Evaluation Of Search Engine

Neha Bhoi
Department of Computer Science
CSULB, 026627560
Neha.Bhoi@student.csulb.edu

Juhi Pathak
Department of Computer Science
CSULB, 026336373
juhi.pathak@student.csulb.edu

Suchitra Chinnamail
Department of Computer Science
CSULB, 026627287
suchitrareddy.chinnamail@student.csulb.edu

***Abstract*—Evaluation is highly important for designing, developing and maintaining effective information retrieval systems as it allows the measurement of how successfully an information retrieval system meets its goal of helping users fulfil their information needs. This paper represent the evaluation of search engine based on some basic evaluation measures. We have developed the search engine which has both boolean as well as ranked retrieval capabilities with four ranking algorithms and vocabulary elimination for inexact ranked retrieval queries. We are evaluating this search engine for ranked retrieval queries using test data set which matches the standard criteria of Buckley and Voorhees's test data set and evaluation measure.**

***Index Terms*—Information retrieval, Rank retrieval, Precision, Recall, Throughput**

I. Introduction

Information retrieval is the process of finding information resources that are relevant from a collection of documents or sources which are usually unstructured. A search engine is a piece of software which comes into the picture when we have to access the information from a corpus, it helps in gathering and sorting the information according to the need of the user and helps in coming up with the most relevant information needed.

We are building a software which can retrieve list of documents which are relevant to the given query from a corpus with around 36,000 documents, for this to achieve and work efficiently we have implemented disk based indexing with optimisation techniques such as gap values for document ids and positions and also implemented various ranking techniques to get different rank based results and also compared the ranked results based on precision, throughput and recall. To increase throughput we also consider vocabulary elimination technique and compared the results with the original results.

This paper is organized as follows, the next section provides a very brief summary of what are the feature are provided by our search engine like rank retrieval, variant ranking formulas, basics of index design, vocabulary elimination and evaluation measures of search engine. Section 3 presents the test environment used in this study along with comparison of results. Section 4 suggests ways to extend the work. While the final section will conclude the paper.

II. Background

A. Features of Search Engine

This search engine is developed in java programming language. It supports boolean as well as ranked queries. Boolean queries consists of AND, OR, NOT queries, wildcard queries and phrase queries. While, ranked queries consists of spelling correction, variant ranking algorithm along with an option of vocabulary elimination for inexact ranked retrieval queries. The search engine reads the document corpus and build positional index for boolean queries, kgram index for wild card and spelling correction queries and store it on disk. For every query evaluation we are reading posting for each term in the query from the disk.

B. How the rank retrieval works?

Ranking is the most important and fundamental concept in information retrieval, which scores the documents according to some criterion so that the best relevant document appears in the result list displayed to the user. In this project, score is assigned to the document according to the relevance of the document to the query given by the user.

Different ranking techniques can be used to score the document which uses the calculation of different values like $W_{d,t}$, $W_{q,t}$ and L_d. These values for each document are the driving force which decides the score of that document. We make sure the score is calculated in a way which preserves the order of the documents returned by few smoothing techniques.

Let us assume q is the query given by the user, t is the term, d is the document and L_d is the length of the document.

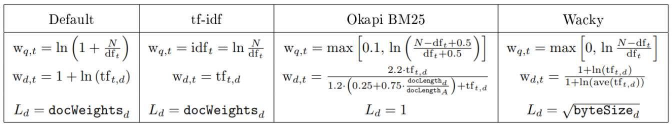
- **$W_{q,t}$:** This value tells us how important the term is to the query. All the words in the query are not given equal importance as they may contain some "stop words" which will have high frequency but have no much importance. For example of, an, and, are, few such stop words. So we can conclude that the words with high frequency are not too important which summarises that the importance of the term is roughly inversely proportional to the frequency of the term in the collection of documents. Usually $W_{q,t}$ is calculated by dividing the total number of documents in the corpus(N) by the frequency of the document. So that the rank of the document is preserved even if some of the words are repeated multiple times.
- **$W_{d,t}$:** This value tells us about the weight of the term t in the document d. The weight assigned to the document tells us how relevant the document is to the term t. There are many ways to assign the weight to the document. But the simplest approach is to decide the relevancy of the document based on the frequency of the term in the document. The search engine algorithm in this project

assigns weight to each document based on the frequency of the term in the document. So to calculate the rank of the document we multiply the $W_{q,t}$ value with $W_{d,t}$ for each term in the query.

