PROJECT REPORT

CS6200 - INFORMATION RETRIEVAL SPRING 2019

SUBMITTED BY:

Ketan Kale, Computer Science, Sem 3
Abhiruchi Karwa, Computer Science, Sem 3
Jay Doshi, Computer Science, Sem 3

INSTRUCTOR:

Rukmini Vijaykumar

INTRODUCTION:

As a part of this project, we have built various information retrieval systems, varying in implementation by the techniques used for query enhancement and for preprocessing the corpus. We have displayed snippets so that they describe the search results in a better manner. Additionally, we have evaluated these retrieval systems based on the results.

PHASE 1:

TASK 1:

In Phase 1, we first parsed and indexed the corpus. No stopping and stemming were done initially, and unigram index was built in the following format:

[TERM]: [(docId1: freq) (docId2: freq) …….]

We implemented 4 baseline runs to rank the documents for all given queries using these models:

- BM25
- Tf-Idf
- JM Smoothed Query Likelihood Model
- Lucene's default retrieval model

Task 2:

Implemented query enhancement using the BM25 run as a baseline.

We used the following approaches:

- Pseudo relevance feedback:
- Semantic Query Expansion using Thesauri/ontologies:
    - For this technique we made use of Wordnet - A large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Task 3:

For task 3, we picked the following three baseline runs:

1. BM25
2. Lucene
3. JM QLM

We performed stopping and then ranked the documents for all the queries using the above techniques. Then, we indexed the stemmed version of the corpus cacm_stem.txt and retrieved the results for queries in cacm_stem.query.

PHASE 2:

Phase 2 involved displaying the snippets generated using H.P. Luhn's approach for results. The query terms were displayed in bold.

PHASE 3:

The following evaluation measures were calculated to determine the performance of different information retrieval models:

a) Mean Average Precision
b) Mean Reciprocal Rank
c) P@K (K=5, K=20)
d) Precision and Recall values tables for all queries for every document for all runs

TEAM'S CONTRIBUTION:

Jay Doshi:

1. Implemented tf-idf
2. Implemented BM25
3. Implemented Lucene
4. Implemented query enhancement using Semantic Query Expansion using Thesauri/ontologies
5. Documented the entire project

Ketan Kale:

1. Created basic parser and indexer
2. Implemented Snippet generation for phase 2
3. Implemented the extra credit retrieval systems
4. Implemented evaluation in phase 3
5. Implemented query enrichment using pseudo relevance feedback

Abhiruchi Karwa:

1. Implemented JM Smoothed Query Likelihood Model
2. Implemented baseline runs after stopping and stemming
3. Implemented the query correction model
4. Implemented Snippet generation for phase 2
5. Performed additional run in Phase 3

LITERATURE AND RESOURCES

- Snippet Generation:

We used H.P. Luhn's approach to rank each sentence in a document by a *significance factor* and then select the top sentence for the summary. The query is split into query terms, called significant terms. Additionally, stopwords are removed from the query because frequency of stopwords in a text will be high and will be of no use to the user. For each sentence, text spans are found which are series of words that contain both significant and nonsignificant terms. There is a limit set, on how many non-significant terms can appear consecutively in a text span, generally a window of 4 is used. On this window of the sentence, Luhn's score called *significance factor* is calculated based on occurrences of significant words. Words occurring in the query are also significant words. Significance factor for these text spans is calculated by the formula:

significance factor: (no. of significant terms / total number of terms)

Above formula is applied on all text spans in a document and then top scoring sentence is selected as snippet.

- Query Enhancement:

For Query enrichment, Pseudo Relevance Feedback (PRF) is performed by query expansion. PRF is a query expansion technique in which, the system assumes that top ranked documents are relevant and then the words that occur frequently in these documents are used to expand the initial query. To weigh the term similarity that appeared in top ranked documents to the query, we used Dice's coefficients.

