**Implementation Summary**

The assignment is implemented in 2 modules:

* Module 1 (Creation of index):

1. The InvertedIndex.py file is responsible for creating the inverted index for given corpus.
2. The corpus file is read line by line.
3. The document number is retrieved using a regex created in program.
4. Each word of document is retrieved which is not digit and added to dictionary until new document is encountered.
5. The same process is repeated till we reach end of file.
6. At the end a dictionary is created where word is a key and value contains another dictionary of doc id and frequency of word in that doc.

e.g. "parker": {"1770": 1, "1889": 1, "3182": 1}

1. Once the dictionary is created it is serialized and dumped to “index.out” file using JSON module provided in python 3. The index.out file is stored in JSON format as key value pair.

* Module 2 (Ranking based on BM25):

1. The BM25.py file is responsible for ranking the documents for given queries by making use of inverted index created in module 1.
2. The inverted index “index.out” file is read and de-serialized to create the dictionary object which can be used in the program.
3. The document length of each doc is calculated and stored in a dictionary.
4. Average document length is calculated based on document length dictionary created above.
5. Queries are read from queries.txt and

for each query

1. BM25 score is calculated for each document which is

sum of BM25 score of each query word



|  |  |
| --- | --- |
| *n = no of docs containing the term* | *R = relevance information* |
| *f = frequency of a term in a particular document* | *dl = length of the document* |
| *qf = query frequency of the term* | *avdl = average length of the document* |
| *r = relevant documents containing the term* | *N = Total no of documents in the collection* |

1. The computed BM score is stored in dictionary.
2. The dictionary is sorted and top N documents specified in input are written in file based on their score.
3. Finally, the output will contain top N ranked documents based on computed score for each query.