Search Engine Design

CS6200: Information Retrieval

Northeastern University

Fall 2017, Prof. Nada Naji

Team Members:
Ayush Shukla
Parmeet Singh Saluja
Varun Sundar Rabindranath

Contents

1. Introduction

1.1 Overview

The goal of the project is to design and build our own information retrieval systems, evaluate and compare their performance levels in terms of retrieval effectiveness. This project designs search engines using four distinct retrieval models as mentioned below:

- BM25
- tf-idf
- Smoothed Query Likelihood Model
- Lucene

Along with these four baseline runs, we have three additional runs:

- BM25 retrieval model along with Pseudo relevance feedback query expansion technique
- BM25 with/without stopping, and with/without stemming
- tf-idf with/without stopping, and with/without stemming
- Smoothed Query Likelihood Model with/without stopping, and with/without stemming

We have also implemented the Snippet Generation and Query Term Highlighting on the results of BM25.We have also evaluated our work by using these retrieval effectiveness measures:

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

1.2 Contribution of Team Members

The detailed contribution of each individual member is elucidated below:

- **Ayush Shukla:** Ayush was responsible for implementing baseline runs of model tf-idf. He was responsible for implementing the various evaluation measures like MAP, MRR, P@K, Precision & Recall. He also analyzed the evaluation results and generated MAP, MRR charts for comparison of the various models, and did query level analysis by generating Precision, Recall graphs for certain queries.
- **Parmeet Singh Saluja**: Parmeet was responsible for implementing baseline runs of model Smooth Query Likelihood Model. He was also responsible for implementing the query expansion technique using Pseudo Relevance Feedback and Rocchios' Algorithm. He was responsible for the design choices, experimenting and parameter settings and documenting the outlooks.
- **Varun Sundar Rabindranth**: Varun was responsible for implementing baseline runs of models BM25 and Lucene. He also designed the search engines which perform stopping and stemming tasks. He was also responsible for developing Snippet Generation, Query Term Highlighting and documenting his design choices.

2. Literature and Resources

- **Pseudo Relevance Feedback:** The techniques used for pseudo relevance feedback is based on the textbook.
  - Croft, W.Bruce; Metzler, Donald; Strohman, Trevor Search Engines: Information Retrieval in Practice. Pearson Education 2015
  - Manning, Christopher D; Raghavan, Prabhakar; Schutze Hinrich An Introduction to Information Retrieval. Cambridge England: Cambridge University Press 2009
  - http://nlp.stanford.edu/IR-book/pdf/09expand.pdf

  The above sources were used to decide some of the parameters involved in Pseudo Relevance Feedback.
  - **Base Model:** We used BM25 as baseline run as the formula has terms which account for relevant documents and term frequency in relevant documents.
  - **N:** Most of the search engines and experiments suggested using 10 as a value for top N documents.
  - **E:** The range of E lies between 10 to 20. We took the median value 15 to expand the query words. E denotes the number of words added to query.
  - **alpha, beta and gamma:** Set of these values also differ largely but we used 1, 0.75 and 0.15 which is one of the most common settings.

- **Snippet Generation:** The techniques used for snippet generation is based out of 2 research papers.
  - Hans Peter Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of research and development 2, 2 (1958), 159–165.
  - Qing Li, K. Selcuk Candan, Yan Qi. January 2007. Extracting Relevant Snippets from Web Documents through Language Model based Text Segmentation

  Luhns paper [1] was used as the base algorithm for snippet generation. Some ideas from [2] was used to expand the query to find more words that can be considered significant word based on a relevance language model. [2] finds the region in the document that is most relevant to the topic and extracts sentences from that region to create snippet. However, considering the fact that most CACM documents are small, we assume that the entire document is relevant to the topic and pick significant words from the entire document. The algorithm implemented can be viewed as Luhn's approach with query refinement using relevance language model.

3. Implementation and Discussion

3.1 Implementation

The intricacies of the implementation techniques are detailed in this section. The implementation revolves around 3 major files,

- corpus.py
- indexer.py
- searcher.py

- **Processing the Corpora**

  The project involves performing retrieval on documents with different text processing techniques. For example, we perform retrieval on,
  - Documents that are cleaned using base text processing techniques (punctuation and case folding)
  - Documents that are cleaned using base text processing techniques (punctuation and case folding) and stopping
  - Documents that are cleaned using base text processing techniques (punctuation and case folding) and stemming

  Due to the variety of text processed documents that are to be subjected to retrieval, we made a design decision that all document processing will be done during the document clean-up (base text processing) phase, as required, and will be independent from the indexer.

- **Processing the Query files**

  The project involves searching various indexes with a variety of differently processed and represented queries. For example, we process queries of following variants,
  - Queries that are not stopped not stemmed (cacm.query.txt)
  - Queries that are stopped only (stopped cacm.query.txt)
  - Queries that are stemmed only (cacm_stem.query.txt)

  Due to the variety of query formats, we made a design decision to represent all variants of the queries in a common space separated “query-id query” format. This way, the searcher.py need only take in as input a query file of a single format.

- **Retrieval Models**

  All retrieval models are implemented as classes. The TFIDF, BM25, Query Likelihood Model (QLM) are all implemented as classes, TFIDF, BM25 and QLM in files tfidf.py, bm25.py and qlm.py respectively. Apart from these baseline models, the Pseudo Relevance feedback algorithm is implemented as a retrieval model in bm25_relevance.py. All these classes provide a function with the same function signature. This enables the searcher.py to just create objects of the user specified retrieval model and invoke the search function of that retrieval model.

