Jaissica Hora
CS6200 Homework 1: Retrieval Models

Implementation

Indexing:

1. I began by searching the corpus for the files in our dataset and then I read a couple of the files to discern the trend.

2. I made a parser to read a file line by line, watch for tag openings and closings, and mark the pertinent notations to indicate when a tag has been reached.

    (a) When <DOC> and </DOC> is encountered it means that the lines between them belong to a file, so I created a new class on seeing open tag and set the text upon close tag.
    (b) When <DOCNO> and </DOCNO> is seen, the number between tags is obtained and set to document object.
    (c) When <TEXT> and </TEXT> is seen, the text between them is obtained and appended to the corresponding document object.
    (d) At the end of document tag, the information is added to elastic search indexer.

3. For Indexing I have used following libraries:
    (a) Stopwords- Used to remove stop words from the corpus.
    (b) PorterStemmer– I am using PorterStemmer library to stem the words.

Retrieval Models:

1. Created a generic class with model specific functions for all retrieval models to simplify the functionality and its understanding.

2. To further simplify the process of comparing and using formulas I am storing the output in separate files for each retrieval model.

Below are the Retrieval Models I have used as mentioned in the Assignment:

I. ES-Built In (Default):
Calls the built-in elastic search API with match query using the query tokens.

II. Okapi TF:
This is a vector space model using a slightly modified version of TF to score documents.
Used the formula described in assignment to compute score.

III. TF-IDF:
This is the second vector space model.
Used the formula described in assignment to compute score.

IV. Okapi BM25:
BM25 is a language model based on a binary independence model.
Used the formula described in assignment to compute score.

V. Unigram LM with Laplace smoothing:
This is a language model with Laplace ("add-one") smoothing. We will use maximum likelihood estimates of the query based on a multinomial model "trained" on the document.
Used the formula described in assignment to compute score.

VI. Unigram LM with Jelinek-Mercer smoothing:
This is a similar language model, except that here we smooth a foreground document language model with a background model from the entire corpus.
Used the formula described in assignment to compute score.

Pseudo relevance feedback:

1. Firstly, I calculated the current query length and calculated the number of words that could be added to improve the search.

(a) If query length is long, I didn’t add many words as that might not be helpful whereas if the query length was short adding words is helpful to improve the score.

2. Top relevant terms in the documents:

    (a) Then I went through the entire document to compute the term frequency and document term frequency for each term for calculating their corresponding scores. To get the idea about rarity of the term in the corpus I multiplied the total term frequency with document term frequency.
    (b) Similarly, to calculate the top k documents : When combined the score for all the terms in all the documents, we scale it based on their rank so that the term with rank 1 is more relevant than the term with rank 2.

3. Then I found the top relevant words which are near to the query words in the document.

4. Elastic search significant terms API:

    (a) Called the Elastic search significant terms API and then I got the top words with highest scores.
    (b) Scores are further derived from the document frequencies in foreground and background sets – Elasticsearch.

5. During the pseudo relevance feedback calculation, I made sure to avoid any stop or common words or words already in the query being used and picking only those which have highest document frequency.

Results:

Model Name	**Mean Average Precision**	**Precision at 10 documents**	**Precision at 30 documents**
ES-Built In	0.2939	0.4320	0.3627
Okapi TF	0.2167	0.3800	0.3040
TF-IDF	0.2820	0.4440	0.3653
Okapi BM25	0.2833	0.4200	0.3493
Unigram LM with Laplace smoothing	0.1955	0.3960	0.3080
Unigram LM with Jelinek-Mercer smoothing	0.2489	0.4040	0.3467
Okapi BM25(After Pseudo relevance feedback)	0.2874	0.4360	0.3600
Okapi BM25(After Pseudo relevance using ElasticSearch aggs “significant terms”)	0.2877	0.4364	0.3604

Results:

- Okapi BM25 best performed amongst all the other models I tried.
- After pseudo relevance there is some improvement in Okapi BM25 which is an indication that the feedback has picked some words related to the search query which helped us to filter out the relevant documents.