Information Retrieval Final Project

CS6200 - Spring 2019

Instructor: Rukmini Vijaykumar

Team members:
Kefei Zhan
Peng Tong
Zheng Zheng

Introduction

Our project utilized core information retrieval concepts introduced in the course to build our own retrieval systems. We used different techniques like stemming, stopping and query expansion to enhance our systems. After that, we implemented snippet generation and query term highlighting techniques to display the results. We also evaluated the performance of theses systems with different measures like Mean Average Precision and Mean Reciprocal Rank.

We tried to distribute the work evenly among team members. Keifei Zhan is responsible for phase1 (built retrieval systems and produced the results of eight runs), extra credit (built a search engine based on the Relevance Model using pseudo-relevance feedback and KL-Divergence for scoring) and the documentation related to his work.

Zheng Zheng is responsible for phase2 (implemented a snippet generation technique and query term highlighting within results in BM25 baseline run) and the documentation related to his work.

Peng Tong is responsible for phase3 (produced a new run by combining a query expansion technique with stopping and implemented MAP, MRR, P@K, Precision&Recall for nine runs), the documentation related to his work and the main structure of final report.

Literature and resources

We built four retrieval systems with Tf/idf, Binary Independence Model, BM25 and Lucene's default retrieval model. To perform query enhancement, we used two distinct approaches which were query time stemming and pseudo relevance feedback. For snippet generation, we used the number of significant words contained in a sentence as the significant factor of the sentence and chosen the sentences with highest factor as snippets. After that, we used a special tag <B>word</B> to highlight the words that belonged to the query term.

We used the textbook recommended in the lecture as the main reference for our technique and algorithm choices. We also learned some concepts and ideas from other research articles and online materials which are cited in the "Bibliography" part of the report.

Implementation and discussion

For the four baseline runs in phase2, we chose to implement BM25, default Lucene model, Tf/idf and Binary Independence Model. For BM25 and Lucene model, we basically just reused the code of previous homeworks. For the Tf/idf, we use the method of tfidf to vectorize each documents and calculate the similarities between document vectors and query vectors. Then rank the documents according to the similarities in descending order. For the Binary Independence Model, we create the index of the relevant documents for every queries and calculate pi and si for each document corresponding to queries, then use the formula provided in class to calculate the score. In the end, we rank the document according to the scores we calculated in descending order.

For the query enhancement in phase1, we chose to apply query time stemming and pseudo relevance feedback. In Query Time Stemming, we choose to use the stem() function in PorterStemmer() in nltk package to process every queries and documents then use our baseline model to create outputs. About Pseudo Relevant Feedback, we re-use some part of the code from HW4 and implement dice's coefficient which we did not use in that homeword. For the part of dice's coef, we convert queries and words in relevant documents (Use BM25 get relevant documents) into bigram letter format, then find the most similar words according to the dice's coef for two bigram letter lists and add to the original queries.

For the snippet generation in phase2, the technique we used is primarily query-dependent. Specifically speaking, for a given query, a list of significant words are generated. All the words that are contained in the query and not contained in the common words list is a significant word. After the list is generated, the original document is retrieved. Since it is an html document, only the tag that contained main content of the article is retrieved. Then the tag is further processed to the the pure content block. After this, the content block is cut into sentences. Then for each sentence, a significant factor is generated so that the sentences can be ranked and the program gets to know which sentence should be chosen for snippet generation. In this case, since the snippet generation technique is highly query-dependent in its essence, we mainly care about the words from the query itself. Therefore in this case, we used the number of significant words contained in a sentence as the significant factor of the sentence. After all the sentences are processed by calculating significant factors, all the sentences are ranked from high to low with regard to significant factor. Then the top one or two sentences are chosen and are displayed as the snippet summary for the document.

For the query term highlighting in phase2, after getting the snippets, we searched the query term in the snippets. If a word belongs to the query term is found, we marked this word with a special tag <B></B> to indicate the word is highlighted.

For the 9th run in phase3, we combined a query expansion technique with stopping based on the Lucene baseline run. To implement query expansion, we used pseudo-relevance feedback. First, we got the top 10 documents with the default retrieval model and counted the words frequency in these documents. The most frequent words are chose as the potential expansion words. Then we combine these expansion words with original query term. After that, we applied stopping on the expanded query term by removing the words that appeared in the given stopping list. Finally, we used the default Lucene model to retrieval results of these expanded and stopped queries.