- **Pseudo Relevance Feedback**
  For performing pseudo relevance feedback using Rocchio's Algorithm (in bm25_relvence.py), perform the following steps:
  - Normal retrieval is performed with the initial query using BM25 as baseline Model.
  - The top k documents (in our case k =10) from the results are included in the relevant set of documents and the rest of the documents are non-relevant set.
  - Rocchio's algorithm is used to generate the new query.
    - Initial query vector with the tf scores and aligned with the inverted index terms is generated.
    - Relevant set of document vector is generated.
    - Non-relevant set of documents vector is generated.
    - Then modified query is generated by the Rocchio's formula.
  - The top 15 terms with the highest weight and which are not present in the query are appended to the original query.
  - Normal retrieval with BM25 and relevance information is performed with the new query and the search results are generated.

- **Snippet Generation**

  The snippet generation program (gen_snippet.py), is designed to take in as input a file containing TREC Eval result strings. This enables the flexibility to use the snippet generation algorithm with any retrieval model.

  As mentioned earlier, the snippet generation algorithm uses [1] and [2] as its base technique. The various parameter choices that we had to make and the reasons for the choices are as follows,

  - **Number of words in the snippet**: The maximum number of words that a snippet can have was evaluated to 50. We settled on this value by experimenting with a popular web search engine. We provided various queries to the search engine, Google, and found that the maximum number of characters per snippet never exceeded 55.
  - **Number to top N relevant documents to consider for relevance language model:** For the relevance language model, we consider top 10 documents of all retrieved documents for a query to be most relevant. As we don't have any feasible way within the scope of the project to experiment and evaluate the different values of N, we assumed 10 is a good guess, based on our literature review for Pseudo Relevance Feedback
  - **Ignoring numbers as significant terms**: As, almost every document in the CACM corpus contains numbers towards the end of the document, the relevance language model would return these numbers as significant terms. Since for the purposes of snippet generation, these numbers are insignificant, we ignore these and don't consider them significant.

- **Evaluation**

  The evaluation.py file computes the various effectiveness measures like Recall and Precision for all queries at each document occurrence in the result list.

  It generates a set of files (corresponding to queries) having recall and precision at each document, a Global measures file which holds the MAP and MRR values for that model, and the p@k file which holds the precision at 5 and 20 for each query.

  - **Creating Relevance Dictionary:** A Dictionary with Query ID's as the key and list of relevant documents as the values is generated in the first step.
  - **Creating Top Documents Dictionary:** A Dictionary with Query ID's as the key and the list of top 100 documents (as per the results) as the values are generated in the second step
  - **Computing Precision and Recall**: After checking presence of relevance information of top 100 documents as per the result, recall and precision is calculated and written to corresponding query files. Precision @5 and @20 is also written to a file and reciprocal rank is computed too.
  - **Computing MAP and MRR:** Finally based on the total average precision and reciprocal rank for all the queries, the Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) is calculated and written to a file "Global Measures.txt"

3.2 Discussion

- **Query by Query analysis**

  We performed query-by-query analysis for the BM25 results with and without stemming. This is due to the fact that stemming improved BM25's results significantly than it did for any other models.

Query 1: "distribut comput structur and algorithm" vs **"distributed computing structures algorithms"**

Here we found that, 2 particular relevant documents (as per cacm.rel.txt), CACM-3137.html and CACM-3148.html, were not ranked in the top 100 for the stemmed version while the documents were ranked 1st and 3rd respectively in the non-stemmed version.

On analyzing the term frequencies of the stemmed and non-stemmed queries, in the stemmed and non-stemmed corpus files, we found that they were similar. Hence, we analyzed the corpus frequencies for the query terms and found that the corpus frequency of the stemmed query terms was significantly higher. As we know BM25 does not favor corpus-wide highly frequent terms, these stemmed query terms, were scored relatively lower than their non-stemmed counter parts.

Corpus frequency in stemmed corpus	Corpus frequency in non-stemmed corpus
“distribut” – 240 times	“distributed”- 52 times
“comput” - 1945 times	“computing”- 195 times
“algorithm” - 2015 times	“algorithms” - 375 times

Query 2: “perform evalu and model of comput system” vs **“performance evaluation modelling computer systems”**

Unlike the previous query, for this query stemming improves the ranking for relevant documents significantly. For example, relevant documents, CACM-2862.html, CACM-3059.html, CACM-1526.html, CACM-1533.html, CACM-1572.html were ranked 10th, 38th, 47th, 54th and 62nd respectively for the stemmed version. While they did not appear in the top 100 ranked documents for the non-stemmed version. This is because, unlike query 1, the term frequencies of the stemmed query in the stemmed corpus is relatively higher than the term frequencies of the non-stemmed queries in the non-stemmed corpus.

Term frequency - Stemmed CACM-2862.html	Term frequency -Non-stemmed CACM-2862.html
“model” - 8 times	“modelling” - 0 times
“comput” - 1 time	“computer” - 0 times
“evalu” - 1 time	“evaluation” - 0 times

All other corresponding query terms, between stemmed and non-stemmed versions appear equal number of times.

Query 3: “portabl oper system” vs **“portable operating systems”**

Here we found that one particular relevant document (as per cacm.rel.txt), CACM-2080.html was ranked significantly higher for non-stemmed version compared to its stemmed version (43 vs 90). On analysis, again we found, as in query 1, that query term frequencies in the documents, between stemmed and non-stemmed versions, were similar and that stemming has diminished query term importance for this query also.

