**Project 3: Query Retrieval and Performance Analysis**

This project is extension of Parsing and Indexing done previously, the finally obtained forward index and inverted index are used to calculate the cosine similarity of documents for the given query.

**Data Structures -**

Initial have used lists to store all the query terms of Title, Description and Narrative separately, then used dictionaries to store the extracted queries with their term frequency , have used defaultdict in some places like calculating the scores for each relevant document.

**Preprocessing –**

Initially all the content in the topics are stored in lists by extracting the terms from respective tags like Topic or so, then same kind of processing is done as we have done before like removal of digits and nonalphanumeric values and split them along space to store in list file, removing stop words and storing term frequency for each query number using counter method from collection.

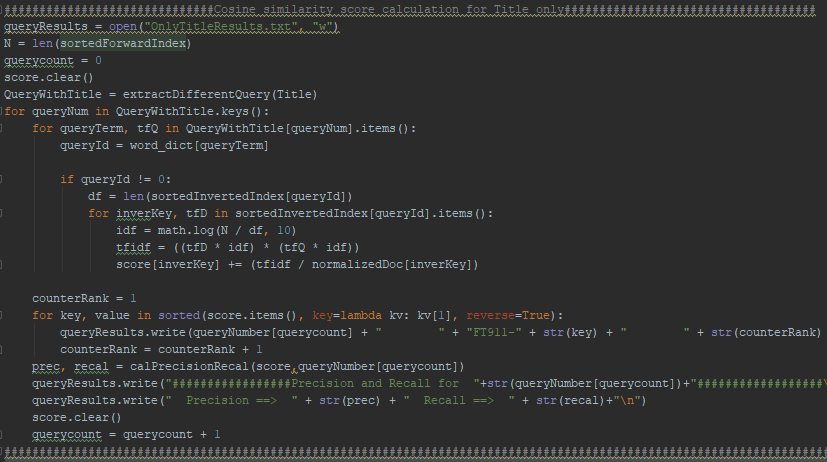
Have implemented **extractDifferentQuery** method to form a dictionary of each query number with term and term frequency of each query term as value in it.

Then have processed main.qrels file extracting the relevant information like details about the documents belonging to FT911 for comparing the implemented query processor performance like precision and recall.

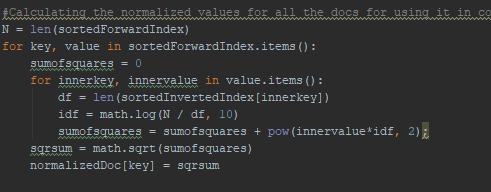
**IR Query Processing design –**

The cosine similarity is done dynamically following the algorithm given in the slides, will run a for loop on each query term accessing its frequency in that topic. By converting query term into respective wordId in the parsed document we access the documents containing the query term using inverted index. Then we iterate through each docId and its term frequency.

Before we calculate the cosine similarity we get document frequency for each query term by counting number of documents for each key value (queryTermId) in Inverted Index as DF.



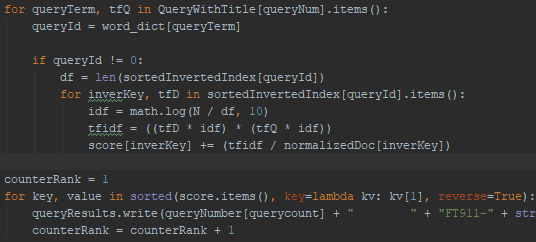
Then also we get the size of collection using length of forward Index and save it as N. Using both these df and N values we calculate the IDF value by dividing N/df and taking logarithm of the result. The same IDF value is used both for the document and query while calculating similarity. Now we multiply the tf-idf values of term in document and query then we divide it with normalization value for that doc calculated in below manner.



To implement the length normalization, we iterate for each docID in forward index and get their term frequencies, we again calculate the tf-idf value as discussed above and then sum the squares of each tf-idf value and finally we do square root of the sum. This way we store the length normalized value for each doc in a dictionary to access them easily while similarity calculation.

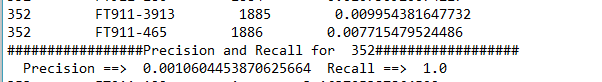
**Dynamically assigning scores to documents that only have the query terms –**

Have implemented the code such that only the similarity is calculated and updated for the documents which have query term. Initially have used defaultdict for dictionary containing term and termId pairs so that if the query term is not present in the dictionary then its value is given as zero and its ignored. Then, if loop allows terms only if they are in word dictionary where for each docId which is having the query term the relevancy score is updated. Here again the dictionary used for calculating similarity is implemented using defaultdict so that initial all documents have zero score and it gets updated for each relevant doc. This was the scores will be getting summed for the whole query in the topic based on the input kind of text like Topic or Narrative or Description or some of them combined. Which is written in to text file. So for each topic the summation all this values for each query in it decides the final score between the given query and document.

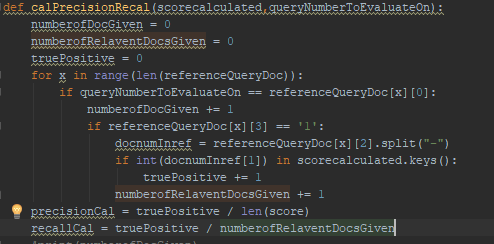


**Calculating Precision and Recall –**

Have implemented claPrecisionRecal method to calculate the precision and recall for each topic which is written in the same file where the score is written at the end of each query number as shown below.



To calculate precision and recall I have extracted all the docs with FT911 beforehand and saved in dictionary, now I am calculating number of relevant documents by checking how many have relevancy as 1 for that particular query number. Then in same if condition, we are calculating how many numbers of documents which have relevancy as 1, are present in score dictionary which has all non-zero scores for relevant docs. Then we can get number of true positives by taking the count of those elements. This way we can calculate the precision by dividing number of true positives by number of docs with non-zero relevance score. Then recall by number of true positive by relevant docs given.



**Precision and Recall values for different combination –**

As we can see with change in kind of query we use the system is performing differently, with a greater number of query terms by taking combination of Topics with description or Narrative we are giving the context and more specific details what we want to retrieve.

As we can see by increasing the query terms, though we are guaranteed that the documents which we are intended to retrieve are getting retrieved but a greater number of documents are retrieved. This can be seen by trend of decrease in precision and increase in recall.

