Part B

Relevance Feedback

[**Deadline:** 14th November 2021 11:59 PM]

You are required to improve the recall of the system built in Part A - Task 2 by adding relevance feedback.

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

Download the zipped corpus file “en_BDNews24.tgz", the query file “raw_query.txt” and the gold-standard file “rankedRelevantDocList” (download as “csv”) containing the ranked list of documents for each query. Unzip the corpus file and place the extracted folder named “en_BDNews24”, along with the other two files in a *Data* folder within your main code directory. You directory structure should look like:

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

Task B1 (Relevance Feedback)

With an expectation of better recall, we can use Relevance Feedback to modify our original queries.

- ➔ Consider the list of top 50 relevant documents (in ranked order) as retrieved by your system in Part A - Task 2 for each query for ***“lnc.ltc”*** (ddd.qqq) weighting scheme, i.e. ***(PAT2_<GROUP_NO>_ranked_list_A.csv).***
- ➔ For each query, consider the ***top 20 documents in the retrieved ranked list.*** You have to implement the following two schemes for expanding the query with an aim to improve the retrieved result set:
    - ◆ **Relevance Feedback (RF):** Consider the documents with ***gold standard relevance judgment score = 2*** as relevant and the ***remaining documents*** as non-relevant.
    - ◆ **Pseudo Relevance Feedback (PsRF):** Consider the top 10 ranked documents as relevant and the set of non-relevant documents as ***null.***
- ➔ Modify the query vector using ***Rocchio’s algorithm*** as follows:

$$q_m = \alpha q_0 + \beta \frac{1}{|D_R|} \sum_{d_j \in D_R} d_j - \gamma \frac{1}{|D_{NR}|} \sum_{d_j \in D_{NR}} d_j$$

Where q_m is the modified query vector, q_0 is the original query vector, D_R and D_{NR} are the sets of relevant and non-relevant documents respectively, and α, β and γ are hyper-parameters.

(Note that for PsRF, the last term in the equation vanishes, since the set of non-relevant documents is null.)

➔ Run these modified queries on your IR system and report the metrics **mAP@20, NDCG@20**, for both schemes of relevance feedback (RF and PsRF) for three values of α, β and γ:

- $\alpha = 1, \beta = 1$ and $\gamma = 0.5$
- $\alpha = 0.5, \beta = 0.5$ and $\gamma = 0.5$
- $\alpha = 1, \beta = 0.5$ and $\gamma = 0$

➔ Due to space and time considerations that occured in Part A - Task 2, you may consider the method where you compute all the updated query vectors first and then re-rank them by comparing with each document vector (calculating them on go).

➔ Your code should output two files - **PB_<GROUP_NO>_rocchio_RF_metrics.csv** and **PB_<GROUP_NO>_rocchio_PsRF_metrics.csv**, the first one containing the calculated metrics for the RF part (all 3 settings of alpha, beta and gamma) and the second one containing the calculated metrics for the PsRF part (all 3 settings of alpha, beta and gamma). The format of each entry in the CSV file should be -

"alpha, beta, gamma, mAP@20, NDCG@20"

➔ Compare the new results with those of the same weighting scheme (***lnc.ltc***) from Part A - Task 2. Further compare the RF and PsRF results. Explain your observations in a text file named: ***PB_<GROUP_NO>_rocchio_report.txt***

➔ Name your code file as: ***PB_<GROUP_NO>_rocchio.py***

➔ Running the file : Your code should take the path to the dataset, inverted index file, i.e. ***model_queries_<GROUP_NO>.pth*** (obtained in Part A - Task 1), ranked list of documents, i.e. ***PAT2_<GROUP_NO>_ranked_list_A.csv*** (obtained in Part A - Task 2) and gold-standard ranked list as input and it should run in the following manner:

```
$>> python PB_<GROUP_NO>_rocchio.py  <path to the en_BDNews24 folder>
<path_to_model_queries_<GROUP_NO>.pth> <path_to_gold_standard_ranked_list.csv>
<path_to_PAT2_<GROUP_NO>_ranked_list_A.csv>
```

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

```
$>>python PB_11_rocchio.py ./Data/en_BDNews24  model_queries_11.pth
./Data/rankedRelevantDocList.csv  PAT2_11_ranked_list_A.csv
```

Task B2 (Identifying words from pseudo relevant documents that are closer to the query)

- For the PsRF scheme, we have considered the ***top 10 documents*** as relevant for each query. You already have the weighted, normalized TF-IDF vectors for each of these documents.

- For each query, obtain a |V|-dimensional vector by averaging the TF-IDF vectors for the above considered documents, where |V| represents the size of the vocabulary.

- The value at each location of the obtained vector represents the average TF-IDF score of a specific word in the vocabulary, when averaged across the top 10 retrieved documents.

- From the obtained vector, report the words corresponding to the locations with the ***5 largest values*** for each query. Save the results in a two-column CSV file named **PB_<GROUP_NO>_important_words.csv** as follows:

  <query id> : <word1>, <word2>, <word3>, <word4>, <word5>

- Name your code file as: ***PB_<GROUP_NO>_important_words.py***

- Running the file : Your code should run in the following manner:
  $>> python PB_<GROUP_NO>_important_words.py <path to the en_BDNews24 folder> <path_to_*model_queries_<GROUP_NO>.pth*> <path_to_*PAT2_<GROUP_NO>_ranked_list_A.csv*>

  For example, for group 11, your code should run for the following command:
  ```
  $>>python PB_11_important_words.py ./Data/en_BDNews24  model_queries_11.pth
  PAT2_11_ranked_list_A.csv
  ```

Submission Instructions:
Submit the files:

PB_<GROUP_NO>_rocchio.py
PB_<GROUP_NO>_important_words.py
PB_<GROUP_NO>_rocchio_RF_metrics.csv
PB_<GROUP_NO>_rocchio_PsRF_metrics.csv
PB_<GROUP_NO>_rocchio_report.txt
PB_<GROUP_NO>_important_words.csv
README.txt

in a zipped file named: **PB_<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 that calculate the vectors or the metrics *(like sklearn)*. If your code is found to use any of such libraries, you will be awarded with zero marks for this assignment without any evaluation.

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.