Information Retrieval Project Three

By Ben Sikora

Complete

25 Hours

This assignment was relatively straight forward. The only thing I wish I had at the start was an example experimental plan, but I was able to figure this out by visiting office hours and the feedback from Professor Goharian. I also wish there were just a few more examples in lecture to help solidify some concepts.

Design Document

General Structure

My goal for this project was to try to integrate as much of query reduction and expansion into project two as possible. This way I would be able to compare the results from my project two with a high degree of confidence and, if there were any changes needed to be made to query scoring, reduction and expansion would not need to be changed. It was because of this that no new classes were created for this project, and the entire structure relies within the **Evaluator** and **queryBuild** classes. For the evaluator class specific query expansion, reduction, and combined methods were added but the relevance calculation methods for BM25 or the language model were not modified. And, for the queryBuild class, there were only changes to the both the query file reading and to adding of term weights. Overall, I was pleased with how integrated my project as I was able to use many of the methods I already had in place from project two.

Overall Design Decisions

1) *Using Only Single Indexing*

I chose only to use single indexing, instead of stemming, phrases, or positional, because since the goal of this project is to examine the effects of query reduction and expansion, I thought it would be best to only use one indexer for comparison. In addition, I was also considering these factors:

a. Phrase and positional indexing don't inherently apply for query expansion.
b. Phrases have incredibly limited terms.
c. Addition of stop words did not make much of a difference in the results of project 2.
d. The Stem indexing results from project 2, were highly inconsistent and did not seem to have that much of an effect

2) *Using Only BM25 and LM*

Again, since the goal for this project is to experiment with reduction and expansion, I thought it was best to minimize other parameters as much as possible and chose only to use BM25 and my language model. In addition, cosine had the worst performance in my project 2 results. As a reminder, for my language model, I used Dirchlet Smoothing.

3) *Default Parameters*

The default parameters for my project are 5 documents retrieved per query and 5 terms per document. The default query threshold is 0.6. These defaults were chosen as they provide the best performance with BM25 across each of the methods (Table 1, Table 5, and Table 7). If you ever want to change these defaults, please consult the read me for this project as you can do so from the terminal.

Program Flow and Design Decisions of Query Expansion

For Query Expansion, I chose to do the Rocchio Vector Space Relevance Feedback without the subtracting of non-relevant terms (thus a y of 0). The entire structure of query expansion lies in the "queryExpansion" method of the **Evaluator Class.**

1) *Reading Files in and Scoring*

All the reading of files and initial scoring was the same as for project two. The relevance method used is either BM25 or the language model.

2) Choosing The Terms to Add to Query
The actual expansion of the query is contained within the "makeExpansionQuery" method. I first rank the documents using the scoring of the specified relevance calculation. Then taking the specified top docs, I loop through each documents' terms and calculate their relative term importance score using the "topTermsInDoc" method. The importance score I used was the product of idf and the number of times the term appears in the selected relevant document set. For each document, I rank its terms using the importance score and select the top terms to be added to the query. When adding expanded terms to the query, I used only a 0.05 beta weight which was represented in the query term frequency (*Note Table 4 for this Design Decision*).

3) Rerunning Scoring with New Expanded Query
After the new queries have been created, I clear and rerun the scoring of the documents with the new queries and print the ranked results.

Program Flow and Design Decisions of Query Reduction
For Query Reduction, I chose to do a percentage total of the query terms based on term idf. The entire structure of query expansion lies in the "queryReduction" method of the **Evaluator Class.**

1) Reading Files In
All the reading of files in was the same as before, except for the reading of the query file. Since we are reading the narrative section for query reduction, I built another method in the **queryBuild** class called "tagParserReduction" to account for this.

2) Modifying and Reducing Query
After creating the initial query terms, the reduction of the query happens in the "makeReducedQuery" method. The query terms are ranked by their idfs, and then a new query is created with the specified query threshold.

3) Rerunning Scoring with New Reduced Query
After the new queries have been created, I score the documents using the specified retrieval method and print the ranked results.

