IR TERM PROJECT
REPORT - PART 3

ABHISHEK S. PUROHIT
19CH10002
GROUP 13

My contribution in part 3 DEFAULT-IR term project:

PART 3A (Relevance feedback) :

In this part, we have re-used the code written by me for generating the Term frequency of corpus using parallelization technique by breaking the corpus into chunks of 5000 files and processing each chunk one by one and at the end, merging the same. Further, I have written a code for reading the ranked list file obtained in part 2, which I processed further to take out top 20 doc-ids corresponding to each query in a separate dictionary “top20_retrieved”. This dictionary is very crucial for this entire part. Further I have written a code to read gold standard relevance file, which I further processed by removing multiple entries of the same document in the same query, taking only the one with highest iteration value. Further, to make our life simpler, I have written a code which converts the gold standard relevance data to the form, goldStandard_dict[query-id][doc-id] = relevance, for easily accessing the relevance values at further stages. Finally, I have written a function for calculating the NDCG values for the given query. Here, I initially got a math error, which I later debugged and found that there are cases when we don’t have any relevant documents in the ground truth data, so for those cases, NDCG couldn’t be calculated (as discussed by Animesh sir in class). So here I made a design choice, that I will be returning -1 for those cases, and in the main code (from where I am calling the NDCG function), if I get

negative value from NDCG function, then I reduce the number of queries by one for calculating average NDCG (because, if NDCG does not exist for a particular query, then there is no point of taking that query while computing the average NDCG values across all the queries). Finally I have written the code Pseudo-Relevance feedback part, and further wrote a code to save the performance metrics corresponding to different alpha, beta and gamma values for PsRF part to a CSV file.

PART 3B (Retrieving important words from pseudo-relevant queries) :

I have written the complete code for this part, wherein I have reused the code for generating the TF-IDF corresponding to all the documents and then I processed the retrieved ranked list which we obtained in part 2A, by taking out top 10 relevant documents corresponding to each query. Now for each query I averaged out the TF-IDF of top 10 documents which we had just retrieved. From that average TF-IDF values, I fetched the 5 most important words (with highest average TF-IDF values) and stored them in the csv file as per the instructions.

Further I have also developed the README for part 3.