Search Engine

CS6200: Information Retrieval
Northeastern University
Fall 2017, Prof. Nada Naji

Team Members:
Varad Choudhari
Tanmay Sinha
Tejas Parab

1.Introduction:

1.1 Overview:

The goal of the project was to design and build our own information retrieval systems, evaluate and compare their performance levels in terms of retrieval effectiveness. The project is mainly a search engine on the CACM test collection. The search engine is based on the following retrieval models:

- BM25 retrieval model
- tf-idf retrieval model
- Smoothed Query Likelihood model
- Lucene Simple Analyzer

Along with these four baseline runs, pseudo relevance feedback was used for query expansion and retrieval for the BM25 retrieval model.

The following was also performed on the BM25 retrieval model, tf-idf retrieval model and Smoothed Query Likelihood model:

- Stopping with no stemming.
- Indexing the stemmed version of the corpus (cacm_stem.txt) and stemmed query subset (cacm_stem.query).

Apart from the above seven distinct runs, we also implemented snippet generation technique and query term highlighting for the BM25 retrieval model.

On completing these eight distinct runs, performance was assessed in terms of the following measures:

- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- P@K measure where K=5 and K=20
- Precision & Recall

1.2 Member Contribution:

- **Varad Choudhari**: Varad was responsible for implementing the BM-25 retrieval model and the Lucene systems. He was also responsible for implementing Pseudo Relevance feedback on the BM25 retrieval system. Varad also designed and implemented the snippet generation and query highlight method for extra credit. Varad was also responsible for the corresponding documentation of design choices for his implementations.

- **Tanmay Sinha**: Tanmay was responsible for implementing the tf-idf and Smoothed Query Likelihood retrieval model. He was also responsible implementing the stemming and stopping techniques incorporated by the search engine. Tanmay was further responsible for the query by query analysis of the stemmed, non-stemmed runs and corresponding documentation of design choices for his implementations.

- **Tejas Parab**: Tejas was responsible for implementing the evaluation measures such as MAP, MRR, P@K and Precision & Recall for the eight distinct runs. He was also responsible for documenting the design choices used by him and completing the documentation for Conclusions and Outlook section of the report.

2. Literature and Resources:

2.1 Overview:

- **Query Expansion**: Pseudo relevance feedback have been used in order to perform query expansion.
- **Stopping**: 'common_words.txt' has been used in order to perform stopping. The words appearing in the list have not been indexed.
- **Stemming**: Stemming has been performed using the BM25 model with the help of 'cacm_stem.txt' and 'cacm_stem.query' which are the stemmed version of the corpus and queries respectively.
- **MAP**: The formula used to calculate the Mean Average Precision is:

$$\text{MAP} = \Sigma \text{ Average Precision} / \text{Number of Queries}$$

- **MRR**: Reciprocal Rank is the Reciprocal of rank of earliest relevant document.

  Mean Reciprocal Rank is the average of the reciprocal ranks of all the queries in the system.

  MRR = Σ Sum of Reciprocal Rank of all queries/ Number of Queries
- **Precision & Recall**: Recall measures how well the search engine is doing at finding all the relevant documents for a query, and precision measures how well it is doing at rejecting non-relevant documents.
  - Precision can be measured as: |Relevant ∩ Retrieved| / |Retrieved|
  - Recall can be measured as: |Relevant ∩ Retrieved| / |Relevant|
- **P@K**: Precision at K is calculated as the number of relevant documents in the top K retrieved documents. We have calculated for K=5 and K=20 in the project.
- **tf-idf**: It stands for term frequency and inverted document frequency. The tf-idf for each document is calculated as the sum, over all query terms, of the number of times each of the query terms occurs in the document multiplied by inverse document frequency (Total number of documents / Number of documents with term t in it). This algorithm is used as a ranking algorithm in one of the baseline runs.
- **BM25 model**: The BM25 Retrieval model is used to calculate the score and rank of document based on their relevance to the query. We have used 'cacm.rel' as the set of relevant documents. Values for 'K', 'k1', and 'k2' are chosen as per TREC standards. This algorithm is used as a ranking algorithm in one of the baseline runs.
- **Query Likelihood model**: In the query likelihood model we rank the documents by the probability that the query text could be generated by the document language model. This is a model of topical relevance, where the probability of query generation is the measure of how likely it is that a document is about the same topic as the query. This algorithm is used as a ranking algorithm in one of the baseline runs.
- **Lucene**: We have used the standard Lucene library with minor modifications to perform indexing and search operations.
- **Snippet Generation**: Query terms were used to score and classify sentences as significant. The more number of query terms the sentence contained the more it was scored. Finally, top 3 ranking sentences were displayed as snippet with query terms being highlighted.

2.2 Resources:

- **Beautiful Soup:** Beautiful soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. It has been used for parsing purpose to process the documents in the corpus and the queries.
- **Lucene Libraries:** The following Lucene libraries have been used in the project:
  - lucene-core-VERSION.jar
  - lucene-queryparser-VERSION.jar
  - lucene-analyzers-common-VERSION.jar

3. Implementation and Discussion:

3.1 Baseline runs:

3.1.1 tf-idf retrieval:

This is the basic retrieval model which serves as a base for many retrieval algorithms. The rank of a document should be directly proportional to the frequency of the query term in a document and inversely proportional to the number of documents the term occurs in the corpus. i.e.,

$$tf= f/|D|$$

$$idf= \log(N/n_k)$$

scores are calculated by multiplying these two terms.

