Christopher Zawora
Information Retrieval
12/16/15

Project 3 - Query Expansion

Cover Page

Project: complete

Hours worked - 30 hours

Things I would like to have known:

At this point in my project, my code, which has been building off previous projects, has become complicated. This is the biggest coding enterprise I have undertaken as an undergraduate and while I do appreciate the open endedness of the project, some tips on organizing code and implementation in general would be beneficial.

This project used the revised qrel.txt file

Design Document

This project builds off of the code I have written for project 2. Queries will be checked against a phrase document threshold and positional document threshold and sent to the respective index if the number of documents passes the threshold. Otherwise they will be sent to the default index, which for this project I will keep consistent as the single term index. This will all be done in Python. The components that are added are additional query parsing and tokenizing, to get the narrative (long) query, as well as the reduction and expansion methods.

First, to add in the additional narrative queries, the loading code will be updated to also grab narratives when reading from the query file. From there, the narratives will be normalized and tokenized in the same way all other documents where processed.

Second, I will update the query processing section of the code to handle expansion and reduction. However, first it is important to note what I plan on measuring when doing query reduction and expansion and which methods of reformulation will be used. For expansion I plan on using relevance feedback wherein I get the top ranked n documents from which I take the top t terms with highest weight (tf-idf). These terms will then be added to the query. The interesting observations will come from exploring which top n documents and how many top t terms produce the best results. On the other hand, for reduction I plan on taking the terms from query which have the highest tf-idf score. The number of terms to take will be explored.

For expansion, the initial results will first need to be calculated and then the ranked documents collected. From those documents the top n will be chosen. The terms from those documents will collected and their tf-idf scores recorded. Then the top t tf-idf scored terms will be added to query and the expanded query results will be calculated.

For reduction, the query terms will be ranked in by their qtf-idf and the top t terms will be collected and reformulated into the new query. The motivation for not simply going by lowest query term frequency is that just by inspecting the long queries there are sometimes more than 10 terms that only appear once or twice. In order to further differentiate the query terms I also decided to include their uniqueness in the collection (idf) as a factor.

Once the expanded or reduced query is produced it is then going to be processed in the same way the original query was handled.

Finally, once each of the query texts and their retrieved documents is evaluated and scored, they will be sorted, written out to a file, and can be evaluated with treceval.

All of this will be handled in Query_Processor, Index, and Query classes. The Index and Query classes are for keeping the indices, lexicons, and doc-term indices and queries organized, respectively. The Query_Processing class is a wrapper class for all the methods and logic used in processing the queries and writing out the search results. The main function will only take in the command line arguments regarding default index and ranking method as well as create an instance of the Query_Processor.

No new classes were added to the flow of the program, however a few new methods, mentioned above, were introduced in the Query Processor class. Additionally, the Query class was updated to hold the narrative query as well.

Analysis

For evaluation metrics I would use both MAP score and number of relevant documents retrieved. These statistics seemed most pertinent because they illustrated both whether query reformulation retrieved more relevant results but also whether query reformulation caused a difference in ranking of the relevant documents.

For analysis data on query expansion, please see the PDF chart. For this experiment, I tried the various combinations of top n docs and top t terms starting with both at 5 and moving down until both values were only 1. I did this for each ranking method..

The purpose here was to compare which number of top docs and top terms would provide evaluation metrics that surpassed the initial query. For cosine similarity, MAP and num_rel_ret were both improved for top doc - top term pairs of (4,5) (4,4) (3,5) (3,4) (3,3). Meanwhile, num_rel_ret improved with most settings but was also negatively impacted when the number of docs used and number of terms used was high.

For bm25, MAP and num_rel_ret were never both improved together. In fact, MAP scores only suffered from query expansion. However, the expansion increased num_rel_ret as top docs and top terms increased.

Lastly, kl divergence saw almost no change in num_rel_ret and had the best MAP score when only a single term from a single doc was added to the query.

These results paint an interesting picture because they illustrate the impact of the choice of ranking method on expansion effectiveness. Since the different ranking methods rank documents differently, the selections of top docs and consequently top terms will also vary.

Now looking at reduction, the data can also be found in the accompanying PDF. For this experiment, I played with different numbers of top terms that were kept from the original long query ranging from a reduced query of 6 terms to a query of 3 terms.

The first thing to notice about the reduction data however, is how poor the MAP and num_rel_ret scores were for the original long queries when phrases were allowed to be found. Most of these queries passed phrase and positional index document thresholds, set to 20 and 100 respectively based on results from project 2, and were therefore processed using those indices. As seen in my project 2 results, the phrase and positional indices did not produce good results. However, when no phrases were allowed and only the single term index was used the number of documents retrieved and the precision of the results was the highest of any queries ever submitted.

Naturally, the reduction data contains some interesting results. The general observation is that reductions ended up not only reducing the query length but also severely reducing the MAP and num_rel_ret scores. This surely has to do with the way terms for the reduction were selected. By only selecting the top terms, the key terms that signified the intent of the query were missed. Yet there is a trend to be seen within the reduction data itself. For cosine similarity and bm25, increasing the number of query terms in the reduction from 3 to 6 increased both the MAP and num_rel_ret scores. However, for kl divergence, while increasing the number of terms in the reduction increased num_rel_ret it, none of the reductions improved MAP significantly.

All in all, a slightly bigger reduction led to more relevant documents being retrieved. But MAP scores varied by rank method.