The top 10 documents ranked using BM25 are assumed as the set of relevant documents. Dice's coefficients is calculated for all the terms in all documents. Top words with higher Dice's coefficient are selected from these documents. We then automatically reformulate the query by adding these terms, and a new ranking is generated using this expanded query.

$$n_{ab}/(n_a + n_b)$$

Dice's Coefficient

n_{ab} - Denotes number of documents containing a and b

n_a - Denotes number of documents containing a

n_b - Denotes number of documents containing b

Both the above techniques have been studied from our textbook "Information Retrieval in Practice" by W. Bruce Croft, Donald Metzler, and Trevor Strohman

IMPLEMENTATION AND DISCUSSION:

TF-IDF Algorithm:

It is a ranking algorithm which gives scores documents based on the frequency of the query words in the document

tf - Term frequency is the number of times the term is present in the given document.
idf - Is the number of documents that contains the given query term. It is calculated by adding 1 to the log of total number of documents in the collection divided by 1 + number of documents in which term appears. 1 is added to prevent it from becoming zero.

$$tf - idf(t, d, D) = tf(t, d) * idf(t, d, D)$$

A document has:

- high score when it contains the term in high frequency and term has low frequency across the collection
- moderate score when the term frequency is moderate and higher frequency across the collection.
- low score when the term frequency is low and the document frequency is high.

BM25 Algorithm:

It is a ranking algorithm that takes into consideration the relevance information and determines the rank for every document that contains at least one term of the query.

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

r_i – no. relevant documents containing the term i
n_i - number of documents containing the term i
N - total number of documents in the corpus
R - total 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
$k1$ - determines how the tf component of the term weight changes as f_i increases
$k2$ - determines the same thing for the query
b - constant, normalizes the document length.

JM Smoothed Query Likelihood Model:

We rank documents by probability that the query text could be generated by the document language model. We calculate the probability of pulling the query terms out of the *bucket* of words in the document. We rank documents based on the probability that query text could be generated by the document language model. We use Bayes Rule and calculate P(D|Q) as,

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

If there are no query term present in the document then the score for that document will be zero, so by using Jelinek-Mercer smoothing,

$$p(q_i|D) = (1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}$$

Thus,

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

Where,

f_{q_i}, D - number of times word q_i occurs in document D

$|D|$ - number of words in D

C_{q_i} - number of times word q_i occurs in the document collection

$|C|$ - total number of word occurrences in the collection

λ - constant, here its value is 0.75 as the queries are longer

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$
$$= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})}{\lambda\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$\stackrel{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|}}{\lambda\frac{c_{q_i}}{|C|}} + 1\right)$$

The top 100 documents are displayed as result

EXTRA CREDIT:

We have used edit distance to calculate the numerical disparity between a query term not in the index(which may be misspelled) with all the index terms. Using the distance and the probability of occurrence of each word in the index, we rank each index term and display up to 6 possible corrections. We have only considered words at an edit distance of 1 and 2

EVALUATION:

Evaluation is instrumental in the process of measuring the efficiency and effectiveness of an Information retrieval system.

We have used the following measures to accurately determine the efficiency of our Information Retrieval System.

1) MAP:
2) MRR:
3) P@K (K= 5 and K=20)
4) Precision and Recall values tables for all queries for every document for all runs

PRECISION:

Precision is an evaluation measure that measures the exactness of the retrieval process (among documents retrieved, how many documents are relevant).

$$Precision = \frac{|Relevant \cap Retrieved|}{|Retrieved|}$$

RECALL:

Recall is an evaluation measure that measures the completeness of the information retrieval process (among documents retrieved, how much of the relevant set of documents are retrieved).

$$Recall = \frac{|Relevant \cap Retrieved|}{|Relevant|}$$

MAP:

Mean Average Precision is the mean of all average precision values of all the queries.
Average Precision is the sum of precision values when a relevant document is found to the total number of relevant documents for that query.

MRR:

Mean Reciprocal Rank is the mean of all reciprocal ranks for all the queries i.e. the reciprocal of the rank at which the first relevant document is found.

P@K (K=5 and K=20):

P@K is a performance measure to calculate the precision after k results. Here, K=5 and K=20, therefore, we measure system performance by evaluating precision of the 5th and 20th document.

