Information Retrieval Course Project

Retrieval Systems Construction, Optimization and Evaluation

Meesam Syed
Neha Shukla
Shravya Raj

Semester: Spring 2018

Instructor: Professor Nada Naji

Institution: Northeastern University

1 Introduction

1.1 Brief Description

In this Information Retrieval system, we have implemented the two important phases of building search engine - Indexing and Querying. Our search engine architecture therefore comprises an Indexer and a Retriever described in the later part of this report.
We have developed three standard retrieval models - BM25, tf-idf, Smoothed Query Likelihood(SQLM) in this retrieval system. Additionally, we have used Lucene (4.7.2) as a classic search engine library in Java to compare with other models. SQLM supports query expansion technique which has been explained in the next section. Moreover, we have implemented snippet generation technique to enhance user experience.
At the end, we have performed evaluation experiments to verify and compare the results of several retrieval models. Evaluation results have been discussed in the later section.

1.2 Contribution

Neha Shukla developed 3 retrieval models, Lucene search engine, Query Refinement approach from Phase 1. Meesam Syed implemented Snippet generation technique and Evaluation Phase from Phase 2 and 3 respectively. Shravya Raj added stopping and stemming techniques on Indexer and Retriever.

2 Literature and Resources

2.1 Query Refinement Approach

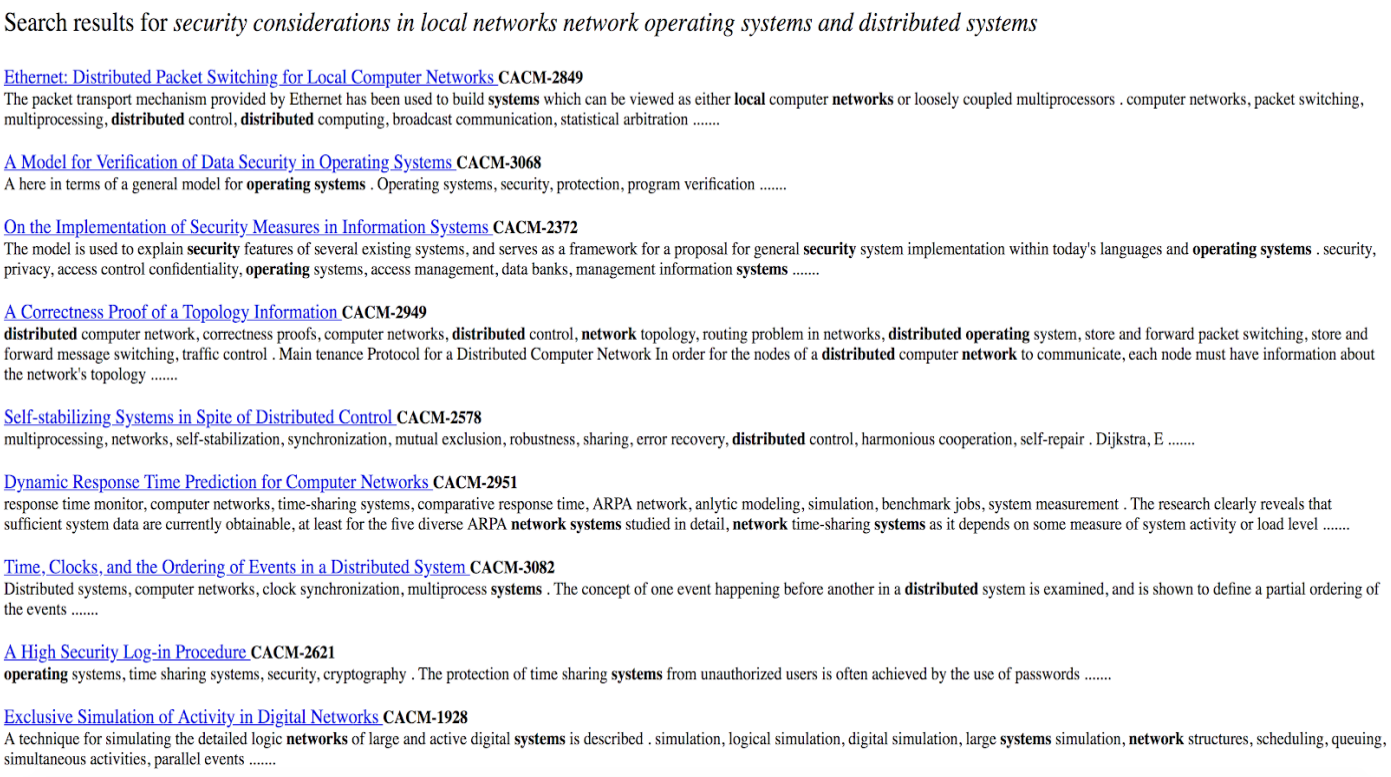
Pseudo Relevance Feedback is one of the most effective and popular techniques for query expansion. In our query expansion strategy, we have assumed that the most significant terms in the documents and queries are less frequent terms in the collection (Luhn (1958) for word significance depends on the frequency). We realized this idea by considering set of unigrams (Sb, where $|Sb| < 10$) from the relevant documents, where unigrams are ranked in the ascending order of their term frequency. We performed stopping on queries and applied the same strategy for choosing the set of unigrams (Sa, where $|Sa| < 5$) from the query.