- **L_d:** Ranking according to the values of $W_{q,t}$ and $W_{d,t}$ has a greater bias towards the documents with greater length. To make the score independent of the length of the documents, we use a normalisation factor L_d, which normalises the score of the documents ensuring that the similar documents to have similar rank. According to the vector space model, the normalisation factor will be the product of the magnitude of the document vector(L_d) and the query vector(L_q). To normalise the product of $W_{q,t}$ and $W_{d,t}$ we divide it with L_d and L_q. But as the L_q value is constant to all the documents we can ignore that as the order is still preserved.

C. Variant Ranking Formulas

Researchers have spent considerable time testing other ways of calculating $w_{q,t}$, $w_{d,t}$, and L_d. Some formulations avoid the use of logarithms for efficiency, others use lessons learned from statistics and language processing to give more accurate results. Figure 1 shows four schemes to calculate these values which we have implemented in our search engine.



Default	tf-idf	Okapi BM25	Wacky
$w_{q,t} = \ln\left(1 + \frac{N}{\mathrm{df}_t}\right)$	$w_{q,t} = \mathrm{idf}_t = \ln \frac{N}{\mathrm{df}_t}$	$w_{q,t} = \max\left[0.1,\ \ln\left(\frac{N-\mathrm{df}_t+0.5}{\mathrm{df}_t+0.5}\right)\right]$	$w_{q,t} = \max\left[0,\ \ln \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\right]$
$w_{d,t} = 1 + \ln(\mathrm{tf}_{t,d})$	$w_{d,t} = \mathrm{tf}_{t,d}$	$w_{d,t} = \frac{2.2 \cdot \mathrm{tf}_{t,d}}{1.2 \cdot \left(0.25+0.75 \cdot \frac{\mathrm{docLength}_d}{\mathrm{docLength}_A}\right)+\mathrm{tf}_{t,d}}$	$w_{d,t} = \frac{1+\ln(\mathrm{tf}_{t,d})}{1+\ln(\mathrm{ave}(\mathrm{tf}_{t,d}))}$
$L_d = \mathtt{docWeights}_d$	$L_d = \mathtt{docWeights}_d$	$L_d = 1$	$L_d = \sqrt{\mathtt{byteSize}_d}$

Fig. 1: Weighting Schemes

D. Basics of Index Design

Indexing is the process by which search engines will organise the information before a search to enable fast responses to queries. There are different type of indies.

- **Inverted Index:** An inverted index is collection of all terms and each term will point to all the documents containing that term. Then, search engines will use tokenization and stemming to reduce words to their core meaning .This is a much faster approach than listing all known documents against all relevant keywords and characters.
- **Positional Inverted Index:** This index will store all occurrences of a term in particular documents. Positional Inverted Index is used for phrase, biword and near queries.
- **Kgram Index:** K-grams are k-length sub-sequences of a string. A k-gram index maps a k-gram to a postings list of all possible terms that will contain the k-gram.
- **Disk Index:** We will write a single posting list on disk at a time in following format:- [df_t, id, tf_{td}, p_1, p_2......] where,
  - **$\mathbf{df_t}$:** Number of documents containing the term t.
  - **id:** Id of the document containing the term t.
  - **$\mathbf{tf_{td}}$:** Number of occurrences of the tem t in a document.
  - **$\mathbf{p_i}$:** i^{th} position of term in document.

  To answer a query first use a dictionary to locate the term byte position of term on disk, seek that position on disk, read postings from disk and return list of postings.
- **Index for rank retrieval:** It is same as inverted index, but for rank retrieval, position of term in document is not needed. Only the terms and document containing those terms are included.

E. Vocabulary Elimination

Vocabulary elimination comes under inexact retrieval which is one of the efficient scoring and ranking method. It uses heuristics to greatly reduce the amount of work for finding the top K documents, hopefully still returning highly relevant documents. The basic idea behind this technique is to ignore the terms while calculating value of accumulator with low inverse document frequency (idf). Low idf means lot of postings and poor discrimination. Here we are ignoring terms which has IDF value less than the threshold idf value.

- **Why does this save time/work?**
  Here we are not considering terms with low idf while calculation the value of accumulator. This needs skipping loops resulting less work.
- **Why are the results now inexact?**
  As we are skipping terms with low idf value, the documents will not get exact score due to skipped terms. Hence the results are inexact.
- **Why are the results are still acceptable?**
  Due to poor discrimination, the value will be so low such that it will not get so may points due to low idf. Thus we can say that the overall calculation will be acceptable.

