Building your own Retrieval system

Course Name: Information Retrieval

Instructor : Nada Naji
Semester : I
Name of group Members:
Rahul Bhat
Rutva Rajdev
Suhani Ladani

Introduction

The project is about designing and building an information retrieval system. It starts from building an indexer and a retrieval system from scratch. The next phase is about displaying the results using snippet generation and query highlighting technique. The final phase is about evaluating and comparing performance levels of various retrieval systems based on the retrieval effectiveness. The test collection used for this project is CACM test-collection which comprises of textual corpus, queries, relevance judgments and stoplist.

Contribution

Rahul Bhat : Indexer and searcher, Baseline smoothed query, with stopping and stemming (for the 3 baseline runs), MAP, Recall, Precision table generation, Extra credit
Rutva Rajdev : Baseline runs tf-idf, pseudo relevance model, MRR table generation, Extra credit
Suhani Ladani : Lucene runs, Baseline runs BM25, snippet generation, MAP, Recall, Precision table generation, Extra credit

These were the main responsibilities that were distributed. But the discussion for all of the things was held together and inputs were given by all. There was constant communication and exchange of ideas throughout. Documentation was done collectively.

Implementation

Phase - I

Task 1

Extraction of data was done by using BeautifulSoup from the given HTML files. And then built a space separated token file for each document using NLTK tokenizer, lower casing the terms and handling punctuations. Using these Space separated files a unigram indexer was built which was stored in a text file in the following format:
international>>CACM-0001 1;CACM-0099 1;CACM-0414 1;CACM-0690 1;CACM-0975 1;CACM-1362 1;CACM-1476 2;CACM-2875 1;CACM-3184 1;

(where “international” is the term in the corpus, “ >>” , ":" are delimiters for separating CACM-0001 is the document which has the word international, “1” is the term frequency. Each of the terms are separated by a by lines in text file
Data structures : We have used the dictionaries for the code to store various information like document length, document frequency(for each term), term frequency(for each term in each document, using 2D array).

The retrieval models have been performed using 4 systems (baseline runs) namely,

- BM25 model: This model is built using the formula as given in [1] (the book Search Engines). The relevance information has been considered in the model which makes use of the set of relevant documents that have been provided to us apriori.

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

  - N : total number of documents in the collection
  - R : number of relevant documents for the query
  - r_i : number of relevant documents containing the term i
  - n_i : number of documents containing the term i
  - f_i : frequency of the term i in the document
  - qf_i : frequency of the term i in the query
  - k 1 : value = 1.2 as per TREC standards
  - k 2 : value = 100 as per TREC standards
  - K : K = k 1 ((1 – b) + b * dl /avdl) where dl is document length, avdl is average length of
  - b : value = 0.75 as per TREC standards

- Tf-idf model:
  - Tf = term frequency in document/length of document
  - idf = log(Total document count / document frequency)
  - The final computation uses the formula: (number of times word appears in the query) * tf * idf

- Smoothed Query model:
  - The formula used

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

  - Lambda as given in the task document.
- Lucene model: The given lucene model used with minor changes in the code

Task 2

Pseudo Relevance feedback model

Pseudo Relevance feedback model provides an automatic way to expand the query for improved information retrieval.

- The top 100 retrieved documents from the BM25 model are further processed for the pseudo relevance feedback
- The top K (10 here) documents are taken and terms in each one of them is arranged in decreasing order of term frequency and added to a temporary list(for expanding query)
- The terms with same term frequency are computed for increasing order of collection frequency (rare terms are important); and added according to this order in the above list of expansion words.
- Now each word is checked, if it is not a stop word then it is used as an expansion word for the query. N (10 here) such words are selected for query expansion.
- BM25 scoring function is used on this expanded query to fetch the top 100 results.

Task 3

With stopping : The queries are stopped and then the system is run on the stopped queries.
With stemming : The stemmed documents are parsed and stored as separate files; stemmed queries are used to run the 3 baseline runs on all the stemmed documents.

Phase - II

Task 1

Snippet generation

For every snippet, we have kept the title and date as a constant information(that is displayed). Given a query, we are first arranging all the query terms in the increasing order of its appearance in the collection frequency. The rare words have a higher probability of highlighting the topic of the document and thus are considered to be important. The words that have the same collection frequency are added to another array, which are processed for document frequency. These words are arranged in increasing order of the document frequency (considering rare as important words), and appended in the increasing order in the list of important terms. We can adjust the number of important terms that should be considered for snippet generation, which is kept 15 for current experiment(through trial and error). For top 100 retrieved documents for each query, we display the sentences that have one or all of the important terms, for each document. The query words are highlighted(uppercasing the whole word) in the snippet. (Text files do not support bolding)

The snippet generation technique has been performed on the BM25 base run.

Phase - III

Task 1

Evaluation

For Evaluation, we have retrieved the top 100 documents for each query and converted them into a list of Relevant and Non-Relevant Documents denoted by R and N respectively. The conversion into R N list is done using the string matching with the given relevant documents list. The top 100 documents are retrieved, and then matched for relevancy using the given CACM.rel list of relevant documents for the respective queries. We are then processing the Precision, MAP, MRR, Recall for the queries.