Now to find the association between the words from these two sets - Sa (from query) and Sb (from relevant documents), we have calculated Dice's coefficient for each words (na) in set Sa with each words(nb) in set Sb by using this formula:

$$2nab / (na + nb)$$

We used the term association scores for these words to analyze and perform the selection of the candidate terms from the relevant documents for query expansion. By doing this, we ensure that only highly associated terms(top 10 rank) are added to the query.
For example, our query expansion words for the cacm query - "What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers?" were – 'time-shared', 'statistics'. These words are taken from result documents, also are the highly associated words with query, selected in the decreasing order of their dice's coefficient score.

2.2 Snippet Generation Approach



Search results for *security considerations in local networks network operating systems and distributed systems*

Ethernet: Distributed Packet Switching for Local Computer Networks **CACM-2849**
The packet transport mechanism provided by Ethernet has been used to build **systems** which can be viewed as either **local** computer **networks** or loosely coupled multiprocessors . computer networks, packet switching, multiprocessing, **distributed** control, **distributed** computing, broadcast communication, statistical arbitration .....

A Model for Verification of Data Security in Operating Systems **CACM-3068**
A here in terms of a general model for **operating systems** . Operating systems, security, protection, program verification .....

On the Implementation of Security Measures in Information Systems **CACM-2372**
The model is used to explain **security** features of several existing systems, and serves as a framework for a proposal for general **security** system implementation within today's languages and **operating systems** . security, privacy, access control confidentiality, **operating** systems, access management, data banks, management information **systems** .....

A Correctness Proof of a Topology Information **CACM-2949**
distributed computer network, correctness proofs, computer networks, **distributed** control, **network** topology, routing problem in networks, **distributed operating** system, store and forward packet switching, store and forward message switching, traffic control . Main tenance Protocol for a Distributed Computer Network In order for the nodes of a **distributed** computer **network** to communicate, each node must have information about the network's topology .....

Self-stabilizing Systems in Spite of Distributed Control **CACM-2578**
multiprocessing, networks, self-stabilization, synchronization, mutual exclusion, robustness, sharing, error recovery, **distributed** control, harmonious cooperation, self-repair . Dijkstra, E .....

Dynamic Response Time Prediction for Computer Networks **CACM-2951**
response time monitor, computer networks, time-sharing systems, comparative response time, ARPA network, anlytic modeling, simulation, benchmark jobs, system measurement . The research clearly reveals that sufficient system data are currently obtainable, at least for the five diverse ARPA **network systems** studied in detail, **network** time-sharing **systems** as it depends on some measure of system activity or load level .....

Time, Clocks, and the Ordering of Events in a Distributed System **CACM-3082**
Distributed systems, computer networks, clock synchronization, multiprocess **systems** . The concept of one event happening before another in a **distributed** system is examined, and is shown to define a partial ordering of the events .....

A High Security Log-in Procedure **CACM-2621**
operating systems, time sharing systems, security, cryptography . The protection of time sharing **systems** from unauthorized users is often achieved by the use of passwords .....

Exclusive Simulation of Activity in Digital Networks **CACM-1928**
A technique for simulating the detailed logic **networks** of large and active digital **systems** is described . simulation, logical simulation, digital simulation, large **systems** simulation, **network** structures, scheduling, queuing, simultaneous activities, parallel events .....

Figure 1: BM25Model Snippet for Query No.9

A concise and meaningful snippet will be helpful to user to understand the contents of the document.
We developed an efficient algorithm to generate snippets which is dependent on the query, the algorithm will search for the most important and meaningful sentences from the document. A score is given to each sentence based on the query which indicates how significant the sentence is in the document with respect to the query.This algorithm is based on ideas provided in Bast and Celikik (2009)

Snippet generation strategy is as follows:

- First and foremost, document text is broken into sentences
- A score is calculated for each sentence in the document which is decided by the frequency of query words in the sentence (higher the number of query words in the sentence, higher will be the score of the sentence). Stop words are removed from the query before the calculation of score.
- Sentence with higher score are selected for the snippet
- The title of the document is the first line present in that document. As the document is best described by its title, it is also considered to be a part of snippet and show as the first line of the snippet followed by document name with rest of the snippet after that.
- Query words which occur in the snippet are highlighted to indicate their presence

A sample result snippet has been presented in the Figure 1 generated for Query No 9. using BM25 Model.

3 Implementation

3.1 Indexer

As mentioned in briefly in the introduction, our retrieval system has been divided into two major components - Indexer and Retriever. Indexer component includes everything that has been done till the Index is ready to be retrieved. It includes - CorpusBuilder and IndexBuilder.

