Jordan Rodak
Pallavi Singh

CS 276 – PA 3 Report

Final Results:

In this assignment, our various ranking functions achieve the following NDCG scores on the training and development data sets:

	Baseline	Cosine	BM25F	BM25F w/windows	BM25F w/extras
Training	0.77612	0.86512	0.89015	0.89257	0.89157
Dev	0.75899	0.84427	0.87703	0.88028	0.87786

Discussion of Weight Tuning:

For the Cosine scorer and basic BM25F scorer, we have the following weights:

task1_W_url task2_W_url	task1_W_title task2_W_title	task1_W_body task2_W_body	task1_W_header task2_W_header	task1_W_anchor task2_W_anchor
8.0	6.0	1.0	7.5	1.2

For the BM25F scorer with smallest window boosting we have the following weights:

task3_W_url	task3_W_title	task3_W_body	task3_W_header	task3_W_anchor
9.0	7.0	1.0	7.5	1.2

To obtain these values, we had the intuition that the body should receive the lowest weight since the other zones are more sure indicators of relevance if query words occur in these zones. As a result, we set W_body = 1.0, and began tuning the other parameters based on NDCG scores on the training data set.

We decided to give significantly higher values to W_url, W_title, and $W\_header_{r,}$ because query terms occurring in these is a better indicator of relevance than a term occurring in the body.

W_url was given the greatest value because often search queries are navigational and directed at reaching a specific website. For example, results for the query “facebook” may include the Facebook site itself but also news articles and other pages discussing Facebook. The query is almost certainly an attempt to reach Facebook.com, so giving the url high weight will help ensure it is ranked higher than all other results. An equivalent example specific to the Stanford domain our ranking system is tested on is the query “Stanford housing”, for which we would want housing.stanford.edu to be the first result.

Based on experimentation we discovered that results were better when W_header > W_title. One possible explanation for this is that, although all documents do not have headers, they are great indicators of the contents of a document when present. A document with a header/section that is very relevant to the query would then generally be ranked higher than a document that is mostly unrelated to the query but may have a title that shares a couple words in common with the query.

Finally, the value of W_achor_{r} is not as critical. Very small and large values for W_anchor result in drops in results. This shows that considering anchor data does help ranking some, but should not be heavily relied upon in the ranking function. On the training data, values of W_anchor from 1.0 to about 1.5 gave acceptable results, with a slight peak around 1.2.

Jordan Rodak
Pallavi Singh

Additional parameters for BM25F:

In addition to the previously mentioned weights for the various fields of the document, we also have the following parameters for length normalization:

task2_B_url	task2_B_title	task2_B_body	task2_B_header	task2_B_anchor
0.6	0.6	1.0	0.6	0.6

task3_B_url	task3_B_title	task3_B_body	task3_B_header	task3_B_anchor
0.8	0.8	1.0	0.8	0.8

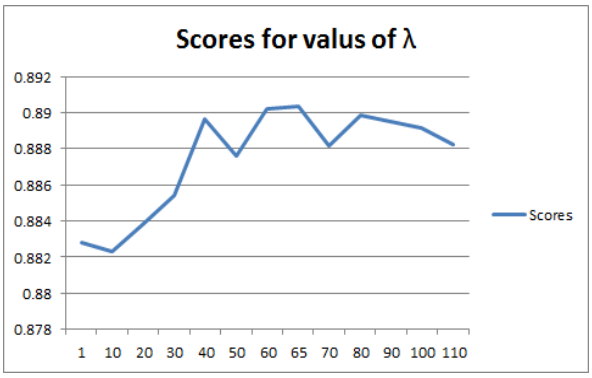
And the following parameters to be used by the function V_j for considering pagerank:

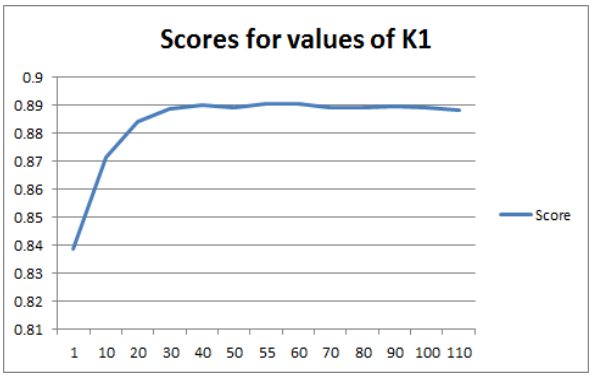
task2_λ	task2_λ'	task2_K_1	task3_λ	task3_λ'	task3_K_1
65.0	1.0	55.0	65.0	1.0	90.0

Using no length normalization (i.e. all B values set to 0), we get a basic BM25F NDCG score of about 0.867 which is slightly lower than our final result. Through incremental testing, we noticed that the NDCG scores were maximized when the body was fully normalized for length while the other zones were normalized less. Exact values for which scores were maximized are in the tables above.

To tune the parameters λ, λ', and K_1 we first had to decide on a V_j function for considering pagerank values. We used log(pagerank + λ'), and the reason for this choice will be discussed further below. We set λ' = 1, so that if pagerank = 0, then also $V_j = 0$. This is the correct choice, because for a page with rank 0, we want to apply no bonus to the score, and then for ranks > 0, we add some small bonus to the score.

Next, to determine reasonable ranges for the values of K_1 and λ, we tested our BM25F scorer for a variety of values and then did more fine tuning once approximate optimal values were found. The results of this testing for BM25 are in the graphs below.



