README

SmallSearchEngine 1.0

IMPLEMENTATION DETAILS:

The solution to the problem is build in python.
There are 2 .py files that drives the requirements of the assignment.

1. indexer.py -builds index for document collection provided in the file tccorpus.txt
2. bm25.py -provide a ranked list of documents for a file with one or more queries

---

indexer.py

It requires the filename of document collection tccorpus.txt to build the index
The required filename is provided as a command line argument
There is one function main() that drives the entire computation.

main()
It passes the following received command line argumentsto function buildindex()
args[0] = tccorpus.txt
argv[1] = index.out

buildindex()
1. It parses the "tccorpus.txt" and stores the document-ID and terms for each document-ID in a dict() "corpus"
2. The dictionary is then filtered to remove any tokens that contain only the digits 0-9 if present as value for each of the document ID.
3. The results are stored in a new dcitionary "new_corpus".
4. Further, inverted index is build and stored as dictionary where
        key is stem word
        value is dictionary that has key as document ID and value as term frequency

e.g. for stem word   orthogon

{orthogon": {"2779": 3, "884": 1, "1345": 1, "765": 1, "1876": 2, "1921": 2, "1895": 1, "2806": 1, "1816": 2, "1837": 2, "2410": 2, "1731": 1, "1730": 1, "1598": 1,}

5. The number of tokens in each document is stored in dictionary "doc_token_count"

6. doc_token_count and new_corpus are stored in a list "result"
6. The list "result "is stored in "index.out" using json.dump

---

bm25.py

It requires the filename of inverted index file, the query file, and the maximum number of document results for top 100 document IDs and their BM25 scores for each test queryent
There is one function main() that drives the entire computation.

main()
It passes the following received command line arguments to function

parseQuery()
computeBM25()
runQuery()

parseQuery()

1. Stores each of the terms contained in each query as element of a list "queries"

runQuery()

1. Reads the data from index.out using json.load and saves in a list "index_and_tokens".
2. The number of tokens in each document is saved in dictionary "doc_token_count" and inverted index is stored in dictionary "index_list"

3. Further, the program uses the following procedure to compute BM25:

-> Retrieve all inverted lists corresponding to terms in a query.
-> Compute BM25 scores for documents in the lists.
-> Make a score list for documents in the inverted lists.
-> Accumulate scores for each term in a query on the score list.

4. The results are stored in list "score_list" where each element of list is a dictionary ("score_list_per_query").
5. score_list_per_query stores BM25 score for each of the terms present in index.out for each query in form of key:value pair.
6. The list "score_list" has 7 elements each corrresponding to the 7 queries provided in queries.txt
7. Further, the score_list is stored by BM25 score and top 100 return are retrieved and stored in results.eval

---

HOW TO RUN:

1) Enter the following command to build the index
./indexer.py tccorpus.txt index.out

Note:- tccorpus.txt must be present in the same directory as of indexer.py
       Also, move index.out, results.eval in a different directory before running the python file.

2) Enter the following command to compute BM25 score for the provided queries.txt and index build from previous command
./bm25.py index.out queries.txt 100 > results.eval

Note:-
The output from this run, with the top 100 document IDs and their BM25 scores for each test query is stored according to the following format:

query_id Q0 doc_id rank BM25_score system_name

system_name is "MOHITG"