CorpusBuilder module is devoted to parse html files,handle punctuation and case-folding, and finally saves corpus files in a folder which will be later on used by IndexBuilder module.
Retriever Component includes RetrievalModel module which is further extended by BM25Model, SmoothedQLM, and tf-idf retrieval models. In this parent-child hierarchy, RetrievalModel(Parent) performs major retrieval tasks such as - loading index, query expansion, producing stemmed queries, saving result, which are common to all the child models. On the other hand, every child retrieval models overrides computeScore() function to implement its own document scoring technique for (Q,D).

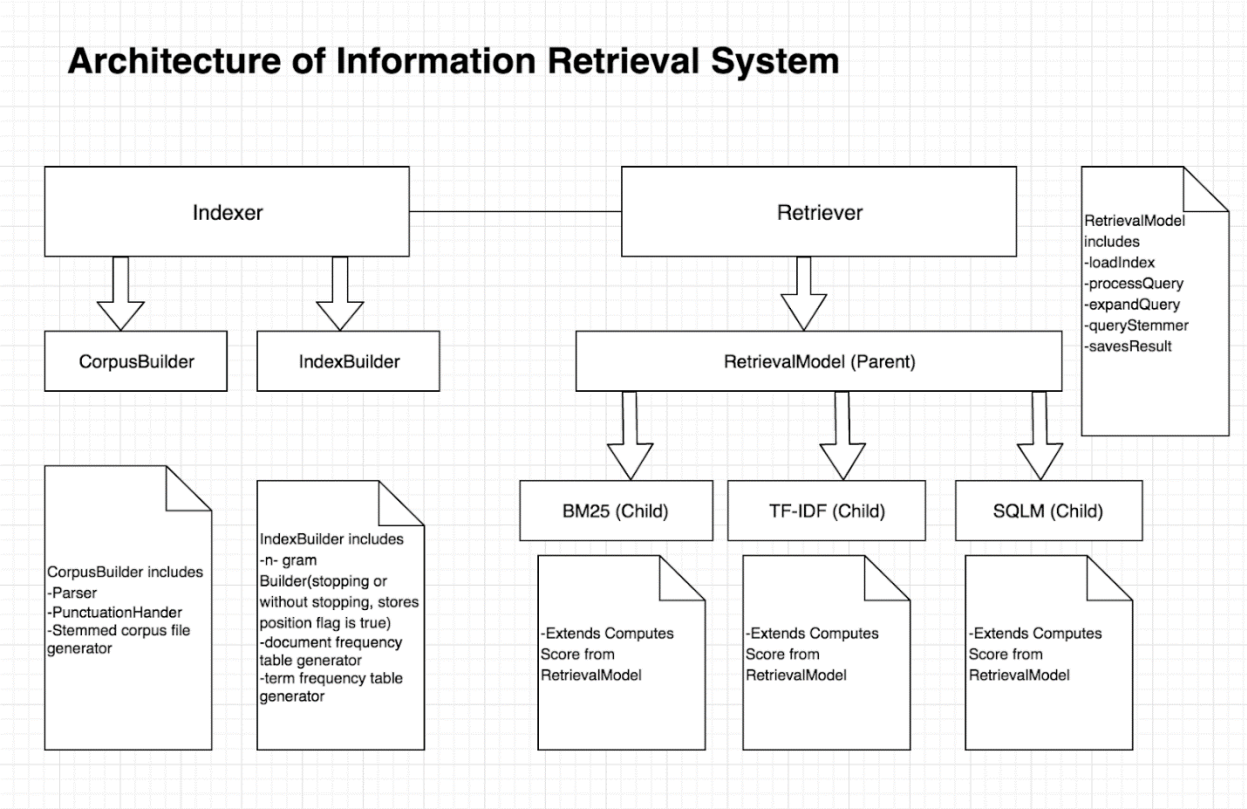


Figure 2. Architecture of Information Retrieval system (Design and Implementation)

3.2 Retrieval Models (BM25, SQLM, tf-idf)

The design of three retrieval models in the Python retrieval system are pretty straightforward. Since the ingredients required to compute document score at each retrieval model has been computed in the common parent class, there is a great amount of flexibility in selecting or extending any new retrieval model in this information retrieval system. Another brownie point in designing this retrieval model is that the inverted index is pre-loaded (run once, use later for n-queries), which reduces the seek time and increases query throughput.

