PROJECT REPORT FOR SEARCH ENGINE

CS6200: Information Retrieval
Northeatern University
Fall 2017, Prof. Nada Naji

TEAM MEMBERS

Aparna Sharma
Megha Suvarna
Mrunal Ghanwat

Table of Contents

1 Introduction

1.1 Overview

This project is on the implementation of information retrieval systems. This system uses multi retrieval models to retrieve the relevant documents based on the query. The search engine also uses query expansion techniques and snippet generation keeping user's experience in mind.

Information retrieval comprises of two main phases, indexing and matching. Additionally, query expansion techniques and snippet generation techniques can be used to increase the efficiency and make search easier.

The indexing step pre-processes documents and queries to obtain keywords to be used in the query. It is important to consider the use of stemming and stop-word lists to reduce related words to their stem, base or root form.

Matching is the process of computing the similarity between documents and queries by weighting terms, the most frequently applied algorithms being the BM25, Lucene, TF-IDF, Query Likelihood algorithms. Retrieval systems return the ranked documents based on the relevancy of the query to the document.

Inverted indexes are created on different sets of corpuses. Corpuses are cleaned to handle punctuations, case-folding and stopping. Above retrieval models are then executed on different result set of inverted indexes. Evaluation of the top results are performed using various parameters such as MAP, MRR, P@K, Precision and Recall.

Query Expansion uses pseudo-relevance technique to improve the efficiency of the retrieval. The most occurring keywords of the top documents previously retrieved is added to the query to re-rank the documents.

1.2 Contribution of team members

The detailed contribution of each team member is explained below:

- *Aparna Sharma:* Aparna was responsible for implementing the query expansion technique using Pseudo relevance feedback. She was also responsible for the implementation of stemming and stopping techniques during the cleaned corpus generation. Additionally, implemented various evaluation measures like MAP, MRR, P@K, Precision and Recall for the 8 distinct runs and contributed in documentation.
- *Megha Suvarna:* Megha was responsible for implementing Tf-Idf measure along with the cosine-similarity and integration of other 3 models. Additionally, she was responsible for query-by-query-analysis for stemmed and non-stemmed runs. She was also responsible for the documentation of the project.
- *Mrunal Ghanwat:* Mrunal was responsible for implementing Query-likelihood model for retrieving the top 100 relevant documents. She was also responsible for implementing snippet generation, query term highlighting and contributed in documentation of her part.

2 Implementation and discussion

2.1 BM25 Model

- Below formula is used to calculate score for each document:

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

    - r_i number of relevant documents containing term i
    - n_i number of documents containing the term i
    - f_i term frequency in the document
    - qf_i frequency of the term i in the document
    - k_1 = 1.2
    - k_2 = 100
    - B = 0.75
    - R number of relevant documents in the query
    - N total number of documents in the collection
    - K = k_1((1-b) + b * dl/avdl)
        - dl -> document length
        - avdl ->average length of document in the collection.
- Sort the documents in descending order as per their BM25 score.
- Write the top 100 documents to a text file.

2.2 TF-IDF Model

Formula to calculate:

- *tf*: total no. of times the term occurring in a document / total no. of terms in the document
- *idf*: 1 + log_e(Total no. of documents / No. of documents with term in it)For each term in the document, its tf-idf is calculated using the above formula.
- For each term in the query, its tf-idf is calculated using the above formula.
- Calculate the cosine-similarity score using tf-idfs of terms in the query and document.
- Sort the documents in descending order as per their score.
- Write the top 100 documents to a text file.

2.3 Lucene

Lucene uses in-built libraries to index and search documents for information retrieval. Indexing involves adding the Documents to an IndexWriter, and searching involves retrieving documents from an index via an Index searcher.

Fields: A document consists of one or more fields. This field is a name-value pair. A common field can be title. In the case of a title Field, the field name is title and value is the title of the content. Indexing involves creating documents comprising one or more fields and adding these documents into an IndexWriter.

Ranking: Query is created using query parser and is handed to the created indexer, which will return the top ranked document based on its score.

Write the top 100 documents to a text file.

2.4 Smooth Query Likelihood Model

In the query likelihood retrieval model, documents are ranked by the probability that the query text could be generated by the document language model.

The retrieval model specifies ranking documents by P(Q|D), calculated using the unigram language model for the document.

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Where qi is a query word, and there are n words in the query and D is the document length.

The major problem with this estimate is that if any of the query words are missing from the document, the score given by the query likelihood model for P(Q|D) will be zero.

To Remove Zero probability problem, “Jelinek-Mercer smoothing” technique is used to calculate probability estimate.

The collection language model probability estimates we use for word qi is cqi/|C|, where cqi is the number of times a query word occurs in the collection of documents, and |C| is the total number of word occurrences in the collection. This gives us an estimate for P(qi|D) of:

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

Each document is scored using the Jelinek-Mercer smoothing for a given query.

The documents are ranked is descendant order based on score.

Top 100 documents are retrieved.

2.5 Pesudo Relevance Feedback

For performing pseudo relevance feedback, perform the following steps:

- Normal retrieval is performed with initial query.
- The top k (in our case k = 10) from the results are included in the relevant set of documents and the rest of the documents are non-relevant.
- Select top 20 words (2 words from each document with highest tf-idf values which are not stop words).
- Add these terms to the initial query and re-issue the query.

Finally return the most relevant documents.

2.6 Query-by-query Analysis

Below are the 3 queries for which the analysis is performed:

1) portabl oper system (non-stemmed version: portable operating systems)
2) parallel algorithm (non-stemmed version: parallel algorithms)
3) appli stochast process (non-stemmed version: applied stochastic processes)

For the first query all the three terms have been changed after stemming. For the second query term 'algorithms' is stemmed to 'algorithm'. And for the last query all the three terms are stemmed, 'applied'-> 'appli', 'stochastic'-> 'stochast', 'processes'-> 'process'. Terms in the query having the same stem class gets replaced by the stem.

We will be using top 10 documents retrieved by the BM25, Lucene and Query likelihood models for stemmed data.

- For the first query, all three systems have '3127' as the topmost document.
  Top two documents are same for BM25 and Lucene, but the ranking is dropped in query likelihood model. Scores in BM25 is dropped sharply when compared to other two models. Analysing the top 10 documents in each system, it can be inferred that BM25 and Lucene has performed better than query likelihood system.
- For the second query, there are 8 common documents between Lucene and BM25 models but with different order. The score drop in all the documents are almost minor in all the systems. If a term is missing in the document, then the score will be dropped to a certain level in query likelihood model. Lucene and BM25 has outperformed Query likelihood model as the documents retrieved by this is not much relevant.
- For the third query, there are 8 common documents between Lucene and BM25 models, and 2 common documents in all three models, but with different order. Lucene and BM25 has outperformed Query likelihood model as the documents retrieved by this is not much relevant.

3 Literature and Resources

3.1 Overview

Below techniques are used for implementing various aspects of the project:

- *BM25 Model:* BM25 score is calculated for each document. Values for K1 = 1.2, b = 0.75 and K2 = 100.
- *Lucene:* Standard Lucene Open Source library is used to perform indexing and matching.
- *Tf-Idf Model:* TF-IDF for each term in a document is calculated by multiplying the normalised frequency of each term and its inverse document frequency. Scoring of the document is calculated using Cosine similarity measure.

  Formula:

$$\text{Cosine Similarity (Query, Document 1)} = \frac{\text{Dot Product (Query, Document 1)}}{||\text{Query}|| * ||\text{Document 1}||}$$

- *Query-likelihood Model:* Query-likelihood Model to estimate P(Q/D) can be computed as:

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

- *Stopping:* Words appearing in the 'common_word.txt' are considered as stop words. These stop words are not indexed.
- *Stemming:* Stemming is performed on corpus 'cacm_stem.txt' and indexed. Results for the stemmed queries from 'cacm_stem.query.txt' is retrieved using BM25, Query-Likelihood and.
- *Query Expansion*: Query expansion is performed using Pseudo Relevance Feedback technique.
- *Snippet Generation:* Once results are retrieved and returned for a given query, a snippet-Good text summery is created for each returning document using Lucene retrieval model to improve user's understanding of the document, and possibly increase the click-through ratio and enhance their interaction with the search engine.
- *MAP:* The formula used for Mean Average Precision is:
  - MAP = ∑Average Precision / Number of Queries
- *MRR:* Mean Reciprocal Rank is the average of the reciprocal ranks over a set of queries. Reciprocal Rank is the reciprocal of a rank at which first relevant document is retrieved.
- *P@K:* Number of relevant documents in the top K retrieved documents. This is calculated for K = 5 and K= 20.
- *Precision:* The formula used to calculate Precision is
  - Precision = | Relevant ∩ Retrieved| / |Retrieved|
- *Recall:* The formula used to calculate Recall is
  - Recall = | Relevant ∩ Retrieved| / | Relevant |

3.2 Third-party tools

The following third-party tools have been used in moderation whenever in need:

- Beautiful Soup: Beautiful soup is used to parse documents and queries.
  - jsoup-VERSION.jar is used for this purpose.
- Lucene Open Source Libraries: Following .jar files have been used:
  - Lucene-core-VERSION.jar
  - Lucene-queryparser-VERSION.jar
  - Lucene-analyzers-common-VERSION.jar

4 Results

4.1 Baseline Runs

BM25BasicOutput.txt   LuceneBasicOutput.txt   tfIdfRanking.txt   QuerylikelihoodBasicOutput.txt

4.2 Query refinement run

BM25PseudoOutpu
t.txt

4.3 Stopping runs

BM25StoppedOutp ut.txt	LuceneStoppedOut put.txt	QuerylikelihoodSto ppedOutput.txt

4.4 Stemmed runs

BM25StemmedOut put.txt	LuceneStemmedOu tput.txt	QuerylikelihoodSte mmedOutput.txt

5 Conclusions and outlook

5.1 Conclusions

Overall findings and conclusions from Eight distinct baseline runs and applying various evaluation measures on their results are outlined as below:

- The best results were obtained from the retrieval model of Lucene with the evaluation measures as MAP: 0.4263 and MRR: 6975 which has produced more relevant documents than any other retrieval model. These results might have been obtained as the fact Lucene incorporates both Boolean retrieval along Vector space model.
- The BM25 runs with Query expansion technique and Pseudo Relevance Feedback seemed to produce similar MRR and MAP evaluation results as BM25 Baseline runs. BM25 produces more good results after Lucene.
- Query-likelihood model produces more significant results than tf-idf. Use of smoothing technique Jelinek-Mercer smoothing produces slightly better results for query-likelihood retrieval.
- Tf-idf retrieval has produced least results for all evaluation measures. Even for the first 5 relevant documents MAP and MRR values of tf-idf varies between 0.000 to 0.003 producing 0 relevant documents retrieved in many cases of queries. Thus tf-idf cannot be considered as good retrieval model.

5.2 Outlook

- Our project incorporates all the basic features of an information retrieval system. However, certain features and functionalities as mentioned below can be included in future to improve the performance of the search engines:
- Alongside considering topical features of a document, we can take into consideration their quality features like ‘incoming links’ ‘update count’ as a part of final ranking using page-rank algorithm to calculate the popularity of a page is also good way to include in this search engine.

- As corpus grows, it will be prudent to use different compression technique like 'Elias-y encoding', 'v-byte encoding' to store the inverted index.
- The primary goal of this IR system is to achieve retrieval effectiveness. In future we can also consider retrieval efficiency into consideration.

6 Bibliography

6.1 Books

- Search Engines: Information Retrieval in Practice
  Book by Donald Metzler, Trevor Strohman, and W. Bruce Croft

6.2 Journals

- https://www.hindawi.com/journals/tswj/2014/132158/
- https://nlp.stanford.edu/IR-book/html/htmledition/using-query-likelihood-language-models-in-ir-1.html

6.3 Websites

- https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
- http://www.lucenetutorial.com/basic-concepts.html
- https://wiki.apache.org/lucene-java/LuceneFAQ
- https://en.wikipedia.org/wiki/Relevance_feedback