INFORMATION RETRIEVAL PROJECT REPORT

CS6200: Information Retrieval
Northeastern University
Spring 2018

Name of the Instructor: Dr. Nada Naji

TEAM MEMBERS
Pratik Devikar
Saran Prasad Ambikapathy
Vishal Patel

Table of Contents

1 Introduction

1.1 OVERVIEW

The goal of this project is to build our own information retrieval system, evaluate and compare their performance levels in terms of retrieval effectiveness. Our dataset is the CACM test-collections corpus. Various operations are performed during the execution such as text transformation, index creation, query processing, document ranking and evaluation results. The project is primarily divided into 3 phases.

Phase 1 of the project involves Indexing and Retrieval. The indexer is a word unigram indexer that was built from scratch along with the retrieval models *BM25*, *Lucene*, *TF-IDF* and Smoothed *Query Likelihood* model. We perform four baseline runs using these four retrieval models. We also perform one *Query enrichment* run which we carry out using the following methods - *stopping*, *query time stemming* and *pseudo relevance feedback*. We also perform three additional baseline runs each with stopping and on the stemmed version of the corpus.

Phase 2 of the project concentrates on implementing *snippet generation* and *query term highlighting* techniques within the results from one of the baseline runs and displaying the results in a webpage resembling a typical search results page.

Finally, in Phase 3, we analyze the performance of the retrieved documents over the eight distinct runs we performed using the following retrieval effectiveness measures: *Recall, Precision, Mean Average Precision* and *Mean Reciprocal Rank.*

Apart from these regular components of the search engine, we have also implemented

1. A synthetic spelling-error generator - to produce noisy queries and compare them to those of the noise-free baselines.
2. A soft-matching query handler - that tries to minimize the impact on effectiveness caused by any unknown error model.

1.2 TEAM MEMBERS' CONTRIBUTIONS

- *Pratik Devikar* - Pratik was responsible for implementing three out of the four baseline runs using BM25, TF-IDF and Smoothed Query Likelihood model. He was also responsible for two of the three baseline runs in Task 3A and 3B each using BM25 and TFIDF model. Additionally, he performed the task 2 of the extra credit part of implementing soft-matching query handler. Lastly, he documented his design choices and performed query-by-query analysis of the three stemmed queries.
- *Vishal Patel* - Vishal was responsible for creating indexes from the CACM corpus. He was responsible for Task 2 query enrichment. He performed Phase 3 Evaluation, for the 8 baseline runs. Additionally, he performed part 1 of the extra credit, wherein he created a synthetic spelling-error generator.
- *Saran Prasad Ambikapathy* - Saran was responsible for research, design and implementation of the snippet generation technique and query term highlighting. He was also responsible for implementing and executing the baseline runs using Lucene for tasks 1, 3A and 3B. He developed a web page template to display the results of snippet generation and was also responsible for the corresponding documentation of his design choices.

2 Literature and Resources

2.1 OVERVIEW

- *BM25 model:* For calculating the scores for the documents, we have used the BM25 as the ranking function. Its parameter values of 'k', 'k1', k2' and b are chosen as per the TREC standards.
- *tf-idf measure:* The tf-idf score for each document is calculated by using the normalized value of the term frequency multiplied by its inverse document frequency.
- *Smoothed Query Likelihood model:* The scores for the documents are calculated by applying the smoothing to the query likelihood model.
- *Lucene:* An open source library that provides indexing and searching functionalities.
- *Query Expansion:* The query expansion in our project is done using query time stemming and pseudo relevance feedback.
- *Stopping:* The standard stop list 'common_words.txt' was used to perform stopping. The queries in this case are not stemmed.
- *Stemming:* We have performed stemming using the BM25 model on 'cacm_query.txt' using the corpus 'cacm_stem.txt'.
- *Precision & Recall:* The formulas used to calculate precision and recall are:

  Precision = |Relevant ∩ Retrieved| / |Retrieved|

  Recall = |Relevant ∩ Retrieved| / |Relevant|
- *MAP:* The formula used to calculate Mean Average Precision is:

  MAP = Σ Average Precision / Number of Queries
- *MRR:* *Reciprocal rank* is reciprocal of the rank at which the first relevant document is retrieved. *Mean Reciprocal Rank* is the average of the reciprocal ranks over a set of queries.
- *P@K:* *P@K* is calculated as the number of relevant documents in the *top K* retrieved documents. We have calculated for K = 5 and K = 20.
- *Snippet Generation:* We have used *query-independent* summary generation technique which identifies significant words in a document using a frequency-based criterion and consequently ranks sentences that contain most significant words to be displayed as snippets.
- *Query Highlighting:* The query terms that match texts in the generated snippets are highlighted in a web page that resembles a typical search results page.
- *Synthetic spelling error generator:* The spelling errors are generated by first selecting at most 40% of the query terms and then for each term randomly shuffling the non-boundary characters.
- *Soft-matching query handler:* This model used 'Edit Distance' to calculate the closest word resembling the wrongly spelled word.

2.2 THIRD PARTY TOOLS

- *NLTK:* NLTK's brown corpus is used as a dictionary of English words for the calculation of the Edit distance between two words.
- *Beautifulsoup:* has been used for parsing of both the documents and the queries.
- *Lucene Libraries:* We used the following Lucene libraries
  a. Lucene-analyzers-common-7.3.0.jar
  b. Lucene-core-7.3.0.jar
  c. Lucene-queryparser-7.3.0.jar

- *JSoup:* A Java equivalent of BeautifulSoup to parse markup documents.
- *JWNL:* Java WordNet Libary
- *Apache POI:* Java library to write output to spreadsheets.

2.3 SCHOLARLY WORK AND RESEARCH ARTICLES

- Clustering in Information Retrieval - https://nlp.stanford.edu/IR-book/html/htmledition/clustering-in-information-retrieval-1.html
- A real time Dynamic Programing approach to Snippet Generation for HTML Search Engines - http://www.cs.pomona.edu/~dkauchak/ir_project/whitepapers/Snippet-IL.pdf
- Query Biased snipped generation in XML search - https://web.njit.edu/~ychen/sigmod08_snippet.pdf

3 Implementation and Discussion

3.1 BASELINE RUNS

Firstly, the text transformation techniques were applied on the raw data of the corpus. Punctuation removal, case folding were the techniques used for the above. Apart from these, the numbers at the end of every document were ignored. Next, we created inverted indexes from the processed text. We stored all the inverted indexes in a different directory. The format of the index is: *termID: Frequency.*

3.2 BM25 MODEL

The formula used to calculate the scores of the documents is the following:

$$\sum \log \{(r_i + 0.5) / (R - r_i + 0.5)\} / \{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)\} * (k_1 + 1) f_i / (K + f_i) * (k_2 + 1) qf_i / (k_2 + qf_i)$$

The summation is done over all the terms of a query.

where,

ri = number of relevant documents containing the term i
R = number of relevant documents for the query
ni = number of documents containing the term i
N = Total number of documents in the corpus (3204)
k1 = 1.2 as per TREC standards
K2 = 0 to 1000
*K = k_1 ((1 – b) + b * dl /avdl) where dl is document length, avdl is average length of document in the collection.*
b = 0.75 as per TREC standards

The information about the top 100 documents are that are retrieved is stored in the following format: *QueryId 'Q0' DocID Rank score sys_name*

3.3 TF-IDF MODEL

The formula used to calculate the scores of the documents using Tf-Idf model is:

- Tf = log(1+(number of times a term occurs in a document) / Document length)
- Idf = 1+ log((Total number of documents) / (Number of documents in which the term appears + 1))

Both these terms are normalized.

