Report

Aryan Singh | 19CS30007 | IR Default Term Project Part 3

Task 3A
Program file involved in this part : Assignment3_10_rocchio.py

My contribution as an individual :

- Custom add and custom cosine similarity functions were implemented:
  For dictionary-based TF-IDF vectors, I implemented the operators add and cosine similarity. I designed both of these functions to iterate over the non-zero-valued terms_id and perform the corresponding operations at that level.
- Modify_query function for Rocchio's algorithm.
  It takes three dictionary-based vectors as input, namely the query base vector, the average of relevant document vectors, and the average of irrelevant document vectors. It then returned the q_m as a dictionary-based vector.

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

- Returning ranked_list in an operable representation
  The get-rank function now needs to return a ranked_list for the queries. A list of lists was created in the getRank function. For every query, after sorting the scores, the top 50 cord ids were saved in ranked_list. This ranked list was returned at the end.
  To conduct further analysis of relevance feedback and pseudo-relevance feedback effectively, it was required to generate the ranked_list.
- Pseudo relevance
  For computing the pseudo-relevant feedback vector, the first K cord-ids were used as relevant documents. The corresponding tf-idf vector was accumulated in the pos_feedback vector. It was then element-wise averaged.
  Finally, we return a tuple containing (avg_pos_feedback_vector, zero_vector).
  The zero neg-feedback vector was added as well for making the function interface uniform as that of relevance_feedback function

Task 3B

ImportantWords.py
I did not contribute to this part of the assignment.