INFORMATION RETRIEVAL

Instructor: Prof. Nada Naji

Members:

Shashikiran Gadhar

Raghavendra Venkatesh

Akash Chidananda Murthy

Semester: First

2. INTRODUCTION

The Web in today's world serves as one of the most reliable and primary source of information. Using the web for information needs is a lot a daily activity for most people. A field that concerns with the structure, analysis, storage, searching and retrieval of information is called Information Retrieval. This project deals with few but important aspects of Information Retrieval. The project is a practical implementation of the core concepts of information retrieval and basically deals with building a search engine from scratch and evaluating the effectiveness of the built search engine by comparing their retrieval effectiveness with a standard result set.

The various phases of implementation include designing and building an Indexer, implementation of query expansion techniques such as Inflectional and/or derivational variants, Pseudo relevance feedback, etc. and also the evaluation of the results obtained in the above mentioned phases.

All the aspects of this project has been successfully implemented in a group of three. However, a detailed description about the contribution of each member of the group is as follows:

ChidanandaMurthy, Akash was responsible for designing of BM25 retrieval model and Cosine Similarity retrieval model.

Venkatesh, Raghavendra was responsible for designing of Lucene, TF-IDF retrieval models and evaluation script.

Gadhar, Shashikiran was responsible for designing the Pseudo relevance model for query expansion using BM25.

3. LITERATURE AND RESOURCES

The various tasks included in this course project are as follows:

3.1 Indexing and Retrieval

3.1.1 Task 1

Indexer - Indexer used here is a unigram indexer. Each file in corpus is parsed for its textual contents and each term in this content is regarded as a unigram. The Index thus created is an Inverted Index of the form "Term: {DOCID, TF}", where DOCID represents the name of the document in which the TERM appears and TF keeps a count of the number of times TERM has occurred in the corresponding document. Thus each term in the index will have several postings where each posting refer to the {DOCID, TF} pair.

BM25 – BM25 is a ranking algorithm that is an extended version of the binary independence model which includes query term weights to rank documents. This algorithm has performed and also influenced the commercial search engines. Our implementation of BM25 is based on the following formula:

$$\sum_{i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

Where, the final score is the summation over all the terms in the query, *N* is the total number of documents in the corpus, *ni* is the number of documents in which the query term *i* appears in, *R* is the number of relevant documents for this query, *ri* is the number of relevant documents containing the term *i*, *fi* is the frequency of the term *i* in the document, *qfi* is the frequency if the term *i* in the query and *k1*, *k2* and *K* are parameters whose values are set empirically.

tf-idf – Term Frequency – Inverse Document Frequency is a statistical feature that indicates the importance of a word to a document in the given corpus. This value increases as the occurrence of the term in the document increases. In our implementation of this task we compute *tf* as

$1 + \log f_{t,d}$ where, *ftd* is the frequency of the term the *t* in the document *d*

And idf as $\log \frac{N}{n_t}$ where, *N* is the total number of documents in the corpus and *nt* is the number of documents in the corpus that contain the term *t.*

Cosine Similarity – This is an implementation of the vector space model. In the general implementation of vector space model, the queries and documents are assumed to be a part of the *t*-dimensional vector space, where *t* is the number of index terms. In Cosine Similarity, a similarity measure is computed to rank documents so that top ranked documents are most similar to the query. This method measures the cosine of the angle between the query and the document vectors.

Lucene – A Java based text search library. Lucene provides functionalities to create index for a corpus, run queries on this index and also return results for the search. Lucene uses Inverted Index to store terms and the documents, thus avoiding the problem of searching the documents directly which is less efficient. A typical Lucene implementation includes field boosting, document boosting, etc.

3.1.2 Task 2

Pseudo relevance feedback – Relevance feedback is a feature where the results from running a query initially is used to provide a final result set. This involves the user to select the relevant documents. However, Pseudo relevance feedback is an automated version of relevance feedback where the top k documents obtained while running a query the first time is assumed to be relevant.

