IMPLEMENTATION REPORT FOR ASSIGNMENT 3

This assignment has two programs.

1. indexer.py
2. bm25.py

indexer.py:

This module is used to create an inverted index file from the document collection. The document collection consists of document id and stemmed words associated with the particular document-id. The general format of document collection is

*# document-id-1  
stem\_word1 stem\_word2 ........  
..............................  
  
# document-id-2  
stem\_word1 stem\_word2........  
..............................  
..  
..*

*# document-id-n  
stem\_word1 stem\_word2........  
..............................*

In order to convert the document collection to inverted index, we parse the file to extract stemmed words and document-ids. The inverted index is then built using document-id and stemmed words. The general structure for inverted index is as follows

{Key -> value} 🡺 {stem-word -> {document-id -> count}}

In the above structure we can observe that the inverted index has a dictionary with stem word as key. The value for key is a dictionary with document-id as key and count as value. Thus for each stem-word, we have a dictionary containing document-id’s in which the term appears and the frequency of the term in that document . Using a dictionary at multiple levels provide following advantages :

1. using a dictionary improves search efficiency . A search in list involves linear search over entire list whereas search in dictionary takes O(1) complexity

The output file is of the form

# stem\_word-1

doc\_id~count

doc\_id~count

.

.

.

.

#stem\_word-n

doc\_id~count

doc\_id~count

doc\_id~count

.

.

bm25.py

The bm25.py files takes the index file , query file and maximum number of documents to be ranked as input and retrieves a list of documents based on bm25 score . The index file is of the format :

# stem\_word-1

doc\_id~count

doc\_id~count

.

.

.

.

#stem\_word-n

doc\_id~count

doc\_id~count

doc\_id~count

.

.

The file is parsed and the data is extracted into a data structure for inverted index . The inverted index is of the form

{key->value} 🡺 {stem\_word -> [(doc-id1,count1) , (doc-id2,count2)……(doc-idn , countn)]}

The data structure for inverted index is a dictionary in which stem\_word is the key and the value is a list of tuples of the type (document\_id , count) . We use list of tuples so that we can efficiently iterate over the list while ranking the documents for queries

Apart from the inverted index , we also have a dictionary of token count in which document-id is the key and frequency of occurrence is the value .

The query file is parsed to obtain a list of words for each query . this word is then compared against the inverted index and BM25 score is calculated iteratively over each frequency term of query.

The documents are displayed in decreasing order of BM25 scores . i.e The document with highest BM25 score is displayed first and document with lesser scores are displayed later