(Extra credit)
Query terms proximity (with and without stopping):

- Given a query, we make an array of the query by tokenizing the query into tokens
- Similarly, we make an array of tokens for a document and store the positions for each word in a dictionary where the key is the term and value is array of positions.
- For each query term, we check whether the term is present in the document. If present, it is added to a list.
- Now the dictionary that has the terms and corresponding positions is checked for the proximity and order. This is checked only for the terms in the list(that we created above).
- The order is checked by finding the difference in the positions of the words. For example if we consider 2 words: operating [5, 20] and system [3, 23]. For each position of the word 'operating', we check for all the positions of the term 'system'; where ever the difference is less than zero (< 0), it could be said that the operating and system appears in the required order as given in the query (i.e 'operating system').
- Find the absolute of the index difference for the terms, which should be less than or equal to 4(at-most 3 terms between any two query terms).
- For each query term pair that appears adjacent and also ordered in the document we have kept a counter 'No of adjacent terms' which is incremented by 1, every time we find ordered match in the document. And a counter for sum of the intermediate terms which is the sum of all the number of intermediate terms for the entire document(for all query words that appear in document)
- Further, we take BM25 scoring function and extend it by incorporating the above calculated parameters 'no of adjacent terms' and number of intermediate terms.
- In order to give importance to the appearance of the query terms in the document in ordered fashion, we multiply the BM25 score by 'no of terms'. While, in order to penalize the score for not appearing completely adjacent (i.e more than 3 non query

terms between two query terms in the document), we divide the BM25 score with that counter(number of intermediate words).

- Thus, the final scoring function would follow the formula:
    - BM25 score *(number of terms/ sum of intermediate terms)
- Now for stopping, we remove the terms which are present in the stop word list.
- Here, stopping the query is not required separately since we are stopping the documents. When we find the query terms which are present in the document also, stop words would be ignored.

Analysis

BM25 gives the best results when we compare the MRR of all the 4 systems. Moreover, we can observe that more relevant documents appear at the top of first 100 retrieved documents in BM25.

When we compare the top few documents out of the 100 retrieved documents for each of the systems, it is observed that they overlap. Though the sequence and ranking do not match exactly but they do have the same documents.

On evaluating the relevant and non-relevant documents using the set of given relevant documents, it could be seen that the tf-idf gives the minimum accuracy since the top retrieved documents are non-relevant. Whereas in case of BM25 and smoothed query, the results are much more accurate in terms relevancy of top retrieved documents. tf-idf would give zero for the query term that is not present in the document, whereas this should have some weight since it has importance in the overall collection. This drawback is handled in the smoothed query model.

In the case of pseudo-relevance feedback, majority of the results appears to be non-reliable. This could be because, if the initial ranking does not contain many relevant documents, the expansion terms found by pseudo-relevance feedback are unlikely to be helpful and, for some queries, can make the ranking significantly worse.[1]

Query by Query Analysis for Stemmed Queries

1. portabl oper system
2. code optim for space effici
3. parallel algorithm

When we observe the stemmed queries, we see that the queries are over stemmed; like operating is changed to ‘oper’ and evaluation is changed to ‘evalu’. The documents are stemmed in a similar manner. This could work in favour as well as become a drawback.

For without stemming, if a query word contains ‘algorithm' and document contains ‘algorithms'; the query is not considered relevant to the document, classifying the document as non-relevant. Here, stemming could work in favour since it would start considering these words which actually should mean same as relevant(resulting in improved results). Thus, in case of parallel algorithm query, it would give comparatively relevant results(with stemming).

Whereas for the words that are over-stemmed, two words that might be meaning different are stemmed to the same word, which could classify non-relevant documents into relevant documents. For example ‘opera’ , ‘operation’ , ‘operating’ , ‘operator’ would all stem into ‘oper’. When we consider the context of ‘operating’ in operating systems and ‘operations’ whose context might not be anywhere related to operating system; but would be classified as relevant.

Extra credit analysis

The effectiveness decreases than BM25(when evaluation is compared for both the systems). The probable reason for this could be that the given queries are vague. For example

> List all articles on EL1 and ECL (EL1 may be given as EL/1; I don't remember how they did it.
>
> I'm interested in mechanisms for communicating between disjoint processes, possibly, but not exclusively, in a distributed environment. I would rather see descriptions of complete mechanisms, with or without implementations, as opposed to theoretical work on the abstract problem. Remote procedure calls and message-passing are examples of my interests.

It is very unlikely, that the given words would appear in the document, that too in the order given in the query. Thus, this approach could be more effective when the queries are more appropriate.

Conclusion and outlook

As discussed in analysis, BM25 gives the best result of all the systems. This could be reasoned with the fact that it takes into consideration the maximum number factors like term frequency in the document, term frequency in the query, and provides the weight to both these using the parameters like k1 and k2 which could be adjusted according to the corpus.

We observe that results of MAP and MRR are more for runs with stopping for smoothed query and tf-idf. Since the stop words are no longer considered, the document is scored according to only the important words which should actually be stressed upon while considering the relevancy of the document for a given query.

Future approach

If the corpus size is reasonably large and uniform, the snippet generation technique could be improved by scoring all the sentences that contains the query words(significant words). Using the approach given in the book Search Engines (by W. Bruce Croft)[1].

For pseudo relevance we could expand the query using terms that are more related to the query rather than randomly choosing the words based on term frequency and document frequency. This could improve the result since it would expand the query based on the query provided by the user and thus reduce the number of irrelevant results. Incorporating the user feedback could improve the results.

Resource and Citations

[1]. Search Engines Information Retrieval in Practice by W. Bruce Croft. Used the text book for referring the concepts and applied our own algorithm using trial error methods to come up with comparatively better solution.
[2]. https://nlp.stanford.edu/IR-book/pdf/08eval.pdf
[3]. http://referaat.cs.utwente.nl/conference/9/paper/6923/generating-snippets-for-undirected-information-search.pdf
[4]. http://ad-publications.informatik.uni-freiburg.de/TOIS_snippets_BC_2014.pdf

Third party libraries

- Beautiful soup
- NLTK Tokenizer