Manthan Thakker
Vineet Trivedi

Information Retrieval Fall 2016
Professor Nada Naji

i. INTRODUCTION

For our project, we have developed a Search Engine, which has the following 3 modules:

1. Cleansing Module:
    a. Remove punctuations and other unwanted characters
    b. Stemming
    c. Stopping
2. Indexer
3. Retrieval
    a. BM25 with Pseudo Relevance Feedback
    b. BM25 without Pseudo Relevance Feedback
    c. Lucene Based Retrieval Model
    d. Cosine-Similarity
    e. tf-idf

The user can run the CACM queries

When the user chooses to run the CACM queries, the aforementioned 7 options are displayed:

1. BM25 with Pseudo Relevance Feedback and Stopping
2. BM25 with Pseudo Relevance Feedback
3. BM25 with Stopping
4. BM25
5. Cosine Similarity
6. tf-idf

The user then chooses one of the options and the following metrics are displayed:

1. Average Precision for each of the 64 queries
2. Precision at 5 for each of the 64 queries
3. Precision at 20 for each of the 64 queries
4. Mean Average Precision for the 64 queries

Manthan Thakker
Implemented:

1. BM25
2. Cleansing Module:
   a. Stemming

Documented:

1. readme.txt
2. First page: Project members' names, course name and semester
3. Introduction: Short description of the project, detailed description of each member's contribution to the project and its documentation.

Vineet Trivedi
Implemented:

1. Pseudo Relevance Feedback
2. Cleansing Module:
   a. Stopping

Documented:

1. Literature and resources: overview of the techniques used, scholarly work and research articles to back the techniques and algorithm choices, resources, third party tools that were used and referred to in the project.
2. Bibliography: citations and links to resources

The following modules were already implemented from previous assignments:

1. Cleansing Module:
   a. Remove punctuations and other unwanted characters
2. Indexer
3. Lucene Based Retrieval Module
4. Cosine-Similarity
5. tf-idf

The following parts of the documentation were discussed and documented together by both teammates:

1. Implementation and discussion
2. Results
3. Conclusions and outlook

ii. LITERATURE AND RESOURCES:

Overview of the Techniques Used (Query Expansion Approaches)

Pseudo Relevance Feedback:
Step 1: Retrieve documents using the underlying retrieval module, in this case BM25.
Step 2: Take the top 10 retrieved documents.
Step 3: For each of these 10 documents, take the top 100 most frequent words, excluding stop-words.
Step 4: Add these terms to the query to get the expanded query.
Step 5: Run the underlying retrieval module again but this time with the expanded query.

Pseudo Relevance Feedback works under the assumption that the underlying Retrieval Model will retrieve relevant documents. Thus it is only as good as the underlying Retrieval Model.

Note: Different variations of Pseudo Relevance Feedback may use more than or less than '10 documents' or '100 most frequent words, excluding stop-words'. However after evaluating the Precision, Reciprocal Rank and Mean Average Precision of several experimental runs we empirically derived the aforementioned values.

Design Choices, Scholarly Work and Research Articles to back Technique and Algorithm Choices:

Algorithm Choice: Use of Pseudo Relevance FeedBack Algorithm for Query Expansion
1. Does not require any dependencies, thesaurus-based requires a thesaurus (example wordnet) and an additional library to access the thesaurus.

2. Its performance is better and more stable

Hersh W, Price S, Donohoe L. Assessing thesaurus-based query expansion using the UMLS Metathesaurus. Proc AMIA Symp; 2000. pp. 344–348. [1]
Link:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2244120/pdf/procamiasymp00003-0379.pdf

Technique Choice: Number of retrieved documents to be used in Query Expansion – 10.
This was decided after
Analyzing results of the following research paper:

T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 162–169, 2006.[2]
Link:
http://delivery.acm.org/10.1145/1150000/1148201/p162-tao.pdf?ip=129.10.110.11&id=1148201&acc=ACTIVE%20SERVICE&key=AA86BE8B6928DDC7%2EC2B8A117C7A71F5A%2E4D4702B0C3E38B35%2E4D4702B0C3E38

B35&CFID=873531861&CFTOKEN=28413710&__acm__=1481220754_15874f2f7761a5a1cae23265d45d5937

where performance degraded as more documents were included.

AND

Performed ran test runs to confirm the results of the paper for our cacm corpus.

Technique Choice: Number of words to expand the query – 100 per document (for each of the 10 documents).
This was decided after analyzing the tables and results of the aforementioned paper. We compared the size of the new terms added to the size of the existing query and looked at the results obtained. We then ran test runs on the cacm dataset.

