# Information Retrieval Systems

Evaluation and Retrieval Effectiveness Comparison

(Submitted in partial fulfillment of the requirements of Northeastern University's CS6200 Fall 2017 course, taught by Prof. Nada Naji)

Athul **Muralidharan**

Meghna **Venkatesha**

Sachin **Haldavanekar**

# 1. Introduction

Throughout the Fall of 2017, the authors were exposed to various core Information Retrieval concepts. In this project, an attempt has been made to put the concepts into practice by building, evaluating and comparing different search engines.

The goal of this project is to build IR systems, make variations to improve the baseline runs, and compare their performance levels in terms of retrieval effectiveness.

The project is mainly coded in Python and Java. Excel has been used for analysis and data visualization. Libraries that are used include BeautifulSoup and Lucene. PyCharms and IntelliJ by JetBrains are the IDEs that were used to aid in coding. Github has been used for version control.

This work on this project has been planned and executed by Athul Muralidharan, Meghna Venkatesha and Sachin Haldavanekar. All of them actively studied various resources to implement IR systems.

Sachin was responsible making the Lucene IR system, evaluation of search engines and the documentation. Meghna was involved in for cleaning the given files, making the tf-idf Retrieval System, query expansion, removing stop words from the corpus and snippet generation. Athul coded the indexer, BM25 algorithm and smoothed query likelihood IR systems.

# 2. Literature and Resources

The techniques used in this project are:

1. Indexing using unigrams
2. Retrieval Modelling with BM25, tf-idf, Smoothed Query Likelihood and Lucene
3. Stopping using a stop list
4. Stemming using a stem list
5. Query Expansion using pseudo-relevance feedback
6. Snippet Generation using static summary

The tools that were used in this project are:

1. PyCharms by JetBrains
2. Sublime Text Editor by Sublime HQ Pty Ltd.
3. Eclipse by Eclipse Foundation, Inc.
4. Github Desktop by GitHub, Inc.
5. Lucene Core by The Apache Software Foundation
6. Excel by Microsoft, Inc.

The scholarly work and research articles (refer Bibliography for URLs) that were referenced include:

1. Selecting Effective Expansion Terms for Better IR
2. Query Expansion by Pseudo Relevance Feedback
3. Query Expansion Using Term Distribution and Term Association
4. Query Expansion using Local and Global Document Analysis
5. A Probabilistic Analysis of the Rocchio Algorithm with TF-IDF for text Categorization
6. Generation of Document Snippets Based on Queries and Search Results
7. Evaluation of Ranked Retrieval Results

## 3. Implementation and Discussion

The given 64 queries were extracted using an XML extractor and put in a new file. Each line had one query, so the new file was of 64 lines. The cleaning that was done for the corpus was done for the queries too.

Cleaning was the first task at hand. Starting with the corpus, the HTML tags were removed. Then, punctuation was removed using regular expressions. Occurrences of "-" were not discarded. For numbers, ",", "." and ":" were also retained. Numbers at the end of the documents seemed non-relevant and thus they were removed.

A single file was provided for the stemmed corpus. The single file was converted into multiple files, one per document. Thus, we had a folder container 3,204 stemmed documents. The stemmed queries were not needed to be modified.

Stop words were removed from the corpus. For each document in the clean corpus, a new document was created that only contained words that were not in the stop words list provided.

Next, the tf-idf scores were calculated for each document in the corpus. The formula used was where

$$tfidf(d) = \log\sum(N/ni) \times tf/dl$$

$N$ is the number of documents in the corpus, $ni$ is the number of documents in which the term occurs in, and $dl$ is the document length of the given document.

After this, the smoothened query likelihood model was implemented using the following query:

$$\text{Log P(Q|D)} = \text{i: \{1: n\}} \sum \text{Log } ((1-\lambda)\ (\text{fqi,D}/|\text{D}|) + \lambda(\text{cqi}/|\text{C}|)$$

Next, Lucene was used to make an index, parse each query, and display the results. The raw queries and corpus were provided, as Lucene has an inbuilt method to handle raw documents. The code that was used in homework 4 was used here, with small modifications.

