Northeastern University

CS6200 Information Retrieval
Spring 2019

Instructor: Rukmini Vijaykumar

Ankitha Kumari Moodukudru

Manisha Sharma

Rishabh Agrawal

I. Introduction:

The goal of this project is to design and build a information retrieval system and evaluate and compare their performance levels in terms of retrieval effectiveness.

Contribution: All team members collaborated and contributed equally for the report and discussing the design choices for implementation.

Ankitha Kumari: Implementation of stemming without stopwords and building index from stemmed corpus, Spelling Corrector, Report.

Manisha Sharma: Implementation of query expansion techniques and evaluation using MAP and MRR, Spelling Corrector,Report.

Rishabh Agrawal: Implementation of retrieval models, Snippet generation and Spelling Corrector, Report.

II. Literature and resources:

The following techniques were implemented in the project

A. Retrieval Models:

BM25: This is a ranking mechanism used in search engines to rank documents based on relevance for a given search query. It is based on a probabilistic retrieval framework. A BM25 score for any document is computed as follows:

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

TF-IDF(Term frequency - Inverse Document Frequency):

The TF of a word is the frequency of a word (i.e. number of times it appears) in a document. The IDF of a word is the measure of how significant that term is in the whole corpus. Higher the TF*IDF score (weight), rarer the term and vice versa.

Query Likelihood (Dirichlet Smoothing): In the query likelihood retrieval model, we rank documents by the probability that the query text could be generated by the document language model.Dirichlet smoothing is one of the methods based on the length of the document and is computed as follows:

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

Lucene: Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java.

B. Query Enhancement

The following query enhancement techniques were implemented:

Stemming : Stemming in the context of information retrieval is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. Porter stemmer is a popular algorithmic stemmer that has been used in the project. It consists of a number of steps, each containing a set of rules for removing suffixes. At each step, the rule for the longest applicable suffix is executed.

Pseudo Relevance Feedback: This is a well known query expansion technique where the terms for expansion are fetched from the relevant documents returned for the query. One way to add terms would be to look at the most frequent words in the retrieved documents.

C. Visualizing Results using Snippet Generation and query term highlighting

A document summary for a search contains the title of the document and a short text summary, or snippet, that is used to convey the content of the document. In addition, query words that occur in the snippet are also highlighted

D. Evaluation

The performance of retrieval systems are evaluated based on their effectiveness using the following measures:

1. MAP (Mean Average Precision) : Summarizes the effectiveness of rankings from multiple queries based on average precision for each query
2. MRR(Mean Reciprocal Rank): It is the average of reciprocal ranks over a set of queries
3. P@K : Precision at k documents is a measure of proportion of top k documents retrieved that are relevant
4. Precision & Recall : Precision is the number of documents retrieved that are relevant and recall is the proportion of relevant documents that are retrieved.

III. Implementation and discussion:

Runs :

1st Run [TF-IDF] : To rank the documents we calculated the tf-idf values associated with each individual term in the query and then added them up to get a score for a single document. The same was done to get the scores for all documents and then the document were sorted according to the values to get the rank.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	0.4048	0	0
QUERY 2	CACM-0204	0.2588	0.2	0.1
QUERY 3	CACM-0929	0.2841	0	0.0

2rd Run [Query-Likelihood Model (Dirichlet Smoothed)] : The implementation for this was taken from Page 282-283 of [1]. The value for **μ** was taken to be the average size of a document as that seemed to give the best result for the model. The same was done and the documents were sorted according to their scores.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	-110.674	0.4	0.1
QUERY 2	CACM-3078	-114.81	0.6	0.15
QUERY 3	CACM-2061	-39.79	0	0

3nd Run [BM25] : The implementation for the 2nd run is same as the 5th assignment where we assume k1=1.2, b=0.75, k2=100. As discussed with the instructors, it was assumed that no relevance information is available.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	19.9912	0.4	0.15
QUERY 2	CACM-3078	17.0053	0.6	0.15
QUERY 3	CACM-2061	7.7019	0	0

4th Run [Lucene] : Lucene's default model was used without any modifications to get the rankings for the documents.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-1519	0.2422	0.4	0.1
QUERY 2	CACM-3078	0.1579	0.6	0.15
QUERY 3	CACM-1988	0.1263	0.	0

5th Run [Query Time Stemming [BM25]] : Porter Stemmer from the NLTK library was used for obtaining the base stem for any word. Then a dictionary was created which had the stem word as the key, and the value contained a list of words for which the key is the stem word.