Corpus frequency in stemmed corpus	Corpus frequency in non-stemmed corpus
“oper” - 572 times	“operating” - 205 times
“system” - 1946 times	“systems” - 701 times

4. Results

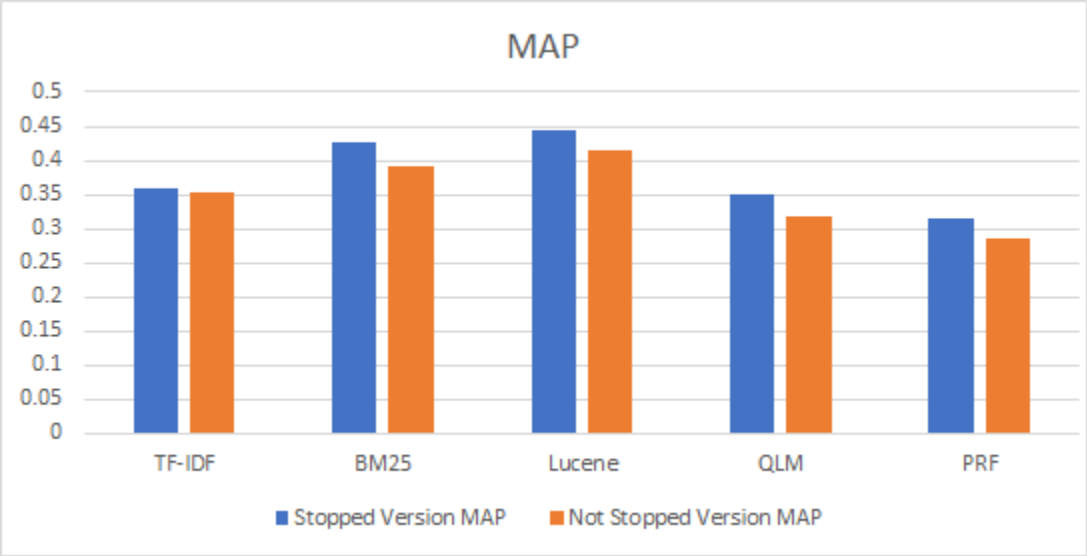
After the output files were generated from the various ranking models, the evaluation was done, using Precision and Recall, and results were generated for each query in a separate text file corresponding to that particular model. Two separate files having Global measures, and Precision @ 5 and @ 20, were also generated apart from files for each query.

(**Note**: All the result tables have been included in the submission inside a separate folder named "resulttables")

4.1 Global Measures Analysis and Comparison

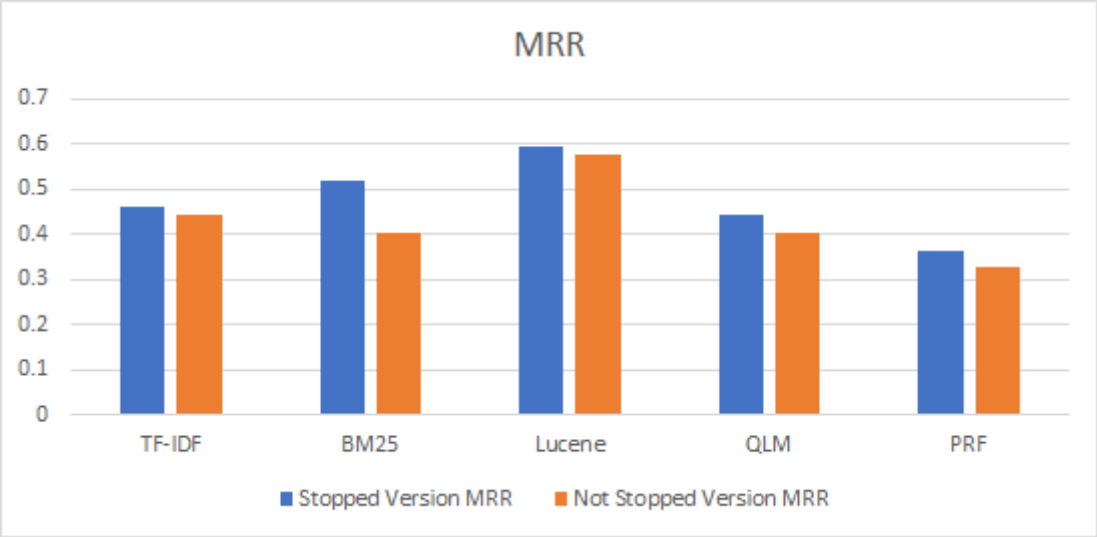
Once the evaluation results were generated, global measures (MAP and MRR) were used to compare the effectiveness of various models. This was done by plotting bar charts for Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), for both standard and stopped versions of the evaluation results of all the models.

- **Chart of Mean Average Precision for various Models-**



MAP Comparison

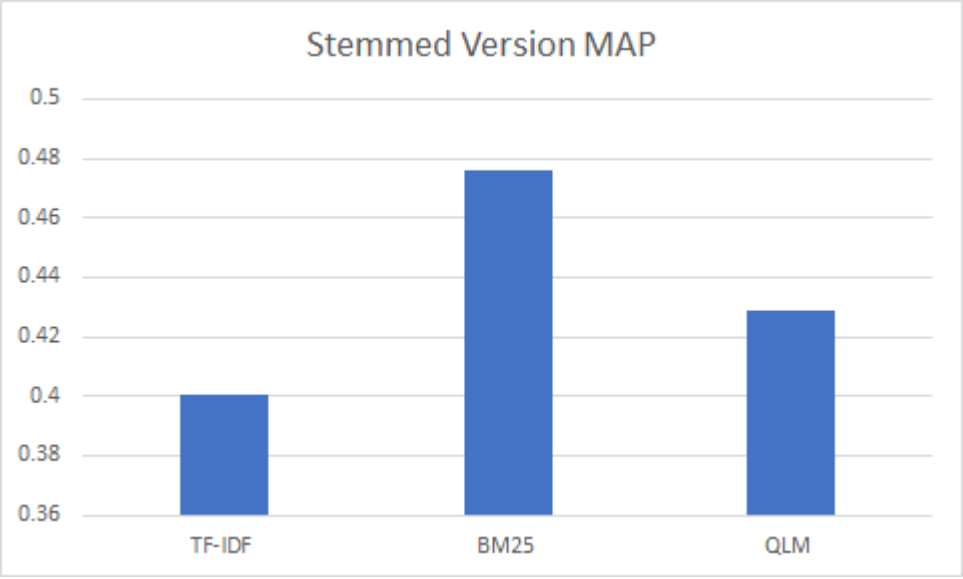
- **Chart for Mean Reciprocal Rank of the various models**