Program Flow and Design Decisions of Combined Query Expansion and Reduction
For the combined Query Reduction and Expansion, I chose to run reduction first and then expansion. The entire structure of combined query reduction and expansion lies in the "queryReduction" method of the **Evaluator Class.**

1) Reading Files In
As I am using query reduction, I read all the files as normal but used the "tagParserReduction" to read the narrative section of the queries.

2) Query Reduction, Expansion, and Results
I run query reduction first and score the documents. Then using the query from reduction and the scored documents, I select the top documents and terms for expansion. After creating the new

query, I calculate the scores one final time and print the results. All the methods used are from above.

Results And Analysis

Report 1: Query Expansion

Table 1: Original results of single indexing BM25 and Dirchlet Smoothing from Project Two. No query expansion or reduction.

Retrieval Method	Relative Retrieved	MAP	Query Processing Time (Sec)
bm25	87	0.4191	1.82611
dirchlet	75	0.4423	3.02106

Table 2: Results of Query Expansion for BM25, using Rocchio Vector Space Relevance Feedback with beta weight of 0.05, across varying parameters on single indexing.

Documents Retrieved for Feedback	Terms Per Document for Feedback	MAP	Number of Relevant Documents Retrieved	Query Processing Time
3	1	0.4325	87	1.863713982
3	3	0.4245	87	1.951806051
3	5	0.4371	89	1.974467336
5	1	0.4189	87	2.091402856
5	3	0.4268	89	1.927354455
5	5	0.4413	91	1.997224642
7	1	0.4116	87	2.004459997
7	3	0.4227	89	2.420348336
7	5	0.3976	92	2.173602424

Table 3: Results of Query Expansion for Dirchlet Smoothing, using Rocchio Vector Space Relevance Feedback with beta weight of 0.05, across varying parameters on single indexing.

Documents Retrieved For Feedback	Terms Per Document for Feedback	MAP	Number of Relevant Documents Retrieved	Query Processing Time
3	1	0.4063	70	4.386746921
3	3	0.3064	62	5.13473256
3	5	0.3366	58	5.752225279
5	1	0.3907	74	4.757823647
5	3	0.3968	77	5.707083803

5	5	0.3658	79	7.119392093
7	1	0.3955	72	4.734209327
7	3	0.3255	84	6.511535622
7	5	0.3078	78	8.337672559

