Part A - Task 2

Group No.: 13

Group Members:

Avijit Mandal	18CS30010
Rajdeep Das	18CS30034
Ashish Gour	18CS30008
Sumit Kumar Yadav	18CS30042

Running the Code:-

Task2A: python PAT2_13_ranker.py ./Data/en_BDNews24 model_queries_13.pth queries_13.txt

Task2B:

python PAT2_13_evaluator.py ./Data/rankedRelevantDocList.csv PAT2_13_ranked_list_A.csv queries_13.txt

python PAT2_13_evaluator.py ./Data/rankedRelevantDocList.csv PAT2_13_ranked_list_B.csv queries_13.txt

python PAT2_13_evaluator.py ./Data/rankedRelevantDocList.csv PAT2_13_ranked_list_C.csv queries_13.txt

Task2A(TF-IDF Vectorization):-

Steps:

1. Imported the inverted index built in task1

2. Created document vectors which store the term frequencies of that particular document.

3. Similarly created query vectors.

4. According to the schemes mentioned in the question, calculated tf-idf weights.

5. Then for each query calculated the value of Cos between the query and each document and sorted the document according to the Cos value and thus got the Top 50 ranked Document for each query.

6. For each scheme, save the result in a csv file.

Task2B(Evaluation):-

Steps:

1. Parsed the file rankedRelevantDocList.xlsx using pandas 1.2.4 library

2. Create a dictionary relevant where every Query_ID would have a dictionary associated with as follows:

relevant[Query_ID] = dictionary of documents having Relevance scores mapped against each Document_ID, if some document is not there in relevant[Query_ID,] this means that it is not relevant against Query_ID

3. Implemented a evaluate function as follows:

Parameters:

ranked_list: dictionary of <Query_ID, {(Document_ID)}>

relevant: dictionary described in step 2

write_file: boolean to indicate if we want to write the result in file

file_name:string value indicating name of file where to write the result