Result obtained for calculating precision:

Precision at Rank	5	20

BM25	0.3653846153846153	0.20673076923076922
BM25 with query enrichment - PRF	0.35769230769230764	0.20480769230769238
BM25 with query enrichment - Thesauri	0.3461538461538461	0.21153846153846162
JM QLM	0.16153846153846158	0.10576923076923075
BM25 using Stopping	0.3653846153846155	0.2288461538461538
JM QLM using Stopping	0.22307692307692314	0.13749999999999998
TF IDF	0.1730769230769231	0.12884615384615383
Lucene	0.33846153846153837	0.20961538461538465
Additional run	0.34230769230769226	0.20000000000000007

Query-by-query Analysis:

First Query:
queryId 1: Stemmed query: “portabl oper system”

BM25 Top 3 documents:
1 Q0 CACM-3127.html 1 14.800182169576432 BM25
1 Q0 CACM-2246.html 2 10.775892919304699 BM25
1 Q0 CACM-1930.html 3 8.956750790227282 BM25
1 Q0 CACM-3196.html 4 8.685342607747508 BM25
1 Q0 CACM-2593.html 5 5.647028206240972 BM25

Lucene Top 3 documents:
1 Q0 cacm-3127.txt 1 1.5456363 lucene
1 Q0 cacm-2246.txt 2 0.7645695 lucene
1 Q0 cacm-3196.txt 3 0.6305411 lucene
1 Q0 cacm-2593.txt 4 0.37098807 lucene
1 Q0 cacm-1930.txt 5 0.33201674 lucene

Among the top documents retrieved by Lucene and BM25, documents 3127, and 2246 are relevant topically. Both the models produce almost similar results for this query. All these documents have high frequencies of the terms “portabl”, “oper”, and “system”.

Second query:
queryId 2: Stemmed query: “code optim for space effici”

BM25 Top 5 documents:

2 Q0 CACM-2748.html 1 11.797168521044295 BM25
2 Q0 CACM-2530.html 2 11.75463708658821 BM25
2 Q0 CACM-2491.html 3 11.59889769783222 BM25
2 Q0 CACM-1947.html 4 10.971037438678582 BM25
2 Q0 CACM-2559.html 5 10.145390063377466 BM25

Lucene Top 5 documents:
2 Q0 cacm-2491.txt 1 0.8643284 lucene
2 Q0 cacm-2748.txt 2 0.7635965 lucene
2 Q0 cacm-2530.txt 3 0.7332481 lucene
2 Q0 cacm-2559.txt 4 0.6133859 lucene
2 Q0 cacm-1947.txt 5 0.59545404 lucene

All these documents have high frequencies of the terms “code”, “optim”, “space”, and “effici”. Although BM25 and Lucene retrieve pretty much the same documents (with different rankings), BM25 produces results that are more topically relevant whereas for Lucene some documents do not seem to be topically relevant. With respect to content, BM25 provides better results for this query.

Third query:

queryId 6: Stemmed query: “perform evalu and model of comput system”

BM25 Top 5 documents:
6 Q0 CACM-2318.html 1 17.029971119264363 BM25
6 Q0 CACM-3048.html 2 13.374285490360535 BM25
6 Q0 CACM-3089.html 3 11.594526231471365 BM25
6 Q0 CACM-3070.html 4 11.594332554762609 BM25
6 Q0 CACM-3119.html 5 11.424610626110226 BM25

Lucene Top 5 documents:
6 Q0 cacm-2318.txt 1 1.9860966 lucene
6 Q0 cacm-3048.txt 2 1.2255543 lucene
6 Q0 cacm-3070.txt 3 1.1999689 lucene
6 Q0 cacm-2741.txt 4 0.99705493 lucene
6 Q0 cacm-2319.txt 5 0.98211074 lucene

BM25 and Lucene retrieve pretty much the same documents (with different rankings), but again BM25 produces results more relevant to the query, as for Lucene some documents seem to be topically not so relevant.

