PROJECT ON INFORMATION RETRIEVAL

Group Members:

Ananta Arora

Shantanu Kawlekar

Siddhant Pasari

Course Name: Information Retrieval

Semester: 1st Semester

Instructor name: Prof. Nada Naji

INTRODUCTION

Summary:

The project consists of two major implementations

The Indexing and Retrieval Phase

- Task1
  In this, we implement the 4 following models and calculate the scores of all the documents in the corpus and output the top 100 ones:
  a. BM25
  b. TF-IDF
  c. Cosine Similarity
  d. Using Lucene

- Task2
  To perform query expansion technique, we use ***Rocchio Algorithm*** which uses the concept of Relevance Feedback.

- Task3
  We do stopping on the corpus using the stopped list provided.
  After that we make an inverted index from the stopped corpus and perform stopping on the query file.
  We also create an inverted index of the stemmed version of the corpus provided to us.

The phase 2 or Evaluation

- Implemented a seventh run that combines query expansion technique with stopping.
- Calculated and tested the performance of the search engine using the evaluation parameters.

Detailed Description of each member's contribution

Initially we did some cleaning process, tokenization, stopping on the corpus and the query file which was done by Siddhant Pasari and Shantanu Kawlekar.

After the corpus was cleaned we implemented the task 1 which consisted of 4 base search engines.

Ananta Arora did the BM25 baseline run, Shantanu Kawlekar did Cosine Similarity and editing of Lucene Search Engines and Siddhant Pasari did the TF-IDF baseline run.

The Query Expansion technique was done by all three of the group members right from understanding the algorithm, researching about it and coding.

Query-to Query Analysis was done by Siddhant Pasari comparing the outputs of the two queries.

The Phase 2 implementation was done by Shantanu Kawlekar and Ananta Arora which calculated retrieval effectiveness of the search engine.

Documentation work was done by Shantanu Kawlekar and Siddhant Pasari.

Readme file was written by Ananta Arora.

LITERATURE AND RESOURCES

The technique used for query expansion is based on a method of relevance feedback found in information retrieval systems called the ***Rocchio's Algorithm***.

The algorithm assumes that most users have a general conception of which documents should be denoted as relevant or non-relevant.

Therefore, the user's search query is revised to include an arbitrary percentage of relevant and non-relevant documents as a means of increasing the search engine's recall, and possibly the precision as well.

We are using the ***Rocchio's Algorithm*** is due to the following reasons:

- The document ranking will result in more precise documents being made available to the user.
- The reason to not consider the ***Thesaurus-based query expansion algorithm*** is because it increases recall but may decrease precision.

There are several research articles that support our Algorithm choice:

- http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap11.htm
  This link explains the concepts and the structure of relevance feedback model, query expansion, importance to the term weighting.
  Rocchio which is based on a method of relevance feedback model has found to have a higher precision value and a lower recall value which is what the best of the information retrieval systems want.

- http://ciir.cs.umass.edu/downloads/SEIRiP.pdf
  The reference book by W. Bruce Croft provided us with the BM25 algorithm, the right values of the constants to be used and an idea as to how we could use those values to tune the search engines to get relevant results.

The third party tools used in the project were **nltk and BeautifulSoup.**

IMPLEMENTATION AND DISCUSSION

Our project starts with building the following 4 search engines from the provided corpus of 3204 documents. The initial step to any search engine is by cleaning the document completely, tokenizing it and finally forming an inverted index for it.

TASK1

BM25

It is a ***probabilistic model*** that considers the document and query term weights while generating the score for a document with respect to a query. While calculating the BM25 score, the parameters R and r are calculated as per the relevance judgement file that is provided. Also, the inverted index of the corpus is stored in the dictionary and is utilized to calculate the score. We have also generated a file that maintains information about the terms appearing in each document. This file is stored in a dictionary and is utilized while calculating the score. The score is calculated as per the formula below:

$$score(q,d) = \sum_{i=1}^{|q|} idf(q_i) \cdot \frac{tf(q_i,d) \cdot (k_1+1)}{tf(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$

TF-IDF

Short form for ***term frequency–inverse document frequency***, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{\mathrm{N}}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j
df_i = total number of documents (speeches) containing i
N = total number of documents (speeches)

Cosine Similarity

Measure of the angle or we can say similarity between two vectors, in our case the two vectors are the document vector and the query vector.

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Lucene Search Engine

Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given Document is to a User's query.

The general idea behind the VSM model is that more times a query term appears in a document relative to the number of times the term appears in all the documents in the collection, the more relevant that document is to the query.