MRR Comparison

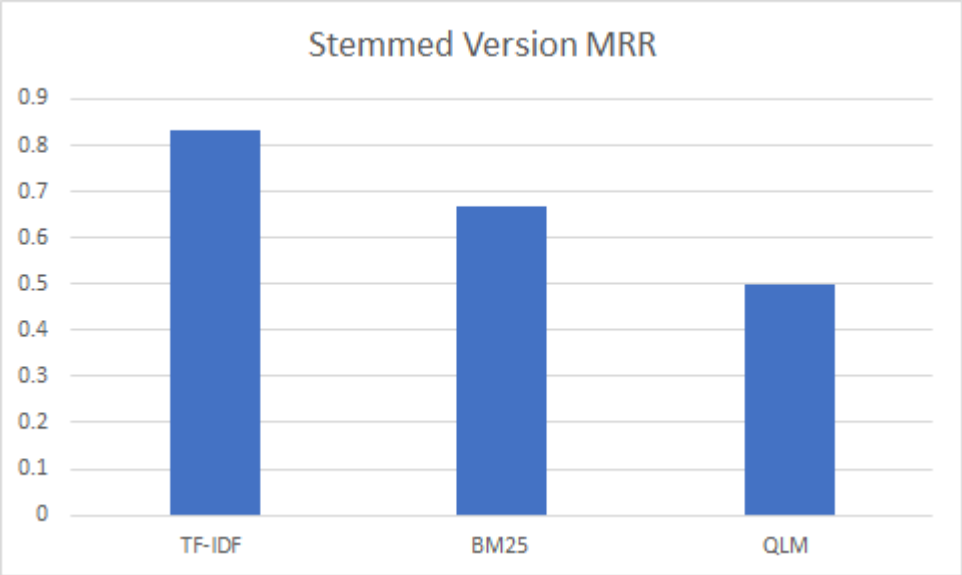
After performing Stemming in Task 3, 3 models (TF-IDF, BM25, QLM) were analyzed for their MAP and MRR values.

- **MAP chart for TF-IDF, BM25 and QLM:**



MAP Comparison (Stemmed Version)

- **MRR chart for TF-IDF, BM25 and QLM:**



MRR Comparison (Stemmed Version)

4.2 Query Level Analysis

For query level analysis, we picked 3 queries

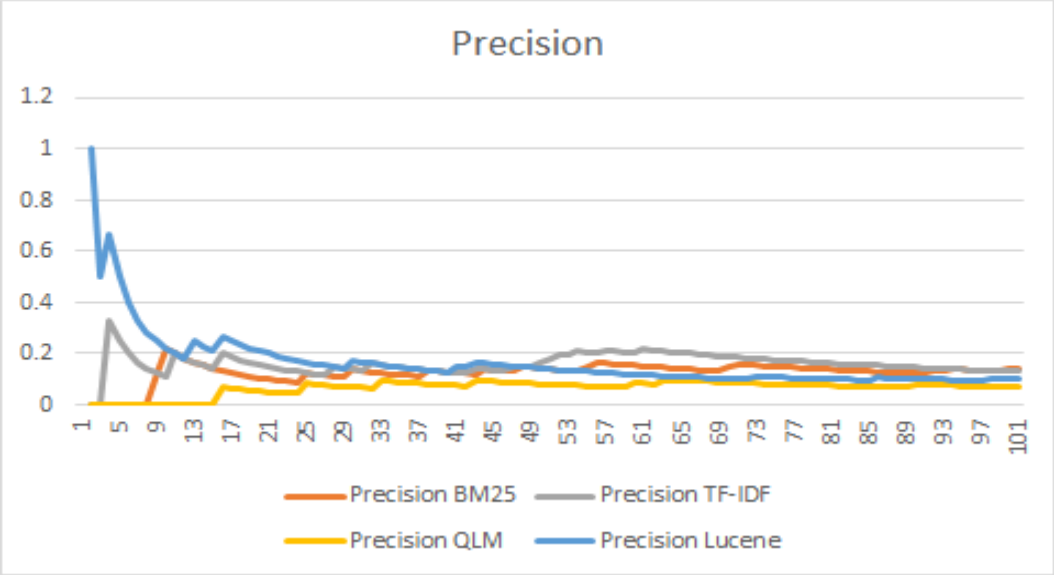
- **Query 14** – "find discussions optimal implementations sort algorithms database management applications"
- **Query 25** – "performance evaluation modelling computer systems"
- **Query 59** – "dictionary construction accessing methods fast retrieval words lexical items morphologically related information hashing indexing methods applied English spelling natural language problems"

The above queries were chosen for the fact that they had the most number of relevance judgements, which made these queries affect the results of most of our ranking models.

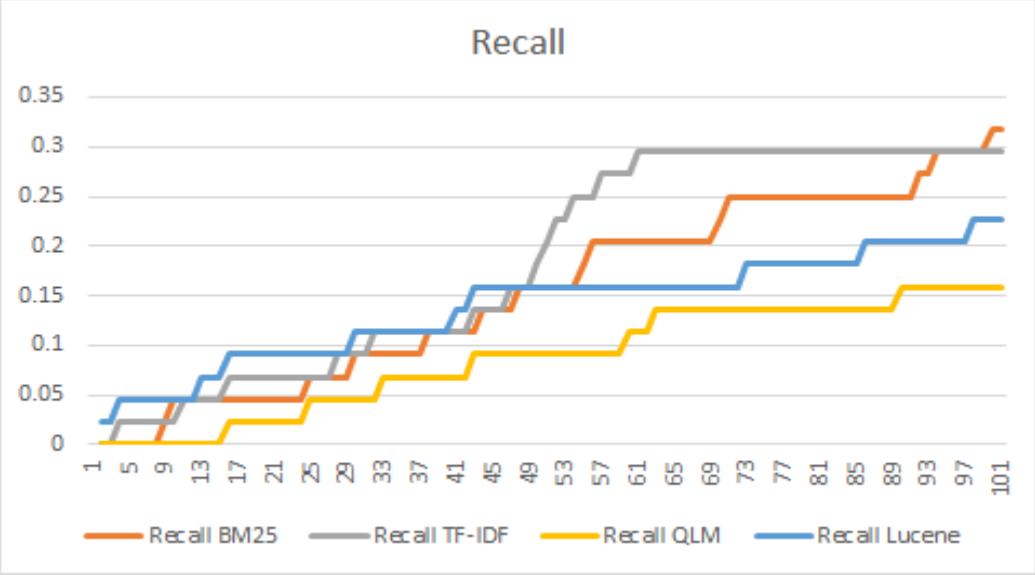
Using the results of precision and recall from these query files, we generated the graphs comparing precision and recall values across ranking models.