Ex : Computer : ['computerization', 'computerized', 'computerize']

Then for each word in the query, a stemmed word was added into the query from this list, and it was done till 10 values. The number of expansion terms can be modified using a parameter value. We chose 10 as the value, as our original queries were already big, so we did not want to expand it too much.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	22.7480	0.2	0.15
QUERY 2	CACM-3078	17.0053	0.6	0.15
QUERY 3	CACM-2652	25.9754	0	0.05

6th Run [Pseudo Relevance Feedback [BM25]] : The implementation here is similar to the one provided on page 233 of [1]. We take the top 100 results from the current query and expand it with the most occuring term from those 100 documents which is not a stop word. The query was expanded upto 10 words.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	17.5058	0.2	0.15
QUERY 2	CACM-3078	11.7894	0.4	0.15
QUERY 3	CACM-2939	4.3686	0	0

7th Run [Stopping [BM25]] : A new corpus was created in which all the stop words [provided by the instructors] were removed. The query also had its stop word removed before fitting in the BM25 model from the 3rd run above.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2319	20.2884	0.2	0.1
QUERY 2	CACM-3078	13.1587	0.6	0.15
QUERY 3	CACM-2061	10.4378	0	0

8th Run [Stopping [TF-IDF]] : A new corpus was created in which all the stop words [provided by the instructors] were removed. The query also had its stop words removed before fitting the TF-IDF model from the 1st run above.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20

QUERY 1	CACM-2319	0.6330	0	0
QUERY 2	CACM-0597	0.2330	0.4	0.15
QUERY 3	CACM-2316	0.3575	0	0

9th Run [Pseudo Relevance Feedback with Stopping [BM25]] : The pseudo relevance from the 6th run was executed with the stopped corpus from the 7th run, to gain a new query and then the final result is the one which we got from running the final query from above. The query was expanded upto 10 words.

	Top Result Returned	BM25 Score of Top Doc	P@5	P@20
QUERY 1	CACM-2358	25.9938	0.4	0.15
QUERY 2	CACM-2356	15.0441	0	0.1
QUERY 3	CACM-2939	23.1496	0	0.05

Task 2:

Snippet Generation and Query Highlighting : Firstly, the query by the user was run using the BM25 Model [3rd Run] and snippets were generated for the top 10 results obtained in them. The number of results can be altered by changing the parameter in the function. The snippet was generated using a sliding window method, where we took a window of 10 [Can be changed in the parameter] to go through the entire document, and generate the snippet which contained the most number of query terms. The query term was highlighted with a yellow background. No snipped is generated if query term is not found in the corpus. An example is presented below:

```
Enter Your Query : algorithms rule


Result: 1
Document Name : CACM-2273
tables by rule mask method without rule mask two algorithms


Result: 2
Document Name : CACM-2239
numerical integration integration rule adaptive integration automatic integration simpsons rule


Result: 3
Document Name : CACM-2074
numerical integration integration rule adaptive integration automatic integration simpsons rule
```

Extra Credit:

Spelling checker was implemented for the spell checking part. If a query word does not exist in the corpus then Levenshtein distance was used to calculate distance of the term from all the words in the corpus and

then the top 6 words with the minimum distance were selected. The same was done for all unrecognized terms in the query and then new query suggestions were provided to the users. A example is shown in the Image. The suggestions are limited to 6 but can be changed by modifying the parameter value in the code.

Query by Query Analysis :

Following queries have been chosen for comparison and analysis:

Query 1: "**what articles exist which deal with tss time sharing system an operating system for ibm computers**"

Most of the runs resulted in top relevant document as CACM-2319 (Operating System Performance), except Lucene that resulted in CACM-1519(A Model for a Multifunctional Teaching System) and Pseudo-Relevance[Stopping] CACM-2358(Multics Virtual Memory: Concepts and Design) .

While most of them did a good job in finding relevant results however Lucene and Pseudo-Relevance with stopping gave results that were irrelevant that is the information in the documents did not match the query.