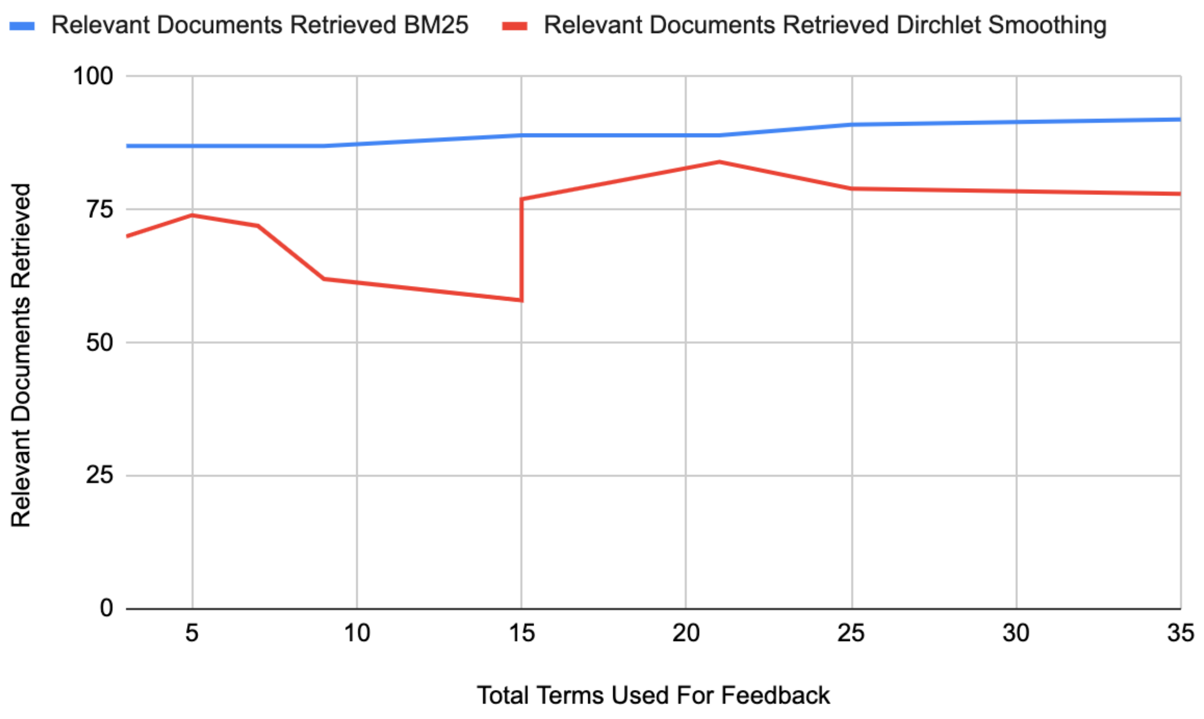

Figure 1: Number of Relevant Documents Retrieved against Total Terms Used for Feedback comparison for BM25 and Dirchlet Smoothing.

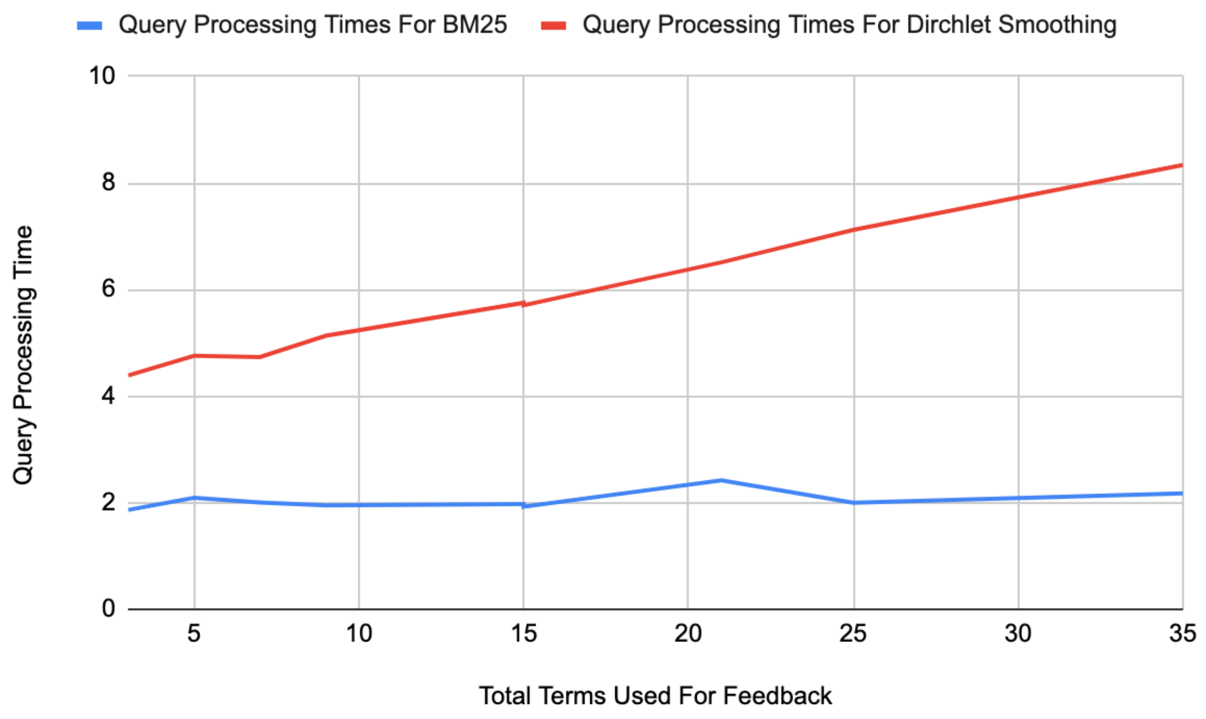

Figure 2: Query Processing Time against Total Terms Used for Feedback comparison for BM25 and Dirchlet Smoothing.