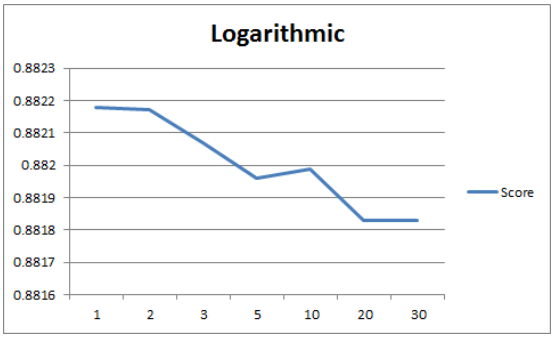


These graphs confirm our initial intuition that values should be moderately large for two reasons. The value of K_1 determines whether term frequencies are treated as boolean (present/ not present) or as actual frequencies. When K_1 is 1, the frequencies are treated as boolean, so we want a large value for K_1. Additionally, a large value for λ creates a larger gap between the bonuses for the different values of pagerank. This is desirable because it may help with ranking two similar documents with different pagerank scores.

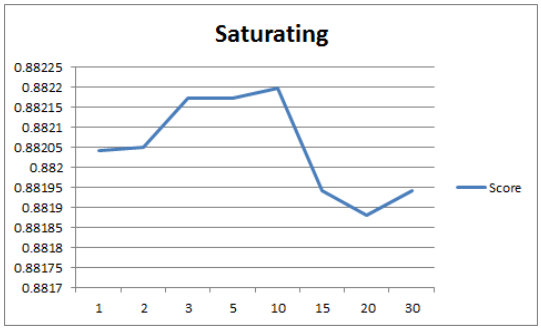
Choice of V_j

To determine an appropriate V_j, we initially set λ = 1.0 and testing three functions for various values of λ' to see which one seemed most promising. Once the function had been determined we tuned λ appropriately.

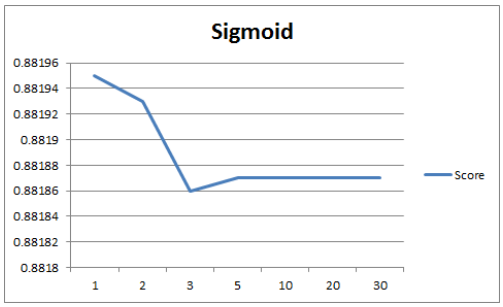
log(pagerank + λ'):



pagerank / (pagerank + λ'):



1/λ' + exp(-pagerank):



These graphs show that the sigmoid function is likely the worst choice of the three and that the maximum values using a logarithmic or saturating function are very similar. Ultimately, because of literature suggesting that a logarithmic function generally works well drove us to choose it for V_j. With V_j chosen we then tuned λ to maximize NDCG scores on the training data.

Smallest Window Boosting:

While tuning the parameters for this scorer and trying various functions, we found that the reasonable range for B was from 0.05 to 0.5. Values much smaller or larger for any function did not give results better than the basic BM25F scorer.

When all of the query words are found in a window the same size as the query, we boost the score of the document by multiplying by 1.0 + B, a parameter tuned for this task. If this window size is greater than the size of the query, we then multiply the score by a factor between 1.0 + B and 1.0. To determine this factor we define a boosting function.

We tried two different boosting functions. The first that we tried considered the difference between the window and query sizes, and then outputs a boosting factor that decreases linearly with the difference in sizes. This function is as follows: 1.0 + B - (win_length - q_length) * boostmod. If the computed value was less than 1.0, 1.0 was used instead.This function gave the following NDCG scores for the following values of B and boostmod:

B	boostmod	NDCG Score
0.5	0.01	0.88989
0.05	0.001	0.89012

For these values and all others we experimented with, the NDCG score did not surpass that of regular BM25F. Then we used the following function: 1.0 + B/(win_length - q_length + boostmod), which gives the following result:

B	boostmod	NDCG Score
0.1	0.5	0.89257

This result, which was the maximum we obtained during tuning outperforms regular BM25F, so we used this boosting function.

Other metrics to consider/ Extra Credit:

We have tried to incorporate various metrics in our algorithms to improve the document rankings as discussed below:

1. Minimum coordinate factor:

This factor gives more scores to documents having the first found term closer to the beginning. So, we calculated the first index of the first appearing query term in body text (ignoring the stop words).
Function used: Score = bm25Score * (1+e^(-c)) where c= closestIndex
Result (Negative): One of the reason, it was not giving better score was because, we already have title and header fields which are assumed to be the most closest text to the beginning of the document and they are already weighted high in bm25 scores.

2. Distinct terms factor:

This factor is a generalized version of smallest window signal. Here, we calculate the number of distinct query words found in document fields. So, the cases where smallest window will give zero boost to the documents, here we can get a positive boost depending on the the extent of the number of distinct found words in document.
Function used: Score = bm25Score * (0.1 * e^u) where u= number of distinct query terms in doc
Result (Positive: 0.15% increase in score)

3. URL relevancy:

When analysing the corpus, we notice that the urls represent the document in a concise and unique way. So, we applied following methods on urls to extract the maximum possible matches with query terms:

A. Split URL on alphanumeric characters.
B. Stemming url terms to find variants of the query term.
C. Find possible query term candidates by splitting url terms on spaces.
D. Ignoring stopwords in url terms.

Score = bm25Score * (1.0 + m*0.01)where m = number of Matches with query terms
Result: 0.25% increase in scores: This signal gave us the maximum boost with bm25 score.

4. Part-of-speech Tagger

We experimented speech tagger on sample query from the resource http://nlp.stanford.edu/software/tagger.shtml. We applied differential boost to different tags and scores sample document. The approach looked promising but we could not complete the full implementation because of lack of time.