3.1.3 Task 3

Stopping –. In our implementation, we are provided with a list of common words that span over the entire corpus. We have considered these common words as stop words and thus we perform stopping on the queries. All queries before being processed undergo stopping. Thus a new query is formed in which there are no stop words, and now this query will be searched against the corpus. Performing stopping at query time will provide more flexibility than doing so at corpus indexing.

Stemming – The technique of reducing a word to its stem or root is called stemming. This decreases the index size. We have used BM25 that was implemented in the previous task with a corpus whose contents are stemmed. We also use a set of stemmed queries. Each word in the query is reduced to its stem. Our implementation has a predefined stemmed corpus and stemmed query set which we use on BM25 ranking algorithm.

3.2 Evaluation

3.2.1 Stopping

We have used the *tf-idf* based search engine to implement stopping. We have used the "common_words" file provided that contains a few of the commonly occurring words throughout the corpus.

A new query is formed by removing the stop words from the current query. This new query thus formed is free of stop words and is checked against the corpus. This way certain frequently occurring words that have little meaning on their own can be avoided and kept away from affecting the performance of a search engine.

3.2.2 Retrieval Effectiveness

Test Collection:

CACM: Titles and abstracts from the Communications of the ACM from 1958–1979. Queries and relevance judgments generated by computer scientists

Collection	Number of queries	Average number of words/query	Average number of relevant docs/query
CACM	64	13.0	16

MAP - The simplest way to summarize the effectiveness of rankings from multiple queries would be to average the average precision of these queries. The MAP measure provides a very succinct summary of the effectiveness of ranking algorithm over many queries.

Precision at K =5 and K=20 - This measure provides information on how well the search engine does at retrieving relevant documents at very high ranks

Mean reciprocal rank - reciprocal rank is defined as the reciprocal of the rank at which the first relevant document is retrieved. The mean reciprocal rank (MRR) is the average of the reciprocal ranks over a set of queries

4. IMPLEMENTATION AND DISCUSSION

4.1 BM25

The implementation of BM25 is straight forward. We have used the following formula to calculate the score for each document

$$\sum_{i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

As mentioned above, each value is query and document specific. We have assumed the value 1.2 for k1, 100 for k2 and 0.75 for b. These are the ideal values for these parameters that are suggested by CMS textbook . From these values it is possible to calculate the value of K as:

$$K = k_1((1 - b) + b \cdot \frac{dl}{avdl})$$

Here, *dl* is the length of the document being scored and *avdl* refers to the average length of the document. This *avdl* is the average of the document length of all the documents in the corpus. In our implementation, the values for r and R are computed every time. This is done by consulting a file called “cacm.rel” which contains the relevant documents for a particular query. If the query does not have any relevant documents then the values for R and r are taken to be 0.

4.2 Query Expansion

The technique of reformulating the initial query to better the performance of the search engine is called Query Expansion. This includes evaluating the user input and expanding the query to find additional documents that are relevant. We have used the BM25 ranking algorithm here and approached query expansion using pseudo relevance feedback. Pseudo relevance feedback is an automated version of the relevance feedback model because it assumes that the top k documents generated are relevant and does not require user

interaction. Our approach towards expanding a query to obtain better results follows the following algorithm

Step 1: Run initial query with BM25 as retrieval model.

Step 2: Consider top 50 documents as relevant documents.

Step 3: Generate snippet for these documents taking significant factor into account. We used Luhn's Law for generating snippets.

Step 4: Determine most frequent terms in the snippet generated for each.

Step 5: Remove stop words and choose top 30 terms from the snippets

Step 6: Add these terms to the original query to produce modified query.

Step 7: Rerun the search using modified query.

We implemented this process with top 10, 20, 30 and 50 terms (step 5). Since, the mean average precision was highest for top 30 terms, we implemented the pseudo-relevance with top 30 terms for the given corpus. We followed the same process with top 10, 20, 30, 50 and 100 documents (step 2).

4.3 Query by Query Analysis

Query - "operating system"

The first rank remains same for both the stemmed and non-stemmed results. For 2nd rank(CACM-1591) document for stemmed corpus the 'oper' term is matched 5 times which are actual terms operating and operation. They are of same context and hence stemming helped here. For the 3rd rank document (CACM-1680) for stemmed corpus this is same case where operations and operating is matched by 'oper', they are of same context.