Query by query analysis for baseline runs with their stopped versions

We can see that for Query 2, BM25 and SQLM retrieved the same document at rank 1, whereas the 2nd ranked document for BM25 was retrieved at the 3rd rank for SQLM. By analyzing each documents that were retrieved, we conclude that all the top 3 documents retrieved for BM25 run were relevant. SQLM was able to retrieve only 2 (2129 does not answer the query 2). Lucene was comparable to SQLM and retrieved only 2 relevant documents For stopped versions of these baseline runs, the retrieval rank for SQLM improved (Doc number 2434 jumped to ranking 2 and was found relevant)

Baseline	**DocID**	**Score**	**Stopped**	**DocID**	**Score**
BM25Model	3078	31.83	BM25 Stopped	3078	26.17
	2434	27.91		2434	24.79
	2863	23.28		2863	20.91
SQLModel	3078	-77.51	SQLModel Stopped	3078	-40.366
	2129	-79.55		2434	-41.02
	2434	-80.06		0397	-41.05
Tf.Idf	3059	0.472	Tf.Idf Model Stopped	2371	0.731
	0727	0.389		1591	0.689
	0335	0.381		1519	0.574
Lucene	2434	0205			

	3078	0188			
	3129	0135			

Query-by-Query analysis for BM25, SQLM, Tf.Idf with stemming

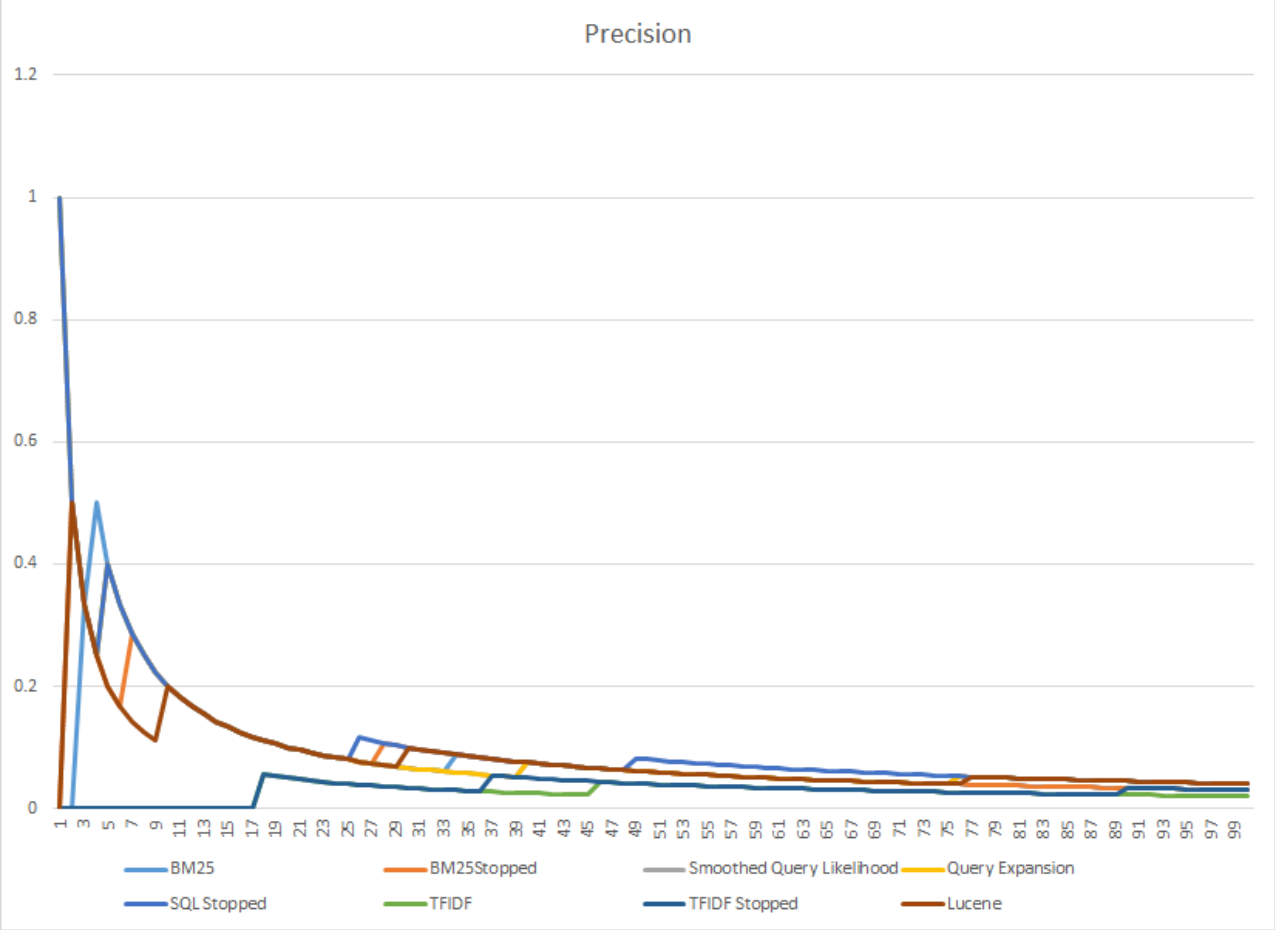
As we can see from the result table, the top 2 results for the Query 1 are same for each retrieval model. Moreover, the results are identical for BM25 and SQLM. After reading Doc 3127, we found it the most relevant for Query 1. Tf.Idf has improved greatly with stemming as compared to its baseline run (2 out of 3 documents are comparable with BM25 and SQLM).

