Information Retrieval Project Report

CS 6200: Information Retrieval

Northeastern University

Fall 2018 semester

Under the guidance of

Prof. Nada Naji

TEAM MEMBERS

Mihir Gandhi

Neha Gundecha

1. Introduction:

The main goal of this project was to design and implement an Information Retrieval System (Search Engine), evaluate and compare its performance based on a set of metrics to find out the retrieval effectiveness of the retrieval system.

Note: This section includes the contribution of each team member to the project.
The Retrieval model was build for the CACM corpus on a set of queries provided.
The project is divided into 3 phases:

1.1 Phase 1: Indexing and Retrieval

This phase consists of 3 tasks:

1.1.1. Task 1**:**

Lucene, TF-IDF – Mihir Gandhi
BM25, JM Smoothing QL– Neha Gundecha

This task aims at building retrieval systems. It makes use of four retrieval systems namely BM25, TF-IDF, JM Smoothed Query Likelihood (lamba 0.35) and Lucene.
This step takes the corpus as the input, tokenizes the input corpus by case-folding and removing the punctuations and generates an inverted index for the corpus. Further it cleans the queries on which the retrieval is to be done and then applies the four systems namely BM25, TF-IDF, JM Smoothing QL and Lucene and calculates the document scores for each of the documents and returns the top 100 retrieved ranked lists for the 4 systems.

1.1.2. Task 2: (Mihir Gandhi)

This task performs Pseudo Relevance Feedback. This re-calculates document scores using BM25 Model considering relevance and returns the top 100 retrieved ranked lists generated by this model.
For this task we need to consider the top 5 words from the top 5 documents. The top 5 words have high chances of being in the common_words list so exclude the words appearing in the common_words.txt. and run the BM25 model on the expanded queries.

1.1.3. Task 3:

This task aims at building the retrieval systems using stopping and stemming.

Part A: (Neha Gundecha)

This part consists of applying stopping on the corpus and queries without stemming. The common_words.txt contains the stop words used for applying stopping.
For applying stopping, we first tokenize the corpus and generate the inverted index as done in Task 1 but this time using stopping. The 3 retrieval models BM25, TF-IDF and JM Smoothing QL is applied on this stopped index and the document scores are calculated. The output is the top 100 ranked documents for each of the retrieval models.

Part B: (Mihir Gandhi)

This part consists of using the stemmed version of the corpus(cacm_stem.txt). For applying stemming, first tokenize the given stemmed corpus, create an inverted index and generate the document scores based on the stemmed queries in cacm_stem.query. The output if the top 100 ranked documents for each of the three retrieval models BM25, TF-IDF and JM Smoothing QL.

1.2 Phase 2: Displaying Results (Neha Gundecha)

This is phase consists of generating snippets for the top 5 documents for each of the 64 queries for the stopped version of the BM25 model. The output of this phase shows the snippets generated for the documents along with query term highlighting. For query term highlighting, the the query terms in the snippet are surrounded by tags.

1.3 Phase 3: Evaluation
Precision, Recall, MRR, MAP, P@5, P@20 – Neha Gundecha
Plotting the graph – Mihir Gandhi

This phase consists of evaluating the results of the retrieval model. These results show how accurate and relevant our retrieval system based on the metrics calculated. The performance of the retrieval model in terms of their effectiveness is given by Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision@5(P@5), Precision@20(P@20), Precision and Recall for all queries and all runs and used term-at-a-time evaluation for getting efficiency.
A graph is also plotted for the points averaged over the precision and recall values for each of the queries for each of the runs individually.

This consists of evaluating the results by calculating the evaluation results for the following retrieval models:

1. BM25 model
2. TF-IDF model
3. JM Smoothed Query Likelihood model
4. Lucene model
5. BM25 considering relevance and and Pseudo Relevance Feedback
6. BM25 with stopping
7. TF-IDF with stopping
8. JM Smoothed Query Likelihood with stopping

1.4 Extra Credit: (Mihir Gandhi)

In this section we created an advance search tool that has the following approaches Exact match, Best match and Ordered best match within proximity N provided by the user. For this task we first tokenized the corpus and created a positional inverted index which stores the terms in the following format: term -> [Doc_ID, term frequency, [position]]. In the program for the advance search we first ask for the query, clean the query provided by the user and give the option of applying exact match, best match or proximity match. For exact match we consider the terms appearing in the exact order as the query, for best match we consider that

atleast one of the query terms appear in the document and for proximity match we consider the match in the windows of N for pairs for the query terms. For all the tasks the positions of the terms play a big role in evaluating the results and for each term we compare the positions in the same documents to do advance search.

