INFORMATION RETRIEVAL (CS6200)

SPRING - 2018

PROJECT REPORT

Team members:

1) Mohan Pothukuchi
2) Varsha Muroor Rao

Instructor:

Prof. Nada Naji

Introduction

Throughout the Spring 2018 semester, various core concepts and processes in the field of information retrieval were introduced. As a part of the coursework, these concepts have been implemented in this project. The main aim of the project is to build search engines using different retrieval models and compare their effectiveness.

For the given dataset, the retrieval system is created starting from cleaning the corpus using various text transformation methods, building index, implementing the retrieval models, namely BM25, Smoothed Query Likelihood Model, tf-idf, Lucene as well as calculating the effectiveness of the search results obtained by using each of these models.

The third party libraries like BeautifulSoup for parsing the raw html documents and Lucene are used. All of the implementation is done in python programming language using PyCharm IDE, except for Lucene IR system which is implemented using Java. The output of the individual runs are stored in text files for analysis and evaluation.

The members of this team project are Mohan Pothukuchi and Varsha Muroor Rao. Mohan was involved in implementing the Smoothed Query Likelihood Model, query enrichment using pseudo relevance feedback, soft-matching query handler and the effectiveness evaluation phase. Varsha was involved in implementing the tf-idf model, stopping and stemming runs, snippet generation and query term highlighting, synthetic spelling error generator and documentation.

Literature and Resources

The following techniques are used to perform query enrichment and snippet generation.

1) To perform Query enrichment, pseudo relevance feedback is used.
   The top 10 documents obtained from the BM25 ranking (simple baseline run without stemming/stopping) are considered for query expansion. The frequency of every term in this set of top 10 documents is calculated. The stop words are filtered out of this list and the terms are ordered according to their frequencies. The top 10 highest frequency terms are used to expand the query.
   This technique is mentioned in the textbook 'Search Engines' by W. Bruce Croft, Donald Metzler, Trevor Strohman'.

2) To perform snippet generation and highlighting, the top 100 documents obtained from the BM25 ranking for each query is considered. Each document is broken down into text windows of 8 terms. Each window is assigned a score based on the occurrence of number of significant query terms. Significant terms are those that do not belong to the stop list. Based on this score, the top 3 text windows are selected to form the snippet and the significant query terms are highlighted. If there are no text windows with significant terms, the text windows that have more number of terms that are present in the query (including stop words) are chosen to form the snippet. The snippets so formed are parsed to check for the occurrence of query terms and such terms are highlighted using tags. The stop words are highlighted only if the snippet formed doesn't contain enough text windows with significant terms. This technique was inspired by the paper 'SnipIt - A real time Dynamic Programing approach to Snippet Generation for HTML Search Engines, Ian Carr and Lucy Vasserman'.
   The BM25 results obtained from the simple baseline run (without stemming/stopping) are used for the above task.

Resources:

1) Stanford.edu. *Pseudo Relevance Feedback*. Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/pseudo-relevance-feedback-1.html
2) Ian Carr and Lucy Vasserman. *SnipIt - A real time Dynamic Programing approach to Snippet Generation for HTML Search Engines*. Retrieved from http://www.cs.pomona.edu/~dkauchak/ir_project/whitepapers/Snippet-IL.pdf
3) W. Bruce Croft, Donald Metzler, Trevor Strohman. (2015) *Search Engines- Information Retrieval in Practice*. PearsonEducation, Inc.
4) Christopher D. Manning Prabhakar Raghavan, Hinrich Schütze. (2009) *An Introduction to Information Retrieval*. Cambridge University Press.

Implementation and Discussion

Input Files

The first step that was involved in the project was to setup the input files. Three different sets of corpus were created, one with only case folding as the text transformation, one with case folding and stopping, and one with case folding and stemming. The first two sets were created by parsing and tokenizing the files in cacm.tar folder and generating text files that contain only the plain textual content of the html files. The last set was generated using the cacm_stem.txt file by storing each document in a separate text file.

Secondly, the queries were extracted from the given cacm.query.txt file using xml parser and stored in a text file which was used as an input later while performing search using various retrieval models. Similarly, stopping was applied on the queries and a separate text file was generated which was an input for the baseline run on the corpus on which stopping was applied.

Retrieval Models

BM25 retrieval model:

The below formula was used to calculate the BM25 score for the documents that had query terms in them.