The information about the top 100 documents are that are retrieved is stored in the following format: *QueryId 'Q0' DocID Rank score sys_name*

3.4 SMOOTHED QUERY LIKELIHOOD MODEL

The formula used to calculate the scores of the documents using the smoothed query likelihood model is:

$$(1-\alpha D)P(qi|D) + \alpha DP(qi|C)$$

where,

αD = coefficient controlling the probability of the unseen words = 0.35
P(qi|D) = Probability of query term qi occuring in document
P(qi|C) = Probability of query term qi occuring in the corpus

The information about the top 100 documents are that are retrieved is stored in the following format: *QueryId 'Q0' DocID Rank score sys_name*

3.5 QUERY ENRICHMENT

1. *Query time stopping* - Stop words were removed from the query as a part of query enrichment.
2. *Query time stemming* - Stems for each word were found using wordnet. These were then added the query.
3. *Pseudo Relevance feedback* - BM25 run was taken for each query. Top 10 documents of those results were then merged, and tokens were sorted based on total frequency. Top 10 frequent words were appended to the query as part of pseudo relevance feedback.

3.6 QUERY-BY-QUERY ANALYSIS

Query number 1: *portabl oper system*

BM25 model:
1 Q0 CACM-3127 1 14.044436693545748 BM25_model_on_stemmed_corpus
1 Q0 CACM-1930 2 9.053699061439582 BM25_model_on_stemmed_corpus
1 Q0 CACM-3196 3 8.96985757979467 BM25_model_on_stemmed_corpus
1 Q0 CACM-2246 4 7.852623898429934 BM25_model_on_stemmed_corpus
1 Q0 CACM-2319 5 5.719068140160503 BM25_model_on_stemmed_corpus

Lucene model:
1 Q0 cacm_stem_3127 1 15.12805 Lucene_Query_Parser_CACM_stemmed_corpus
1 Q0 cacm_stem_2246 2 10.532982 Lucene_Query_Parser_CACM_stemmed_corpus
1 Q0 cacm_stem_1930 3 8.6281595 Lucene_Query_Parser_CACM_stemmed_corpus
1 Q0 cacm_stem_3196 4 8.560251 Lucene_Query_Parser_CACM_stemmed_corpus
1 Q0 cacm_stem_3068 5 6.213706 Lucene_Query_Parser_CACM_stemmed_corpus

TFIDF model:
1 Q0 CACM-2542 1 1.7680392935508102 TFIDF_model_Stemmed
1 Q0 CACM-1665 2 1.6871382522652563 TFIDF_model_Stemmed
1 Q0 CACM-1719 3 1.6871382522652563 TFIDF_model_Stemmed
1 Q0 CACM-2188 4 1.6871382522652563 TFIDF_model_Stemmed
1 Q0 CACM-2319 5 1.6871382522652563 TFIDF_model_Stemmed

From the top 5 results from the 3 baseline runs for the query there is no document that is common for all 3 runs. However, 4 out of 5 documents are same for BM25 and Lucene

(although their rankings are shuffled). Also, there is 1 common document between TFIDF and BM25 model. For BM25 and Lucene there is a sharp decrease in the scores between rank 1 and rank 2, although the decrease becomes steady thereafter. After looking at the documents of the top results we were able to conclude that the retrieved documents are indeed relevant. So as users we would be satisfied by the results generated.

Query number 3: *parallel algorithm*

BM25 model:
3 Q0 CACM-2714 1 6.802703930764863 BM25_model_on_stemmed_corpus
3 Q0 CACM-2664 2 6.480659354835745 BM25_model_on_stemmed_corpus
3 Q0 CACM-2785 3 6.35556498014273 BM25_model_on_stemmed_corpus
3 Q0 CACM-2973 4 6.200391076361175 BM25_model_on_stemmed_corpus
3 Q0 CACM-2266 5 6.186738177926703 BM25_model_on_stemmed_corpus