F. Evolution Measures

Below are the measures which is used to evaluate search engine:

*1) **Precision @K**:* Precision (P) is the proportion of retrieved documents that are relevant. Precision measures the system's ability to reject any non-relevant documents in the retrieved set. Precision at k (P@k) is defined as the fraction of the top k ranked documents that are relevant, for the input query. When P@k is computed over a set of queries, the mean per-query P@k is returned. P@k a single query is defined in the figure 2

$$P@k = \frac{|Res\,[1..k] \cap Rel|}{k}$$

Fig. 2: P@k

*2) **Recall @ K**:* Recall (R) is the proportion of relevant documents that are retrieved. Recall measures the system's ability to find all the relevant documents. R@K is the proportion of relevant documents that are retrieved in the top-K. When R@k is computed over a set of queries, the mean per-query R@K is returned. R@K a single query is defined in the figure 3

$$R@k = \frac{|Res\,[1..k] \cap Rel|}{|Rel|}$$

Fig. 3: R@K

*3) **Average Precision (AP):*** Average precision is approximately the area under a precision-recall curve.The metric average precision accounts for precision and recall without having to set K. It consider ranking mistakes at the top of the ranking which are more influential as well as ranking mistakes at the bottom of the ranking which are still accounted for. AP is defined in the figure 4

$$AP\,(q) = \frac{1}{|Rel|} \sum_{i=1}^{|Res|} relevant\,(i) \cdot P@i$$

Fig. 4: Average Precision

*4) **Mean Average Precision (MAP):*** Mean average precision is defined over a set of queries, it is the mean of the average precision (AP) measurements for each query, in a set of queries. Let Q be a set of queries to test the search engine with. Then MAP is defined in the figure 5

$$MAP\,(Q) = \frac{1}{|Q|} \sum_{q \in Q} AP\,(q)$$

Fig. 5: Mean Average Precision

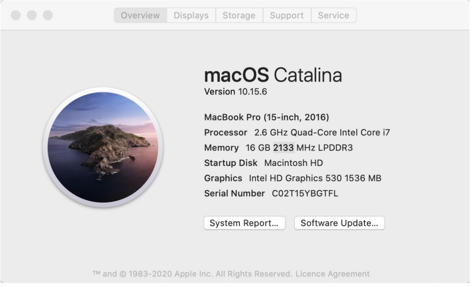
*5) **Precision-Recall Curves:*** In some situations, we want to understand the trade-off between precision and recall. In a precision-recall graph, precision is plotted as a function of recall. Then for each K examine the set of top K retrieved documents and calculate the precision and recall for the set. These values are then plotted on a graph of precision versus recall.

*6) **Mean Response Time (MRT):*** Mean response time is the average amount of time to answer one query. Typically this is measured as an average across many hundreds of thousands of requests.

*7) **Throughput:*** Throughput is the number of queries per second. This is simply the reciprocal of mean response time.

III. Experimental Result

1) Experimental Setup: To evaluate our search engine we have considered a test collection dataset which has 1400 documents. It also consist of total 225 queries with the expected relevance result. We are going to calculate mean average precision, mean response time, and throughput using these 225 queries and retrieving top 50 document for queries. We carried out this experiment on hardware refer the figure 6.



macOS Catalina
Version 10.15.6

MacBook Pro (15-inch, 2016)
Processor 2.6 GHz Quad-Core Intel Core i7
Memory 16 GB 2133 MHz LPDDR3
Startup Disk Macintosh HD
Graphics Intel HD Graphics 530 1536 MB
Serial Number C02T15YBGTFL

