Search Engine Design
CS6200: Information Retrieval
Northeastern University Spring 2017, Prof. Nada Naji

Members:

1. Veera Venkata Sasanka Uppu
2. Yatish Kadam
3. Jayanth Gangadhar

4/20/2017

Contents

1. Introduction: 2

1.1 Overview: 2

1.2 Contribution of team members: 2

2. Literature and resources: 3

2.1 Overview: 3

3.Implementation and Discussion: 4

3.1 Baseline Runs: 4

3.1.1 tf-idf measure: 4

3.1.2 BM25 Model: 5

4. Query Expansion:…… 6

4.1 Query-by-query analysis: 6

4.2 Snippet Generation (Extra Credit): 7

4.3 T-Test(Extra Credit): 7

5. Results: 8

6. Conclusions and outlook: 8

6.1 Conclusions 8

6.2 Outlook 9

7. Bibliography: 9

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 three distinct retrieval models as mentioned below:

- ❖ BM25 retrieval model
- ❖ tf-idf retrieval model
- ❖ Lucene Default model

Along with these three baseline runs, we have two additional runs for:

- ❖ BM25 retrieval model along with Pseudo relevance feedback (query expansion technique)
- ❖ tf-idf retrieval model along with Pseudo relevance feedback (query expansion technique)

Two runs for stemming:

- ❖ BM25 retrieval model on the stemmed version of the corpus 'cacm_stem.txt' using 'cacm_stem.query'
- ❖ tf-idf retrieval model on the stemmed version of the corpus 'cacm_stem.txt' using 'cacm_stem.query'

Two runs for stopping:

- ❖ BM25 retrieval model on the stopped version of the corpus using 'common-words'
- ❖ tf-idf retrieval model on the stopped version of the corpus using 'common-words'

We assess the performance of these seven distinct runs (excluding stemming ones) in terms of the following 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:

Veera Venkata Sasanka Uppu was responsible for designing of BM25 retrieval model, Query expansions for Bm25, Tfidf models, corpus cleaning.

Yatish Kadam was responsible for designing of Lucene, Snippet generation, T-test, evaluation scores.

Jayanth Gangadhar was responsible for designing the TfIdf model and its stopping,stemming versions

2. Literature and resources:

2.1 Overview:

The overview of the techniques used for implementing various aspects of the project is outlined below:

- <u>tf-idf measure</u>: 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.
- <u>BM25 Model</u>: For BM25 model, we have used 'cacm.rel' file as the set of relevant documents. The values of 'K', 'k1' and 'k2' are chosen as per TREC standards.
- <u>Lucene</u>: We have used the standard Lucene open source library with minor modifications to perform indexing and search operations.
- <u>Query Expansion</u>: Pseudo relevance feedback –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.
- <u>Stopping</u>: The standard stop list – 'common_words.txt' has been used to perform stopping. Any word appearing in the above-mentioned stop list has not been indexed.
- <u>Stemming</u>: 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
- <u>Precision & Recall</u>: The formulae used for calculations of precision and recall are:
  - Precision = |Relevant ∩ Retrieved| / |Retrieved|
  - Recall = |Relevant ∩ Retrieved| / |Relevant|
- <u>MAP</u>: The formula used for calculation of Mean Average Precision is:
  - MAP = Σ Average Precision / Number of Queries
- <u>MRR</u>: 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.
- <u>P@K</u>: 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.

3.Implementation and Discussion:

The intricacies of the implementation techniques are detailed in this section. It also contains the query-by-query analysis comparing the results of stemmed and non-stemmed runs.

3.1 Baseline Runs:

3.1.1 tf-idf measure:

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF: Term Frequency**,
  TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
- **IDF: Inverse Document Frequency**,
  IDF(t) = 1+ log(Total number of documents) / Number of documents with term t in it + 1.

<u>Steps to calculate tf-idf</u>:

A. For each query in ‘query.txt’ file, perform the following steps:
   a) For each term in the query, calculate tf*idf scores for all the documents in the corpus containing the term.
   b) Calculate the total tf*idf scores of documents for all the terms in the query.
   c) Sort the documents as per their scores in decreasing order.
   d) Print the top 100 documents in ‘tfidf_output.txt’ file of ‘doc_score’ directory in the below format:
      QueryID 'Q0' DocID Rank tf_idf_score 'System_Name'

3.1.2 BM25 Model:

BM25 ! Popular and effective ranking algorithm based on binary independence model – adds document and query term weights

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

– k1, k2 and K are parameters whose values are set empirically
– dl is doc length
– Typical TREC value for k1 is 1.2, k2 varies from 0 to 1000, b = 0.7

- ri is the # of relevant documents containing term i (set to 0 if no relevancy info is known)
- ni is the # of docs containing term i
- N is the total # of docs in the collection
- R is the number of relevant documents for this query (set to 0 if no relevancy info is known)
- fi is the frequency of term i in the doc under consideration
- qfi is the frequency of term i in the query
- k1 determines how the tf component of the term weight changes as fi increases. (if 0, then tf component is ignored.) Typical value for TREC is 1.2; so fi is very non-linear (similar to the use of log f in term wts of the vector space model) --- after 3 or 4 occurrences of a term, additional occurrences will have little impact.
- k2 has a similar role for the query term weights. Typical values (see slide) make the equation less sensitive to k2 than k1 because query term frequencies are much lower and less variable than doc term frequencies.
- K is more complicated. Its role is basically to normalize the tf component by document length.
- b regulates the impact of length normalization. (0 means none; 1 is full normalization.)