But here the 3rd rank document retrieved from non-stemmed corpus seems to be more relevant since it takes only operating into consideration and it matches with phrase "operating system". All relevant documents for given query are fetched in top 50 ranks in non-stemmed version, whereas these documents are dispersed in stemmed version.

In Conclusion non-stemmed version is much more successful in retrieving relevant document in top ranks than stemmed version. Below are the relevant document positons for stemmed and non-stemmed results that prove this.

12 Q0 CACM-3127 1 17.023126367 BM25-SRA

1 Q0 CACM-3127 1 14.7752368657 BM25-Stemmed-SRA

12 Q0 CACM-2246 2 10.5453369746 BM25-SRA

1 Q0 CACM-2246 55 7.73707334579 BM25-Stemmed-SRA

12 Q0 CACM-2629 36 6.25684057824 BM25-SRA

1 Q0 CACM-2629 13 8.46984031592 BM25-Stemmed-SRA

12 Q0 CACM-2080 42 6.03070017943 BM25-SRA

1 Q0 CACM-2080 84 7.23035384824 BM25-Stemmed-SRA

Query2 - "Applied stochastic processes"

Below are the relevant documents fetched:

query id: 24 (non-stemmed version)

query id: 5 (stemmed version)

24 Q0 CACM-1892 14 2.72357710798 BM25-SRA

5 Q0 CACM-1892 37 2.03088970508 BM25-Stemmed-SRA

24 Q0 CACM-3078 28 1.99626179839 BM25-SRA

Top 3 documents are same for both versions. In this case the stemming has led to loss of a relevant document from the top 100 ranking document. Even in this cases,
non-stemmed version has performed better than stemmed version of the search engine

4.4 Snippet Generation – Extra Credit

For generating snippets, we followed the procedure mentioned in the book 'Search engines Information Retrieval in practice by W. Bruce Croft'. We took each sentence in the document and calculated the significant factor using Luhn’s Law. We considered word to be significant, if that word is present in the query. We took the top 6 sentences and presented it as snippet to the user. We have highlighted the significant words with RED font color.

5. RESULTS

Below are the results for queries 1 to 10, for all the 5 runs.

For query level results and rest of the run results, please refer Evaluation.xls

- BM25

Query-ID	RR	Precision	Recall	Avg-Precision	Precision@5	Precision@20
1	1	0.02	0.4	0.642857143	0.2	0.1
2	1	0.03	1	1	0.6	0.15
3	0.142857143	0.03	0.5	0.123376623	0	0.1
4	1	0.04	0.333333333	0.379462643	0.2	0.15
5	1	0.05	0.625	0.326130952	0.2	0.1
6	0.5	0.03	1	0.353333333	0.4	0.1
7	1	0.11	0.392857143	0.556895713	0.8	0.35
8	1	0.03	1	0.458937198	0.2	0.1
9	1	0.07	0.777777778	0.358168119	0.4	0.15
10	1	0.24	0.685714286	0.682629396	0.8	0.7

Mean Average Precision = 0.532078651861

Mean Reciprocal Rank = 0.805204517705

- TF-IDF

Query-ID	RR	Precision	Recall	Avg-Precision	Precision@5	Precision@20
1	0.5	0.04	0.8	0.369601889	0.4	0.15
2	0.5	0.03	1	0.212191358	0.2	0.05
3	0.05	0.03	0.5	0.059851552	0	0.05
4	0.142857143	0.03	0.25	0.124489796	0	0.1
5	0.5	0.05	0.625	0.194336633	0.2	0.15
6	0.333333333	0.03	1	0.157070707	0.2	0.05
7	1	0.1	0.357142857	0.650780052	1	0.3
8	0.5	0.03	1	0.291316527	0.2	0.1
9	0.2	0.08	0.888888889	0.212163836	0.2	0.2
10	1	0.21	0.6	0.538417945	0.6	0.55

