Jerry Lee

Professor Goharian

12/9/2016

Status of Assignment:

Incomplete:

I did not finish implementing a query reduction algorithm. However, I did finish implementing query expansion for my Cosine VSM search engine using the pseudo-relevance feedback algorithm detailed in class. As such, I have written out my experiment plans and results only for query expansion.

The query expansion program allows you to specify any value for N and T to use with my Cosine VSM engine. I improved my cosine VSM search engine from project 2 to reach a MAP of 0.3745, allowing it to be used to assess query expansion. My analysis discusses how Mean Average Precision varied at different levels of N (number of top documents selected for query feedback) and T (number of top terms selected for query feedback). These were evaluated using the TREC Eval software.

Time spent on assignment:

45 Hours

Things you wish you had been told prior to being given the assignment:

Design Document

The query expansion process is executed in the queryExpansion.py program. The program first parses the query an initial time, and then passes it to an object of the pseudoRelator() class, where query expansion and subsequent retrieval occurs. I implemented the pseudo-relevance feedback approach detailed in the class slides. The expansion process in the pseudoRelator() class is as follows:

1. **Perform an initial Cosine VSM Retrieval:** First, for every query, we retrieve the N most relevant documents using our Cosine VSM engine. This initial search is performed by the initialCosineVSM() class, which is called in the pseudoRelator() class. initialCosineVSM() has the same design as my cosine VSM engine from project 2, but it returns an array of the N most relevant documents for each query rather than outputting it to a document. The user specifies N and T using command line arguments.
2. **Count up the top T terms for each Query:** After our initial retrieval, we go through each of the N documents corresponding to a query and count up the most common terms in the documents using a python dictionary. After doing so, we select the top T terms by frequency in each document, recorded them into a string, and appended the string to their corresponding original queries.
3. **Perform a second Cosine VSM Retrieval with Expanded Queries:** After our query expansion occurs, we then evaluate each expanded query using the Cosine VSM engine from project 2 and output the results to to a .txt file to be tested with TREC EVAL. The results are all written into the /rankings folder under the name "expandedCosineVSM.txt"
4. **Evaluate Precision with Trec Eval:** Once the output file has been created, we then evaluate the search precision with Trec eval.

Experimental Plan:

My experiment plan was to compare the Mean Average Precision of my Cosine VSM retrieval engine using TREC Eval at varying levels for N and T. The results are summarized in line graph and table below. We evaluated for N = 1, 3, and 7 and for T = 1, 2, 3, 4, and 5. By doing so, we could see the effect of changing T while holding N constant, as well as the effect of changing N across various T levels.

The baseline metric was the cosine VSM engine run on just the queries without any expansion. All the queries processed were the "title" queries in the TREC dataset.

My expectation was that MAP would increase up to a certain T value, before encountering diminishing returns. I also expected the same relationship for increasing the value of N: a greater value for N would correspond with a higher MAP across all values for T before reaching a point of diminishing returns. Finally, I believed that introducing query expansion would provide considerable increases in precision compared to not having any degree of query expansion for a wide range of values of N and T.

