INDEXER.PY

This program outputs an inverted index of the given corpus.

Input:

Corpus file in the below format:
doc ID
tokens in the document

Output:

File containing an inverted index of the corpus and number of tokens of each document in the corpus.

Programming Language Used : Python 2.7

Libraries Used : sys, json, re

Data Structures Used :

docTokenCount : Dictionary to store token count number for each doc ID.
Key -> Document ID, Value -> Number of tokens in each doc.

docTokenCollection : Dictionary to store words for each doc ID.
Key -> Document ID, Value -> list of tokens

invertedIndex : Dictionary for Inverted Index
Key -> word, Value -> term freq, {docID : tf } for key

data : List with Inverted Index and Number of tokens for each document
List to dump json [invertedIndex, docTokenCount]

Pseudo Code:

1. Read the input file, split each line based on "#" and docTokenCount and docTokenCollection is initialized.
2. To build an inverted index:
   For each document in docTokenCollection:
   For each word in document:
   If first occurrence of document or word, initialize the invertedIndex dictionary, else increase the counter of the word.
3. List data is used and output is written using json.dump

BM25.PY

This program is used to build a small search engine using the BM25 ranking model.

Input :

An inverted Index of the corpus, queries file.

Output :

Top 100 document IDs and their bm25 scores for each query in the file in the following format:

query_id Q0 doc_id rank BM25_score system_name

Programming Language Used : Python 2.7

Libraries Used : sys, json, math, collections

Data Structures Used :

bm25 : Dictionary to store bm25 scores.
Key -> DocID, value -> bm25 score

data, invertedIndex, docTokenCount (Output of indexer.py) as input.

Pseudo Code :

1. Load the invertedIndex and docTokenCount dictionaries from the Index.out file.
2. The bm25 formula is given below:

$$\sum_{i \in Q} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$$

where the summation is now over all terms in the query;
R, r = 0
ni = number of documents in which term i found
N = Total number of documents
fi = frequency of term i in the document
qfi = frequency of term i in the query

$$K = k_1((1-b) + b \cdot \frac{dl}{avdl})$$

k1=1.2, b=0.75, k2=100

3. Calculate N, avdl
   For each query in queries file:
   For each term in query:
   Get tf, i.e., term frequency for the term from invertedIndex
   For each docID in tf:
   Calculate dl for the docId, dfi, bm25 score using
   Above formula and update the bm25 dictionary
4. Sort the results based on bm25 scores and write the top 100 results into output file for each query.