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

The value of K can be calculated using

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

where, N is the number of documents in the collection

n_i is the number of documents containing term i

f_i is the frequency of the term i in the document

qf_i is the frequency of term i in the query

r_i is the number of relevant documents containing term i

R is the number of relevant documents for the query

k_1 , k_2 , b , and K are parameters whose values are set empirically

dl is the length of the document

avdl is the average length of the document in the collection

The values chosen are k_1 =1.2 , k_2 = 100 , b = 0.75, R =r_i = 0

The documents are ranked in the non-increasing order of the score obtained.

tf.idf Retrieval model:

The term frequency of every term k in the document D_i is calculated using

$$tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$$

where f_{ik} is the number of occurrences of term k in the document

The inverse document frequency idf_k for term k can be calculated using the below formula, where N is the number of documents in the collection, and n_k is the number of documents in which term k occurs.

$$idf_k = \log \frac{N}{n_k}$$

The product of term frequency and inverse document frequency gives the tf.idf score for each term in the document. Every document is represented by a vector of tf.idf scores of index terms. A similar approach is used to represent the query as a vector, where every term in the query is assigned a tf.idf score.

The documents are then ranked by calculating the similarity measure between the document vector and the query vector using cosine similarity measure.

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$

Smoothed Query Likelihood model:

The probability of generating the query text from the document language model is calculated for every document using the formula:

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

where, c_{qi} is the number of times a query word occurs in the collection of documents,

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

$f_{qi,D}$ is the number of times word q_i occurs in document D,

$|D|$ is the number of words in D

λ Is a coefficient controlling the probability assigned to unseen words.

The value of λ is chosen to be 0.35

Based on the values obtained using the above formula, the documents are ranked.

Lucene IR system:

The third party library provided by apache Lucene are used to rank the documents for each query. This implementation is done in Java. Lucene uses a combination of tf.idf model, Boolean model and Vector Space model.

Effectiveness evaluation:

For all the runs that were obtained using the above described retrieval models, the Mean Average Precision, Mean Reciprocal Rank, Precision and recall tables were found out. Precision and recall at every rank for every query can be calculated using the below formulae:

$$Recall = \frac{|A \cap B|}{|A|}$$
$$Precision = \frac{|A \cap B|}{|B|}$$

where A is the relevant set of documents for the query, B is the set of retrieved documents, $A \cap B$ is the set of documents that are both relevant and retrieved.

The average precision for each query is calculated by averaging the precision values from the rank positions where relevant document was obtained. The mean average precision for every run is calculated by averaging the individual average precision values for each query.

For every run, the Mean Reciprocal Rank is calculated by averaging the reciprocal of ranks at which the first relevant document was retrieved for every query.

Design and technical choices:

Since the project was implemented using Python, it was very convenient to store the intermediate results in json files using the data structure dictionary.

For the snippet generation task, the documents were converted to lower case first so as to make the matching of query terms easier. Also, terms containing hyphen in them are considered as a single word and will be highlighted only if query contains the hyphenated word.

Query by query analysis for stemmed version of the corpus with stemmed queries

The top 10 results given by each model is sampled and compared.

1) portabl oper system

First 10 documents and scores given by BM25 model

Document ID	Rank	BM25 Score
CACM_3127	1	14.485361213978987
CACM_2246	2	10.494243414031136
CACM_1930	3	8.684875463471087
CACM_3196	4	8.47912635557923
CACM_3068	5	5.535635557529355
CACM_2593	6	5.50166923746054
CACM_2319	7	5.403346726025001
CACM_2740	8	5.335882089610984
CACM_2379	9	5.209859749372077
CACM_1591	10	5.165198818191222

First 10 documents and scores given by Smoothed Query Likelihood Model

Document ID	Rank	SQLM Score
CACM_3127	1	-11.57577815181015
CACM_2246	2	-15.136051953792228
CACM_3196	3	-15.29763444274979
CACM_1461	4	-17.222520872262095
CACM_1930	5	-17.267883307032697
CACM_2593	6	-17.492160326488495
CACM_3068	7	-17.66055952422734
CACM_2796	8	-17.828541773198676
CACM_2319	9	-17.922398627511072
CACM_1755	10	-17.99240273111952

First 10 documents and scores given by tf-idf model

Document ID	Rank	Cosine similarity Score
CACM_3127	1	0.4767644415952218
CACM_2246	2	0.4030816695672952
CACM_1930	3	0.36164866665327716
CACM_3196	4	0.13520034832993888
CACM_2319	5	0.13201633899993317
CACM_1680	6	0.11362334796197139
CACM_3068	7	0.11246015048214487
CACM_2379	8	0.11143531560608085
CACM_1591	9	0.10699075489371393
CACM_2593	10	0.10665956677605615

About 70-80% of the documents in the above three tables are the same. The documents obtained in the first two positions are the same for all the baseline runs. There is a very slight difference in the rankings of the rest of the documents that are common in the tables. They differ by a maximum of 3 positions.