Lucene:
3 Q0 cacm_stem_2714 1 7.4759502 Lucene_Query_Parser_CACM_stemmed_corpus
3 Q0 cacm_stem_2973 2 7.290684 Lucene_Query_Parser_CACM_stemmed_corpus
3 Q0 cacm_stem_2266 3 6.76527 Lucene_Query_Parser_CACM_stemmed_corpus
3 Q0 cacm_stem_950 4 6.6894546 Lucene_Query_Parser_CACM_stemmed_corpus
3 Q0 cacm_stem_2785 5 6.632615 Lucene_Query_Parser_CACM_stemmed_corpus

TFIDF:
3 Q0 CACM-2714 1 3.59370447589263 TFIDF_model_Stemmed
3 Q0 CACM-2266 2 2.525205864955467 TFIDF_model_Stemmed
3 Q0 CACM-2401 3 2.331101543414897 TFIDF_model_Stemmed
3 Q0 CACM-3075 4 2.331101543414897 TFIDF_model_Stemmed
3 Q0 CACM-3156 5 2.331101543414897 TFIDF_model_Stemmed

From the top 5 results from the 3 baseline runs for the query there are 2 documents that are common for all 3 runs. However, 4 out of 5 documents are same for BM25 and Lucene (although their rankings are shuffled). Also, there are 2 common documents between TFIDF & BM25 model and also 2 in between TFIDF and Lucene. For BM25 and TFIDF there is a sharp decrease in the scores between rank 1 and rank 2, although the decrease becomes steady thereafter and for Lucene there is sharp decrease in between rank 2 and 3. After looking at the documents of the top results we were able to conclude that the retrieved documents are indeed relevant. As users we would be satisfied by the result generated.

Query number 7: parallel processor in inform retriev

BM25 model:
7 Q0 CACM-1811 1 12.68751516592611 BM25_model_on_stemmed_corpus
7 Q0 CACM-2967 2 10.884003205299779 BM25_model_on_stemmed_corpus
7 Q0 CACM-3075 3 10.84801306805624 BM25_model_on_stemmed_corpus
7 Q0 CACM-2664 4 10.825637569526174 BM25_model_on_stemmed_corpus
7 Q0 CACM-2973 5 10.57925178563439 BM25_model_on_stemmed_corpus

Lucene model:
7 Q0 cacm_stem_1811 1 13.137129 Lucene_Query_Parser_CACM_stemmed_corpus
7 Q0 cacm_stem_2714 2 11.668196 Lucene_Query_Parser_CACM_stemmed_corpus
7 Q0 cacm_stem_2973 3 11.624806 Lucene_Query_Parser_CACM_stemmed_corpus
7 Q0 cacm_stem_2664 4 11.383755 Lucene_Query_Parser_CACM_stemmed_corpus
7 Q0 cacm_stem_3075 5 11.341561 Lucene_Query_Parser_CACM_stemmed_corpus

TFIDF model:
7 Q0 CACM-1811 1 5.53404639442928 TFIDF_model_Stemmed
7 Q0 CACM-2175 2 3.846612767730573 TFIDF_model_Stemmed
7 Q0 CACM-2838 3 3.8366013274602597 TFIDF_model_Stemmed
7 Q0 CACM-2973 4 3.8366013274602597 TFIDF_model_Stemmed
7 Q0 CACM-1665 5 3.3776948480386855 TFIDF_model_Stemmed

From the top 5 results from the 3 baseline runs for the query there is 1 document that is common for all 3 runs. However, 4 out of 5 documents are same for BM25 and Lucene (although their rankings are shuffled). For all the models there is a sharp decrease in the scores between rank 1 and rank 2, although the decrease becomes steady thereafter. After looking at the documents of the top results we were able to conclude that the retrieved documents are indeed relevant. As users we would be satisfied by the result generated.

