Homework 1

The following python scripts have been included:

findDocLen.py: Uses termvector function of elasticsearch and finds the term frequencies of each document. It then adds up the term frequencies of all the terms and computes the document length. The document name and length are stored in the file document_length.txt.

parseDoc.py: Takes all the input files and parses them to retrieve the document id and text fields. Creates index using elasticsearch.

wordStemmer.py: Referred post 65 of piazza and took the program from http://tartarus.org/~martin/PorterStemmer/python.txt. It returns the stem of a word given to it.

senseScripts.py: Has the elasticsearch scripts used to calculate the TF and TFF.

findOkapi_TfIdf_Score.py: Uses the getTF function defined in senseScripts.py and finds the terms required for computing the Okapi TF and TF-IDF scores. Computes the Okapi and TF-IDF values looping through each query for all the documents that contain the terms in the query.

findOkapiBM.py: Uses the getTF function defined in senseScripts.py and finds the terms required for computing the Okapi BM25 score. Computes the Okapi BM25 values looping through each query for all the documents that contain the terms in the query.

findUnigramLM.py: Uses the getTF function defined in senseScripts.py and finds the terms required for computing the Unigram LM with Laplace Smoothing values. It loops through all the queries, stores the document ids that have the query terms and the TF for each query term in the document. It then loops through all the documents that has the query and computes p_laplace(w/d) for all the words in the query. It sums the log(p_laplace(w/d)) values of each word to get the lm_laplace for the document and the query.

findUnigramJM.py: Uses the getTF and getTTF functions defined in the senseScripts.py and finds the terms required for computing the Unigram LM with Jelinek-Mercer smoothing. It loops through all the queries, stores the document ids that have the query terms and the TF and TTF for each query term in the document. It then loops through all the documents that has the query and computes p_jm(w/d) for all the words in the query. It sums the log(p_jm(w/d)) values of each word to get the lm_jm for the document and the query. The smoothing parameter Lambda is taken as 0.3 to achieve a good average precision using trec_evals script. It has been observed that as Lamda decreases from 1.0 to 0.0, the average precision increases. I have tried running the script for values 0.9, 0.5, 0.3, and the highest precision was obtained for Lambda = 0.3.

Following is the summary of output obtained from the above scripts:

Model	Output filename	Average Precision	Exact Precision	Execution time (minutes)
Okapi TF	okapi_output.txt	0.1463	0.1716	1.42
TF-IDF	tf_idf_output.txt	0.2172	0.2503	1.42
Okapi BM25	okapi_bm_output.txt	0.2099	0.2327	1.37
Unigram LM with Laplace Smoothing	unigram_lm_output.txt	0.1377	0.1762	11.32
Unigram LM with Jelinek-Mercer Smoothing	unigram_jm_output.txt	0.1807	0.2104	17.01