Later, a search engine with BM25 as a retrieval model was made. With relevance information provided in cacm.rel, values of $ri$ and $R$ were obtained. The file did not have the relevance information for all the queries, so for queries with no relevance information, $ri$ = $R$ = 0 was assumed.

The output of BM25 is fed to the Query expansion module which uses KL Divergence to compute additional terms to add to the initial query.

There were 11 runs which were as follows:

a. 4 baseline runs (Lucene, BM25, Tf-idf and Smoothened Query Likelihood Model)
b. 3 stopped runs (Lucene, BM25 and Smoothened Query Likelihood Model)
c. 3 stemmed runs(Lucene, BM25 and Smoothened Query Likelihood Model)
d. 1 Pseudo Relevance Feedback with BM25

BM25 was again applied on the corpus with no stop words and again to the stemmed corpus and queries to check how well it performs with this variation.

Finally, MAP, MRR, precision, and recall were used to evaluate the performance of the search engines.

In Pseudo relevance terms for which this score is largest are selected for expansion.

We considered the Query term occurrence in each sentence of relevant documents. Relevant set compromises of the top 15 documents for each query as obtained by BM25 baseline run. Each sentence is assigned a score based a score based on the number of query term occurring in it. The two sentences with the maximum score are selected for representing the snippet for that document relevant to a query. Here we assume that while there may be some better words that can help differentiate the two sentences. But we consider that the user will be more interested for document whose snippet reflect more query terms. Thus we have selected the query term occurrence to differentiate between sentences. This choice was inspired by the paper [SnipIt - A real time Dynamic Programing approach to Snippet Generation for HTML Search Engines by Ian Carr and Lucy Vasserman]. We represent a sentence only up to 100 characters for long sentences. These sentences have "…" at the end to denote that it has more content to it.

We read each document in the relevant set. store all the sentences in lower case in a dictionary. We eliminate the stop words from query to avoid its impact on the sentence score. We select the top ranked sentences.

Query highlighting is represented as <query - term> by traversing to the query term occurrence in the selected sentences and replacing it as the representation mentioned above.

Query-by-query analysis

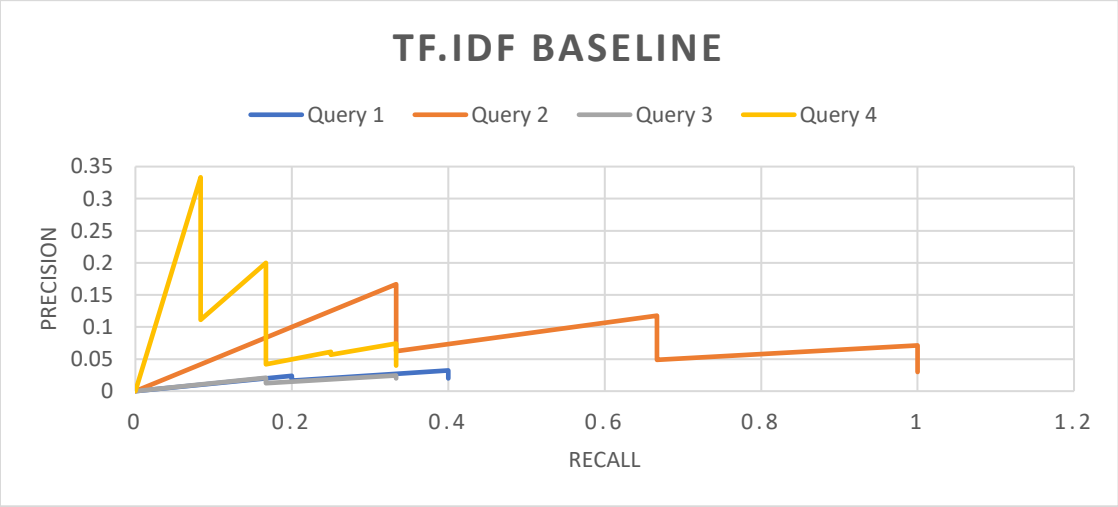
The query “portable operating systems” becomes “potable oper system” when stemmed. Because “system” occurs in a lot of documents, a lot more documents match the stemmed query than the un-stemmed query. So, many non-relevant documents are shown in the result.