RESULTS:

The results for all the metrics can be found in the following folders:

- Phase 1
    1. BM25: src/phase1/task1/BM25_results
    2. JM QLM: src/phase1/task1/JM_QLM_results
    3. Lucene: src/phase1/task1/lucene_results
    4. Tf-Idf: src/phase1/task1/TF_IDF_results
    5. Query Expansion using Thesaurus: src/phase1/task2/Query Expansion/query_expanded_thesaurus.txt
    6. Query Expansion using Pseudo Relevance: src/phase1/task2/Query Expansion/terms_for_expansion.txt
    7. BM25 using expanded query terms from PRF: src/phase1/task2/BM_Results_after_qe_PRF
    8. BM25 using expanded query terms from thesauri src/phase1/task2/BM_Results_after_qe_thesauri
    9. Lucene with stemming: src/phase1/task3/Stemmed_Lucene_results
    10. BM25 with stemming: src/phase1/task3/Stemmed_BM25_results
    11. JM QLM with stopping: src/phase1/task3/Stopped_JM_QLM_results
    12. BM25 with stopping: src/phase1/task3/Stopped_BM25_results
- Phase 2
    1. Snippet generation: src/phase2/snippet_results/
- Phase 3
    1. Evaluations – BM25: src/phase3/evaluations/BM25_results_evaluations.txt
    2. Evaluations – BM25 after query expansion using PRF: src/phase3/evaluations/BM25_Results_after_qe_PRF_evaluations.txt
    3. Evaluations – BM25 after query expansion using thesauri: src/phase3/evaluations/BM25_Results_after_qe_thesauri_evaluations.txt
    4. Evaluations – JM QLM: src/phase3/evaluations/JM_QLM_results_evaluations.txt
    5. Evaluations – BM25 after stopping: src/phase3/evaluations/Stopped_BM25_results_evaluations.txt
    6. Evaluations – JM QLM after stopping: src/phase3/evaluations/Stopped_JM_QLM_results_evaluations.txt
    7. Evaluations – TF-IDF: src/phase3/evaluations/TF_IDF_results_evaluations.txt
    8. Evaluations – Lucene: src/phase3/evaluations/lucene_results_evaluations.txt
    9. Evaluations – Additional Run using BM25 with stopping and query expansion: src/phase3/evaluations/Additional_results_evaluations.txt
    10. Additional Results: src/phase3/Additional_results/

- All query-level results i.e., MAP, MRR can be found at the bottom of query-64 results, they are also shown in the table:

	BM25	BM25 with query enrichment using PRF	BM25 with query enrichment using Thesauri	JM QLM	BM25 using Stopping	JM QLM using Stopping	TF IDF	Lucene	Additional Run
MAP	0.44379	0.399917	0.449030	0.21911	0.4644815	0.308677	0.2274711	0.383858	0.3990708
MRR	0.71061	0.661436	0.680393	0.37213	0.7142017	0.5261327	0.3629142	0.638687	0.6628262

CONCLUSIONS AND OUTLOOK:

CONCLUSION:

- On average, TF-IDF returned the more irrelevant documents as compared to the other models
- JM QLM model returned the more relevant documents at the top of the search results.
- For query enrichment, we chose PRF and Thesauri. But we observed that the top ranked documents for either of them did not improve the relevance significantly.
- Since PRF assumes that the top 'k' documents we chose are all relevant to the information need of the user. However, there are many documents that cover various topics within themselves. Thus, if PRF decides to choose such document for generating expansion terms the resulting terms might have very less relevance to the actual information need of the user. On the other hand, if there is a document that is solely focused on one topic which is the exactly relevant to the information need, then PRF will pick that document for generating expansion terms which indeed may result in improving the rank of that document and identifying more relevant documents to the information need.

OUTLOOK:

1) Incorporating Relevance feedback into the system to ensure that user gets relevant results. We can ask the user to provide feedback and then produce better ranking based on that.
2) We can perform clustering to present all documents that appear to be on the same topic in one cluster, rather than presenting all documents separately.
3) We can also incorporate searching in multiple languages where the user enters the queries in languages other than English.

BIBLIOGRAPHY:

- https://nlp.stanford.edu/IR-book/html/htmledition/pseudo-relevance-feedback-1.html
- https://nlp.stanford.edu/IR-book/html/htmledition/using-query-likelihood-languagemodels-in-ir-1.html
- https://en.wikipedia.org/wiki/Okapi_BM25
- https://www.crummy.com/software/BeautifulSoup/
- https://pythonprogramming.net/wordnet-nltk-tutorial/
- http://wordnetweb.princeton.edu/perl/webwn