The graphs obtained are as follows:

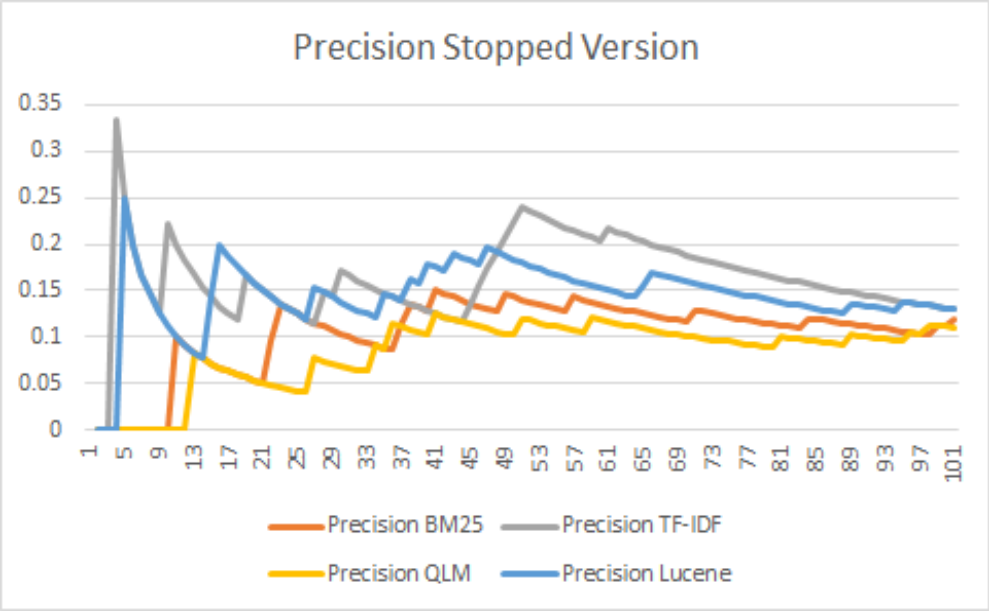
- **Graphs for Query 14:**



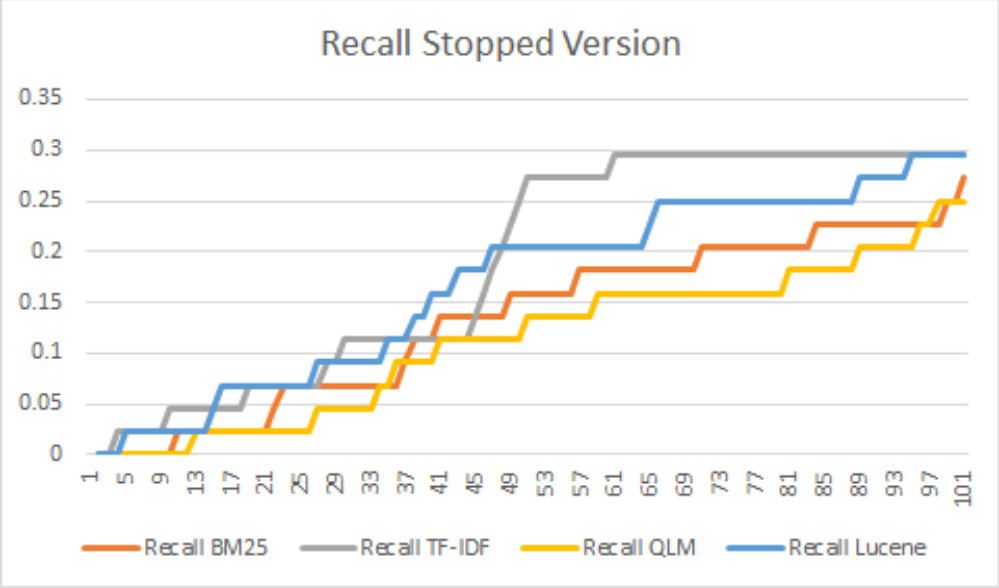
Precision Graph for Query 14