The query “performance evaluation and modelling of computer systems” stems down to “perform evaluation and model of computing  system”. The precision increases by 0.05 and the recall increases by 0.1. It may be because ‘perform’ and ‘performance’ basically mean the same thing, and all its verb forms would be stemmed down to ‘perform’ as well. Converting ‘systems’ to ‘system’ also helps as it broadens the search scope a bit.

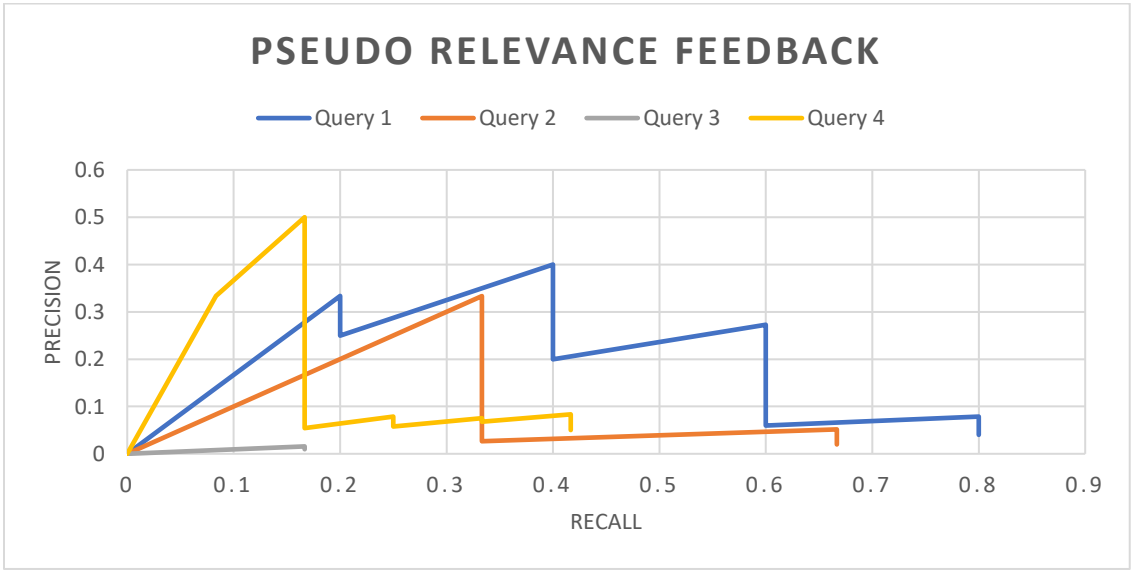
## 4. Results

Here are a few results from the eight runs which provide the numbers for precision and recall.

