COMPSCI546 Assignment-2 Report

Jaideep Rao

September 2020

1 System Design

This system is built on top of the previous implementation, and makes use of the files and lookup tables generated by the indexer. The query retrieval component has been updated to support 4 different retrieval models based on provided input parameters.
Each retrieval model type (vector sapce, Bm25, QL) has its own method in order to simplify the process and keep it intuitive. Each model uses document-at-a-time processing. Both QL methods are handled in a single function since they are very similar, and only differ in the value of α_D used in both methods, hence it made sense to take advantage of that to reuse code wherever possible. The trade-off for this was to duplicate some code for document-at-a-time processing in other functions.
I had a number of questions regarding the vector space implementation at the beginning, around whether we had to create the query and document vectors for each iteration and perform the dot product with normalization for our calculations since that is how the equations go, but that complicated the implementation a lot. This was dealt with when I followed the simpler pseudo code found in the slides
I also faced some issues of mismatching results in my language models but that was due to a bug I discovered and fixed in my implementation of getCollectionTermFrequency(), which was not resetting the value each time it was called, resulting in the same query term producing higher values for its term frequency as the document number increased, causing the latter documents to get higher scores.
Software libraries used in this project were restricted to the java collection library for simple abstractions like HashMaps and ArrayLists to capture information appropriately in the postingListsMap and the postingList itself. I also used the Math library for its log function to help with calculations involving the product of small numbers.

2 Do you expect the results for Q6 to be good or bad? Why?

I expect the results to be bad, since this particular query requires that we take positional information into consideration because of the way it has been structured. Effective results would need to take into account that the repetition of the query terms has more to it than to just bump up the individual query term frequency values. The strategic placement of the repeated instances of words contains key contextual information. Since all the models we have implemented are bags-of-words, they only take occurrence frequency into account disregarding any positional information, the resulting top ranked documents don't contain the complete phrase

3 Do you expect the results for a new query setting the scene to be good or bad? Why?

I expect the results to be bad because the word "scene" occurs as the first word in every document. As a result of this, it will have a very low IDF value, which is punishing to all documents. I expect that the high frequency of this word would skew the resulting rankings of all documents and provide bad results.

4 What will have to change in your implementation to support phrase queries or other structured query operators?

I suspect the best way to support phrase queries and related structured query operators would be to incorporate positional information within our model to enforce restrictions on how far/close query words should occur from each other for a relevant result.
This can probably be achieved by expanding our implementations to output n-grams rather than just unigrams. This would allow us to try and generate whole phrases and sub-phrases from the query which could then be used to generate the likelihood of a particular document containing them. With the help of whole-phrase searching we could look for direct matches for the query phrase, and the sub phrases could be used to conduct any query operations such as "tropical fish NOT saltwater fish" etc.

5 How does your system do? Which method appears to be better? On which queries? Justify your answer.

Broadly speaking, the probabilistic models appear to perform better than the vector space model. Internally, Bm25 and the QL models seem to perform comparatively for the most part as far as queries with unique words and no duplicates are concerned. The resulting ranked lists produced by all 3 models tend to show large overlaps and comparative scores for all the documents.
In terms of dealing with edge case queries such as single word queries (Q7), my observation was that the dirichlet model outperformed the others and produced a list where the top 2 entries contained the highest occurrence counts for the query term as compared to the other models' top 2s. This may be because QL models are inherently good at scoring relevant documents higher than irrelevant documents (due to the extra contributing term in the equation) as compared to the other models, but dirichlet also takes into account the document length. I believe it may have performed better because longer documents tend to produce smaller negative scores (when combined with term frequency) and have a higher likelihood of containing the most number of occurrences of the term .
I also noticed that all none of the models performed great on the query containing repeated terms (Q6) due to all of them being bag of words models and none of them taking positional information into account (Ql-dir still ranked it the highest amongst the 4)