Table of Contents

1. Project Details 2
2. Introduction 3
3. Literature and Resources 4
3.1 Retrieval Models 4
3.2 Query Expansion 4
3.3 Evaluation 4
4. Implementation and Discussion 5
4.1 Retrieval Models 5
4.2 Query Expansion 6
4.3 Stopping and Stemming 6
4.4 Evaluation 6
4.5 Query by Query Analysis 7
5. Results 8
6. Conclusion and Outlook 9
7. Bibliography 10

1. Project Details

Course Details:
Course Name: CS6200 Information Retrieval
Semester: Spring-2017
Instructor: Prof. Nada Naji

Project Members':

1) Dhavalkumar Patel
2) Dhritiman Ganguly
3) Oneil Contractor

2. Introduction

We have built three search engines using three different retrieval models (i.e. BM25, tf.idf and Lucene's default retrieval model) and evaluated the effectiveness of each search engines using various evaluation measures like precision, recall, MAP, MRR etc. We have applied some techniques to improve the effectiveness like query expansion on the baseline version of the search engine. We have used Pseudo Relevance Feedback approach for query expansion. We have also measured the effectiveness of each search engine after applying stemming and stopping too.

The work distribution for this project is as shown below.

Task1:
Implementation of Search Engines using tf-idf, BM25 and Lucene as retrieval model: Dhaval & Oneil

Task 2:
Implementation of Pseudo-Relevance Feedback query expansion: Dhaval & Oneil

Task 3 - A & B:
Implementation: Dhaval & Dhritiman

Phase2:
Evaluation and Documentation: Dhaval, Oneil & Dhritiman

3. Literature and Resources

3.1 Retrieval Models

The presentations slides that were provided by Professor Nada Naji[1] and all the in-class explanation regarding the Tf.Idf, BM25 and Lucene were the main resources used. For implementing the TF.IDF and BM25, we referred to the textbook Search Engines: Information Retrieval in Practice by Croft, Metzler and Strohman[2]. And also in that we focused more on the chapter regarding retrieval models for understanding the concept of BM25 and Tf.Idf.

For understanding Apache Lucene, we referred to the HW4 provided by Prof. Nada Naji. In that assignment, we had implemented a retrieval model using Lucene. Along with that we have also looked over at the Lucene 4.7.2 API[3], so as to understand various APIs which can be used for implementing a retrieval model for the given set of queries.

3.2 Query Expansion

In this task, we were required to implement query expansion technique for the provided queries. Query expansion is the technique to improve the query which is a better representation of the user need and to improve the overall effectiveness of the search engines. We have used pseudo relevance feedback technique for implementing the query expansion. We have followed the same steps for pseudo relevance algorithm as given in the slides provided by Prof. Nada Naji[1] and also referred to the textbook Search Engines: Information Retrieval in Practice by Croft, Metzler and Strohman[2].

3.3 Evaluation

For Evaluation, we have gone through the class presentation slides provided by Prof. Nada Naji[1]. The calculation for precision, recall, MAP and MMR was very well explained by professor during the lectures and we have referred to the class notes. Some guidance was taken from the textbook[2] and from Piazza discussions[4] in order to find the boundaries for each value.

4. Implementation and Discussion

4.1 Retrieval Models

In this project, we are scoring the CACM documents for the given CACM queries using the Tf.Idf, Lucene and BM25 retrieval models and shortlist the top ranked (100) documents for each query in below format:

QueryID Q0 DocumentID Rank Score SystemName

1. **Tf.Idf** (Term frequency–Inverse document frequency) is a numerical statistic that is intended to reflect how important a word is to a document in collection or corpus[5]. In Tf.Idf retrieval model, score of a specific document for specific query term t is calculated by following equation:

   Score(D,t) = TF(t)*IDF(t) where TF and IDF as defined below for document D:

   - **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:
     TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

   - **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
     IDF(t) = log(Total number of documents / Number of documents with term t in it).

2. **BM25** is a probabilistic model and an extension of a binary independence model to include document and query term weights. It uses the following formula to calculate the scores of a document:

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

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

In this equation, ri is the number of relevant documents containing term i, ni is number of documents containing term i, N is total number of documents in the collection, R is the number of relevant documents for this query, fi is the frequency of term i in the document, qfi is the frequency of term i in query and k1, k2, and K are the parameters. We are using standard values for these parameters as given in the textbook like K1 = 1.2, K2 = 100, b = 0.75. We have calculated R and ri from the relevance judgments information of the query. Here, effect of fi will be very non linear, similar to the use of log f in term weights due to k1=1.2. And k2 is used for the same purpose but for query terms and b is used for document length normalization.

3. **Apache Lucene** is an open-source and free retrieval model library which will create an index for the document corpus and produce and list of the top 100 documents for a given query. We used Lucene's default retrieval model (SimpleAnalyzer for indexing) [3].

4.2 Query Expansion

We have used pseudo relevance feedback technique to expand the query for Tf.Idf and BM25 retrieval model. It is an automated technique to expand the query where instead of asking the user to identify relevant documents, the system simply assumes that the top-ranked documents are relevant. Words that occur frequently in these documents are then be used to expand the initial query. For our project, we are considering top 10 documents as relevant from the first search and add the top 10 most frequent terms to the query (discarding the stop words using given stop list) such that query will have 10 additional terms after query expansion. We come to these values by analysing the behaviour of the retrieval models with different values and measured the effectiveness of the retrieval model by calculating MAP and MRR values. We tuned these values such that it gives best MAP and MRR values for the particular retrieval model.