Retrieval Model	**DocID**	**Score**
BM25Model stemmed	3127	14.695
	2246	11.405
	1930	9.471
SQLModel stemmed	3127	-9.485
	2246	-10.467
	1930	-11.055
Tf.Idf Stemmed	3127	0.409
	2246	0.283
	2311	0.235

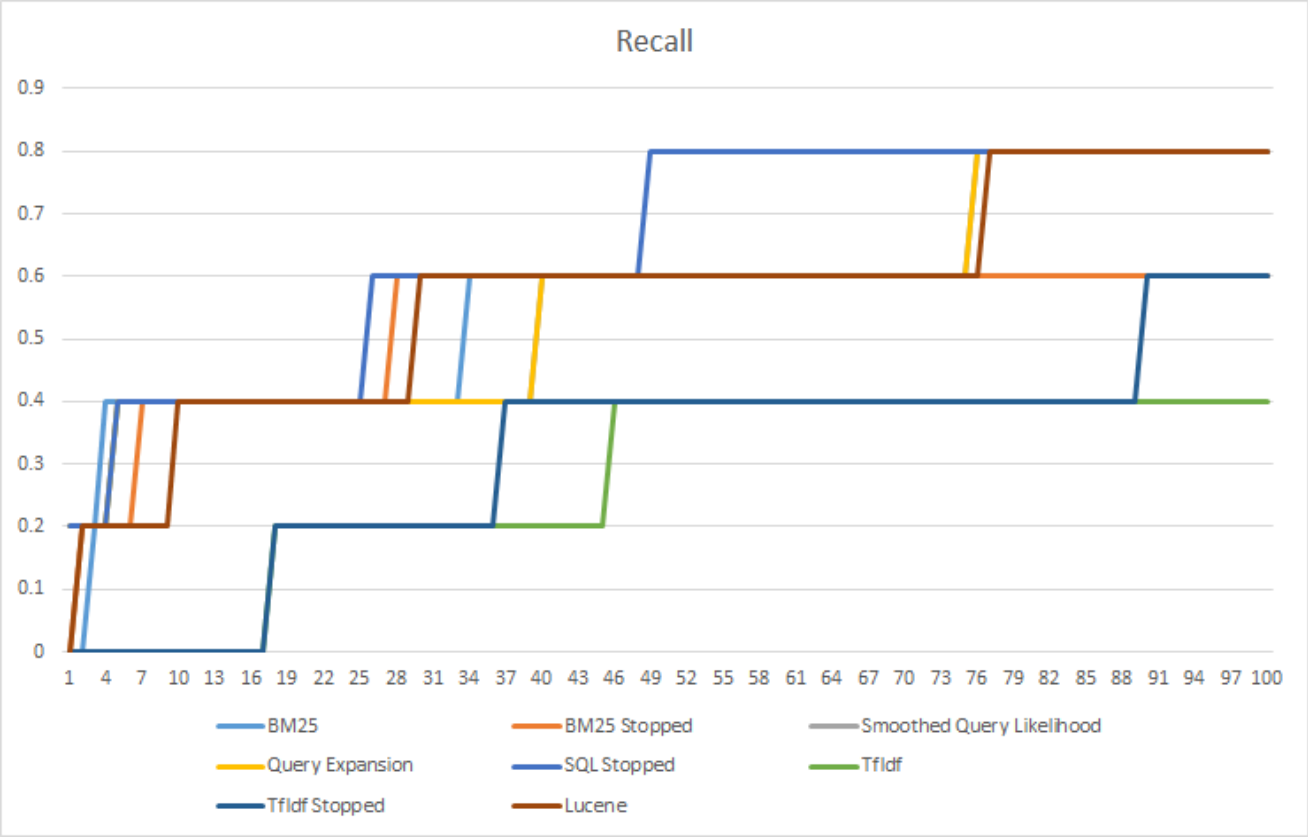
4 Evaluation

4.1 Precision and Recall

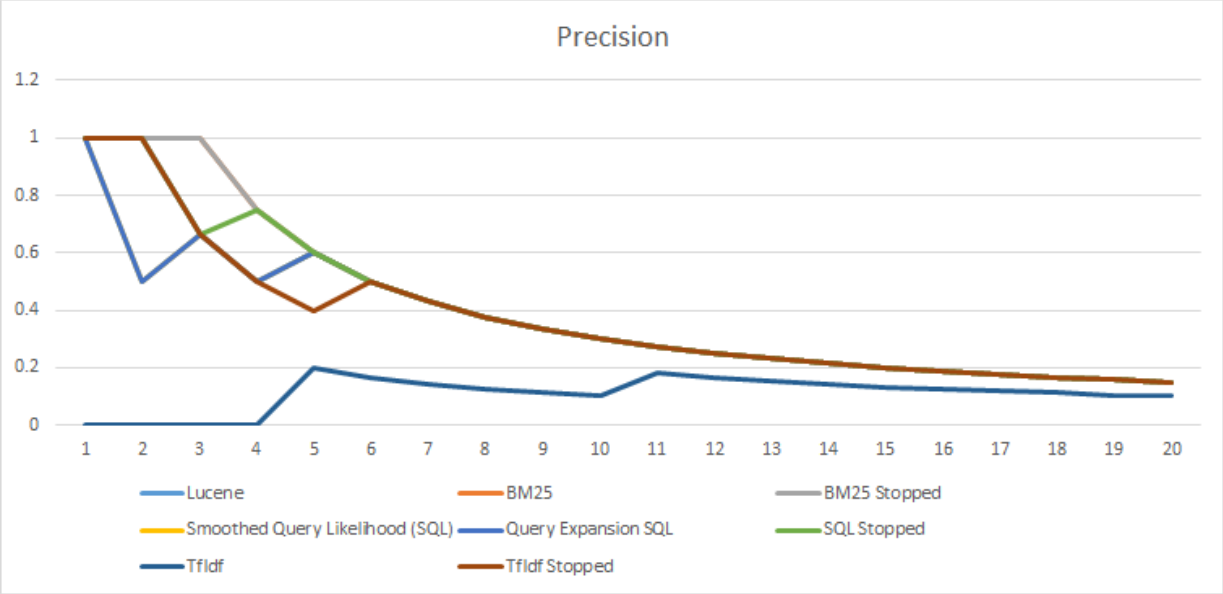
Precision is defined as proportion of retrieved document that are relevant whereas Recall is the proportion of relevant documents that are retrieved. Precision and Recall graphs are shown in Figure 1 and Figure 2 respectively for Query 1.From Figure 3 and 4, we can see that BM25 and Smoothed Query Likelihood retrieval models are better retrieval model compared to others. For, Smoothed QL, Query Expansion, SQL stopped, the first document retrieved is relevant. Overall BM25 and SQL have good precision and recall values. Whereas Tf.Idf models have the worst performance compared to other models. Both its variants have a low precision and low recall value meaning most of the top documents that are returned are not relevant. Moreover, the precision and Recall values for the top 10 documents is 0.Moreover when we look at the top 100 results BM25 and Smoothed Query Likelihood achieves a Recall close to 1.0 . Similar observations are obtained for Query 2 also