Algorithm Choice: BM25 as the Retrieval Model for run 5, 6 and 7
Pseudo Relevance Feedback depends on the efficiency of the underlying Retrieval Model. After obtaining performance metrics for all baseline runs we found that BM25 gives the most relevant results.

The above design choice was backed by the following papers:
I. Cantador, A. Bellogín, D. Vallet. Content-based recommendation in social tagging systems. Proceedings of the fourth ACM conference on Recommender systems (RecSys'10), ACM, New York, USA (2010), pp. 237–240[3]
Link:
http://delivery.acm.org/10.1145/1870000/1864756/p237-cantador.pdf?ip=129.10.110.11&id=1864756&acc=ACTIVE%20SERVICE&key=AA86BE8B6928DDC7%2EC2B8A117C7A71F5A%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=873531861&CFTOKEN=28413710&__acm__=1481220427_be7f2748b0da08730699ea4952b78be2

Turtle, H.R. and Croft, W.B. A comparison of text retrieval models. C0mput. J. 35, 3 (1992), 279-290. [4]
Link:
http://comjnl.oxfordjournals.org/content/35/3/279.full.pdf+html

Shaw Jr, W. M.; Burgin, R.; and Howell, P. 1997. Performance standards and evaluations in IR test collections: Cluster-based retrieval methods. Information recessing and management 33:1- 14, 1997. [5]
Link:
http://www.sciencedirect.com/science/article/pii/S030645739600043X
N. Fuhr and C. Buckley. Probabilistic document indexing from relevance feed-back data. In Proceedings of the 13th International Conference on Research and Development in Information Retrieval, edited J.-L. Vidick, pp. 45-61 (1990) [6]
Link:

http://delivery.acm.org/10.1145/100000/98008/p45-fuhr.pdf?ip=129.10.110.11&id=98008&acc=ACTIVE%20SERVICE&key=AA86BE8B6928DDC7%2EC2B8A117C7A71F5A%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=873531861&CFTOKEN=28413710&__acm__=1481220181_cabe8bd17040c9e9bc9c6cf5f21fd91b

J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In Proceedings of SIGIR 2008, pages 347–354. [7]
Link:
http://delivery.acm.org/10.1145/1400000/1390394/p347-elsas.pdf?ip=129.10.110.11&id=1390394&acc=ACTIVE%20SERVICE&key=AA86BE8B6928DDC7%2EC2B8A117C7A71F5A%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=873531861&CFTOKEN=28413710&__acm__=1481219909_62b838e47cc4450ed0af907c7befc7e0

Resources and Third Party Tools used:

1. Jsoup.jar – Used for parsing documents in the corpus.
2. Eclipse – IDE
3. NetBeans – IDE

iii. IMPLEMENTATION AND DISCUSSION

Cleansing:

All punctuations from the CACM corpus and queries had to be removed. Exceptions:

1. Hyphens were not removed.
2. Hyphens and Dots were not removed if they appeared between digits.

A combination of regex and character by character reading was utilized to achieve this.

Stopping:

The list of stop-words provided in common_words.txt had to be removed from the CACM corpus and queries.

Stemming:

The stemmed corpus had to be cleaned and split into files. A file was created for each document so that the indexer code for the un-stemmed corpus could be reused.

Indexing:

The CACM corpus was indexed. An inverted index was created for each unique word.
Inverted Index Format:
Term = {DocID1=tf1; DocID2=tf2}

Where: tf1 is the term frequency of the term in the document with DocID1.
Data Structure used to store this information: HashMap_A <String, HashMap_B>
where HashMap_B is HashMap_B <String, Long>

HashMap was used because:

1. It is extensible
2. It stores data as Key-Value pairs, so if an inverted list of a particular term had to be retrieved only the term(Key) had to be given and the inverted list(Value) was directly retrieved.

The inverted index was stored to disk as a serializable file as opposed to a text file so that it could easily be read from disk whenever required without having to worry about parsing issues.

Retrieval Model:

BM25

To perform BM25 we had to compute the following for each query term.

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

where:
r_i = Total number of relevant documents in which the term i was present
R = Total number of relevant documents
n_i = Total number of documents in which the term i was present

N = Total number of documents in the corpus
qf_i = frequency of term i in the query
f_i = frequency of term i in the document

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

where:
b, k_1 and k_2 are empirically derived for each corpus
dl = length of document
avdl = average length of a document in the corpus

Value of b=0.75, k1=1.2 and k=100 as per 'Bruce Croft, Donald Metzler, Trevor Strohman, Search Engines: Information Retrieval in Practice, Addison-Wesley Publishing Company, 2015.'[8]

The relevant documents for each query provided in the file 'common_words.txt' were checked against each retrieved document, for each query, to determine if the retrieved document is relevant.

