**Report describing your implementation**

The file **Indexer.java** takes in a command line argument which is filename.

It read the tccorpus.txt and creates an inverted index

**static** Map<String, HashMap<Integer, Integer>> *wordDetails* = **new** HashMap<String, HashMap<Integer, Integer>>();

**static** HashMap<Integer,Integer> *HmDocAndLength* = **new** HashMap<Integer,Integer>();

In order to store the data of the index, I have used two HashMaps to store the details of each word in the file.

*wordDetails* contains the word and a hashmap within it. The inner hashmap contains the documentId as key and the frequency of the word within that document.

*HmDocAndLength* contains the document Id as key and the length of the document within the corpus.

**Steps:**

1. The tccorpus.txt file read line by line.
2. Each line is then split into words
3. For each word, I check if it is present in *wordDetails* hashmap or not. If the word is not present, I create an entry for the word. Else, I simple increment the term frequency corresponding to the current document.
4. Simultaneously, as each line is read the length of the corresponding document is calculated. Once a document is read, its documentId and length is added to the hashmap *HmDocAndLength.*
5. Both these hashmap are then printed to the file index.out so that they can be processed by the bm25 algorithm.

The file **Bm25.java** takes in command line arguments which are index.out queries.txt and 100 which is the number of documents which are the displayed.

It read the index.out and queries.txt files and stores them in the following data structures.

**static** Map<String, HashMap<Integer, Integer>> *wordDetails* = **new** HashMap<String, HashMap<Integer, Integer>>();

**static** HashMap<Integer,Integer> *HmDocAndLength* = **new** HashMap<Integer,Integer>();

In order to store the data of the index, I have used two HashMaps to store the details of each word in the file and the document details.

*wordDetails* contains the word and a hashmap within it. The inner hashmap contains the documentId as key and the frequency of the word within that document.

*HmDocAndLength* contains the document Id as key and the length of the document within the corpus.

Steps:

1. The*readFileAndFillWordDetails* methodwill read the both the files.
2. The queries.txt file is read and all the queries are stored in the data structure

**static** ArrayList<String> *listOfQueries* = **new** ArrayList<String>();

1. Now the index.out file is read one line at a time and then both the hashmaps *wordDetails* and *HmDocAndLength* are filled.
2. I then calculate the avdl or avg document length and store it in the variable *avdl.*
3. For each query in *listOfQueries,* we get all the words in that query. We also get a list of all the document Ids for that particular query.
4. For each document Id in the list, we execute the bm25 algorithm according to the formula given in the slides.
5. These document Ids and their corresponding Bm25 scores are then saved in the Hashmap HmDocAndBmScore.
6. This Hashmap is then sorted as per the bm25 scores and the top 100 documents for a particular query is displayed.