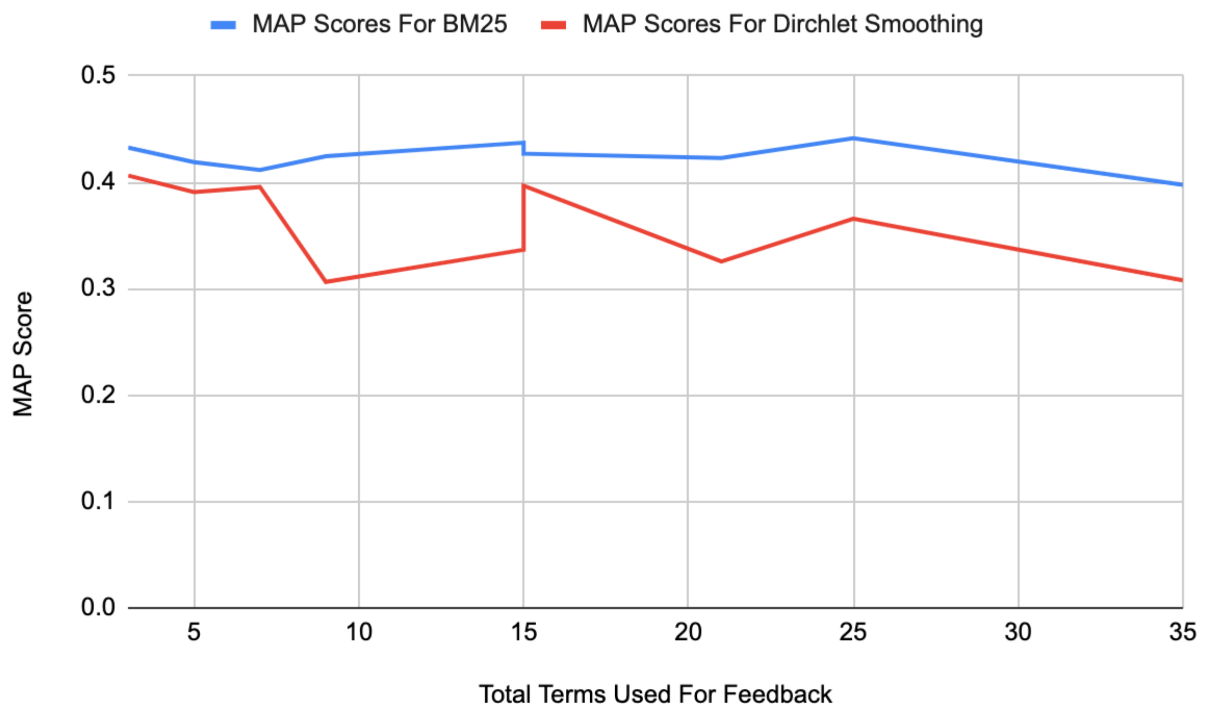

Figure 3: MAP Scoring against Total Terms Used for Feedback comparison for BM25 and Dirchlet Smoothing.

Goal of Query Expansion
My goal for query expansion was to increase the number of relevant documents retrieved—i.e obtaining a higher recall—while minimizing drops in precision. I am expecting lower precision because increasing the number query terms inherently comes with the chance of bringing in a non-relevant term. The parameter modifications are all aimed at finding an effective balance of recall and precision.