Query 2: "**i am interested in articles written either by prieve or udo pooch prieve b pooch u**"

Most of the runs resulted in top relevant document as CACM-3078(Analysis of the Availability of Computer Systems Using Computer- Aided Algebra) , except TF-IDF that resulted in CACM-0204(Proving Theorems by Pattern Recognition I), TF-IDF with stop that resulted in CACM-0597 and Pseudo-Relevance[Stopping] CACM-2356 (A Technique for Software Module Specification with Examples)

While most of them did a good job in finding relevant results however both version of TF-IDF and Pseudo-Relevance with stopping gave results that were irrelevant that is the information in the documents did not match the query.

Query 3: "**intermediate languages used in construction of multi targeted compilers tcoll**"

BM25, Stopping BM25 and QLM returned CACM-2061: An Algorithm for the Construction Of Bounded-Context Parsers with seems relevant as parsers use compiler.

TF-IDF returned CACM 0929 :Glossary Construction which was irrelevant in terms of the query.

Results

	MAP	MRR
BM25	0.3231	0.7265
TF-IDF	0.2007	0.5466
Query Likelihood	0.2409	0.6068
Lucene	0.3871	0.6462
Stemming	0.3174	0.7202
Pseudo-Relevance	0.2759	0.6306
Stopping-BM25	0.3085	0.6841
Stopping-Tf-idf	0.2145	0.5379
Pseudo Relevance [stopping]	0.2516	0.5746

Query 3 for different runs :

Run 1: `intermediate languages used in construction of multi-targeted compilers tcoll`

Run 2: `intermediate languages used in construction of multi-targeted compilers tcoll`

Run 3: `intermediate languages used in construction of multi-targeted compilers tcoll`

Run 4: `intermediate languages used in construction of multi-targeted compilers tcoll`

Run 5: `intermediate languages used in construction of multi-targeted compilers tcoll language use constructed compiler using construct compilation useful constructing compiling`

Run 6: `intermediate languages used in construction of multi-targeted compilers tcoll 4 5 programming cacm 1978 jb language 6 1098 2939`

Run 7: `intermediate languages construction multi-targeted compilers tcoll`

Run 8: `intermediate languages construction multi-targeted compilers tcoll`

Run 9: `intermediate languages construction multi-targeted compilers tcoll 4 5 programming cacm 1978 jb language 6 1098 2939`

IV. Conclusions and outlook:

The best MAP value is obtained by the Lucene model followed by the BM25 Model, while the best MRR is obtained by the original BM25 model, followed by the stemming in Run 5. The worst MAP and MRR are provided by the Stopping Model [TF-IDF] in the Run 8. If we see the values for both MAP and MRR we can say that all in all BM25 performed the best as it ranked 1st and 2nd in the sections respectively.

As query implementation was implemented using both pseudo relevance feedback and stemming words. The stemming was the one which performed better as it used different forms of the same word, when compared to the pseudo relevance model which just blindly added the words.

Pseudo Relevance on the stopped corpus didn't seem to provide satisfactory results when compared to the other models, the reason for which might be the loss of information because of performing stopping.

Outlook :

- Although results for BM25 are good enough, but they can be increased even more by adding relevance information to the models, to provide more accurate and relevant results.
- Removing words by stopping led to a decrease in performance, so removing words might not be a good decision in terms of information retrieval.
- By adding new words with stemming the results were far better and more relevant documents to the queries were fetched.
- While expanding queries with the Pseudo Relevance feedback, we need to be extra careful in which terms to append, as choosing the words carefully can lead to a huge change in the effectiveness of the model.
- Additionally we could add a page rank for the pages and use the rankings together with our model to gain more effective and relevant results.
- Additional page details like freshness of the page, update rate etc. could be used to provide more up-to-date information to the user.

V. Bibliography:

1. Croft, W.B., Metzler, D. and Strohman, T., 2010. *Search engines: Information retrieval in practice* (Vol. 520). Reading: Addison-Wesley.
2. https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model1.html
3. http://mlwiki.org/index.php/Smoothing_for_Language_Models#Dirichlet_Prior_Smoothing
4. https://www.nltk.org/
5. https://www.cuelogic.com/blog/the-levenshtein-algorithm

VI. Appendix

```
Enter your Query : algorthm computre is good tiem
Did you Mean :  algorithm compute is good them

Did you Mean :  algorithms computers is good ties

Did you Mean :  algorithmic computer is good tied

Did you Mean :  althm compare is good the

Did you Mean :  algol computed is good ibm

Did you Mean :  aegerter computes is good time
```