1.5 Documentation – (Neha Gundecha)

2. Literature and Resources

Task 2: Query Refinement
This task performs Pseudo Relevance Feedback. This re-calculates document scores using BM25 Model considering relevance and returns the top 100 retrieved ranked lists generated by this model.
For this task we need to consider the top 5 words from the top 5 documents. The top 5 words have high chances of being in the common_words list so exclude the words appearing in the common_words.txt. and run the BM25 model on the expanded queries.

Phase 2: Snippet Generation
This is phase consists of generating snippets for the top 5 documents for each of the 64 queries for the stopped version of the BM25 model. The output of this phase shows the snippets generated for the documents along with query term highlighting. For query term highlighting, the query terms in the snippet are surrounded by tags.

Resources:

1. Croft Textbook
2. Slides provided by the professor
3. Beautiful soup documentation
4. Python documentation

3. Implementation and Discussion:

3.1 Literature of the concepts used

1. Indexing
   Indexing is the main part of building a search engine. Text processing is done to create an index where the words are converted into index terms. For the purpose of this project we use the unigram indexer:
   Format: index_term -> [doc_ID , term frequency]

2. BM25 Model
   BM25 is a document scoring algorithm that scores documents based on the query terms appearing in each document. This model extends the binary independence model to include the document term and query term weights.

The scoring formula for BM25 is –

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

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

ri – Number of relevant documents containing term i
R – Number of relevant documents for the given query
ni – Number of document containing the term i
N – The total number of documents in the corpus
fi – Frequency of the term i in the document
qfi – frequency of the term i in the query
k1, k2 – determines how tf changes
b – Regulates the length normalization impact

3. TF-IDF Model
   Term frequency and Inverse Document Frequency determines the importance of a word in the document.

$$tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$$

$$idf_k = \log \frac{N}{n_k}$$

tf – Term frequency
idf – Inverse Document frequency
fik – Number of occurances of term k in the document
N – Total number of documents in the collection
nk – Number of documents containing term k

4. JM Smoothing Query Likelihood

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

Lamba = 0.35

|D| - Number of documents
|C| - Number of word occurances in the collection
fqid – Frequency of query term in document D
cqi – Number of times a query word occurs in the collection of documents

5. Lucene Model
   This is a free and open source information retrieval library that is written in Java. It is used widely by many search engines for its indexing and search capabilities.

6. Stopping
   This is task of removing the stopping words from the corpus and queries which are of very little importance when it comes to retrieving the best documents through the retrieval system. Stopping words are the identified common words that occur very frequently in the corpus or queries. By creating stop-lists the index size is reduced improving the effectiveness of the retrieval system.

7. Stemming
   Stemming is used for reducing the words appearing in the corpus or queries to its stem converting the words into common stems. This improves the the index building and ranking of documents.

8. Evaluation
   There are various metrics used for evaluating the retrieval systems and measuring their effectiveness.

   1. Recall and Precision

      Precision – The proportion of retrieved documents that are relevant

      $$Precision = \frac{Number\ of\ Relevant\ documents\ Retrieved}{Total\ Documents\ Retrieved}$$

      Recall – The proportion of relevant documents that are retrieved

      $$\text{Recall } = \frac{Number\ of\ Relevant\ documnets\ Retrieved}{Total\ Relevant\ documents}$$

   2. Mean Average Precision

      $$Mean \text{ Average Precision (MAP) } = \frac{Sum\ of\ Average\ Precision\ for\ each\ query}{Total\ number\ of\ queries\ executed}$$

3. Mean Reciprocal Rank

$$Mean\ Reciprocal\ Rank\ (\text{MRR}) = \frac{Sum\ of\ Reciprocal\ Rank\ for\ each\ query}{Total\ number\ of\ queries\ executed}$$

4. P@K – Precision@K

$$Precision\ @\ K = Precision\ for\ Kth\ rank\ document\ in\ the\ result\ of\ a\ query$$

5. Average Precision

$$\text{Average Precision} = \frac{Sum\ of\ Precision\ at\ Relevant\ documents}{Total\ Relevant\ documents}$$

6. Average Recall

$$\text{Average Recall} = \frac{Sum\ of\ Recall\ at\ Relevant\ documents}{Total\ Relevant\ documents}$$

3.2 Query by Query Analysis

Query 1: What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers?
Top relevant documents for Query 1:
CACM-1410
CACM-1572
CACM-1605

Query 2: Graph theoretic algorithms applicable to sparse matrices
Top relevant documents for Query 1:
CACM-1563
CACM-2695
CACM-2986

Query 3: Number-theoretic algorithms, especially involving prime number series, sieves, and Chinese Remainder theorem.
Top relevant documents for Query 1:
CACM-2372
CACM-2632
CACM-2870