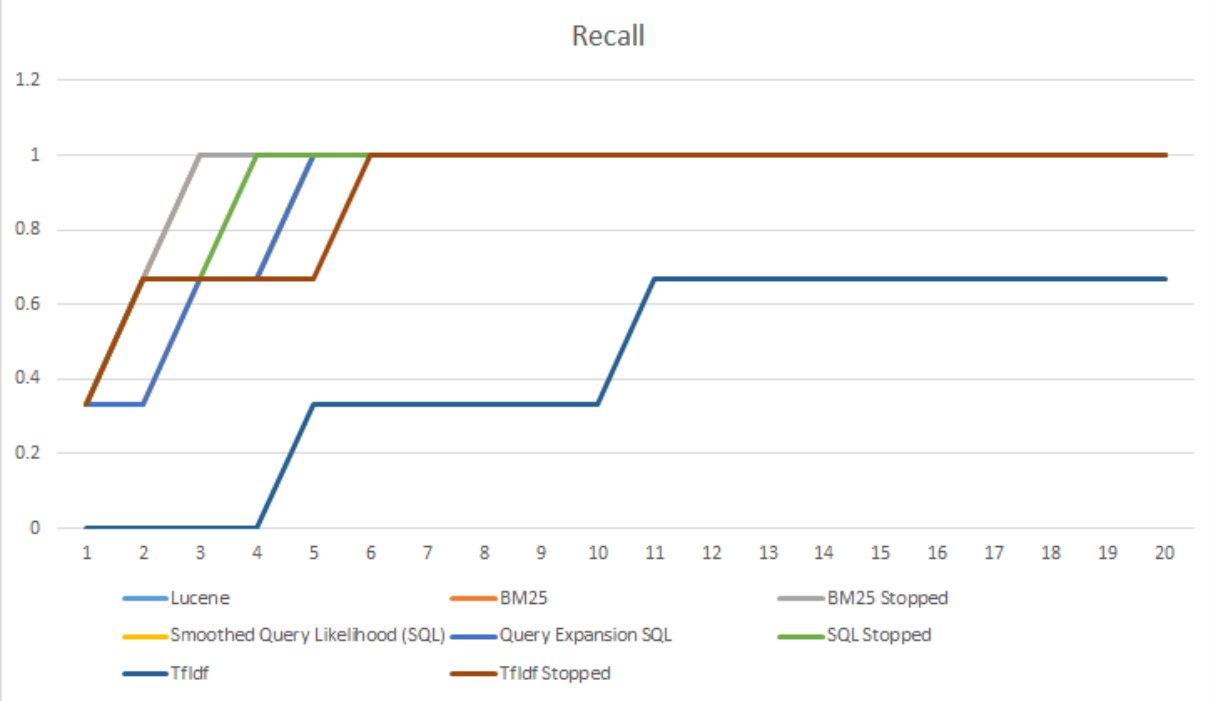
Precision for Query 1
Figure 3



Recall for Query 1
Figure 4



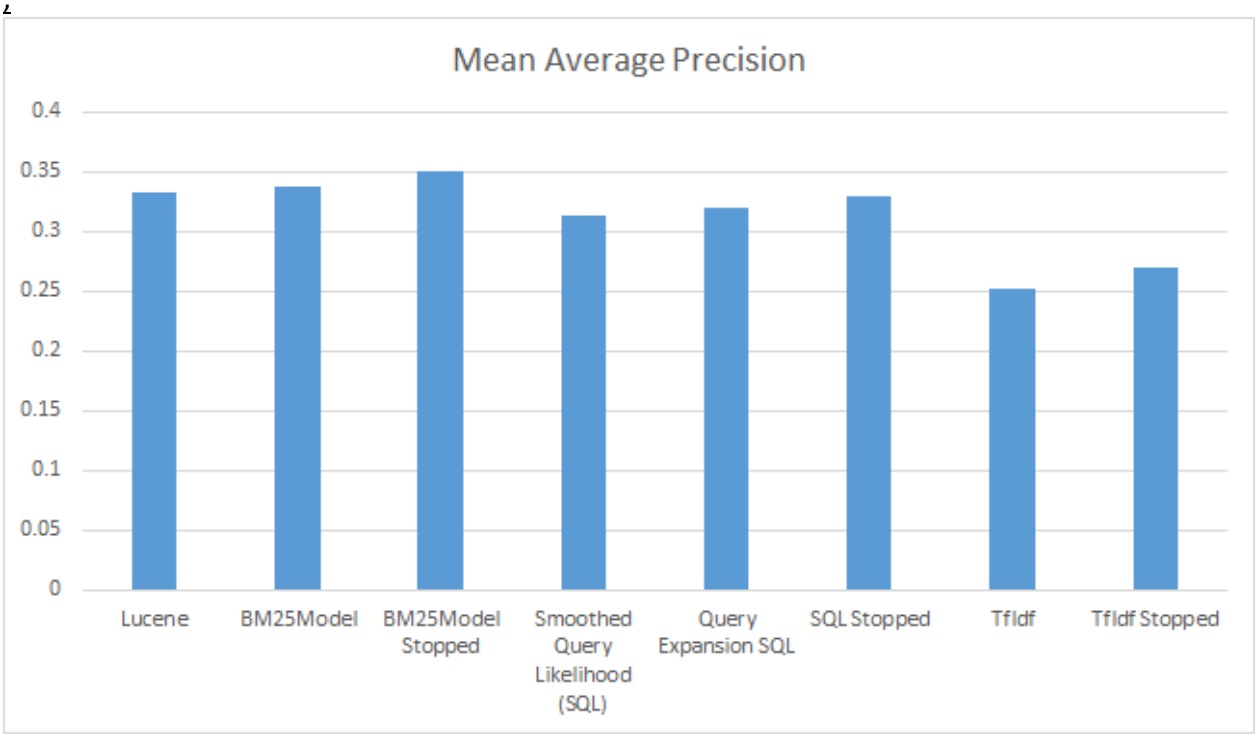
Precision for Query 2
Figure 5



Recall for Query 2
Figure 6

4.2 Mean Average Precision

Mean Average Precision is defined as the mean of average precision across queries for a particular retrieval system. From Figure 5, we can clearly see that all models of BM25 outperforms other retrieval models. Lucene comes after all the BM25 models. Again, it is observed that both models of Tf.Idf one without stopping and the one with stopping performs worst as compared to other model with mean average precision of approximately 0.26-0.29