™ and © 1983-2020 Apple Inc. All Rights Reserved. Licence Agreement

Fig. 6: Hardware Specification

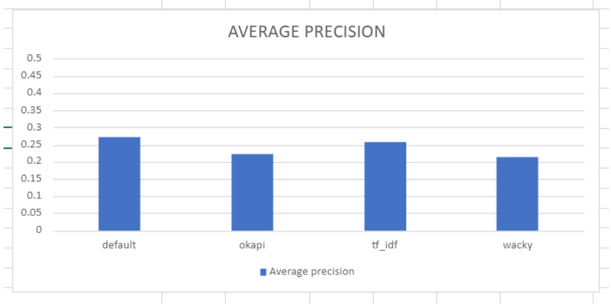


Fig. 7: Average precision for given query for variant formula

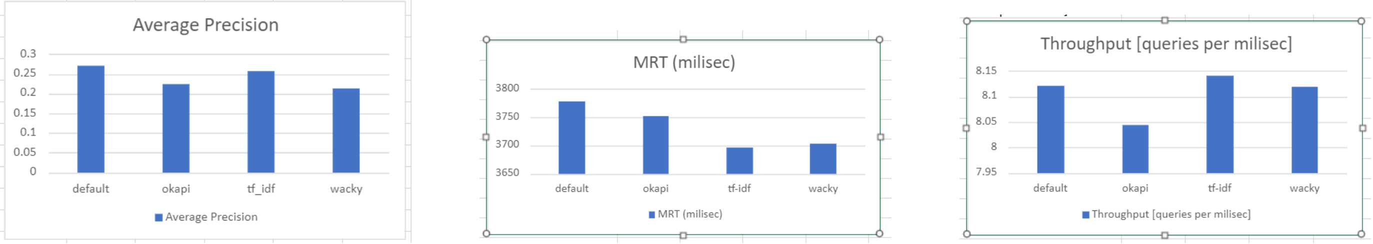
2) Results for Variant ranking formulas: We have consider query **”what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft .”**. We have calculated precision, recall value for the return document based on relevance document list in data set and plotted precision-recall curve form the computed results. We also computed average precision for all variant formulas for the same query. Figure 12 shows the precision-recall curve for this query for variant ranking formulas. Also, figure 7 shows average precision for the same query for precision@50. We have executed the same query for thirty times and calculated the mean response time and throughput. Figure 8 shows the result for thirty iterations.

We execute all 225 queries to observe mean average precision, throughput, mean response time on the test data set. Figure 9 shows the mean average precision, throughput, mean response time for variant formula.

3) Results for Vocabulary Elimination: For this we have consider various values for $W_{q,t}$ threshold. We executed all 225 queries from test data set for these values of $W_{q,t}$ threshold and computed mean average precision, throughput, mean response time. Figure 11 shows the mean average precision, throughput, mean response time for vocabulary elimination. We executed query **”what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft .”** for thirty time for list of $W_{q,t}$ threshold value and calculated the mean response time and throughput. The figure 10 shows the same results.

A. Comparing Results

1) Variant ranking formulas:

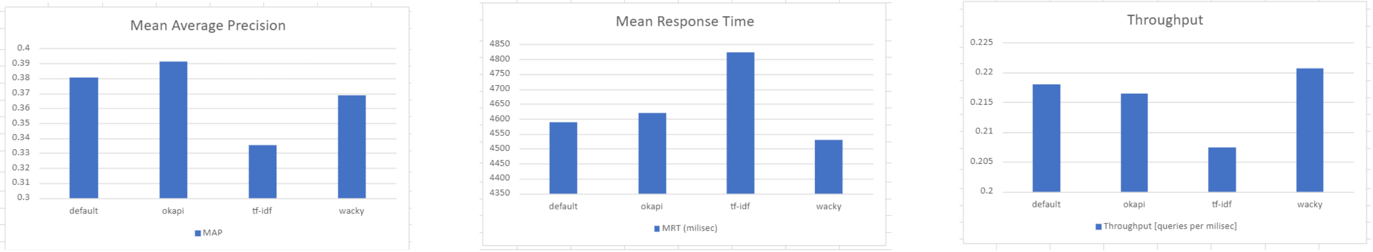


(a) Average Precision

(b) Mean Response Time

(c) Throughput

Fig. 8: Single query average precision, mean response time and throughput values for variant formulas (30 iteration)

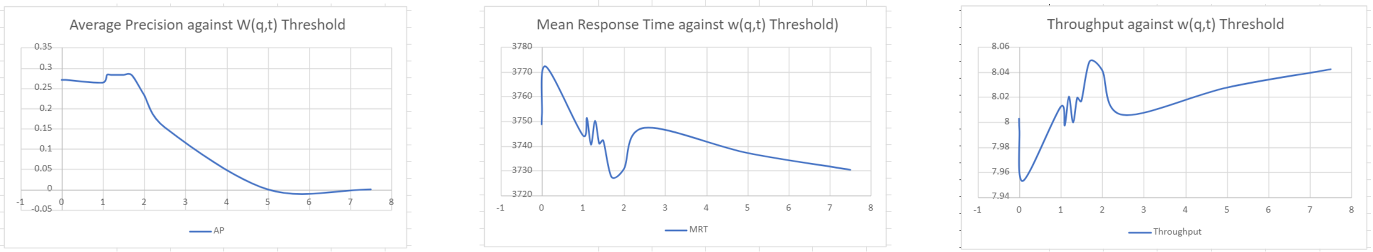


(a) Mean Average Precision

(b) Mean Response Time

(c) Throughput

Fig. 9: Measures for variant formulas (for all 225 query)