Mean Average Precision = 0.343502198731

Mean Reciprocal Rank = 0.577960368309

- Cosine Similarity

Query-ID	RR	Precision	Recall	Avg-Precision	Precision@5	Precision@20
1	1	0.04	0.8	0.468181818	0.2	0.2
2	1	0.03	1	1	0.6	0.15
3	0.03030303	0.03	0.5	0.048989899	0	0
4	0.5	0.05	0.416666667	0.236989796	0.4	0.1
5	0.5	0.05	0.625	0.249465338	0.4	0.15
6	0.5	0.03	1	0.221668512	0.2	0.05
7	1	0.13	0.464285714	0.588887964	0.8	0.45
8	0.5	0.03	1	0.246503497	0.2	0.1
9	0.166666667	0.09	1	0.170392814	0	0.15
10	1	0.21	0.6	0.520880636	0.6	0.45

Mean Average Precision = 0.383189585745

Mean Reciprocal Rank = 0.62855976606

- Lucene

Query-ID	RR	Precision	Recall	Avg-Precision	Precision@5	Precision@20
1	0.5	0.03	0.6	0.288359788	0.2	0.1
2	0.076923077	0.01	0.333333333	0.076923077	0	0.05
3	0.037037037	0.03	0.5	0.054889539	0	0
4	0.25	0.01	0.083333333	0.25	0.2	0.05
5	0.5	0.07	0.875	0.178205698	0.2	0.15
6	0.25	0.03	1	0.147397397	0.2	0.1
7	1	0.11	0.392857143	0.690794465	0.8	0.4
8	1	0.03	1	0.445520581	0.2	0.1
9	0.25	0.08	0.888888889	0.235620366	0.4	0.2
10	1	0.23	0.657142857	0.593650945	0.8	0.55

Mean Average Precision = 0.400350055305

Mean Reciprocal Rank = 0.66409322234

- Query Expansion

Query-ID	RR	Precision	Recall	Avg-Precision	Precision@5	Precision@20
1	1	0.04	0.8	0.456168831	0.4	0.1
2	1	0.03	1	1	0.6	0.15
3	1	0.04	0.666666667	0.394117647	0.2	0.2
4	1	0.09	0.75	0.355099343	0.4	0.2
5	1	0.07	0.875	0.499090758	0.6	0.25
6	1	0.03	1	0.743589744	0.4	0.15
7	1	0.17	0.607142857	0.677711012	1	0.6
8	1	0.03	1	0.703703704	0.4	0.1
9	1	0.09	1	0.474953078	0.4	0.35
10	1	0.25	0.714285714	0.633221615	0.8	0.65

Mean Average Precision = 0.543927844372

Mean Reciprocal Rank = 0.903991841492

6. CONCLUSIONS AND OUTLOOK

For the given collection, following was the observation.

- BM25 retrieval model was most effective compared to other implemented models.
- The pseudo relevance query expansion boosted the effectiveness of BM25, Since the BM25 has better precision at top ranking, it was right combination with pseudo relevance query expansion.
- Stopping had negative impact on effectiveness of BM25, but stopping boosted the effectiveness of Tf-idf retrieval system.

Below are the some of the methods for improving this project

- We can incorporate query logs and user feedback in the project. Using query logs and user feedback we can improve the effectiveness of retrieval systems.
- Use of more general test collection, like TREC to better evaluate the effectiveness of retrieval models.
- Use of better Stop word list suitable to specific retrieval model through empirical analysis.

7. BIBLIOGRAPHY

- https://en.wikipedia.org/wiki/Query_expansion
- https://en.wikipedia.org/wiki/Stemming
- https://en.wikipedia.org/wiki/Relevance_feedback
- https://en.wikipedia.org/wiki/Stop_words
- http://nlp.stanford.edu/IR-book/html/htmledition/pseudo-relevance-feedback-1.html
- “Search Engine Information Retrieval in Practice” by W. Bruce Croft, Donald Metzler, Trevor Strohman.
- “An Introduction to Information Retrieval” by Christopher D Manning, Prabhakar Raghavan, Hinrich Schutze.
- https://www.youtube.com/watch?v=XFIKE34HafY
- https://www.youtube.com/watch?v=9Cvtu9wmrDg