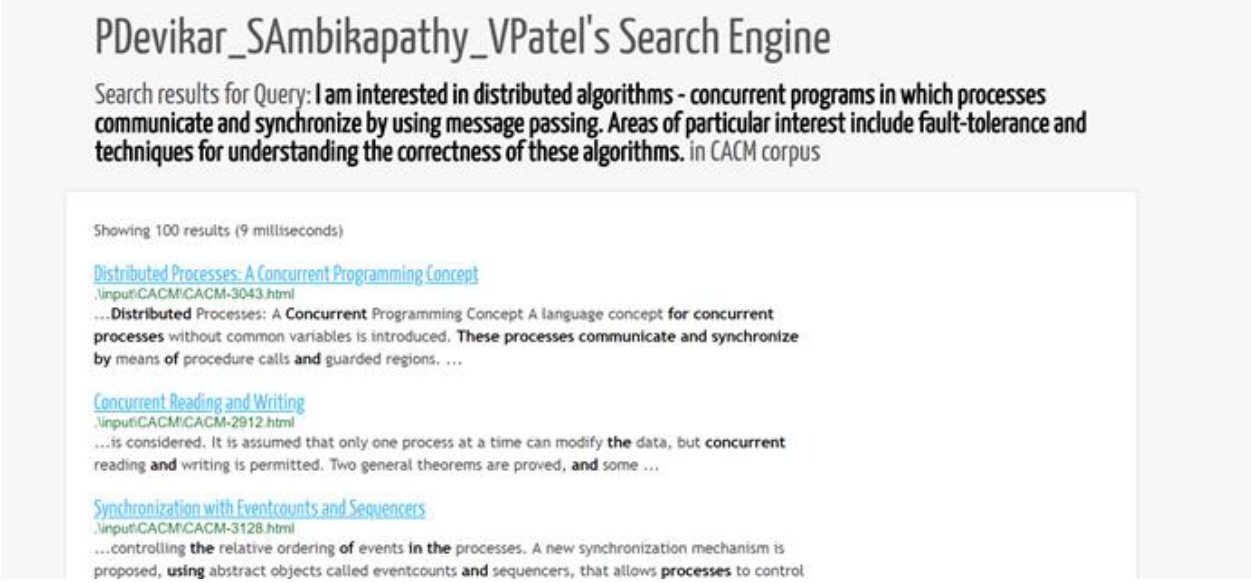
3.7 SNIPPET GENERATION AND QUERY TERM HIGHLIGHTING

The *query-independent* summary generation technique which we used identifies significant words in a document using a frequency-based criterion. A word w is a significant word if

$$f_{d,w} \geq \begin{cases} 7 - 0.1 \times (25 - s_d), & \text{if } s_d < 25 \\ 7, & \text{if } 25 \leq s_d \leq 40 \\ 7 + 0.1 \times (s_d - 40), & \text{otherwise,} \end{cases}$$

Where, $f_{d,w}$ - frequency of word w in document d.
s_d - number of sentences in document d.
For simplicity, we used a snippet window of size = 30 words to denote a sentence length. Thus the snippets are generated in advance, during the indexing process, exerting less load during the querying phase. This is the main reason why we chose a query-independent technique post analyzing other query-dependent techniques like Query biased documents snippets methodology.

To highlight the query terms, we did a case-folded, punctuation-free matching of each query term with the snippet text and highlighted them using HTML markups. The following is a snapshot of one of the results from the CACM baseline run.



3.8 SYNTHETIC ERROR GENERATOR

The objective of this task is to generate error in terms of the query. Based on given specifications. We picked a number between *1 and 40%* of the query length. Then we shuffled that many terms from the query keeping the starting and the ending character untouched.

3.9 SOFT MATCHING QUERY HANDLER

The objective of this task was to find a correct word closely resembling a wrongly spelled word. We have used *Edit distance* as a similarity measure *and probability of word occurrence* to implement the above task.

Here, we are assuming the length of the incorrect word and its closest correct words are the same, i.e. *max(Edit Distance) = length(incorrect word).*

We calculated the edit distance of a given word with every word in the brown corpus. The edit distance recurrence relation using the Dynamic Programming approach is given below:

$$D(i, j)=\min[D(i-1,j)+1, D(i, j-1) + 1, D(i-1,j-1) +t(i, j)]$$

Where D is the optimal edit distance of the instance i,j of the two words.

