Benjamin Au (bau227)
Mikhail Sushkov (msushkov)

CS 276
5/15/14

PA3 Report

Task 1: Cosine

In task 1, we implement the cosine scorer. To do so, we calculate the term frequencies from the PA1 corpus. To normalize the terms, we remove non-alphanumeric characters and convert the strings to lowercase. We calculate baseline performance by using values of 1 for each of the W parameters and 500 for the *smoothingBodyLength* parameter to reach performance of 85.94% train; 83.77% dev. Tuning parameters based on intuition of field importance boosts performance to **87.74% train**; **84.21% dev**, providing significant boost of +1.80% train; +0.44% dev. The parameters that give the optimal accuracy for the training set are below.

task1_W_url = 1.5
task1_W_title = 1.3
task1_W_body = 0.3
task1_W_header = 1.5
task1_W_anchor = 1.5
task1_W_smoothingBodyLength = 30000

Figure 1. Optimal parameters for task 1 (cosine).

Question 1

What was the reasoning behind giving the weights to the url, title, body, header and anchor elds for the three tasks? Were there any particular properties about the documents that allowed a higher weight to be given to one eld as opposed to another?

Our intuition suggested that body would have the least signal, given that it is the least structured piece of content, whereas the others (url, title, title, header) provide more signal given that they are more succinct. In addition, we suspected that body length would be a noisy signal, so reducing its impact by increasing the body length smoothing parameter would improve performance. We did not suspect the tuned smoothing parameter to be so high (30,000), but that could be due to a reduced body length signal improving overall performance.

Question 2

What other metrics, not used in this assignment, could be used to get a better scoring function from the document? The metrics could either be static (query-independent, e.g. document length) or dynamic (query-dependent, e.g. smallest window).

Other query-independent parameters:

- Social Media Impact: Number of Likes/Shares/Comments on Facebook/Twitter/Google+.
- Number of total page views, frequency of page views (based on time).
- Frequency of page update.
- Statistics on time spent per page visit per user (mean, median, etc.)
- Number of past visits from a particular user for a given page.

Other query-dependent parameters:

- Document frequency for each term in the document vector (and its associated document vector length normalization) in order to implement the full Cosine scoring model.
- Data from previous queries from the same user (detecting user search patterns).

    - Common query misspellings of a given user.
    - Common query tokens of a given user.
- General (non user-specific) spelling correction/suggestions for the query.
- Query expansion (running a parallel query by replacing query terms with similar, more popular terms to see if the results produced are more relevant).

Task 2: BM25

In task 2, we implement the BM25F scorer, tuning the various parameters and normalization functions accordingly. Baseline BM25F performance with default parameters based on Task 1 results yielded performance of 88.37% train; 85.79% dev.

task2_W_url = 1.5
task2_W_title = 1.3
task2_W_body = 0.3
task2_W_header = 1.5
task2_W_anchor = 1.5
task2_W_smoothingBodyLength = 30000
task2_B_url = 0.75
task2_B_title = 0.75
task2_B_body = 0.75
task2_B_header = 0.75
task2_B_anchor = 0.75
task2_k1 = 1.2
task2_pageRankLambda = 1.2
task2_pageRankLambdaPrime = 1.2
task2_pageRankLambdaDubPrime = 1.2

Figure 2. Baseline parameters for task 2 (BM25).

After tuning the parameters, given in Figure 3, we arrived at performance of **88.69% train; 86.50% dev**, providing a tuning performance boost of +0.22% train; +0.81% dev. This tuning boost is surprisingly small, but still a significant improvement from cosine scoring: +0.95% train, +2.29% dev.

task2_W_url = 1.5
task2_W_title = 1.3
task2_W_body = 0.3
task2_W_header = 1.5
task2_W_anchor = 1.5
task2_W_smoothingBodyLength = 30000
task2_B_url = 2
task2_B_title = 1.0
task2_B_body = 0.75
task2_B_header = 0.5
task2_B_anchor = 0.1
task2_k1 = 1
task2_pageRankLambda = 1.2
task2_pageRankLambdaPrime = 0.5
task2_pageRankLambdaDubPrime = 4.0

Figure 3. Optimal parameters for task 2 (BM25).

Question 3

In BM25F, in addition to the weights given to the fields, there are 8 other parameters, Burl; Btitle; Bheader; Bbody; Banchor, L, L' and K1. How do these parameters affect the ranking function?

For the w parameters for each field type, we used optimal parameters found in Task 1 as the baseline. In addition, we used the recommended parameters of 0.75 for b-type parameters, and 1.2 for k1 and lambda parameters.

We performed manual grid-search for to find optimal values for the parameters. For each of the parameters not already tuned from Task 1, we ran grid search with step-size of 0.2 and range of +/- 3 and arrived at our optimally tuned parameter set.

Given that B-weights are denominator-based constant, their weights are inversely associated with importance of each field type. That is, anchor and header are relatively more important than the other parameters. This makes intuitive sense, given that the information stored in each of these types is more structured and thus hits in any of these types should provide stronger signal.

Question 4

In BM25F, why did you select a particular Vj function?

We tested the three Vj recommended functions using optimally tuned parameters, namely (1) $log\,\lambda_i' + f_j$ (83.32% train), (2) $\frac{f_j}{\lambda_i' + f_j}$ (83.31% train), and (3) $\frac{1}{\lambda_i' + exp(-f_j\lambda_j'')}$ (88.69% train) and found that equation (3), the inverse negative exponential function, boosted performance significantly: +5.37% performance boost from baseline log function.

Task 3: Smallest Window

In task 3, we implement the smallest window scorer and applied it to the cosine scorer from task 1. We precompute the smallest windows for a given query, document pair at initialization so that at lookup we can do a simple hash table lookup to find the minimum window size for a given query and document. In addition, we design the following damping function for smallest window boost weights:

$$Boost = 1 + (B-1) \times exp(-(minWinLen - Q))$$

Figure 4. Damping function for smallest window size boost.

task3_W_url = 1.5
task3_W_title = 1.3
task3_W_body = 0.3
task3_W_header = 1.5
task3_W_anchor = 1.5
task3_W_smoothingBodyLength = 30000
task3_W_B = 2.0

Figure 5. Optimal parameters for task 3 (smallest window + cosine).

In this function, Q is the number of unique terms in the query. This equation is derived from the intuition that Boost range should be between 1 and B, *minWinLen* range is between Q and Infinity, and we want an exponential decay from (B, Q) to (Infinity, 1). This Boost equation models these requirements accordingly.

Question 5

For a function that includes the smallest window as one component, how does varying B and the boost function change the performance of the ranking algorithm?

We similarly tune B using step size 0.5 range 1-10, and find the optimal value of B = 2. This means that having an optimal smallest window in a document doubles the document signal, which is reasonable. We took the pre-tuned w-type parameters from Step 1 and achieved performance of **87.90% train**; **84.28 dev**. This is a minor increase in performance from the baseline cosine scorer with tuned parameters (+0.16% train; +0.07% dev).