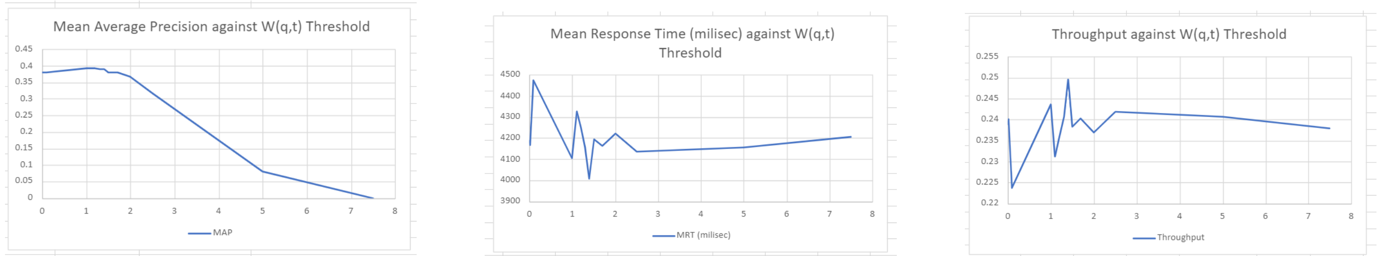


(a) Average Precision

(b) Mean Response Time

(c) Throughput

Fig. 10: Single query average precision, mean response time and throughput values for vocabulary elimination (30 iteration)