Results from Pseudo Relevance Feedback

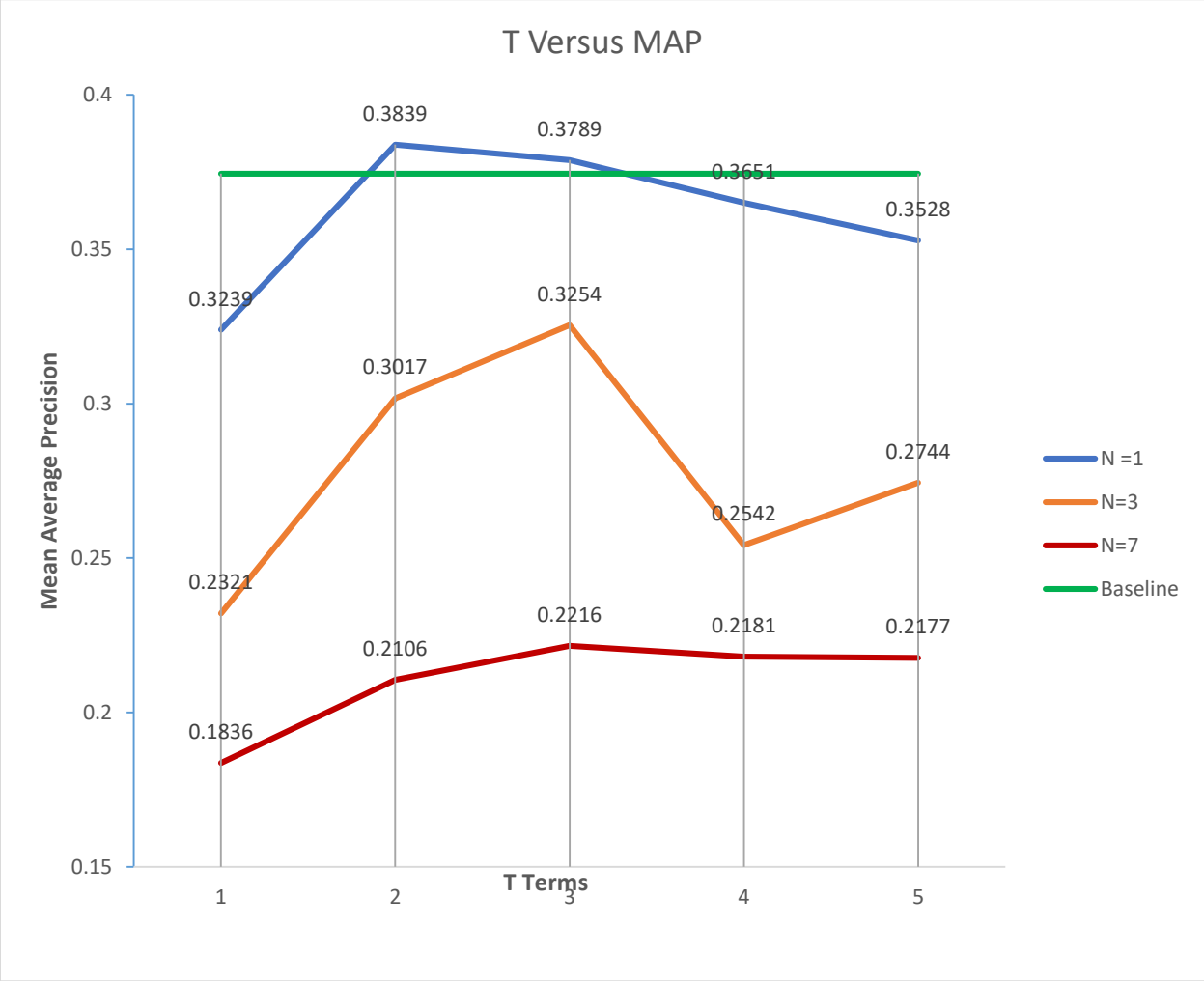

Figure 1. A line graph comparing the MAP of the Cosine VSM search engine at different levels of N and T. T varies from 1 to 5 terms. N was tested for values of 1, 3, and 7 documents. Each line represents MAP at different levels of T for a set N value. The baseline is the green, horizontal line with a MAP of 0.3745 to be compared for any T level.

Expansion with varying N and T	Mean Average Precision	R-Precision, Exact	Runtime (seconds)
Baseline: Cosine VSM on title	0.3745	0.3224	129.7974
N = 1, T =1	0.3239	0.2602	259.1515
N = 1, T=2	0.3839	0.3347	293.6800
N = 1, T = 3	0.3789	0.3316	309.2316
N = 1, T = 4	0.3651	0.3297	292.4645
N = 1, T = 5	0.3528	0.3156	302.4047
N = 3, t = 1	0.2321	0.1797	302.3958
N = 3, t = 2	0.3017	0.2412	311.4694
N = 3, t = 3	0.3254	0.2850	320.8777
N = 3, t = 4	0.2542	0.1683	320.6779
N = 3, T = 5	0.2744	0.2166	329.3613
N = 7, t =1	0.1836	0.1407	297.3713
N = 7, t = 2	0.2106	0.1498	344.8245
N = 7, t = 3	0.2216	0.1679	331.5881
N = 7, t = 4	0.2181	0.1617	348.9288
N = 7, t = 5	0.2177	0.1506	339.4044

Figure 2. A table of mean average precision, R-precision, and runtime for different values of N and T.

From the above chart and table, we find that my implementation of Pseudo-Relevance feedback actually reaches a point of diminishing returns very quickly for both N and T. Increasing the number of docs retrieved, N, only improves search precision over our initial non-expanded query search when N = 1 and T = 2 or 3. Our greatest mean average precision value was 0.3839, which was achieved when N = 1 and T = 2.

Increasing N past 1 seems to decrease the mean average precision of our search engine. The higher the value of N, the lower our mean average precision scores were across all values of T. On the topic of number of terms retrieved, T, it appears that precision for any given value of N was highest when T was equal to 2 or 3. Before or after those thresholds for T, precision was not optimized at a given N level.

From my results, it appears that pseudo-relevance feedback query expansion has a very narrow optimal range for number of documents re-fed and number of top terms re-fed. The only cases where my search engine with query expansion outperformed by baseline search engine was when the algorithm retrieved the top document for each query (N=1) and the top 2 or 3 terms (T = 2, T= 3). In all other cases, the search engine performed better without any query expansion. Query expansion is also a much more time-consuming process, and the search retrievals took considerably longer when we included expansion. We must note, however, that we did not test for N = 2, which may have had higher MAP scores than N=1 across the T values. This should perhaps be tested later on.