Summary of Results
Overall, it appears that query expansion was able to retrieve a good number of relevant documents with only minor drops and even some improvements in my MAP scoring. For BM25, while the parameters that added less query terms (7 documents and 1 term or 3 documents and 3 terms) had no relative change in performance, there was around two to five more relevant documents retrieved for higher number of terms added (5 documents and 5 terms).

For dirchlet smoothing, the results were not as clear. The MAP score was consistently much lower than the baseline, sometimes dropping 0.12 below. And the number of retrieved documents actually decreased with more terms being added until about 15 total terms. Only after this 15-term mark, does the model return to around 2 or 3 more relevant documents being retrieved with an enormous 9 relevant document increase at specifically the 7 doc and 3 term mark. Unlike BM25, dirchlet's also had massive increases in query processing time compared to the baseline. As the total number of terms increases in dirchlet, the processing time would continue to rise and maxing out at about 5 more added seconds (Figure 2). This aligns with what I expected, since dirchlet smoothing calculates a score for every document and does not skip terms that are not in a specific document, adding more terms dramatically would dramatically increase the number of calculations and subsequently the processing time.

Overall Takeaways of Query Expansion
Overall, query expansion performed as expected. As more terms were added to the query, more relevant documents were retrieved (Figure 1), as more relevant terms could find more relevant terms, but after a certain a point there would be significant drop-off in performances (Figure 2 and 3) as non-relevant terms would begin to dominate and the query would become too large. For BM25, I believe that the best parameter was 5 docs and 5 terms, as this was able to bring in second highest number of relevant documents retrieved and increased the MAP score by 0.0222. And for dirchlet smoothing the best model was 5 docs and 3 terms because this brough in 2 more relevant documents, didn't have too much of an increase of query time, and had the least amount of drop in precision. Overall, I would prefer BM25 because I was able to increase the recall without sacrificing any precision.

Differences in Performance of BM25 and Dirchlet Smoothing
I believe that the drastic differences in BM25 and dirchlet smoothing performance have to do with what these retrieval methods are prioritize in their scoring. Dirchlet smoothing, unlike BM25, does not consider query weights. Therefore, one non-relevant term that is added to the query has a much greater chance of poisoning or ruining the results. This explains why dirchlet smoothing struggled retrieving documents with less total terms added (Figure 1) because adding a couple non-relevant term in expansion would make up a larger proportion of the total pool. Dirchlet scoring also explains why the 5 document and 3 term parameters had a significantly

higher relevant documents retrieved than the 3 document 5 term parameters, despite having the same number of total terms added. Since the first few terms across many relevant documents are most likely more relevant than many terms in only a few documents, it makes sense why parameters that have more documents and less terms perform better under dirchlets.

BM25 did not have as much a drop in performance with non-relevant terms being added because BM25 can consider query weights. The original query terms, with their higher query weight, still dominate the scoring and why I believe the MAP did not really see a drop in performance. The added expansion query terms, with the smaller query weight, simply serve to score relevant documents so that they are retrieved but are not highly ranked. The lower weights also serve to dilute non-relevant terms because duplicate terms added, which are more representative of relevant terms, are stacked and then given higher preference in scoring. Overall, as BM25 can consider query weights and had such a better increase in performance, it is by far the stronger retrieval method for query expansion.

Drawbacks of Rocchio Vector Space Model and Possible New Solutions
One inherent drawback of the Rocchio Model is that we are marking documents as relevant by only using the scoring of the retrieval methods, and that we are arbitrarily choosing the relevant terms from each document. There is therefore a high chance, under this model, of adding non-relevant documents to the query. I would be interested in exploring two other query expansion methods to work around this. First, since we know that the original query contains relevant terms, I would be interested to see how a thesaurus-based query expansion would perform. As a thesaurus expansion approach would not add duplicate terms or any completely non-relevant terms, I would predict that dirchlet smoothing would also have a better overall performance. Another possible query expansion would be to have the exact same process for Rocchio Space Model but pause for user feedback. This way the user could pick out for themselves non-relevant terms and perhaps even change the weights of terms to match what they are looking for. Both of a thesaurus and user-based query expansion carry with them their own assumption, but I still would be really interested to compare their performance with the Rocchio Model and within the limitations of the BM25 and dirchlet methods.