2)parallel processor in inform retriev

First 10 documents and scores given by BM25

Document ID	Rank	BM25 Score
CACM_1811	1	12.255609879050214
CACM_1613	2	10.679596449530873
CACM_2714	3	10.655377532014716
CACM_2973	4	10.576186203979473
CACM_3156	5	10.376388261278693
CACM_2664	6	10.359368497426425
CACM_3075	7	10.352521394816984
CACM_2967	8	9.809310338711226
CACM_2114	9	9.031908496999034
CACM_2288	10	8.989165828341301

First 10 documents and scores given by Smoothed Query Likelihood Model

Document ID	Rank	SQLM Score
CACM_2967	1	-25.572883144576636
CACM_1811	2	-26.505215122775585
CACM_634	3	-26.584973431480996
CACM_2664	4	-26.749567489172247
CACM_2973	5	-27.201943636337386
CACM_2714	6	-27.39510119115182
CACM_3075	7	-27.506212480587383
CACM_1927	8	-27.584252125659336
CACM_1457	9	-27.64964319068606
CACM_3156	10	-27.840162868209536

First 10 documents and scores given by tf-idf model

Document ID	Rank	Cosine similarity Score
CACM_1811	1	0.5275968284461722
CACM_1613	2	0.34934557904276087
CACM_2664	3	0.32766943875432997
CACM_3134	4	0.2852837914043326
CACM_1457	5	0.28379706334862487
CACM_3075	6	0.27764931615014354
CACM_2973	7	0.27378077934109973
CACM_891	8	0.26577929933445527
CACM_1825	9	0.26525822019592465
CACM_1699	10	0.2566586470658482

There is about 50% match in the results in the above three tables. However, the results obtained from Query likelihood and BM25 models match by about 60%. The document CACM_1811 consistently occurs in the top two in the list. The document CACM_3075 is at 6 or 7th rank in all runs.

3) appli stochast process

First 10 documents and scores given by BM25

Document ID	Rank	BM25 Score
CACM_1696	1	8.938408092141279
CACM_268	2	7.890535245126738

CACM_1410	3	6.558778361995578
CACM_1233	4	5.670350558913919
CACM_20	5	5.640775251133785
CACM_2535	6	5.586787600563176
CACM_2882	7	5.5529860829190465
CACM_3020	8	5.488579269108091
CACM_1540	9	5.399678054012767
CACM_2065	10	5.275097598098065

First 10 documents and scores given by Smoothed Query Likelihood Model

Document ID	Rank	SQLM Score
CACM_1410	1	-18.163937350196402
CACM_1194	2	-18.31341809885319
CACM_2535	3	-18.763680803199545
CACM_268	4	-18.890885254507786
CACM_1696	5	-19.05505751182522
CACM_1233	6	-19.51500947492002
CACM_20	7	-19.534709215750404
CACM_1892	8	-19.553631019187815
CACM_2065	9	-19.590331794985723
CACM_3120	10	-19.876192917646932

First 10 documents and scores given by tf-idf model

Document ID	Rank	Cosine similarity Score
CACM_1696	1	0.3721966589946635
CACM_268	2	0.2986860154865694
CACM_2882	3	0.1335704645816255
CACM_1540	4	0.1246108106357784
CACM_1410	5	0.1114873635539812
CACM_1435	6	0.11135396042856141
CACM_2535	7	0.10916809194644095
CACM_2080	8	0.10522675235621175
CACM_2727	9	0.09604687947208895
CACM_111	10	0.09596129120335449

There is a 70% match in the results obtained from BM25 model and Query likelihood model. There is about 50% match in the results obtained from tf.idf model compared with the other two models. The documents that occur in all the three tables differ by a maximum of 3 ranks.

