IR ASSIGNMENT 2

Question 1)

Pre-processing :

- Following pre-processing operations are done on the dataset as well as the input query :

- All type of special characters and digits are removed from the data.

- Punctuations, single quote, blank spaces are removed.

- All of the tokens are converted to lower case.

- Tokens are being stemmed to root word.

Methodology :

- Our approach is to build the positional index data structure and then using the data structure we will be providing the support to search for query.

- Basic algorithmic steps to build positional index data structure :
- Traverse through each document and performing the required pre-processing steps on it.
- For each of the new token encountered, we have incremented its count occurence which is total term freqeuncy for respective token.\
- Accordingly, a dictionary is maintained for each token. Dictionary contains key as the document id and value as the position of token in document.

- We are ready with the data structure.

- Now, for a particular query, we have build up one logic inorder to retrieve the relevant document from corpus.

- Logic :
- Taking query as input and performing the pre-processing operations on it.
- For each of the token in query, we will fetch the list of documents which contain token.

- Now we took intersection of all the list obtained above.
- Now we have list of document which contains all the tokens present in query.
- For each token, we will fetch the positions of token in the respective document obtained above.
- Lists obtained above will be arranged w.r.t the occurence of token in query.
- Starting from 2nd list, we will subtract 1 from 2nd list, then 2 from 3rd list and similarly n-1 from nth list.
- Now we will take intersection of all the list after performing above operations and if intersection is not null then the current document will be added to result.
- Following above logic, we have retrieved the list of all documents for a phrase query.

Question 2)

a) Jaccard Coefficient :

Pre-processing :

- Following pre-processing operations are done on the dataset as well as the input query :
- All type of special characters and digits are removed from the data.
- Punctuations, single quote, blank spaces are removed.
- All of the tokens are converted to lower case.
- Tokens are being stemmed to root word.

Methodology :

- We take query as input and perform pre-processing.
- Then we take each document and perform pre-processing on the document.
- For each of the document, using jaccard coefficient formula, we find coefficient value w.r.t each of the document.
- Now we have coefficient value w.r.t all document for given query, we will simply sort the coefficient value list in descending order.
- After sorting the list, we will retrieve the top 5 document names from the list.

b) TF-IDF weighting scheme :

Pre-processing :

- Following pre-processing operations are done on the dataset as well as the input query :
- All type of special characters and digits are removed from the data.
- Punctuations, single quote, blank spaces are removed.
- All of the tokens are converted to lower case.
- Tokens are being stemmed to root word.

Methodology :

- Initially, we had to create TF-IDF matrix using different weighting schemes.
- First of all, we have retrieved all the unqiue words from the whole dataset.
- Then we have evaluated the idf value (inverse document frequency) for each of the word.

- We traversed through each document. Then for each of the term in vocabulary, we evaluated term frequency for that term from the corresponding document.

- Using different weighting schemes given, we have evaluated term frequency value. Accordingly, the tf*idf value is filled in the respective matrix.

- In this fashion, we have obtained 5 different tf-idf matrices for each different weighting scheme.

- Now for a particular query, initially, we pre-processed the query. Then we evaluate the tf-idf score for each query term for each of the corresponding document.

- Then, after obtaining the tf-idf score for each document, we sort the score in descending order and retrieved the top 5 documents.

- Similarly, using other weighting schemes, we evaluate the tf-idf score and retrieved the top 5 documents.

c) Cosine similarity:
Pre-processing :

- Following pre-processing operations are done on the dataset as well as the input query :

- All type of special characters and digits are removed from the data.

- Punctuations, single quote, blank spaces are removed.

- All of the tokens are converted to lower case.

- Tokens are being stemmed to root word.

Methodology :

- The tf-idf matrices obtained above for each of the weighting schemes are used here in order to evaluate cosine similarity.

- Initially, we take the query and perform pre-processing operations on it.

- Then we generate vector of length vocabulary size for the given query. Logic is present in code file.

- Now we evaluate the cosine similarity between query vector and each of the document vector.

- After obtaining similarity, we sort them in descending order and then retrieved the top 5 documents as result.

- Similarly, results are retrieved using other weighting schemes also.

Pros :
Scheme 1: Computation cost is less.

Scheme 2: Computation cost is less.

Scheme 3: Normalization might lead to better results.

Scheme 4: Computation cost is less.

Scheme 5: Involvement of 0.5 factor in Scheme 3 lead to small change in retrieval of document. Possibly the relevant document might be retrieved.

Cons :
Scheme 1: It doesn’t give imp to term frequency.

Scheme 2: It only considers raw count of token but doesn’t normalize it which lead to changes in retrieval of document.

Scheme 3: Computation cost is high.

Scheme 4 : Involvement of logarithmic factor lead to less variance among TF-IDF score.

Scheme 5: Computation cost is high.

Question 3)

b)
Pre-processing :
From the given dataset, we have to work only on queries with id 4. Therefore, remaining all of the queries are removed from dataset.

Methodology :

- For each of the query, we are given relevance value.

- Now we have to find max DCG for given query-url pair.

- In order to find max DCG, we will sort the query ids in descending order of their relevance.

- Now we have relevance in descending order, we simply applied the formula in order to calculate DCG.

- The value obtained above is maximum DCG.

- Now to find all possible files with maximum DCG can be found by applying permutation logic. Wherever the relevance is same, in that particular vicinity, we can do internal sorting of the files has it won’t affect the DCG value.

- In this fashion, we have obtained the all possible files with max DCG.

c)
Pre-processing :
From the given dataset, we have to work only on queries with id 4. Therefore, remaining all of the queries are removed from dataset.

Methodology :

- In order to compute the NDCG at rank 50, we simply calculate the DCG value at rank 50.

- Now we want DCG value for ideal ranking, so we will sort the query ids in descending order of their relevance.

- Now we simply calculate the DCG value at rank 50 for ideal ranking.

- Now we simply divide DCG value for given ranking by DCG value for ideal ranking.

- Similarly, NDCG for whole dataset can be compute in similar fashion by calculating DCG for whole dataset for normal and ideal ranking.

d)
Pre-processing :
From the given dataset, we have to work only on queries with id 4. Therefore, remaining all of the queries are removed from dataset.

Methodology :

- In this question, we had to plot precision-recall curve for query id 4.

- We are already given relevance value for each of the url.

- We simply followed the definition of precision and recall and evaluated the values for both at each rank.

- Precision is fraction of no of relevant documents retrieved out of total documents retrieved till moment.

- Recall is fraction of no of relevant documents retrieved out of total relevant document.

- Then we plotted the curve using values obtained above.