Recall Graph for Query 14

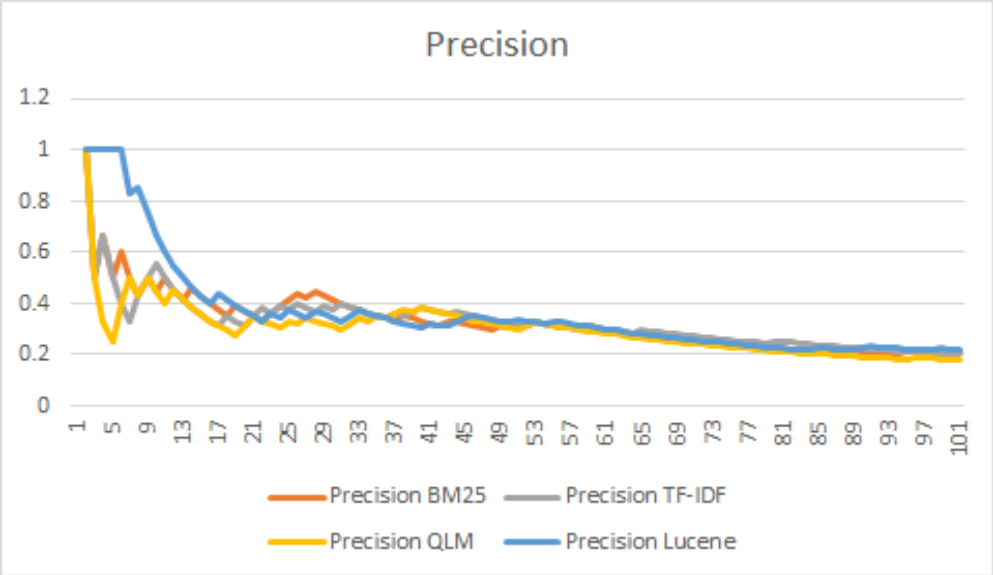


Precision Graph for Query 14 (Stopped Version)

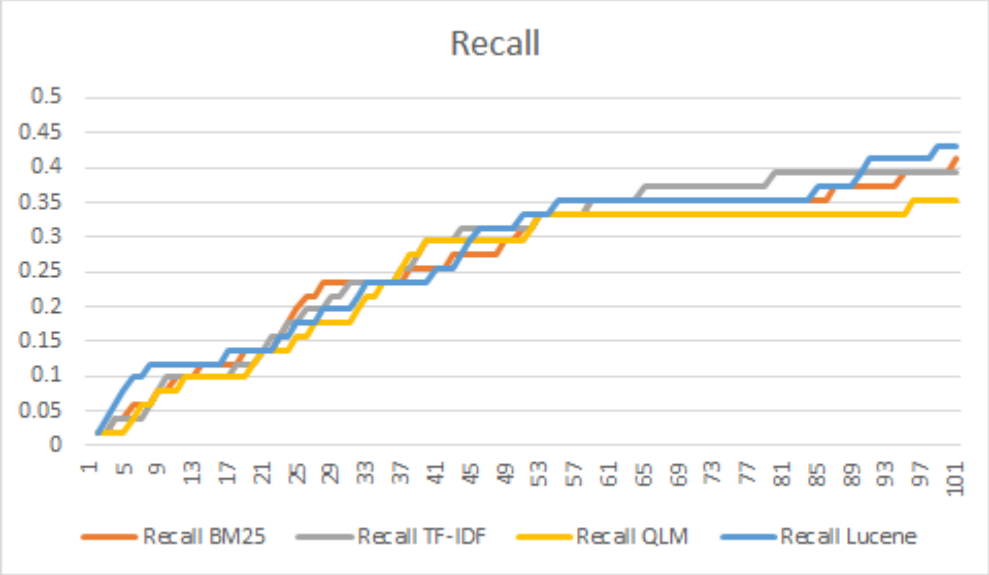


Recall Graph for Query 14 (Stopped Version)