❖ Sort the documents as per their scores in decreasing order.
❖ Print the top 100 documents in ‘BM25_Output.txt’ file of ‘doc_score’ directory in the below format:
QueryID 'Q0' DocID Rank BM25_Score 'System_Name'

4. Query Expansion

The technique of reformulating the seed query to improve the performance of the search engine is called Query Expansion. This query expansion involves evaluating a user's input (what words were typed into the search query area, and sometimes other types of data) and expanding the search query to match additional documents. We have used the BM25 and tf-idf ranking algorithms 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

1. Run initial query with the retrieval model.
2. Consider around top 50 documents as relevant documents.
3. Generate snippet for these documents taking significant factor into account. We used Luhn's Law for generating snippets.
4. Determine most frequent terms in the snippet generated for each.
5. Remove stop words and choose upto top 30 terms from the snippets
6. Add these terms to the original query to produce modified query.
7. Rerun the search using modified query.

The above steps has been performed on both BM25 and the tf-idf models.

4.1 Query-by-query analysis:

For this task we have chosen the following queries:

- Parallel algorithms
- Portable operating systems
- code optimization for space efficiency

The query "Parallel algorithms" after stemming becomes – "parallel algorithm". For this query, we can see that the stemmed version and non-stemmed version are almost same except for 'algorithms' which gets stemmed to 'algorithm'.

The query "Portable operating systems" after stemming becomes – "portabl oper system". The query is stemmed to a great extent the query word operating gets stemmed to "oper".

The query "code optimization for space efficiency" after stemming becomes –"code optim for space effici". This query also is stemmed to a great extent. The words optimization and efficiency are stemmed to "optim" and "efficiency".

In case of the stemmed run, all terms belonging to the same stem class are getting replaced by the stem.

For query "code optimization for space efficiency" we see that the doc CACM-2897 is the only doc present in the top 15 ranks for the two Outputs for the BM25 models. On the other hand, in case of the non-stemmed run, each distinct term is considered. For example, the term 'optimization' is different from 'optima', 'efficiency' is different from 'effici' etc. As both the queries and documents are not stemmed, each word (even though they might belong to the same stem class) will have different weightage.

For the first query we find that only few docs are present in both top 20 ranks. Although the stemmed version is not that far from the non-stemmed, this happen because stemming has a very little impact on the query terms. In this case the stemming has led to loss of a relevant document from the top 100 ranking document. Even in this case, non-stemmed version has performed better than stemmed version of the search engine.

4.2 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 upto top 6 sentences and have output it as snippet to the user with the query words present in Red color Bold.

4.3 T-Test (Extra Credit):

The t-test is implemented using the formulae

$$t = \frac{\overline{B-A}}{\sigma_{B-A}}.\sqrt{N}$$

where B−A is the mean of the differences, σB−A is the standard deviation of the differences, and N is the size of the sample (the number of queries)

σ is calculated as:

$$\sqrt{\sum_{i=1}^{N}(x_i-\overline{x})^2/N},$$

The t-test enables us to reject the null hypothesis and conclude that ranking algorithm B is more effective than A. As seen when running the t-test file given. We see that the BM25 model is more effective than the tf-idf model.

5. Results:

Below are MAR and MAP values obtained in different runs for BM25 algorithm:

	BM25	BM25 with stopping	BM25 with query expansion
MAP	0.537939381631	0.538353924502	0.59423078818
MRR	0.8209249084 2	0.842902930403	0.868131868132

Below are MAR and MAP values obtained in different runs for TfIdf algorithm:

	TfIdf	TfIdf with stopping	TfIdf with query expansion
MAP	0.335548079794	0.411604014537	0.308328834662
MRR	0.5687910903	0.679395604396	0.497669299411

Below are MAR and MAP values obtained in different runs for Lucene algorithm:

	Lucene
MAP	0.411670863189
MRR	0.675527803714

As we can see Stopping boosts the search engine scores. Query Expansion does boost BM25 but in case of TfIdf however it has shown a slight decrease in the overall scores.

6. CONCLUSIONS AND OUTLOOK

6.1 CONCLUSIONS

For the given collection, following was the observation.

- BM25 retrieval model was most effective compared to other implemented models.
  MAP= 0.53793938 | MRR= 0.82092491
- Even the base run of BM25 yielded good results in terms of MAP, MRR and was more effective than Lucene and tf-idf measures.
- 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 feedback query expansion. Whereas, Tf-Idf had a slight negative effect when used with pseudo relevance feedback query expansion
- Stopping had small boost on effectiveness of both BM25 and Tf-Idf.

6.2. OUTLOOK

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.
- Alongside considering topical features of a document, we can take into consideration their quality features like ‘incoming links’, ‘update count’ etc. as a part of the final ranking. Using ‘PageRank’ algorithm to calculate the popularity of a page is also a viable method which can be incorporated in our search engines.
- 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/Stemming
- https://en.wikipedia.org/wiki/Query_expansion
- 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.
- https://www.youtube.com/watch?v=9Cvtu9wmrDg