(a) Mean Average Precision

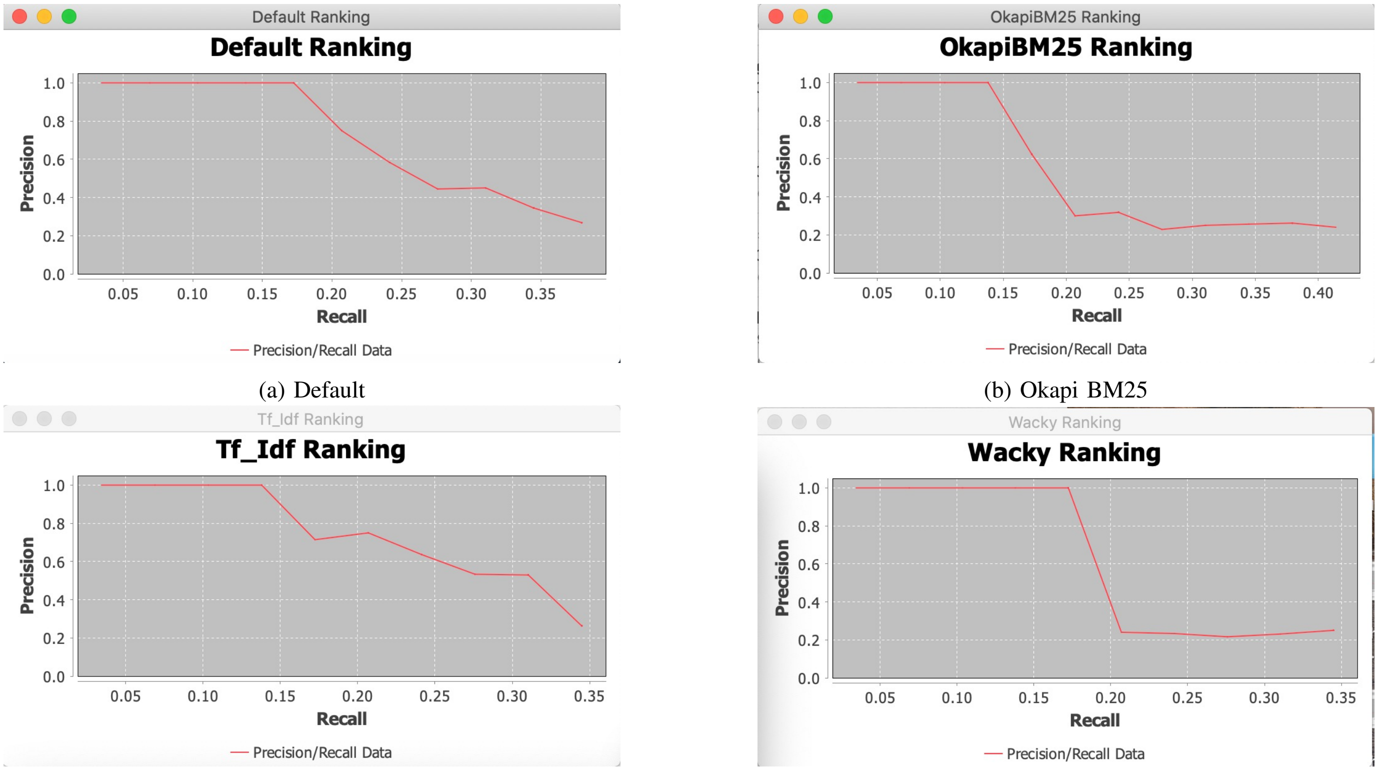
(b) Mean Response Time

(c) Throughput

Fig. 11: Measures for vocabulary elimination (for all 225 query)

- For a single query default strategy has maximum value of average precision, See the figure 7.
- For all 225 queries wacky strategy has smallest mean response time, See the figure 9b.
- For all 225 queries okapi BM25 strategy has maximum value for mean average precision, See the figure 9a.
- For all 225 queries wacky strategy has highest value for throughput, See the figure 9c.
- Mean response time for tf-idf is the lowest for 30 iterations of same query.
- Throughput for tf-idf is the maximum for 30 iterations of same query.

From above results we can say wacky perform well for the given test data as web search prefers faster query resolution



(a) Default

(b) Okapi BM25

(c) tf-idf

(d) Wacky

Fig. 12: Single query Precision-Recall Curve for variant formula

instead of most accurate solution.

2) Results for Vocabulary elimination:

- For all 225 queries the mean average precision value is maximum for $W_{q,t}$ threshold value 1.2, See the figure 11a.
- For all 225 queries the throughput value is maximum for $W_{q,t}$ threshold value 1.4, See the figure 11c.
- For all 225 queries the mean response time is minimum for $W_{q,t}$ threshold value 1.4, See the figure 11b.
- The Average precision is maximum for $W_{q,t}$ threshold 1.2 - 1.7 for single query.
- The Mean response time is minimum for $W_{q,t}$ threshold 7.5 for 30 iterations of same query. See the figure 10.
- The throughput is maximum for $W_{q,t}$ threshold 7.5 for 30 iterations of same query. See the figure 10.

According to recommendation of Buckleyand and Voorhees' for web search precision at a cut-off level of 10 or 20 is an appropriate evaluation measure. We have consider precision at cut-of level of 50 . For MRT and throughput $W_{q,t}$ threshold value 1.4 gives optimal result but for MAP $W_{q,t}$ threshold value 1.2 gives optimal result. Generally we expect fast results from web search instead of exact results. Hence we recommend 1.4 for $W_{q,t}$ threshold value for the given test data.

If we compare the result of vocabulary elimination with $W_{q,t}$ threshold value 1.4 with default strategy results, The mean average precision is increased by 2.73%, mean response time is reduced by 12.69% and Throughput is increased by 14.53%. Hence we can say that vocabulary elimination technique reduces the calculation workload for ranked retrieval with optimal results.

IV. Future Work

For future work we are considering to implement various inexact retrieval techniques like Impact Ordering, Cluster Pruning, Tiered Index etc. and then will compare the results with vocabulary elimination for better inexact retrieval technique.

V. Conclusion

We have evaluated our search engine based on standard evaluation measure such as mean average precision, mean response time, average precision, throughput etc. We have implemented vocabulary elimination which will save time and effort as we don't consider the terms with low idf. Results were satisfactory but still it could be improved more by adding indexing techniques like cluster pruning, impact ordering, and tiered index to decrease response time for a query. For rank retrieval we just take into consideration term frequencies, but if we consider other factors such as page rank algorithm, phrase query score , then rank retrieval results can be improved.

References

[1] Buckley, Chris and E. Voorhees. "Evaluating evaluation measure stability." SIGIR Forum 51 (2000): 235-242.
[2] Information retrieval. Wikipedia.
[3] Information Retrieval: how Search Engines retrieve Data.