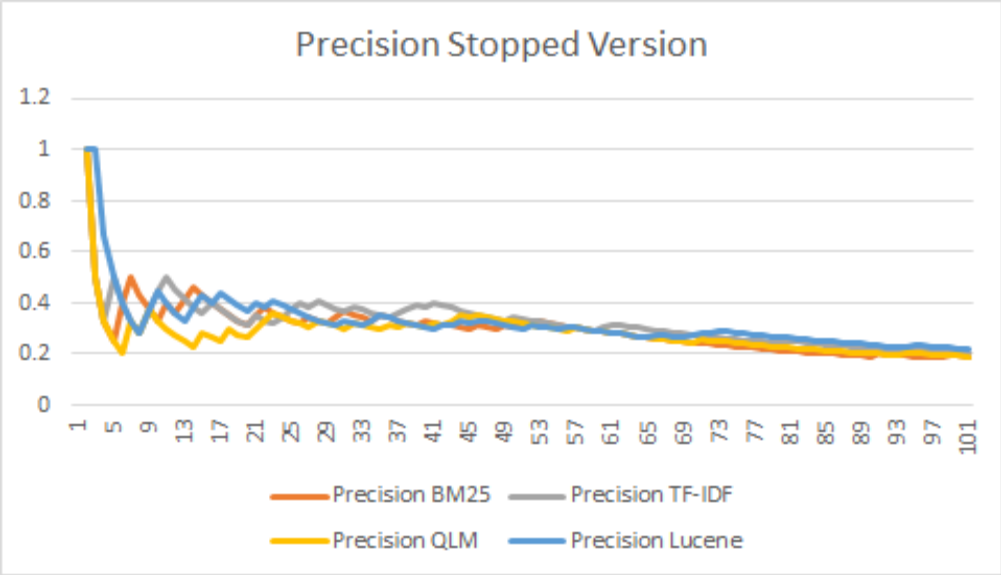
- **Graphs for Query 25:**



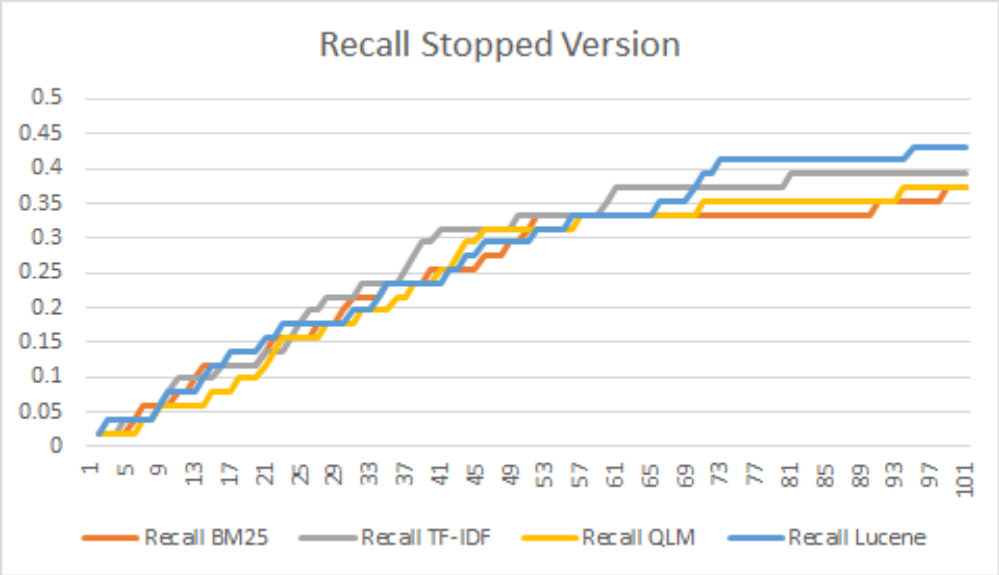
Precision for Query 25



Recall for Query 25

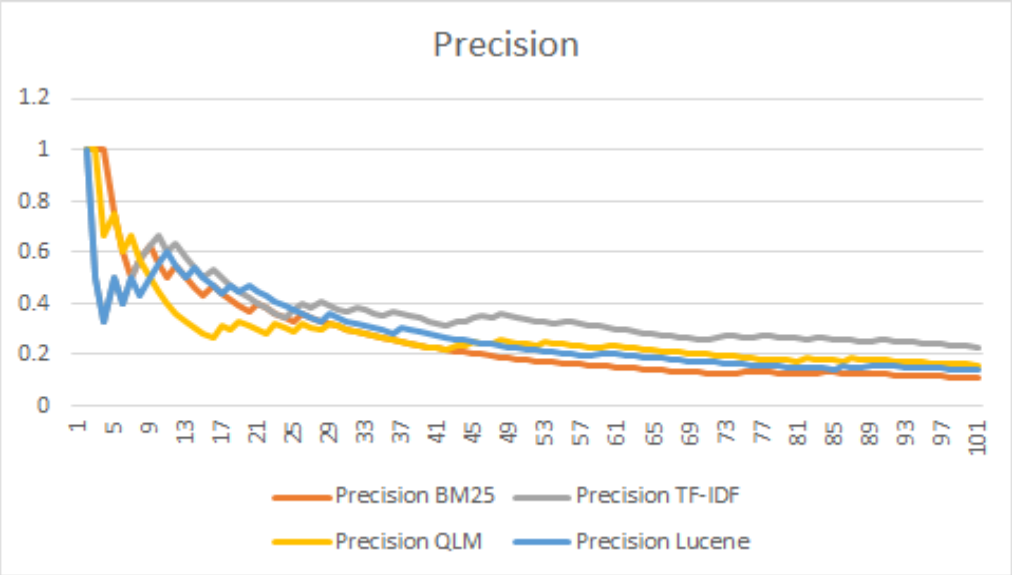


Precision for Query 25 (Stopped Version)

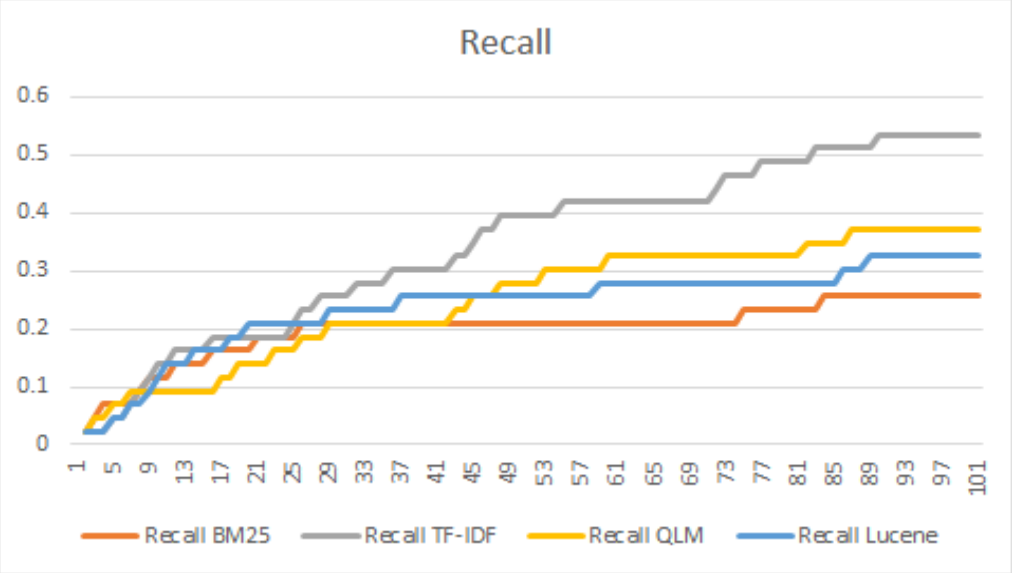


Recall for Query 25 (Stopped Version)