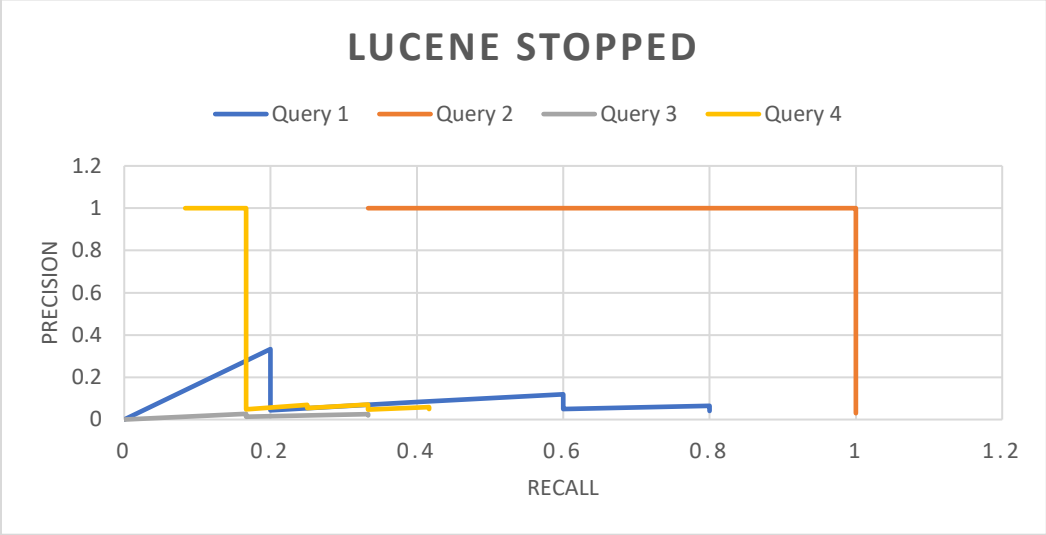
| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 4 | Q0 | CACM-2844 | 1 | 0.290852 | NR | TF_IDF_Baseline | 0 | 0 |
| 4 | Q0 | CACM-3059 | 2 | 0.274468 | NR | TF_IDF_Baseline | 0 | 0 |
| 4 | Q0 | CACM-3043 | 3 | 0.267474 | R | TF_IDF_Baseline | 0.333333 | 0.083333 |
| 4 | Q0 | CACM-1943 | 4 | 0.257227 | NR | TF_IDF_Baseline | 0.25 | 0.083333 |
| 4 | Q0 | CACM-2377 | 5 | 0.226901 | NR | TF_IDF_Baseline | 0.2 | 0.083333 |