Once we find the set of closest words, we check their occurrences in our CACM corpus. The term with the *maximum probability* is selected as the closest 'corrected' word of the given word. Eventually, the corrected queries are run on the BM25 model.
The information about the top 100 documents are that are retrieved is stored in the following format: *QueryId 'Q0' DocID Rank score sys_name*

4 Results

Following is a summary of evaluation results from different phases of the project sorted by their Mean Average Precision values.

Retrieval Model	Mean Average Precision	Mean Reciprocal Rank
Stopping - Lucene	0.46361999193607384	0.7293574481074481
Stopping - BM25	0.448247054400717	0.6943523890639276
Lucene - CACM	0.43751759366952475	0.7049387934003318
BM25 - CACM	0.41489300129333895	0.681697673241791
Stopping - TF-IDF	0.4075005879099344	0.6288163225663225
TF-IDF - CACM	0.35414839663461256	0.5905157164772549
Query Enrichment - BM25	0.37874427399451593	0.6289397924013309
SQLM - CACM	0.021567410845849806	0.03384227726706532

Below are the links to spreadsheet results obtained in all runs for all required metrics.

- Phase 1 – Task 1 Baselines – a. Lucene | b. BM25 | c. tf-idf | d. SQLM
- Phase 1 – Task 2 Baselines – Query Enriched Run
- Phase 1 – Task 3 Baselines – a. Lucene | b. BM25 | c. tf-idf

5 Conclusion and Outlook

5.1 CONCLUSION

From our research, design choices, implementation and the results of the 8 runs we arrived at the following conclusions:

- The best results were obtained from the Lucene run with stopping. As per its MAP(0.46361999193607384) values and MRR(0.7293574481074481) values it was the best result closely followed by BM25 baseline model (MAP-0.448247054400717 and 0.6943523890639276). This observation might be attributed to the fact that Lucene incorporates both Boolean model along with Vector Space Model.
- Smoothed Query Likelihood model and tf-idf model with baseline runs scored significantly low scores compared to BM25 and Lucene models. (MAP scores - 0.021567410845849806 and 0.35414839663461256 respectively).
- Every model scored a higher score when stopping was used as compared to the its baseline runs. This observation indicates that the list of stop words might be including certain relevant high frequency words which is having a negative impact on the overall search effectiveness.
- Lucene again scored the highest in the baseline runs amongst all the models. (MAP is -> 0.43751759366952475, MRR is -> 0.7049387934003318)
- Smoothed Query Likelihood model measures have produced the least values for all the evaluation techniques. Even the p@5 and p@20 values for many queries has been 0 indicating that zero relevant documents were retrieved in top 5/20 ranked documents. Hence, ideally, tf-idf does not serve as a good measure for retrieval model, at least for the CACM corpus.

5.2 OUTLOOK

Based on our inference, we feel the following features can be included to this project as future modules

1. Unsupervised Machine learning algorithms such as *Clustering* can be made use to improve the effectiveness of the search results.
2. *Query Log* along with *Pseudo Relevance Feedback* to improve performance of search results retrieval.
3. *PageRank* can be leveraged to evaluate a document's quality and eventually use this information in the ranking.

6 Bibliography

6.1 BOOKS

- Search Engines: Information Retrieval in Practice by Croft, Metzler and Strohman.
- Lucene in Action, 2nd Edition, by McCandless, Hatcher and Gospodnetic.

6.2 WEBSITES

- Lucene documentation - https://lucene.apache.org/core/7_3_0/index.html
- NLTK - https://www.nltk.org/
- JSoup documentation - https://jsoup.org/apidocs/org/jsoup/nodes/Document.html
- Snippet generation - Literature
- https://nlp.stanford.edu/IR-book/html/htmledition/results-snippets-1.html
- Edit Distance - https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/
- Query Biased Snippet Generation - https://people.eng.unimelb.edu.au/jzobel/fulltext/ecir09.pdf