code or logic that you wish to convey.

IMPORTANT: PLEASE FOLLOW THE EXACT NAMING CONVENTION OF THE FILES AND THE SPECIFIC INSTRUCTIONS IN THE TASKS CAREFULLY. ANY DEVIATION WILL RESULT IN DEDUCTION OF MARKS. PLEASE NOTE THE EVALUATION GUIDELINES ON PAGE 5.

NOTE: YOUR Assignment1_<ROLL_NO>_results.txt FILE WILL NOT BE EVALUATED UNLESS YOUR CODE WORKS PROPERLY.

This assignment is on building a TF-IDF based ranked retrieval system to answer free text queries.

You have to use the Python programming language for this assignment.
The total marks for this assignment is 60.

Dataset:

For this assignment, you will be using the CORD-19 Dataset [1]. The relevant files will have to be downloaded from the given links-

- Dataset– https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/historical_releases/cord-19_2020-05-26.tar.gz
- Relevance file (https://ir.nist.gov/trec-covid/data/qrels-rnd1.txt)
    - The format of a relevance judgments file ("qrels") is as follows-
      `topic-id iteration cord-id judgment`
      where *judgment is 0 for not relevant, 1 for partially relevant, and 2 for fully relevant*
- Query file (https://ir.nist.gov/trec-covid/data/topics-rnd1.xml)
    - Use **ONLY** the query field.

For more details, visit here and here.

[1] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, et al.. 2020. CORD-19: The COVID-19 Open Research Dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.

<u>Task A (TF-IDF Vectorization)</u>

For this assignment, you have to first create an Inverted Index (similar to what you built in Assignment 1). You can use the functions/code from Assignment 1. 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 to Lecture 8 [Check Course website] 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 Lecture 8.

- 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>

  There will be <u>EXACTLY one</u> 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.Ltc**"
  Assignment2_<ROLL_NO>_ranked_list_C.txt for "**anc.apc**"

- <u>Name your code file as</u>: ***Assignment2_<ROLL_NO>_ranker.py***

- <u>Running the file:</u> Your code should take the path to the dataset, and inverted index file, i.e. ***model_queries_<ROLL_NO>.bin*** (similar to what was prepared 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 B (Evaluation)

- For each query, consider the top 20 ranked documents from the list obtained in the previous step.
- For each query, calculate and report the following metrics with respect to the gold-standard ranked list of documents provided in "Relevance judgements":
    - Average Precision (AP) @10.
    - Average Precision (AP) @20.
    - Normalized Discounted Cumulative Gain (NDCG) @10.
    - Normalized Discounted Cumulative Gain (NDCG) @20.

- 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.

- 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 filename ***Assignment2_<ROLL_NO>_metrics_<K>.txt*** and systematically save the values of the above mentioned evaluation metrics (both query-wise and average).

- Name your code file as: ***Assignment2_<ROLL_NO>_evaluator.py***

- 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>
```

EVALUATION GUIDELINES

1. Task A [30 marks]
    a. Correctly creating DF and TF-IDF vectors: **5 marks**
    b. Correctly implementing each the three schemes: **5*3 = 15 marks**
    c. Correctly performing Cosine Similarity and creating ranked list files: **1 + 3*3 = 9 marks**
2. Task B [25 marks]
    a. Calculating Avg. Precision and NDCG @10, @20 : **4*5 = 20 marks**
    b. Calculating mean Avg. Precision and Avg. NDCG: **5 marks**
3. README [5 marks]
4. Deductions:
    a. Plagiarism: **-60 marks**
    b. Using libraries that are not allowed: **-60 marks**
    c. Not following naming conventions: **-2 marks for every violation**
    d. Small bugs in code, etc. that are beyond the overall logic of the code workflow, e.g., not following the input format specification for running code or anything else that does not fall into the marking scheme above: **-2 marks for every violation**