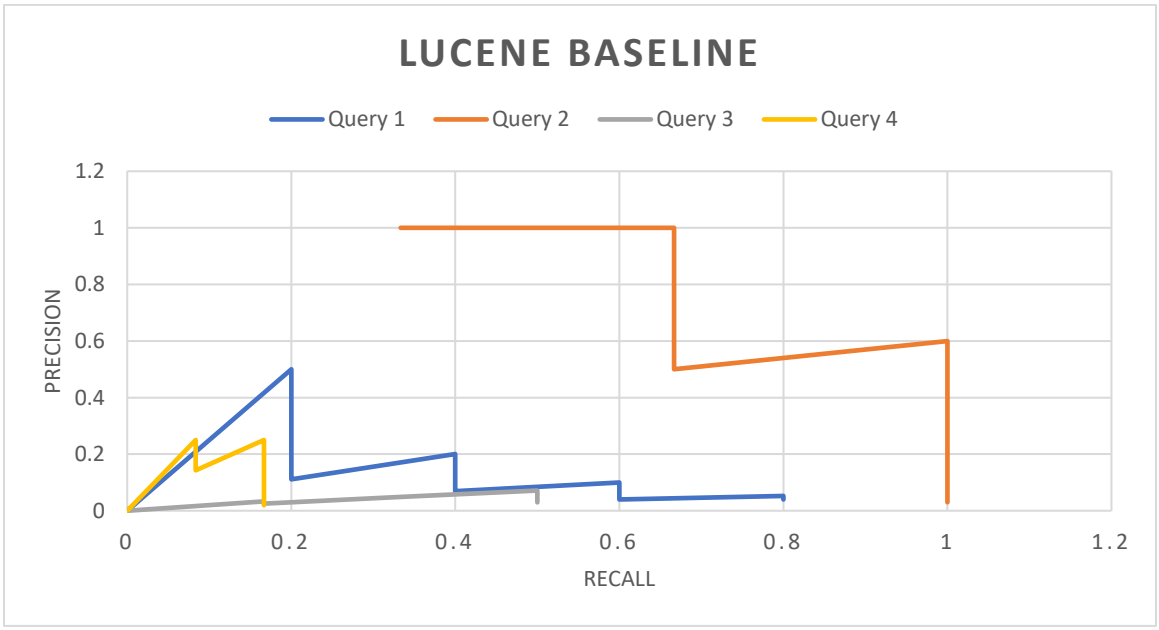

| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Na | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-2319 | 1 | 22.74289 | NR | PRF | 0 | 0 |
| 1 | Q0 | CACM-2379 | 2 | 20.20955 | NR | PRF | 0 | 0 |
| 1 | Q0 | CACM-1605 | 3 | 19.05892 | R | PRF | 0.333333 | 0.2 |
| 1 | Q0 | CACM-1698 | 4 | 18.85394 | NR | PRF | 0.25 | 0.2 |
| 1 | Q0 | CACM-1410 | 5 | 17.02935 | R | PRF | 0.4 | 0.4 |



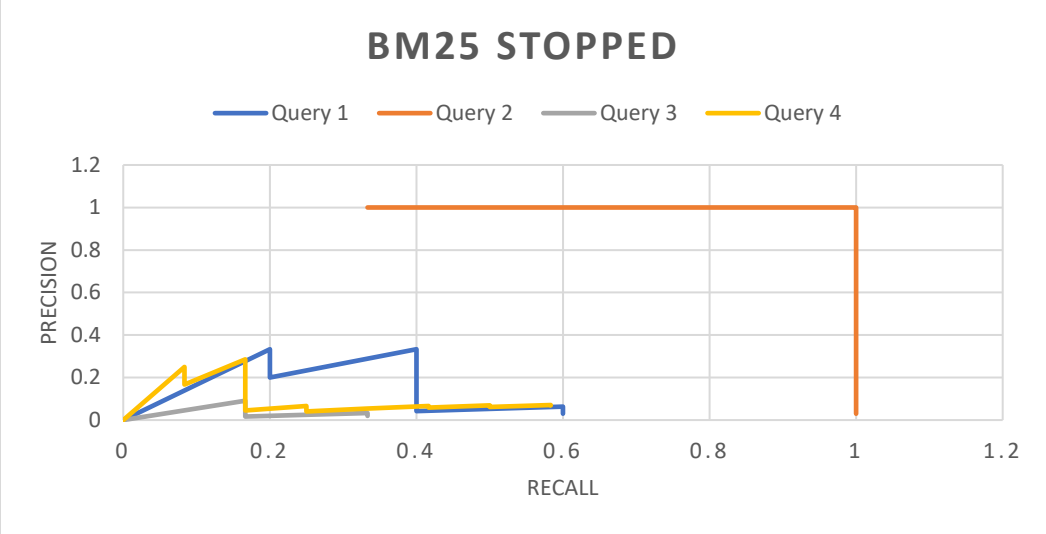
| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-2629 | 1 | 0.31605625 | NR | Lucene_Stopped | 0 | 0 |
| 1 | Q0 | CACM-1657 | 2 | 0.29048225 | NR | Lucene_Stopped | 0 | 0 |
| 1 | Q0 | CACM-1410 | 3 | 0.28220344 | R | Lucene_Stopped | 0.333333333 | 0.2 |
| 1 | Q0 | CACM-2319 | 4 | 0.27287894 | NR | Lucene_Stopped | 0.25 | 0.2 |
| 1 | Q0 | CACM-2151 | 5 | 0.25199133 | NR | Lucene_Stopped | 0.2 | 0.2 |