Table 4: Beta Weight Comparison of Query Expansion using the top 5 terms of the top 5 documents. Single Indexing BM25.

Beta Weight	MAP	Number of Relevant Documents Retrieved
0.05	0.4413	91
0.25	0.404	90
0.5	0.3974	90
0.75	0.391	90
1	0.3874	90

Choosing a Beta Weight for Rocchio Vector Space Model
Initially, I was not planning to look at Beta Weight, but I found that it had significant impacts on my MAP scores and even slight changes on the relevant documents retrieved. For examining weight, I chose specifically to look at the parameters that gave the best performance for BM25

expansion which were 5 documents used and 5 terms per document. After trying various weights, I decided to move forward with a beta weight of 0.05 because it had the highest MAP and the highest number of relevant documents retrieved. I believe that the lowest beta weight ended up working the best, because the weight is still large enough to distinguish scoring of higher documents and rank documents that otherwise would have been discarded, but it is also low enough it does not impact the scoring of the original queries which remained at a weight of 1. I did not need to test beta weights for dirchlet because, as dirchlet does not take into account query weights, the changes had no effect on my results.

Report 2: Query Reduction

Table 5: Query Reduction BM25 using single Indexing across various query thresholds.

Query Threshold	MAP	Number of Relevant Documents Retrieved	Query Processing Time
0.2	0.1939	44	1.813435457
0.4	0.389	92	1.826233339
0.6	0.4006	95	1.989952179
0.8	0.4227	95	2.244598812
1	0.4589	97	2.571617148

Table 6: Query Reduction Dirchlet Smoothing using Single Indexing across various query thresholds.

Query Threshold	MAP	Number of Relevant Documents Retrieved	Query Processing Time
0.2	0.1753	31	2.276869216
0.4	0.2927	63	3.196094559
0.6	0.3294	71	5.542914768
0.8	0.2723	63	9.590574548
1	0.2664	61	17.86199462