The above parameters are then computed.
Finally, the score for each document for the given query is computed and stored in a HashMap<String, Double>.
The HashMap is sorted in descending order of its values.
The results are displayed.

Cosine Similarity
To perform cosine similarity the weight of each query term and document term had to be calculated.
Weight vector for Document-1:
$\text{Document-1}_{weight} = [\text{Term-1}_{weight}\ \text{Term-2}_{wieght} \ldots\ \text{Term-n}_{weight}]$

where:
{Term-1 Term-2 … Term-n} are the terms in Document-1
$\text{Term-1}_{weight} = tf_1 * idf_1$

where:
tf_1= (term frequency of Term-1 in Document-1) / (Sum of term frequencies of all terms in Document-1)
idf_1= $\log_{10}(N/n)$

where:
idf_1 is Inverse Document Frequency of Term-1
N is the total number of documents in the corpus
n is the number of documents in which the given term has appeared

The weight vector for each document is computed and stored in:
$HashMap_A$ <String, $HashMap_B$>
where $HashMap_B$ is $HashMap_B$ <String, Double>

The same is done for the query.

Once we have the document and query vectors the cosine similarity score for each (Query, Document) pair is computed as:
co_sim = ($Query_{weight}$ * $Document_{weight}$) / (|$Query^2_{weight}$| * |$Document^2_{weight}$|)

The scores are stored in a HashMap<String, Double>.
The HashMap is sorted in descending order of its values.
The results are then displayed.

tf-idf
In this method, the tf-idf score for each query term is computed for each document. The score of a document is equal to the sum of the tf-idf scores of all the query terms present in the document. The tf-idf score is computed as mentioned in Cosine Similarity above.

The scores are stored in a HashMap <String, Double>.
The HashMap is sorted in descending order of its values.
The results are then displayed.

Lucene-Based Retrieval Model
The Lucene-Based Retrieval Model code from HW-4 was used. The CACM corpus was given as the input.

<u>Query Expansion:</u>
Pseudo Relevance Feedback
Explained above in section iii overview of the techniques used (Query Expansion Approaches)

<u>Snippet Generation:</u>
Takes input as a query and a document.
Parses the document sentence by sentence.
Identifies significant words (in this case query terms) and computes the score of the snippet between the significant words (including both the significant words) as:

[s w s w w s w s s]

(Total number of significant words)^2 / (Total number of words)
$5^2/9 = 25/9$.
where:
s = significant word
w = non-significant word.

The score of a sentence is the maximum score any snippet contained within it. The sentences are then arranged in descending order of their scores and the top 3 snippets are displayed.

<u>Query by Query Analysis:</u>
Stemmed Query-1 and Un-Stemmed Query-12

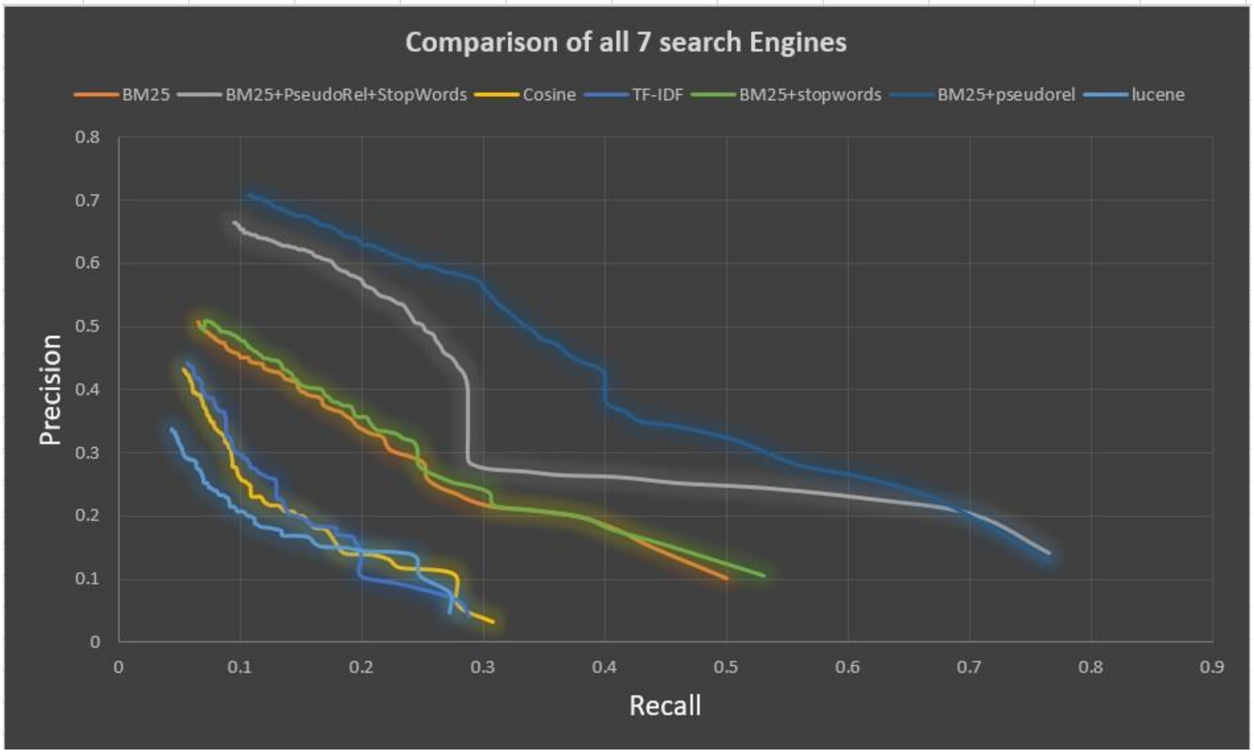
1. The scores decline more gradually as compared to the un-stemmed query scores in which there is sudden dip from 23(Value at Rank 1) to 12(Value at Rank 2).
2. The overall quality of results is mostly same however the un-stemmed version performs slightly better.