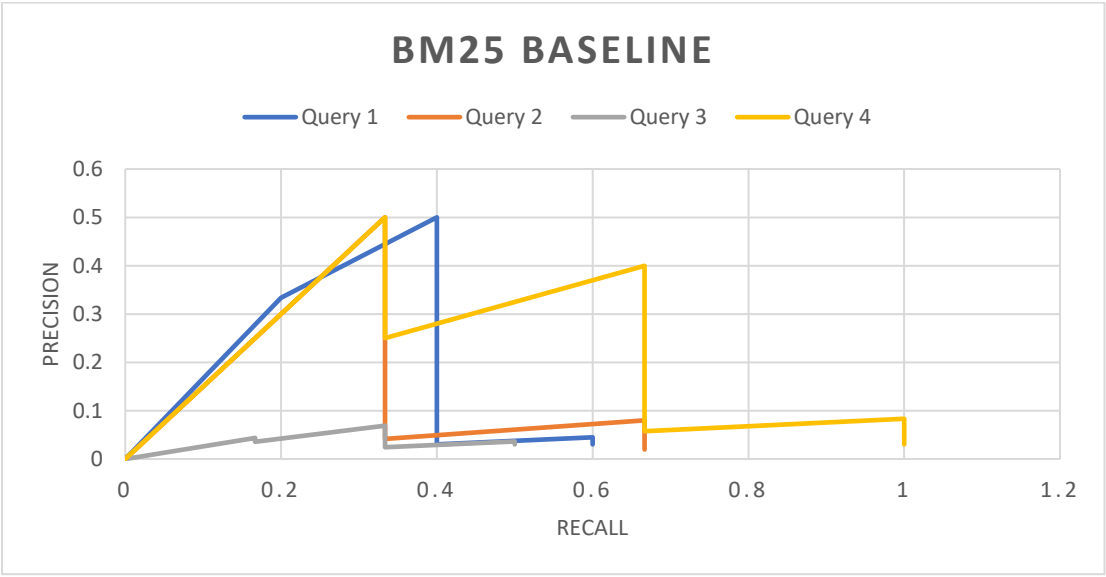

| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-1657 | 1 | 0.33750585 | NR | Lucene | 0 | 0 |
| 1 | Q0 | CACM-1410 | 2 | 0.32017016 | R | Lucene | 0.5 | 0.2 |
| 1 | Q0 | CACM-2319 | 3 | 0.3107554 | NR | Lucene | 0.333333333 | 0.2 |
| 1 | Q0 | CACM-1938 | 4 | 0.2908424 | NR | Lucene | 0.25 | 0.2 |
| 1 | Q0 | CACM-1827 | 5 | 0.2730184 | NR | Lucene | 0.2 | 0.2 |



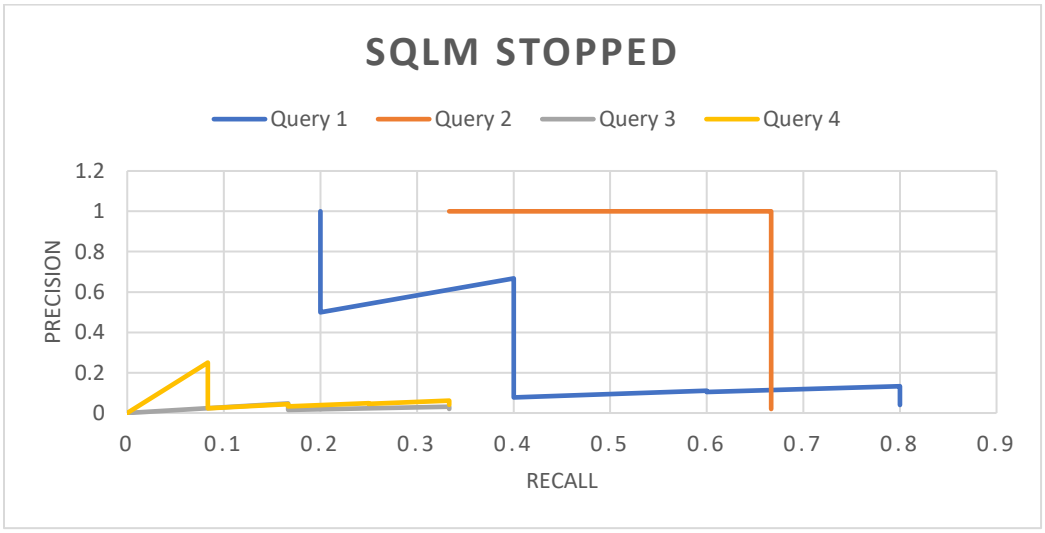
| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-1519 | 1 | 22.19541994 | NR | BM25_Stopped | 0 | 0 |
| 1 | Q0 | CACM-1161 | 2 | 20.40037586 | NR | BM25_Stopped | 0 | 0 |
| 1 | Q0 | CACM-1410 | 3 | 20.37326821 | R | BM25_Stopped | 0.333333333 | 0.2 |
| 1 | Q0 | CACM-1680 | 4 | 20.03348583 | NR | BM25_Stopped | 0.25 | 0.2 |
| 1 | Q0 | CACM-3048 | 5 | 19.73756033 | NR | BM25_Stopped | 0.2 | 0.2 |