Lucene uses boost to get high quality results

Document level boosting

Document's Field level boosting

Query level boosting

TASK2:

Query Expansion

The query expansion technique we have used is a combination of Relevance and Pseudo Relevance Feedback. For implementation purpose, we have used BM25 retrieval model. To generate the initial results, we have utilized the relevance judgement file. For the second pass, where we expand the query, we have considered the top 10 documents of the results generated from the first pass as our set of relevant documents. In our case, the document term weight of the terms is the tf-idf score of the terms. We have used Rocchio Algorithm for getting the terms to be added to the query for expansion. Following is the formula for the algorithm:

$$q'_j = \alpha.q_j + \beta.\frac{1}{|Rel|}\sum_{D_i \in Rel} d_{ij} - \gamma.\frac{1}{|Nonrel|}\sum_{D_i \in Nonrel} d_{ij}$$

α, β and γ are the parameters that control the effect of each component. q_j is the initial query vector for which we have considered idf as 1 so the query vector is represented by the term frequencies of the query terms. Rel is the set of relevant documents and Nonrel is the set if nonrelevant documents. d_{ij} is the weight of j^{th} term in the document i. We have three vectors each with a length same as the vocabulary of the corpus. And $\beta > \gamma$ as we give more importance to the positive feedback (i.e relevant documents). We have considered the values for α, β and γ as 8,16 and 4 respectively. And after getting a list of terms with their scores, we are removing the stop words and the original query terms from the list and then taking the top 15 terms from the list to add it to our original query thereby expanding the query. All the expanded queries are written to a file and then fed into the BM25 algorithm that generated the initial results for us. Finally, we get the results of top 100 documents with their precision and recall calculated as well.

TASK3:

In the third task, we do stopping on the corpus provided to us by removing all the common stop words from it.

After the corpus is successfully stopped, we create an inverted index of it. We have also created the number of terms for each document as that is required by our

algorithm. The Query file has also been stopped using the same stop list provided to us. Thereby, applying the same processing on the list of queries as well.

The stemmed corpus was provided to us in a single file for which we generated individual 3204 files by processing the above-mentioned corpus.

We have used the stemmed query list provided to us for query-by-query analysis.

The same process of making an inverted list is followed on the stemmed corpus provided to us.

PHASE 2 : Evaluation

For the seventh run, we have used the Lucene Search Engine on the stopped corpus using the inverted list created for unigrams from the above-mentioned corpus.

We have used the stopped queries for running the search.

Now that we have seven distinct runs, we assess the performance of our search engines based on the retrieval effectiveness using the following measures:

1) **MRR**
2) **MAP**
3) **P@K, K=5 & 20**
4) **Precision and Recall**

Here are the measures for the seven runs: -

(Note that we are considering the Query_id 2 as a sample for all the runs below)

BM25 baseline run

Mean Average Precision (MAP): 0.32469337471344517

Mean Reciprocal Rank (MRR): 0.7892818055927813

Precision @ K= 5 for query number: 2 is: 0.6

Precision @ K= 20 for query number: 2 is: 0.15

Precision after the 100th document: 0.03

Recall after the 100th document: 1.0

TF-IDF baseline run

Mean Average Precision (MAP): 0.17122969009651487

Mean Reciprocal Rank (MRR): 0.5235438573178958

Precision @ K= 5 for query number: 2 is: 0.0

Precision @ K= 20 for query number: 2 is: 0.05

Precision after the 100th document: 0.02

Recall after the 100th document: 0.66

Cosine Similarity baseline run

Mean Average Precision (MAP): 0.11873370675214742

Mean Reciprocal Rank (MRR): 0.4640330670966733

Precision @ K= 5 for query number: 2 is: 0.0

Precision @ K= 20 for query number: 2 is: 0.05

Precision after the 100th document: 0.01

Recall after the 100th document: 0.33

Lucene Search baseline run

Mean Average Precision (MAP): 0.2718683075594305

Mean Reciprocal Rank (MRR): 0.6799758876875811

Precision @ K= 5 for query number: 2 is: 0.4

Precision @ K= 20 for query number: 2 is: 0.15

Precision after the 100th document: 0.03

Recall after the 100th document: 1.0

Query Expansion with BM25 run

Mean Average Precision (MAP): 0.34640765131942125

Mean Reciprocal Rank (MRR): 0.7866221422255906

Precision @ K= 5 for query number: 2 is: 0.6

Precision @ K= 20 for query number: 2 is: 0.15

Precision after the 100th document: 0.03

Recall after the 100th document: 1.0

BM25 on stopped corpus

Mean Average Precision (MAP): 0.38951893547156435

Mean Reciprocal Rank (MRR): 0.752960927960928

Precision @ K= 5 for query number: 2 is: 0.6

Precision @ K= 20 for query number: 2 is: 0.15

Precision after the 89th document: 0.033

Recall after the 89th document: 1.0

Lucene on stopped corpus

Mean Average Precision (MAP): 0.25425482124329163

Mean Reciprocal Rank (MRR): 0.1822293377373798

Precision @ K= 5 for query number: 2 is: 0.6

Precision @ K= 20 for query number: 2 is: 0.15

Precision after the 89th document: 0.033

Recall after the 89th document: 1.0

Analysis for query done using BM25 Search Engine

Consider the stemmed query "***portabl oper system"*** and the unstemmed query ***"portable operating systems"***