Stemmed Query-2 and Un-Stemmed Query-13

1. The quality of documents retrieved is more or less the same.

Note: In most cases the document retrieved at Rank 1 for the un-stemmed version is the same as the document retrieved at Rank 2 for the stemmed version.

iv. RESULTS



METRICS FOR EACH RUN

SR NO	SEARCH ENGINE TYPE	MAP	MRR
1	BM25+PseudoRel+StopWords	0.5012	0.5777
2	BM25+PseudoRel	0.600981814	0.60465
3	BM25+StopWords	0.326	0.5604
4	BM25	0.312	0.322
5	Lucene	0.152148958	0.5046
6	Cosine	0.139	0.143
7	TF-IDF	0.14	0.15

v. CONCLUSIONS AND OUTLOOK:

Conclusions

Retrieval Models Ranking

1. BM25 with Pseudo Relevance Feedback
2. BM25 with Pseudo Relevance Feedback using stopping
3. BM25 with stopping
4. BM25
5. Lucene Based Retrieval Model
6. Cosine-Similarity
7. tf-idf

We initially started the baseline runs with tf-idf which gave us inferior quality results. We then moved onto the vector space model, the Cosine-Similarity Model, which is more sensitive towards the terms in the documents as opposed to only the document length(t-idf). This produced better results. The Lucene-Based Retrieval Model uses several other parameters to refine its search and in turn returns better results as compared to Cosine-Similarity or tf-idf model. When we implemented the probabilistic model, BM25 we got the best results. It takes into account the relevant documents for a given query and is in-turn able to score the documents more accurately. When stopping was applied to this most of the noise from documents got cancelled out and the results got further refined. Pseudo Relevance Feedback when applied to the already well performing BM25 was able to thrive on it and return the best results amongst all the runs. BM25 with Pseudo Relevance Feedback and stopping retrieved similar although slightly inferior results.

Outlook

The project can be improved in the following manner:

1. Use query logs to give alternative query suggestions which can yield better results.
2. Use query logs to determine frequently asked queries. Their results can be cached and reused to significantly reduce response time.
3. Include user behavior and statistics to personalize the search and retrieve results that are relevant to the specific user.

vi. BIBLIOGRAPHY:

[1] Hersh W, Price S, Donohoe L. Assessing thesaurus-based query expansion using the UMLS Metathesaurus. Proc AMIA Symp; 2000. pp. 344–348.

[2] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In SIGIR ’06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 162–169, 2006.

[3] I. Cantador, A. Bellogín, D. Vallet. Content-based recommendation in social tagging systems. Proceedings of the fourth ACM conference on Recommender systems (RecSys’10), ACM, New York, USA (2010), pp. 237–240

[4] Turtle, H.R. and Croft, W.B. A comparison of text retrieval models. C0mput. J. 35, 3 (1992), 279-290.

[5] Shaw Jr, W. M.; Burgin, R.; and Howell, P. 1997. Performance standards and evaluations in IR test collections: Cluster-based retrieval methods. Information recessing and management 33:1- 14, 1997.

[6] N. Fuhr and C. Buckley. Probabilistic document indexing from relevance feed-back data. In Proceedings of the 13th International Conference on Research and Development in Information Retrieval, edited J.-L. Vidick, pp. 45-61 (1990)

[7] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In Proceedings of SIGIR 2008, pages 347–354.

[8] Bruce Croft, Donald Metzler, Trevor Strohman, Search Engines: Information Retrieval in Practice, Addison-Wesley Publishing Company, 2015.