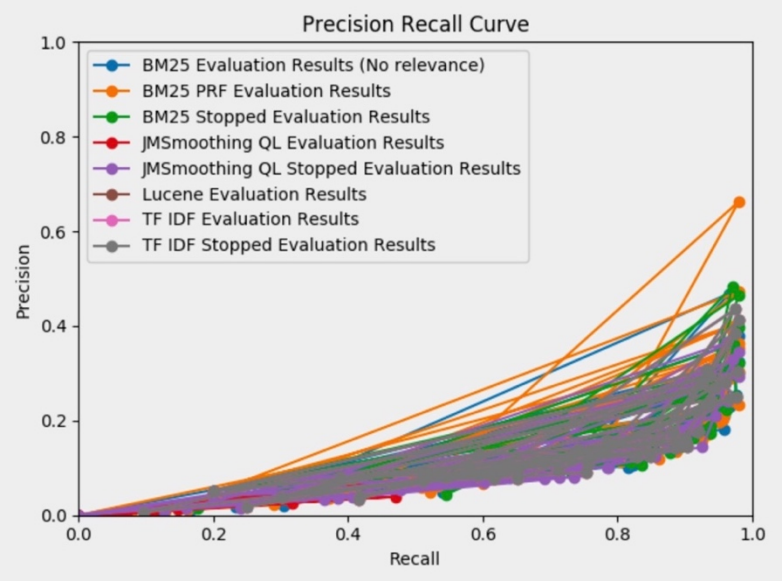
Query	BM25	TF-IDF	JM Smoothing QL
Query 1	CACM-1410 (Rank 6) CACM-1572 (Rank 11) CACM-1605 (Rank 12)	CACM-1410 (Rank 64) CACM-1572 (Rank 31) CACM-1605 (Rank 62)	CACM-1410 (Rank 28) CACM-1572 (Rank 17) CACM-1605 (Rank 12)
Query2	CACM-1563 (Rank 4) CACM-2695 (Rank 1) CACM-2986 (Rank 2)	CACM-1563 (Rank 2) CACM-2695 (Rank 37) CACM-2986 (Rank 30)	CACM-1563 (Rank 32) CACM-2986 (Rank 14)
Query3	CACM-2372 (Rank 3) CACM-2632 (Rank 72) CACM-2870 (Rank 22)	CACM-2372 (Rank 9) CACM-2870 (Rank 36)	CACM-2372 (Rank 8) CACM-2632 (Rank 40) CACM-2870 (Rank 12)

4. Results

The following table shows the evaluation results for the retrieval models implemented in this project:

Retrieval Model	Mean Average Precision (MAP)	Mean Reciprocal Rank (MRR)
BM25 No Relevance	0.441202	0.653113
TF-IDF	0.144948	0.243673
JM Smoothing QL	0.120698	0.136232
Lucene	0.39708	0.65320
BM25 using stopping	0.490334	0.735402
TF-IDF using stopping	0.347754	0.587328
JM Smoothing QL using stopping	0.257859	0.463735
BM25 Pseudo Relevance Feedback	0.379848	0.545574

The following graph shows the Precision vs Recall graph for the retrieval models implemented for this project:



Precision vs Recall Graph

The following table shows the query level results for the top 1, 2, 5 and 10 documents. This part was implemented using Pseudo Relevance feedback for query expansion.

No. of Documents	**Mean Average Presicion (MAP)**	**Mean Reciprocal Rank (MRR)**
1	0.378967	0.545574
2	0.378893	0.547883
5	0.379848	0.545574
10	0.379849	0.545577

5. Conclusions and Outlook:

Looking at the results and analyzing on them we find that BM25 works best amongst other retrieval models. It has better MAP and MRR values and also brings relevant documents at the top more consistently. Also we observe that the retrieval models applied with stopping have better results than the ones without stopping for each corresponding retrieval system especially BM25 when considered with stopping. Lucene provides considerate results is also equally stable.

The performance of this retrieval system can be improved if the retrieval considered the user's search history which will improve the query refinement and relevance of documents to the user. Thus query expansion considering user's search history and query logs could serve better than using Pseudo Relevance feedback or query expansion. In future even NLP techniques could be applied to get results based on a closer meaning than just considering the words and their stems for finding relevant documents.

6. Bibliography

1. Search Engines: Information Retrieval in Practice by Croft, Metzler, Strohman. (Text book)
2. Python documentation [https://docs.python.org/3/]
3. Beautiful Soup documentation [https://www.crummy.com/software/BeautifulSoup/bs4/doc/]