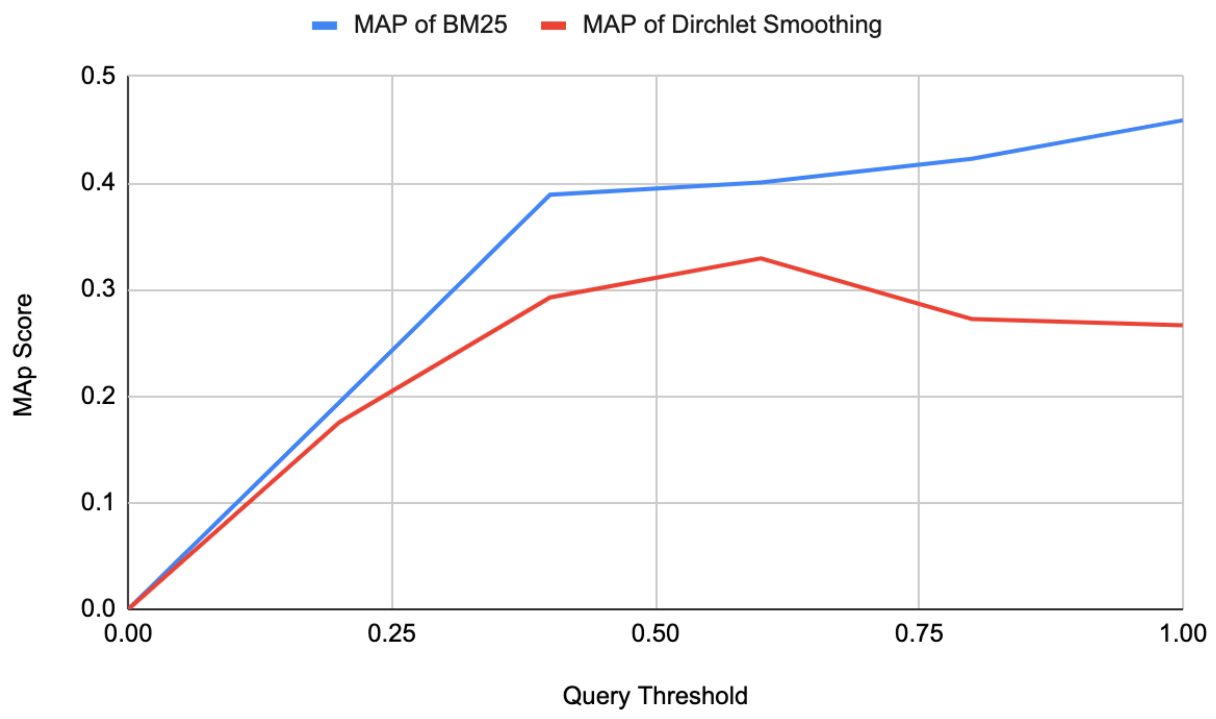

Figure 4: MAP Scoring against Total Terms Used for Feedback comparison for BM25 and Dirchlet Smoothing.

Goal of Query Reduction
My goal for query reduction is to minimize the query size in an effort to decrease processing time while still maximizing performance as much as possible. I am expecting that the top query terms will have significant increases in performance, as they are the most relevant terms, but as more less relevant query terms are being added the performance increase rate will start to drop off.

Results of Query Reduction and Takeaways
While initially both BM25 and dirchlet smoothing performed as expected, with their performance increase rate starting high and slowly decreasing, after the 0.6 query threshold both had unexpected trends. For BM25, the additions of the last 0.2 and 0.4 of query terms increased the MAP score at a higher rate than earlier jumps. And for dirchlet smoothing, the MAP score actually began to decrease after the 0.6 threshold. Also, while the processing times of dirchlet smoothing had dramatic differences between the thresholds with a 15 second total range, the differences in query processing times of BM25 were minimal with a total range of only 0.7 seconds.

Analysis of Query Reduction
Again, I believe that the unexpected trends and differences of BM25 and dirchlet smoothing depend on the constraints of their scoring. As mentioned in Report 1, Dirchlet smoothing does significantly more calculations than BM25, calculating scores for every document and not skipping terms when they are not in a document, so it makes sense why the spread of processing times is much larger. And although a beta weight was not used for reduction, query term

frequencies were, and I believe this is what contributed to the varying trends for both methods towards the higher thresholds. For dirchlet smoothing, since it does not consider query term frequencies, it cannot distinguish between the importance of terms in the query and as a result cannot distinguish queries with a high number of terms. This is the same reason I believe there was drop-off for more total terms (*Figure 3*) and why dirchlet had an overall worst performance (*Table 3*) in Report 1. For BM25, the rate change is a little more complicated. BM25 does depend on query term frequencies to determine term importance, but term frequencies were not included to rank the query terms, only idf. Therefore, terms that were selected as more important by the ranking algorithm, might have been crucial for BM25 performance because of their frequency. So, when those terms were eventually added, towards the end of the threshold, they delivered drastic increases in MAP performance.

Takeaways For Query Reduction
Overall, for BM25, I believe the optimal threshold was 0.6 as this was still under the 2 second mark and gave a high MAP scoring and relevant documents retrieved. Yet, if you are not incredibly concerned with processing time, I could still see an argument for 0.8 or even keeping the entire query as these the delivered the highest performance and still had a relatively low processing time. For dirchlet smoothing, the only two viable options are a 0.4 and 0.6 threshold. While 0.6 threshold had the best overall performance and an average processing time, I would ultimately choose a 0.4 threshold as this was a whole two seconds less in processing with only 8 fewer relevant documents and a 0.03 drop in MAP scoring.

Report 3: Combination of Query Reduction and Expansion

Table 7: Combination of query Reduction and Expansion using BM25 on single indexing. The Model Ranks of reduction are— 1: 0.6 and 2: 0.8. The Model Ranks of expansion are—1: 5 docs and 5 terms 2: 3 docs and 5 terms.