Results

Below are the precision and recall values for the first 5 results obtained for query 19 (parallel algorithms)

1) BM25 model with regular corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3075	R	2	1.0	0.18
CACM-2266	R	3	1.0	0.27
CACM-3156	R	4	1.0	0.36
CACM-1601	R	5	1.0	0.45

2) BM25 model with stopped version of the corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3075	R	2	1.0	0.18
CACM-0950	NR	3	0.67	0.18
CACM-1601	R	4	0.75	0.27
CACM-2266	R	5	0.8	0.36

3) Smoothed Query Likelihood model with regular corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3075	R	2	1.0	0.18
CACM-2266	R	3	1.0	0.27
CACM-2557	NR	4	0.75	0.27
CACM-3156	R	5	0.8	0.36

4) Smoothed Query Likelihood model with stopped version of the corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3075	R	2	1.0	0.18
CACM-0950	NR	3	0.67	0.18
CACM-2266	R	4	0.75	0.27
CACM-1601	R	5	0.8	0.36

5) tf-idf model with regular corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-2884	NR	2	0.5	0.09
CACM-0371	NR	3	0.33	0.09
CACM-3166	NR	4	0.25	0.09
CACM-2226	R	5	0.2	0.09

6) tf-idf model with stopped version of the corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3166	NR	2	0.5	0.09
CACM-2884	NR	3	0.33	0.09
CACM-3075	R	4	0.5	0.18
CACM-2950	NR	5	0.4	0.18

7) Pseudo Relevance Feedback with BM25 and regular corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-1658	NR	2	0.5	0.09
CACM-2401	NR	3	0.33	0.09
CACM-2289	NR	4	0.25	0.09
CACM-3112	NR	5	0.2	0.09

8) Lucene model with regular corpus

Document ID	Relevance	Rank	Precision	Recall
CACM-2973	R	1	1.0	0.09
CACM-3075	R	2	1.0	0.18
CACM-2266	R	3	1.0	0.27
CACM-3156	R	4	1.0	0.36
CACM-2557	NR	5	0.8	0.36

Conclusions and outlook

Below are the MAP and MRR values for all the 8 runs.

Retrieval Model Type	Mean Average Precision	Mean Reciprocal Rank
BM25 - Stopped corpus	0.3726437615362634	0.42408678275295203
Lucene	0.35548304485648885	0.350858927850249
BM25 - regular corpus	0.34481252055389244	0.41391377840716065
Query likelihood model - Stopped corpus	0.33832654308275556	0.3845750142199788
Query likelihood model- regular corpus	0.31269504841803253	0.32725767688173696
Pseudo relevance feedback with BM25 -regular corpus	0.29201123282298946	0.2839914341930471
Tf.idf – Stopped corpus	0.24098812204347872	0.22154472185790441
Tf.idf – regular corpus	0.15136713287720308	0.2590033675595173

It was found that the stopped version of the corpus gives the best results as compared to the plain and stemmed corpus. The MAP value for BM25 model for stopped version of the corpus has the highest value. Hence, in general, BM25 model has the best effectiveness. Since lucene internally uses a scoring function which is a combination of vector space model, tf-idf and Boolean model, it takes the second position in the list w.r.t MAP value.

When the p@5 and p@20 values for different queries for BM25's run on stopped corpus was compared with Query Likelihood model's run on stopped corpus, it was observed that neither QLM consistently outgrows BM25 nor the vice versa. But, as a whole, when the MAP was calculated, BM25 had an upper hand. The observation with the unstopped corpus was also the same. Also, when the recall values were observed closely, it was found that relevant documents were retrieved at an earlier rank in BM25 than in QLM.

Similarly, the p@5 values for BM25 was compared to p@5 values for Pseudo Relevance Feedback (PRF) using BM25. It was found that BM25's precision values were better than that of PRF. This might be because the terms that were used for query expansion using PRF were based on an initial ranking that didn't have many relevant documents, thus making the ranking worser.

Additionally, the results produced by the stemmed corpus and the unstemmed corpus for BM25, Smoothed Query Likelihood Model, tf.idf model, and Lucene all were very much similar for the corresponding runs. When the top 10 documents were sampled, about 80-85% of the results were the same. For example, the top 5 documents for BM25 retrieval with both stemmed and unstemmed query (portabl oper system) was found to be CACM_3127, CACM_2246,

CACM_1930, CACM_3196, CACM_3068. The possible reason for this could be that the corpus was very much similar in terms of document length, and the queries were also stemmed accordingly. The only difference was that the BM25 scores were higher for the unstemmed version, owing to factors like document length, which was found to be slightly lesser for the unstemmed corpus, according to the corpus cleaning technique that was applied.