4.3 Stopping and Stemming

For stopping, we created a lookup table of all the stop words (from "common_words.txt") and the tokens generated from the corpus are then looked up and compared with this stop word lookup table. If a token is already present in the stop word table, then it is removed, else we keep it. This process is repeated for the tokens in all the documents and the query. For task B, we used the stemmed corpus ("cacm_stem.txt") as an input to BM25 and Tf.Idf retrieval model and generated top ranked documents for the given stemmed queries.

4.4 Evaluation

Each evaluation run on any search engine's query result will generate following three files.

1) File to capture precision and recall value at each step for each query in below format:
   <QueryId Rank Precision Recall>
2) File to capture precision at 5 and precision at 20 along with average precision and reciprocal rank of each query (with relevance information) in below format:
   <QueryId PrecisionAt5 PrecisionAt20 AveragePrecision ReciprocalRank>
3) File to capture MAP and MRR for specific run

MAP: For computing the Mean Average Precision, we find the average precision values of the relevant documents for every query who has relevance information and then took average of the for each retrieval model. If the search engine does not find any relevant document for a query which has relevance information, then its average precision is considered 0 (as a penalty) and it is considered for calculating MAP.

MRR: The reciprocal rank is calculated of all the queries with relevance information and then we took average of it. If the search engine does not find any relevant document for a query which has relevance information, then its reciprocal rank is considered 0 (as a penalty) and it is considered for calculating MRR.

PRECISION at Kth Rank:

Specifically, the precision at rank 5 and rank 20 are considered. Also the precision and recall at all the hundred rank position were calculated for all seven runs.

4.5 Query by Query Analysis

For query "portabl oper system", BM25 and TfIdf both have same first document and which is relevant too. This is because query terms are occurring at high frequency in this document. Second document is different and BM25 finds the relevant one whereas TfIdf finds the non relevant document at second place. This is because all the other documents have very less frequent query terms in them and the score of all the other documents is very similar using TfIdf model. Whereas in BM25, due to relevance information, score of the relevant document gets a boost and they come before non relevant one.

So we can say stemmed query terms rarely appeared in the document corpus and the top ranked documents were those which had the query terms in high frequency. The rank of a document is almost commensurate to the query term frequency found in the document. But stemming a token may cause problems like over-stemming and we may retrieve non-relevant documents too. In our query, due to over-stemming operating to oper, a lot of non-relevant documents like CACM-1930 will also get retrieved and appear as a top document in the results list. Hence we see the difference in the ranked document list of the stemmed and non-stemmed versions.

For query "distribut comput structur and algorithm", both search engines gave non relevant document as first document. This is because the other non relevant document had some of the query terms with very high frequency, e.g. Document with id 2949 is on "distribut comput network" and have a very high frequency of "distribut" and "comput" terms. This means then rather than just looking at the query term weights, we should consider the term positioning too in the document.

For query "parallel processor in inform retriev", due to absence of relevance information, both BM25 and Tf.Idf produced slightly similar results and they are not that effective as some documents with other parallel topic has higher frequency of terms and they got the higher score.

5. Results

Below are the evaluation results (MAP and MRR) for all runs:

Run-No	Run-Name	MAP	MRR
Run-1	BM25 Baseline	0.547084042	0.849038462
Run-2	TfIdf Baseline	0.292083264	0.517013033
Run-3	Lucene Baseline	0.422040115	0.709344475
Run-4	BM25 with Query Expansion	0.566636652	0.88548344
Run-5	TFIdf with Query Expansion	0.338021047	0.537605774
Run-6	BM25 with Stopping	0.555712361	0.879807692
Run-7	TfIdf with Stopping	0.365626699	0.596904616

For query level results, we have submitted EvaluationResults.xlsx file which covers results for all seven runs in different tabs.

6. Conclusion and Outlook

We found that the retrieval model is a very crucial part in a search engine but at the same time the text pre-processing (de-punctuation, stemming and stopping) plays a key role too. We found that BM25 is the most effective retrieval model for the given corpus and this is because it uses the relevance information while calculating the score of a document. Other approaches are dependent on term frequency like features only. As we can see in all search engines results, query expansion technique improved the effectiveness of all the retrieval runs. Removing stop word also increased Map and MRR values as more relevant documents are retrieved at higher rank. The query expansion technique is not that effective as it might introduce the noise in the query that is expanded and that may result in the reduced precision as more non relevant documents will be retrieved at higher ranks.

As we found in our query analysis, rather than looking only at term frequency, we should also consider the positioning of the words too and for this we can use bigram or trigram instead of unigram. For un-stemmed corpus, we would like to introduce the query time stemming and would like to generate the stemmed class by measuring word co-occurrence analysis. This will expand the query with some more related words from newly generated stemmed classes. We can also design the GUI for the search engine and from the real user query logs and using click throughput we can come up with more accurate relevance information.

7. Bibliography

1. Course Material provided by Prof. Nada Naji
2. Textbook: Search Engines: Information Retrieval in Practice by W. Bruce Croft, Donald Metzler and Trevor Strohman
3. http://lucene.apache.org/core/4_7_2/
4. Piazza Discussions
5. http://www.tfidf.com/