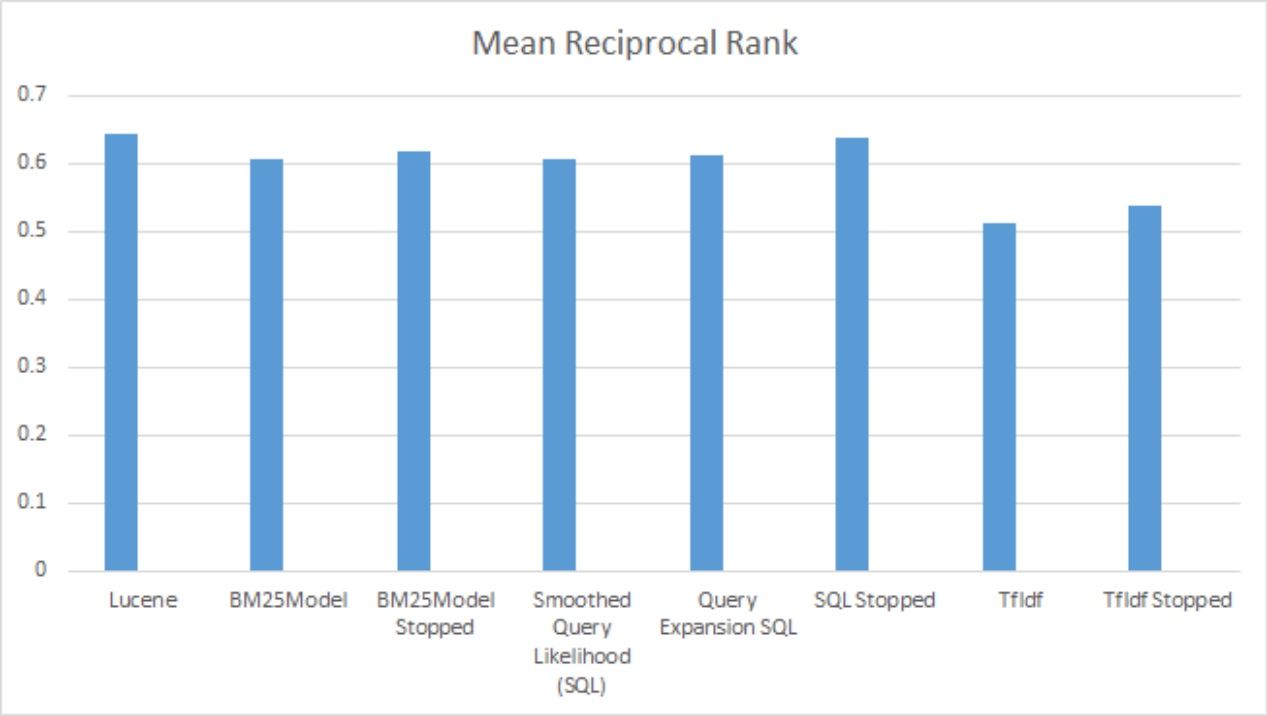


MAP for eight baseline model

Figure 7

4.3 Mean Reciprocal Rank

Mean Reciprocal Rank is defined as the mean of reciprocal rank across queries for a particular retrieval system. Reciprocal rank is the reciprocal of the rank at which the first relevant document is found.
From figure 6 we can see that BM25 models, Smoothed Query Likelihood and Lucene have high values of MRR, Once again the worst performance is given by Tf.Idf models with MRR values as low as 0.50



MRR for eight baseline models
Figure 8

4.3 P@K

Precision at rank k is defined as its name says Precision value at Rank k. Figure 7 and Figure 8 show the precision at rank 5 and 20 respectively. Once again BM25 model and Smoothed Query Likelihood model outperforms other. Tf.Idf as usual gives worst performance.

Retrieval Model	Query 1	Query 2	Query 4	Query 5
BM25	0.4	0.6	.2	0.2
BM25 Stopped	0.2	0.6	0.2	0.2
Smoothed Query Likelihood	0.4	0.6	0.1	0.2
Query Expansion	0.4	0.6	0.4	0.2
SQL stopped	0.4	0.6	0.4	0.2
Tf.Idf	0	0.2	0.2	0.2
Tf.Idf Stopped	0	0.4	0.2	0.2
Lucene	0.2	0.6	0.2	0.2

P@5 for first 4 queries for eight baselines
Table 3

Retrieval Model	Query 1	Query 2	Query 4	Query 5
BM25	0.1	0.15	0.1	0.1
BM25 Stopped	0.1	0.15	0.1	0.1
Smoothed Query Likelihood	0.1	0.15	0.1	0.1
Query Expansion	0.1	0.15	0.1	0.1
SQL stopped	0.1	0.15	0.1	0.1
Tf.Idf	0.05	0.1	0.1	0.05
Tf.Idf Stopped	0.05	0.15	0.15	0.05
Lucene	0.1	0.15	0.1	0.15

P@20 for first 4 queries for best baseline models
Table 4

Extra Credit:

1.Synthetic spelling error generator: This algorithm first calculates the number of words(totalsize) in the query and calculates the length of each word. Then it chooses 0.4*totalsize number of words (words are chosen based on their length, words with larger length are chosen). After selecting top 40% words, a random shuffler is used to shuffle all the character in word expect for the first and last character in it.

2. First of all, each query word is checked with the inverted index, if that query word is present in the index, we proceed with the normal processing. Else, we try to find words which are closely related to it by using levenshtein. If a similar word is found, we proceed with that otherwise the error word is altogether ignored from the processing

5. Conclusion

We have implemented BM25, BM25 with stopping, stemming, Lucene, Smoothed QL, Smoothed QL with stopping, stemming, Lucene, Tf.Idf, Tf.Idf with stopping, stemming. Some of the systems have high precision at higher ranks whereas some have high recall. Among all the systems, BM25 models are the best specifically BM25 Model with stopping because it has high precision, recall, MAP, MRR values. To, improve this project, we would like to include a spelling error corrector with high accuracy

References

Croft, W. B., Metzler, D., and Strohman, T. (2010). Search engines: Information retrieval in practice, volume 283.

Bast, H. and Celikik, M. (2009). Efficient index-based snippet generation.