The same was the case with the results obtained using stopped corpus and unstopped corpus. The results produced using different retrieval models using stopped corpus version matched for about 70% of the document occurrences in the top 10 sample obtained using the corpus which was not subjected to stopping (for the same run). The possible reason could be that, the queries used with the stopped version of the corpus were also subjected to stopping. Hence any score that was contributed by the stopwords were ignored by almost the same amount, leading to similar results.

When the results obtained from tf idf model was closely observed, it was found that documents with lesser document length were favored even if the appearance of query terms were less in that document. For example, for the query 'parallel algorithms', CACM-1601 is at rank 40 and CACM-3156 is at rank 19. CACM-1601 had 4 occurrences of 'parallel' and 1 occurrence of 'algorithms', with document length 144. CACM-3156 had 3 occurrences of 'parallel' and 1 occurrence of 'algorithms'. Inspite of this, CACM-3156 was ranked better owing to its lesser document length. Thus the overall ranking , MAP and MRR values get affected, thus pushing tf-idf scoring model to the end of the list.

Outlook

1) The time taken for the search process using tf.idf model and the time taken for snippet generation could be reduced by introducing multithreading.
2) Better snippet generation techniques can be implemented, which take into consideration the position of the query terms in the document and the context in which they are used.
3) Techniques to automatically update the index on addition of new documents to the corpus can be implemented.
4) Query based stemming can be implemented to enrich the query results.

Extra credit

1) Approach used for generating noise in the query:
The length of every query is calculated and the number of terms that can be affected is found out, based on the given threshold of 40%. The query terms are then sorted according to their lengths in non-increasing order. The terms that are to be manipulated by the noise are randomly selected from the first half of this sorted list, so that the longer terms in the query get a higher probability of getting affected.
The words chosen in such a manner are then individually shuffled while keeping the boundary characters intact. Using trial and error, shuffling every other character in tokens of length larger than average token of the query yielded approximately desired results. As a design choice, terms with length 3 and digits are not shuffled.

2) Approach used for implementing soft-matching query handler:
The Brown corpus provided by NLTK is used to get the corpus of words in English dictionary. For every query that has noise induced in it, each term of the query is subjected to correction. The correction module ignores the digits, terms containing hyphen, and terms with 3 character length, and then calculates the possible corrections of the words, limited by a certain time. The probability of occurrence of each of these words in the English dictionary is found out and the one with the maximum probability is used to replace the query term.

Analysis:

Type of run	**MAP value**	**MRR value**
Noise-free baseline with BM25	0.34481252055389244	0.41391377840716065
Noise-induced baseline with BM25	0.1782936842802936	0.20859319599573758
Noise-minimized baseline with BM25	0.19786070709532863	0.2152479619398847

The MAP value for the Noise-free baseline is the highest because of the input of queries whose terms match with the indexes. The MAP value for the noise minimized baseline lies between that of noise-free baseline and noise-induced baseline. BM25

gives better results with noise free queries and worst results with noise induced queries.

Resources:

1) *Implementing a spelling corrector*- Retrieved from https://norvig.com/spell-correct.html
2) *Using brown corpus* – Retrieved from https://www.nltk.org/book/ch02.html
3) https://stackoverflow.com/questions/7370801/measure-time-elapsed-in-python

Bibliography

1) Stanford.edu. *Snippet generation* -Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/results-snippets-1.html
2) Stanford.edu. *Vector Space Model*. Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/scoring-term-weighting-and-the-vector-space-model-1.html
3) *Lucene scoring function using elastic search* - Retrieved from https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html
4) Stanford.edu. *Pseudo Relevance Feedback*. Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/pseudo-relevance-feedback-1.html
5) Ian Carr and Lucy Vasserman. *SnipIt - A real time Dynamic Programing approach to Snippet Generation for HTML Search Engines*. Retrieved from http://www.cs.pomona.edu/~dkauchak/ir_project/whitepapers/Snippet-IL.pdf
6) W. Bruce Croft, Donald Metzler, Trevor Strohman. (2015) *Search Engines- Information Retrieval in Practice*. PearsonEducation, Inc.
7) Christopher D. Manning Prabhakar Raghavan, Hinrich Schütze. (2009) *An Introduction to Information Retrieval*. Cambridge University Press.