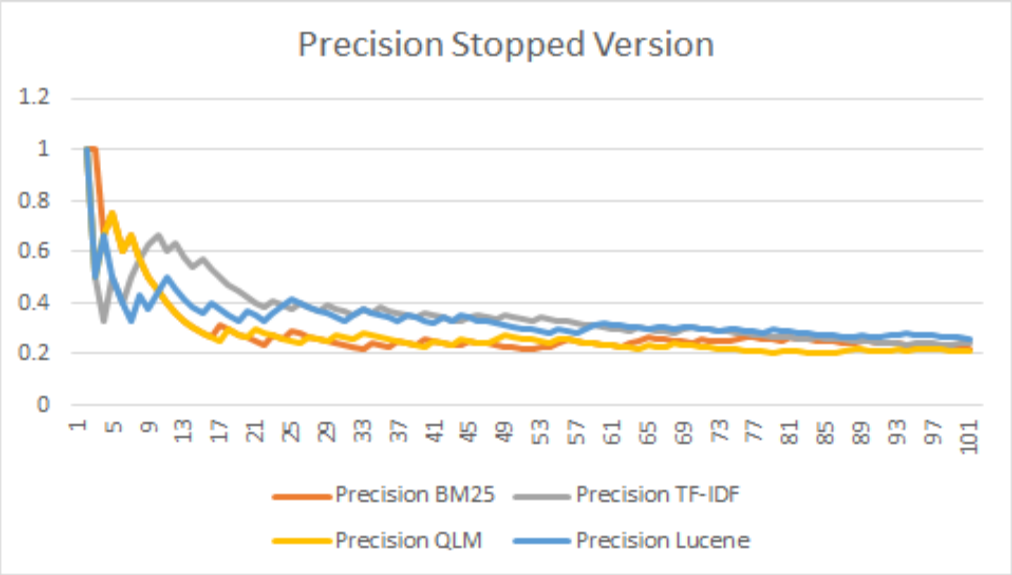
- **Graphs for Query 59:**



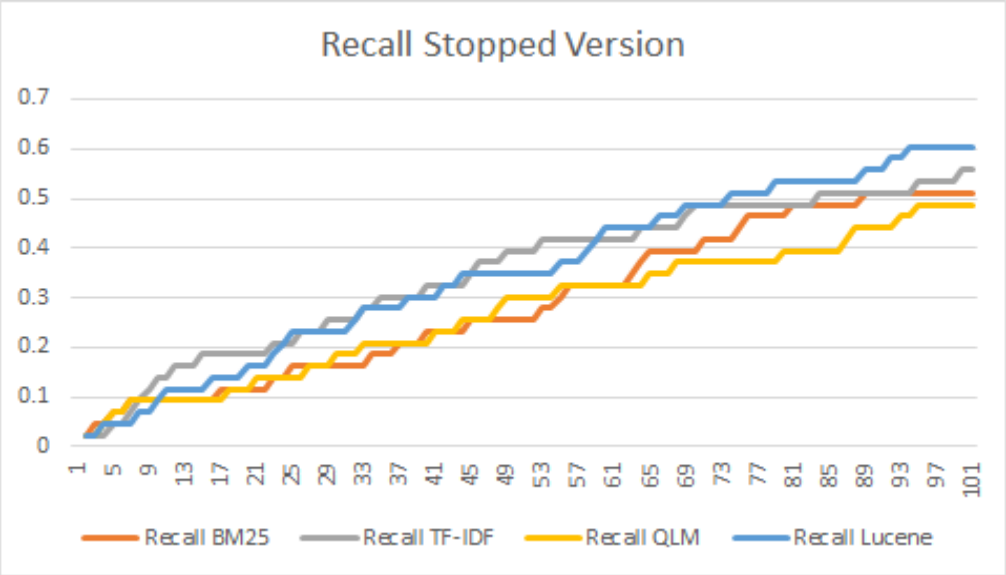
Precision Graph for Query 59



Recall Graph for Query 59



Precision Graph for Query 59 (Stopped Version)



Recall Graph for Query 59 (Stopped Version)

5. Conclusions and Outlook

5.1 Conclusion

Our overall findings and conclusions from the eight assigned runs and after applying various evaluation measures on their result set are outlined below:

- The best results were obtained from the BM25 (not considering Lucene) run which incorporated stopping. As per its MAP scores (0.426421) and MRR scores (0.519230), this run was undisputedly better than all other retrieval models
- Even the base run of BM25 yielded good results in terms of MAP scores (0.391936) and MRR scores (0.403846).
- As stopping increased the efficiency of BM25 model we can observe that list of stop words included words which were non-relevant to the queries and resulting in positive effect on the overall search effectiveness.
- PRF measures have produced the least values for all the evaluation techniques. This may happen due to some unrelated terms in new query.
- Lucene provided best result overall it may be because it is using Boolean model and vector space model with some query normalization.
- In all the models we can see that stopping has produced better results thus stop lists were having non-relevant high frequency words.

5.2 Outlook

Our project has all the basic features of an information retrieval system. However, certain features and functionalities as mentioned below can be included in future to improve the performance of the search engine designed in this project:

- **Topical features of a document:** We can take into consideration the quality features like 'incoming links', 'update count' etc. of pages as a part of the final ranking.
- **Spell Checking:** We can also incorporate this feature for query refinement.
- **Compression Techniques**: As the corpus grows, it will be prudent to use different compression techniques like 'Elias-γ encoding', 'v-byte encoding' to store the inverted index.
- **Distributed Processing**: We can also incorporate Distributed Processing feature for faster query processing.
- **Multi Lingual Searching Ability**: Currently our system works on pages related to English language we can also incorporate Multi lingual search ability in future.
- **Efficiency**: Primary purpose of project was learning and implementation. We did not focus on efficiency a lot. In future improving the retrieval efficiency can be one aspect.

6. Bibliography

6.1 Books

- Croft, W.Bruce; Metzler, Donald; Strohman, Trevor Search Engines: Information Retrieval in Practice. Pearson Education 2015
- Manning, Christopher D; Raghavan, Prabhakar; Schutze Hinrich An Introduction to Information Retrieval. Cambridge England: Cambridge University Press 2009

6.2 Scholarly Articles

- Hans Peter Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of research and development 2, 2 (1958), 159–165. http://courses.ischool.berkeley.edu/i256/f06/papers/luhn58.pdf

- Qing Li, K. Selcuk Candan, Yan Qi. January 2007. Extracting Relevant Snippets from Web Documents through Language Model based Text Segmentation. https://pdfs.semanticscholar.org/7771/0bd1e8aa754e897a53093abd98cc572fde11.pdf

- http://nlp.stanford.edu/IR-book/pdf/09expand.pdf