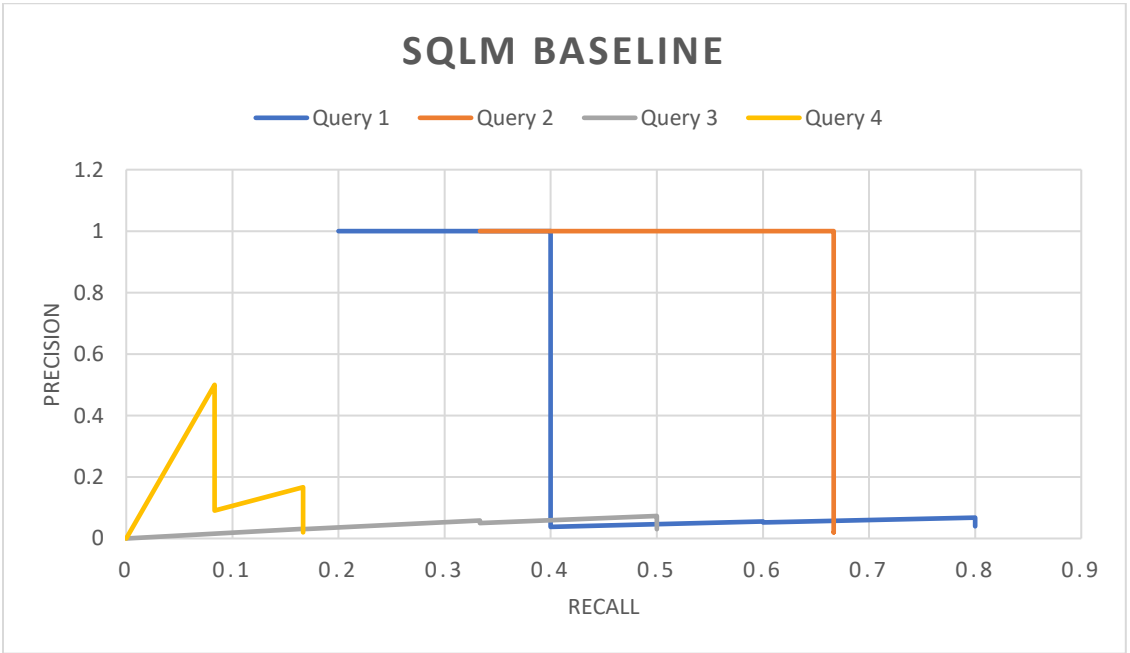

| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-1519 | 1 | 24.53394 | NR | BM25 | 0 | 0 |
| 1 | Q0 | CACM-1161 | 2 | 23.0243 | NR | BM25 | 0 | 0 |
| 1 | Q0 | CACM-1605 | 3 | 22.92594 | R | BM25 | 0.333333 | 0.2 |
| 1 | Q0 | CACM-1410 | 4 | 22.21637 | R | BM25 | 0.5 | 0.4 |
| 1 | Q0 | CACM-1033 | 5 | 22.11632 | NR | BM25 | 0.4 | 0.4 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-1410 | 1 | -77.70070764 | R | SQLM_Stopped | 1 | 0.2 |
| 1 | Q0 | CACM-1519 | 2 | -80.36741302 | NR | SQLM_Stopped | 0.5 | 0.2 |
| 1 | Q0 | CACM-1605 | 3 | -81.49590615 | R | SQLM_Stopped | 0.666666667 | 0.4 |
| 1 | Q0 | CACM-1506 | 4 | -81.96891909 | NR | SQLM_Stopped | 0.5 | 0.4 |
| 1 | Q0 | CACM-2379 | 5 | -82.61943728 | NR | SQLM_Stopped | 0.4 | 0.4 |