- The first relevant document that appeared for the stemmed query was at rank **66** with a precision rate of **0.015151515151515152**
- The first relevant document that appeared for the unstemmed query was at rank **1** with a precision rate of **1.0**

Consider the stemmed query ***"code optim for space effici"*** and the unstemmed query "***code optimization for space efficiency"***

- The first relevant document that appeared for the stemmed query was at rank **5** with a precision rate of **0.2**
- The first relevant document that appeared for the unstemmed query was at rank **1** with a precision rate of **1.0**

From the above observation, we can conclude that the results for the unstemmed queries are much better than the stemmed queries.

RESULTS

1) *BM25 baseline run*

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-2947	1	19.422237613740162	NR	0	0
1	Q0	CACM-2371	2	18.7773679183087	NR	0	0
1	Q0	CACM-2500	3	18.41205345004912	NR	0	0
1	Q0	CACM-2535	4	18.192174348362062	NR	0	0
1	Q0	CACM-2950	5	17.706799326386218	NR	0	0

2) TF-IDF *baseline run*

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-1938	1	0.14728720364499878	NR	0	0
1	Q0	CACM-2371	2	0.10072965524832189	NR	0	0
1	Q0	CACM-1071	3	0.09999427386170949	NR	0	0
1	Q0	CACM-1657	4	0.09890012929714707	NR	0	0
1	Q0	CACM-0971	5	0.0864681296080474	NR	0	0

3) COSINE-SIMILARITY *baseline run*

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-1938	1	0.04942843532269551	NR	0	0
1	Q0	CACM-0971	2	0.04327912343436488	NR	0	0
1	Q0	CACM-3048	3	0.04205344013580225	NR	0	0
1	Q0	CACM-1680	4	0.041444535550277155	NR	0	0
1	Q0	CACM-1071	5	0.04121019659450415	NR	0	0

4) LUCENE SEARCH ENGINE *baseline run*

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-2319	1	0.3903907	NR	0	0
1	Q0	CACM-1657	2	0.3183247	NR	0	0
1	Q0	CACM-2378	3	0.27641648	NR	0	0
1	Q0	CACM-2629	4	0.27382436	NR	0	0
1	Q0	CACM-2054	5	0.2707414	NR	0	0

5) BM-25 STOPPED

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-2371	1	16.36859556040743	NR	0	0
1	Q0	CACM-2439	2	15.218542717404173	NR	0	0
1	Q0	CACM-2522	3	14.50645850058987	NR	0	0
1	Q0	CACM-2951	4	14.304288514153153	NR	0	0
1	Q0	CACM-2500	5	14.161225376461188	NR	0	0

6) LUCENE STOPPED

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-2629	1	0.4175381	NR	0	0
1	Q0	CACM-2319	2	0.41148937	NR	0	0
1	Q0	CACM-1657	3	0.3881113	NR	0	0
1	Q0	CACM-2151	4	0.35625854	NR	0	0
1	Q0	CACM-2218	5	0.31387976	NR	0	0

7) ROCCHIO FOR QUERY EXPANSION

Query Id	*Q0*	*Document Name*	*Rank*	*Score*	*Relevance*	*Precision*	*Recall*
1	Q0	CACM-2371	1	36.20204695709471	NR	0	0
1	Q0	CACM-2500	2	32.547032929069196	NR	0	0
1	Q0	CACM-2439	3	31.505326385379647	NR	0	0
1	Q0	CACM-2947	4	28.521636501085904	NR	0	0
1	Q0	CACM-2950	5	28.432042358461285	NR	0	0

(Note that we have just included the top 5 results for the query id 1. The rest of the output for other queries are present in our output files)

CONCLUSION AND OUTLOOK

After careful analysis of all the seven runs of the search engines, we observed that for various queries the relevant documents appeared at top ranks in both ***BM25 baseline run*** and ***BM25 using query expansion run***.

The ***BM25 using query expansion run*** however gave better results considering the top 10 documents for various queries, also we know that the users generally care about the top 5 to 10 results and by using the above-mentioned run, more relevant results are obtained.

- We observed that by changing various ***parameters*** like alpha, beta, gamma, K1 or K2 we could tune search engines to obtain better results.
- Features like ***snippet generation, query highlighting*** can help the user to get a gist of the documents during the search results itself.
- Certain features like ***spell checking, auto-correct, suggestions*** can be incorporated in the system to help the user get relevant information.

We could have incorporated the above-mentioned points to improve our project.

BIBLIOGRAPHY

1. http://ciir.cs.umass.edu/downloads/SEIRiP.pdf
2. https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
3. http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html
4. https://www.youtube.com/watch?v=yPd3vHCG7N4
5. http://www.sfs.uni-tuebingen.de/~parmenti/slides/slides9-1x4.pdf