For the evaluation part in phase3, the key was creating the precision and recall tables for each query of each run. To do that, for each query, we iterated the top 100 documents produced from phase2, remembering the position of first relevant document and keeping tracking the total count of relevant documents. After getting the precision and recall tables, we can easily compute the MAP, MRR and Precision at Top k.

For the extra credit, we first build the relevant model from query and top ranked documents according to the probability derivation from class, then rank documents by similarity of document model to relevance model using KL-divergence. The relevant information is from baseline output in Phase1 Task1. Then we rank the documents according to the KL-divergence: The larger the document's KL-divergence is, the more relevant the query and document are.

Results

It is difficult to represent all the results in this report, thus we included the result tables under the folder of each phase. For instance, ../phase1/result.

For phase1, we got the top 100 results for each query of each run. And there were totally 8 runs among three tasks, while each run had 64 queries. And the result was represented with format "query_id Q0 doc_id rank score system_name".

For phase2, for each query, we generated the snippet of each document and indicated the highlighting words with tag <B>words</B>. Words inside the tags would be highlighted. Since for displaying results we chose baseline run 'BM25' result as the resource, all the 64 files are based on BM25 baseline run results. For example, 'Query1.txt' contains the top 100 documents retrieved for query 1, with the snippet summary generated for each document.

For phase3, we represented the results of MAP, MRR, P@K, precision scores and recall scores in separate tables, while each table had a title to indicate the meaning of each column. Especially, for MAP, MRR and P@K, we displayed the results of all nine runs in one file, while for precision and recall scores, we displayed the result of each run in a separate file.

Conclusions and outlook

After analyzing the results of phase1, we found out that the result is satisfying for most of the queries except those queries which are not easy to find the key words. For example, in the result of query 2, the ranking is very different in most algorithm. If there are some sensitive words in the query, the result is pretty relevant. There is an interesting phenomenon in Phase1 Task3: I selected two queries which contains "parallel" in both of them, and we can find some similarities in the result. In document CACM-2664 (whose id in my program is 2663) contains both "parallel", "process" and it was ranked in both queries ("parallel algorithm", "parallel processor in inform retriev").

After analyzing the evaluation results of phase3, we found out that BIM system has highest MAP score and MRR score, following is default lucene system and lucene system with query expansion and stopping. It means these three systems have higher possibility to find more relevant documents and find these relevant documents in the top results. Since the ability of finding relevant documents is the key factor of evaluating a retrieval system, these three systems tend to have better performance in practical application.

After comparing the results of extra credit with phase3, we found out that the kl-divergence had lower MAP and MRR score than 4 base runs(Lucene, BM25, BIM, tf/idf). It means that kl-divergence had no advantage on finding relevant documents based on query terms.

There are still a lot of space for improvement. For instance, currently, different phases of the system are kind of independent. Besides, even though it can reproduce the results, there is no interfaces for users to perform queries except running the codes in certain IDE. To improve the system, we consider to provide User Interfaces that user can perform actions like choosing retrieval models, choosing whether to apply query enhancement techniques or not, performing query and getting results.

Bibliography

Following are the references we used while implementing this project:

[1] Croft W B , Metzler D , Strohman T . Search engines: Information retrieval in practice[M]
[2] Roelleke T, Wang J, Robertson S. Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model[M]// Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model. , 2009.
[3] Albitar S, Fournier S, Espinasse B. An Effective TF/IDF-Based Text-to-Text Semantic Similarity Measure for Text Classification[C]// International Conference on Web Information Systems Engineering. 2014.
[4] Dupret G, Piwowarski B. A user behavior model for average precision and its generalization to graded judgments[C]// International Acm Sigir Conference on Research & Development in Information Retrieval. 2010.
[5] Yu H, Liu Z, Yi C. Query biased snippet generation in XML search[C]// Acm Sigmod International Conference on Management of Data. 2008.
[6] Qiu Y, Frei H P. Concept based query expansion[C]// 1993.
[7] https://link.springer.com/chapter/10.1007/978-3-642-45068-6_5
[8] https://nlp.stanford.edu/IR-book/html/htmledition/the-binary-independence-model-1.html