| Query id | Delimiter | Doc Id | Rank | Score | Relevance | System_ Name | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| 1 | Q0 | CACM-1410 | 1 | -113.4270021 | R | SQLM | 1 | 0.2 |
| 1 | Q0 | CACM-1605 | 2 | -115.4305125 | R | SQLM | 1 | 0.4 |
| 1 | Q0 | CACM-2379 | 3 | -116.6164277 | NR | SQLM | 0.666666667 | 0.4 |
| 1 | Q0 | CACM-1033 | 4 | -116.7523326 | NR | SQLM | 0.5 | 0.4 |
| 1 | Q0 | CACM-1506 | 5 | -116.9528264 | NR | SQLM | 0.4 | 0.4 |



# 5. Conclusion and Outlook

| RUNS | MAP | MRR |
|---|---|---|
| Lucene | 0.333476798 | 0.553305018 |
| Lucene Stopped | 0.354978632 | 0.572724171 |
| BM25 | 0.227497445 | 0.326248188 |
| BM25 - Stopped | 0.359107166 | 0.522492707 |
| SQLM | 0.267784488 | 0.505911781 |
| SQLM Stopped | 0.270877602 | 0.466962989 |
| Tf-Idf | 0.220822693 | 0.393891719 |
| PRF | 0.293041034 | 0.503461356 |

From the results, BM25 triumphed as the best retrieval model for all the models we tried implementing. The precision and recall values were better than other models, and it was quick to give results too. Having a file which had data about relevant documents for different queries helped BM25 to perform better than others, as information about $R$ and $r$ can be deduced from it.

Even after experimenting with various k value the statistics didn't improve and thus expansion have not much of a positive effect on the result

The output form BM25 stemming was quiet good compared to the baseline run but it contains only 7 queries and its performance may degrade when dealing with the complete query set.

SQLM and tf-idf yields quiet poor result and from all these observation we conclude that bm25 baseline is better for the given corpus.

There is a lot of scope for improvement here. Automation of index updating, pre-processing frequent queries and keeping a log of user patterns are a few things which can be done in the future. Stemming can be replaced with Lemmatization. For bigger corpuses, parallel computing can be explored for maximizing resource utilization

# 6. Bibliography

Iman, Hazra, and Aditi Sharan. "Selecting Effective Expansion Terms for Better Information Retrieval." International Journal of Computer Science and Applications 7.2 (2010): 52-64. Web. 4 Dec. 2016. <http://www.tmrfindia.org/ijcsa/v7i24.pdf>.

Pal, Dipasree, Mandar Mitra, and Kalyankumar Datta. "Query Expansion Using Term Distribution and Term Association." N.p., 4 Mar. 2013. Web. 5 Dec. 2016. <https://arxiv.org/abs/1303.0667>.

Xu, Jinxi, and W. Bruce Croft. "4. Query Expansion Using Local and Global Document Analysis." N.p., 02 Oct. 2010. Web. 05 Dec. 2016. <http://www.eng.utah.edu/~cs7961/papers/XuCroft-SIGIR96.pdf>.

Joachims, Thorsten. "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization." N.p., 1997. Web. 4 Dec. 2016. <https://www.cs.cornell.edu/people/tj/publications/joachims_97a.pdf>.

Verstak, Alexandre A., and Anurag Acharya. Generation of Document Snippets Based on Queries and Search Results. Patent US 8145617 B1. 27 Mar. 2012. Print.

Evaluation of Ranked Retrieval Results <https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html>