In order to perform ranking by stopping, we performed stopping on the queries and the corpus and scored them.
The above process was repeated while ranking documents using stemmed queries.

3.1.2 BM25 retrieval model:

The following formula was used for scoring documents by bm25 algorithm:

$((k_2 + 1)q)/((k_2 + q)) * ((k_1 + 1)f)/((K + f)) * \log((r + 0.5)(N - n - R + r + 0.5))/((n - r + 0.5)(R - r + 0.5))$

Where $K = k_1(bL + (1 - b))$
'k_1', 'k_2' and 'b' are normalization constants.
Relevance information was not considered while scoring. Hence, R=r=0.
L is the normalized document length (i.e. the length of this document divided by the average length of documents in the collection).
q is the frequency of the term in the query.
f is the frequency of the term in the document.
n is the number of documents in which the query term occurs.

By using the above formula scores for each document were calculated for a query. Top 100 documents were retrieved for each query were retrieved.

In order to perform ranking by stopping, we performed stopping on the queries and the corpus and scored them.
The above process was repeated while ranking documents using stemmed queries

3.1.3 Query Likelihood model:

In order to prevent the score to be equal to 0 in case a query term does not appear in a document, we used Jelinek-Mercer Smoothing.

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

where P(Q|D) denotes the probability that a query can be generated from a document.
Here, $\lambda=0.35$
$f_{qi,D}$ is the frequency of the term in the document
|D| is the length of document D.
c_{qi} is the frequency of a term in the entire corpus C.
|C| is the total word occurrences in the corpus.

In order to perform ranking by stopping, we performed stopping on the queries and the corpus and scored them.
The above process was repeated while ranking documents using stemmed queries.

3.1.4 Lucene SimpleAnalyzer:

We used the inbuilt API provided by Lucene to score the documents for the queries. We used the simple analyser which is a combination of vector space and Boolean model.

3.2 Pseudo Relevance Feedback:

- For implementing pseudo relevance feedback we have used the technique mentioned in 'Search Engines – Information Retrieval in Practice' textbook which states that instead of asking the user to identify relevant documents, the system assumes that the top-ranked documents are relevant. Words that occur frequently in these documents may then be used to expand the initial query.
- Following are the steps to perform query expansion using pseudo relevance feedback:
    1. Initial run using the BM25 retrieval model. The results generated from this run are used to perform pseudo relevance feedback.
    2. Relevant set of documents is generated by assuming the top 5 documents to be relevant. (k = 5)
    3. From the relevant set of documents, top 4 words which are not stop words are selected from each document and added to a list. (n = 4)
    4. The words in this list are then appended to the original query.
    5. Normal retrieval is performed with the new query using the BM25 retrieval model.

3.3 Snippet Generation:

- Since the CACM corpus contains documents of minimal length, finding significant sentences of conventional length was not appropriate.
- Hence, we decided to consider a sequence of 5 words to be considered a sentence. We checked the occurrence of each query term in a sentence and accordingly scored them.
- We excluded the stop words from the query.
- If a non- stop word query term appears in a sentence we add 1 to it's initial score of 0. This process continues after we have checked all query terms in the query.
- We displayed the top 3 sentences, with the query terms occurring in them highlighted as *<query term>*, as the result.

3.4 Evaluation:

- Evaluation has been performed on all 8 distinct runs with results for all 64 queries.
- Following are the steps to perform evaluation:
    1. 'cacm.rel' has been used to get relevance judgement beforehand. Since 'cacm.rel' does not have information on all the queries, the presence of such queries has been ignored (as discussed with the Professor).

2. For each query, a Precision Recall table has been maintained which also contains information about Average Precision, Reciprocal rank, P@5, and P@20.
3. After processing all the queries, all the generated files are read again to compute the summation of precision and reciprocal rank. This step is needed to calculate the mean average precision (MAP) and Mean reciprocal rank (MRR).
4. To calculate the mean, divide the summation by the number of queries for which the relevance judgement is present.
5. Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are stored in 'summary.txt' for each system.

4 Results

Results for various runs and evaluations can be found in the locations mentioned in the ReadMe.txt file located in the root directory.

5 Conclusion and Outlook

5.1 Conclusion:

- Lucene model is working best for retrieval of relevant documents as the MAP and MRR are highest for it
- BM25 comes at 2nd place followed by tf-idf and query likelihood.
- Using pseudo relevance feedback for query expansion actually did not help in improving MAP or MRR. This could be due to the fact that the terms added aided in retrieval of non-relevant documents.
- Performing stopping on the queries did not make the results better. This could be possible if the stop words in the query were aiding in the retrieval of relevant documents.
- Query Likelihood yielded the worst results. Most of the P@K values were observed to be zero. This could be due the incorrect retrieval which depends on the corpus frequency of a term.

5.2 Outlook:

The search engine developed in course of this project incorporates most of the features and functionalities of a basic information retrieval system. However, certain features can be added in future to improve the project. The features are mentioned as below:

- Query expanding techniques such as spellcheck could be implemented in the future in order to improve the efficiency of the search engine and reduce the influence of misspelled words.
- A ‘PageRank’ algorithm can be used in order to calculate the popularity of a page. By counting the number of inlinks and outlinks to a page, we could estimate how important the website is. Thus, improving the search results further.
- UI could be added in the future for better experience for the user to enter queries and see the retrieved results.

6 Bibliography:

BOOKS:

- Search Engines – Information Retrieval In Practice by W.Bruce Croft, Donald Metzler, Trevor Strohman, 2015
- An Introduction to Information Retrieval by Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze