Search Engine Design

CS6200: Information Retrieval

Northeastern University

Fall 2016, Prof. Nada Naji

Team Members:

Partha Sarathi Jena

Rahul Pyne

Tridiv Nandi

Contents

1. Introduction: 3
  1.1 Overview: 3
  1.2 Contribution of team members: 3
2. Literature and resources: 4
  2.1 Overview: 4
  2.2 Third Party Tools: 4
  2.3 Research Article References: 5
3.Implementation and Discussion: 5
  3.1 Baseline Runs: 5
    3.1.1 tf-idf measure: 5
    3.1.2 Cosine Similarity measure: 5
    3.1.3 BM25 Model: 6
  3.2 Pseudo Relevance Feedback 7
  3.3 Query-by-query analysis: 7
4. Results: 8
5. Conclusions and outlook: 9
  5.1 Conclusions 9
  5.2 Outlook 9
6. Bibliography: 10
  6.1 Books: 10
  6.2 Scholarly articles: 10
  6.3 Websites: 10

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 four distinct retrieval models as mentioned below:

- BM25 retrieval model
- tf-idf retrieval model
- Cosine Similarity retrieval model
- Lucene Systems

Along with these four baseline runs, we have three additional runs:

- *BM25* retrieval model along with *Pseudo relevance feedback* query expansion technique
- *BM25* retrieval model along with *Stopping* technique
- *Cosine Similarity* retrieval model along with *Stopping* technique

Now we assess the performance of these seven distinct runs 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

- Moreover, we perform another run using *BM25* retrieval model on the stemmed version of the corpus *'cacm_stem.txt'* using *'cacm_stem.query'*.

1.2 Contribution of team members:

The detailed contribution of each individual member is elucidated below:

- Partha Sarathi Jena: Partha was responsible for implementing the query expansion technique using *Pseudo relevance feedback*. Additionally, he has developed the search engines which incorporate the stemming and stopping techniques. He was also responsible for the corresponding documentation and query-by-query analysis for stemmed and non-stemmed runs.
- Rahul Pyne: Rahul was responsible for implementing the four baseline runs using tf-idf measure, BM25 Model, Cosine Similarity Model and Lucene Systems. He was also responsible for documenting his design choices and developing the Snippet generation logic.
- Tridiv Nandi: Tridiv was responsible for implementing the various evaluation measures like MAP, MRR, P@K, Precision & Recall for the seven distinct runs. Moreover, he made notable contributions in documenting the conclusions and outlooks from the findings, observations and analyses of the results.

2. Literature and resources:

2.1 Overview:

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

- tf-idf measure: For calculating tf-idf for each document, we are using the normalized value of term frequency for the document multiplied by its inverse document frequency.
- Cosine Similarity measure: For calculating cosine similarity, we have used tf*idf as the term weights for the query and document terms.
- BM25 Model: 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.
- Lucene: We have used the standard Lucene open source library with minor modifications to perform indexing and search operations.
- Query Expansion: The query expansion for our project has been done using *Pseudo Relevance Feedback* technique and *Rocchio* algorithm.
- Stopping: 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.
- Stemming: We have performed stemming using the BM25 model with the help of the stemmed version of the corpus '*cacm_stem.query*' and query '*cacm_stem.query*'.
- Precision & Recall: The formulae used for calculations of precision and recall are:
    - Precision = |Relevant ∩ Retrieved| / |Retrieved|
    - Recall = |Relevant ∩ Retrieved| / |Relevant|
- MAP: The formula used for calculation of 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: The technique used by us to generate the snippets is called *Query biased document snippets methodology*. It uses the individual scores of each document for a given search engine model and creates, either trigrams, bigrams or unigrams of the query terms to get context in the snippet from the given document. The snippets are displayed in the console with query terms being highlighted using *Colorama* with blue background and yellow foreground.

2.2 Third Party Tools:

The following third party tools have been used in moderation as and when required:

- Beautiful SOUP: Beautiful SOUP has been used for parsing purpose to process both the documents in the corpus and the queries.
- Colorama: Colorama has been used in Snippet Generation to highlight the query terms in a snippet when displayed in a terminal.
- Lucene Libraries: We need the following referenced libraries for Lucene:
    - lucene-core-VERSION.jar
    - lucene-queryparser-VERSION.jar
    - lucene-analyzers-common-VERSION.jar

2.3 Research Article References:

The following articles were consulted for the various evaluation measures:

- http://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf
- https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/10-EvaluationMetrics.pdf

The following article was consulted for Query Expansion Technique using Pseudo Relevance Feedback:

- http://nlp.stanford.edu/IR-book/pdf/09expand.pdf

The following article was consulted for designing the four baseline runs:

- http://nlp.stanford.edu/IR-book/essir2011/pdf/11prob.pdf

The following article was consulted for designing the snippet generation:

- http://dl.acm.org/citation.cfm?id=1376651

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:

For tf-idf measure, the formulae used are given below:

tf = no. of times a term occurs in a document/ document length

idf = 1 + log (Total no. of documents) / (number of documents in which term appears) + 1

- For inverse document frequency calculation, we are adding 1 to the denominator to prevent it from becoming 0 when the term does not appear in any document. We are also adding 1 to the log values to prevent entire idf value from becoming 0.

<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_doc_score.txt*' file of '*doc_score*' directory in the below format:
       QueryID 'Q0' DocID Rank tf_idf_score ' tf_idf_Model'

3.1.2 Cosine Similarity measure:

In our vector space model, documents and queries are assumed to be a part of a *t-dimensional vector space* where *t* is the number of index terms(unigrams).

- A Document D_1 is represented by a vector of index terms:
  $D_1 = (d_{11}, d_{12}, d_{13}, ..., d_{1n})$ where d_{11}, d_{12}... etc. are the weights of the terms
- Queries are also represented the same way as documents. For example, a query Q is represented by a vector of *t* weights:
  $Q = (q_1, q_2, q_3, ..., q_t)$
  If a term does not appear in the query, its weight is replaced by 0.
- Among the many different weighting schemes available, we have used tf*idf as the term weights for the query and the document terms.
- The tf component reflects the importance of a term in a document D or query and the idf component reflects the importance of the term in the collection of documents.
- We have also normalized the term frequencies as in reality each document will be of a different size.

Steps to calculate cosine similarity:

A. For each query in '*query.txt*' file, perform the following steps:
   a. For each unique query term, retrieve all the inverted lists corresponding to that term from the inverted index
   b. Calculate tf*idf scores for all the documents in the retrieved inverted lists and all the unique terms in the query.
   c. Calculate cosine similarity scores for all the documents in the retrieved inverted lists and the query using the following formulae:
      Cosine Similarity (Q, D) = Dot Product (Q, D) / |Q| * |D|
      |D| -> Magnitude of Query vector
      |Q| -> Magnitude of Document vector
   d. Sort the documents as per their cosine similarity scores in decreasing order.
   e. Print the top 100 documents in '*VSM_doc_score.txt*' file of '*doc_score*' directory in the below format:
      QueryID 'Q0' DocID Rank tf_idf_score ' Vector_Space_Model'

3.1.3 BM25 Model:

In our BM25 Model, to calculate the scores of documents use the following scoring function:

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

- The summation is done over all the query terms.
- r_i -> number of relevant documents containing the term i
- n_i -> number of documents containing the term i
- N -> total number of documents in the collection
- R -> number of relevant documents for the query
- f_i -> frequency of the term i in the document
- qf_i -> frequency of the term i in the query
- k_1 -> the value is taken as 1.2 as per TREC standards
- k_2 -> the value is taken as 100 as per TREC standards
- K -> $K = k_1((1 - b) + b * dl / avdl)$ where dl is document length, avdl is average length of document in the collection.
- The value of b is taken as 0.75 as per TREC standards.

- Sort the documents as per their scores in decreasing order.

- Print the top 100 documents in '*BM25_doc_score.txt*' file of '*doc_score*' directory in the below format:

  QueryID 'Q0' DocID Rank tf_idf_score ' BM25_Model'

3.2 Pseudo Relevance Feedback

For performing pseudo relevance feedback using *Rocchio* Algorithm, perform the following steps:

A. Normal retrieval is performed with the initial query
B. The top k documents (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 set.
C. Rocchio algorithm is used to generate the new query.
    a. Initial query vector with the tf scores and aligned with the inverted index terms is generated.
    b. Relevant set of document vector is generated.
    c. Non-relevant set of documents vector is generated.
    d. Then modified query is generated by following the formula:
    e. The algorithm proposes using the modified query qm:
       $qm = \alpha q0 + \beta / |Dr| \sum dj - \gamma / |Dnr| \sum dj$
       Where q0 is the original query vector, Dr and Dnr are the set of known relevant and non-relevant documents respectively, and α, β, and γ are weights attached to each term.

       For our project, we have chosen $\alpha = 1.0$, $\beta = 0.75$, and $\gamma = 0.15$
D. The top 20 terms with the highest weight and which are not present in the query are appended to the original query.
E. Normal retrieval is performed with the new query and the search results are generated.

3.3 Query-by-query analysis:

The two queries that we have chosen are

- Parallel algorithms / parallel algorithm
- Performance evaluation and modelling of computer systems / perform evalu and model of comput system

For the first query, we can see that the stemmed version and non-stemmed version are almost same except for *'algorithms'* getting stemmed to *'algorithm'*. However, we can see that for the second query, five terms ('*Performance'-> 'perform'; 'evaluation'* -> *'evalu'; 'modelling'-> 'model'; 'computer*' -> '*comput*'; *'systems'* -> *'system'*) have changed after stemming.

In case of the stemmed run, all terms belonging to the same stem class are getting replaced by the stem. For example, the terms '*Performance*', '*performing*', '*performer*', '*performed'* etc. will be replaced by the stem '*perform'*. Similarly, the terms '*computer*', '*computing*', '*compute*', '*computed*' etc. will be replaced by '*comput*'. So, the documents having any of the terms that will get transformed to the same stem by the stemming algorithm will have high scores.

On the other hand, in case of the non-stemmed run, each distinct term is considered. For example, the term '*Performance'* is different from '*perform*', '*computer*' is different from '*compute*' 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.

In case of the first query, if we consider the top 20 documents retrieved by the BM25 model during the stemmed and non-stemmed runs, we will find that there are eleven common documents *('CACM-0950',* *‘CACM-1262’, ‘CACM-2714’, ‘CACM-2700’, ‘CACM-2266’, ‘CACM- 2685*’ etc.). This happen because stemming has a very little impact on the query terms.

In case of the second query, if we consider the top documents retrieved by them during the stemmed and non-stemmed runs, we will find that there are only six common documents (*‘CACM- 3048’, ‘CACM- 3070’, ‘CACM-2988’* etc.). This happen because stemming has considerable impact on the query terms.

4. Results:

Given below are the results obtained for different runs:

- Base Runs:

  BM25_runs.xlsx tfidf_runs.xlsx VSM_runs.xlsx Lucene_runs.xlsx

- Query Expansion Run:

  BM25_pseudo_rel_feedback_runs.xlsx

- Stopping Technique Run:

  BM25_stoplist_runs.xlsx

- 7th Run (Cosine Similarity with stopping):

  VSM_stoplist_runs.xlsx

- Stemming Technique Run:

  BM25_stemmed.xlsx

- Comparison of MRR(s) and MAP(s):

Comparison of MRRs.txt

Comparison of MAPS.txt

5. Conclusions and outlook:

5.1 Conclusions:

Our overall findings and conclusions from the eight assigned runs and after applying various evaluation measures on their result set are outlined below:

- The best results were obtained from the *BM25* run which incorporated *Query Expansion* technique. As per its *MAP* scores (*0.479730025988*) and *MRR* scores (*0.699869791667*), this run was undisputedly better than all other retrieval models.
- Even the base run of *BM25* yielded good results in terms of MAP, MRR and was more effective than Lucene, Cosine Similarity and tf-idf measures.
- However, *BM25* along with stop lists caused a slight decrease in the values of effectiveness measures. 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.
- tf-idf 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. So, ideally tf-idf does not serve as a good measure for retrieval model.
- Lucene provides slightly better MAP and MRR values than cosine similarity measures. This observation might be attributed to the fact that it incorporates both Boolean model along with Vector Space 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 hone 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’* 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.
- As the corpus grows, it will be prudent to use different compression techniques like ‘*Elias-γ encoding*’, ‘*v-byte encoding*’ to store the inverted index.
- The primary motivation of our IR system was to achieve retrieval effectiveness. Going forward, we can also take *retrieval efficiency* into consideration.

6. Bibliography:

6.1 Books:

- Croft, W.Bruce; Metzler, Donald; Strohman, Trevor *Search Engines: Information Retrieval in Practice.* Pearson Education 2015
- Manning, Christopher D; Raghavan, Prabhakar; Schutze Hinrich *An Introduction to Information Retrieval.* Cambridge England: Cambridge University Press 2009

6.2 Scholarly articles:

- http://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf
- https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/10-EvaluationMetrics.pdf
- http://nlp.stanford.edu/IR-book/pdf/09expand.pdf
- http://nlp.stanford.edu/IR-book/essir2011/pdf/11prob.pdf
- http://dl.acm.org/citation.cfm?id=1376651

6.3 Websites:

- https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
- https://www.youtube.com/watch?v=hiiB9LKwfVg