Query Expansion Model Rank	Query Reduction Model Rank	MAP	Number of Relevant Documents Retrieved	Query Processing Time
1	1	0.3751	95	2.439514018
2	1	0.4139	94	2.294585997
1	2	0.4529	96	2.890993044
2	2	0.4498	96	2.78585649

Table 8: Combination of query Reduction and Expansion using Dirchlet Smoothing on single indexing. The Model Ranks for reduction are— 1: 0.4 and 2: 0.6. The Model Ranks for expansion are—1: 5 docs and 3 terms 2: 5 docs and 5 terms.

Query Expansion Model Rank	Query Reduction Model Rank	MAP	Number of Relevant Documents Retrieved	Query Processing Time
1	1	0.2279	71	6.352766261
2	1	0.2701	73	7.059128391
1	2	0.2072	63	9.980236998
2	2	0.2109	59	11.71948262

Goal of Query Reduction and Expansion
My goal for query reduction and expansion was ultimately to try to take the best features of both: limiting the number of query terms, to minimize processing time, while still getting the benefits of increased recall in expansion.

Analysis of Results of Query Reduction and Expansion
Overall, I was really surprised with the results of combined query reduction and expansion. For BM25, there was no improvement to recall, and the MAP scores saw an overall increase. As query reduction does not involve term weighting and expansion does, I believe that their addition helped to distinguish between some documents at the top of the ranking and ultimately what led to the MAP scoring differences. My goal for query reduction and expansion, however, was to increase recall and that was where I saw the least improvement. I initially thought it was perhaps that I wasn't returning enough documents, as I could now return significantly more documents with the higher number of query terms, but when I reran the table with a higher number of documents returned it made no difference in comparison. I did also notice, however, that I am also approaching the total number of relevant documents in the entire collection, 109. It could just be that I need a more powerful expansion algorithm, such as thesauri, user-based, or even neural networking, to obtain those last relevant documents.

For dirchlet smoothing, the results were incredibly unexpected. My higher model query reduction rank showed the highest increase in recall, retrieving around 10 more relevant documents, but my lower model query reduction rank had a significantly decreased recall, losing about 10 relevant documents. The overall performance of both models also fell from their respective baselines in Table 6. The query processing time increased by 3 to 5 more seconds and the MAP score dropped from 0.05 to 0.1. Again, I believe the inconsistent performance of dirchlet's has to do with its scoring. By leaving out query weights, and only increasing the overall term pool, it is hard to distinguish term importance and the precision and processing times of dirchlet's suffers. The reason the lower reduction model rank had a higher recall is because it started off with a lower term of terms and thus the addition of more had a lower chance of ruining performance.

Takeaways of Query Reduction and Expansion
To conclude, while you can see some performance improvements with the addition of query expansion to reduction, in all cases it is much better to stick with query reduction and just raise the threshold. The increased processing time is much better utilized for more consistent performance in reduction, rather than the results of the expansion. And intuitively this makes sense because, by increasing the threshold, we are adding terms to the query that we know are relevant compared to using expansion and only guessing at relevant terms to add.

Conclusion
Overall, I am really pleased with how this project went. The actual coding was much more manageable compared to the other projects, and it was actually really fun to play around with all these different parameters and try to infer what was going on. As with every project, I feel I also have a much more solidified understanding of the concepts and exactly how query reduction and expansion are implemented. Now I only wish that I could see other people's experimental plans and results to see how different implementations or parameters affected their performance.

Also, since this is the end of the year, I just want to give a big thank you to Professor Goharian, Shabnam, and Sajad! I was initially really intimated, at the start of this class by the amount of work and content but you all really made this class manageable and fun to learn in. I always felt that if I was ever lost or confused, I could stop by office hours for help, and that reassurance gave me the confidence to try my best. Thank you all again for a great semester!