# Programming Assignment 3

SUNetID: pchase, chuang39

## Task 1 - Cosine Similarity

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

Web page consists of several sections including url, title, header, body and anchor. Intuitively, each factor has different effect on the relevance/ranking. All factors are independent. In the ranking algorithm, we need to assign weights to each of independent factor and assess the relevance based on them.

The data is partitioned into two sets: training set and dev set. We first tuned the parameters on the training set and then tested the parameters on the dev set to make sure that we weren't overfitting the training set. To tune the parameters, we first guessed the weight of each parameter and started with {urlweight=6, titleweight=3, bodyweight=1, headerweight=1.5, anchorweight=1.5}, which gave us a base NDCG of .8492. We then varied each parameter and moved it to the place that led the highest NDCG, and then moved on to the next parameter. This process can be seen below. For example, for URL weight we varied the parameter and found that W_url = 8 led to the best NDCG. For each parameter, the range with the best performance on the training set is marked in bold. At the end, we tested the parameters on the test set to make sure we didn't overfit the training data. We knew we were not overfitting the training set because the NDCG on the training and dev set were roughly the same.

| url weight | 3 | 7 | **8** | **9** | 12 |
|---|---|---|---|---|---|
| NDCG | 0.849 | 0.8503 | 0.8506 | 0.8506 | 0.849 |

| title weight | 2 | **9** | **10** | **11** | 13 | 8 |
|---|---|---|---|---|---|---|
| NDCG | 0.8494 | 0.8498 | 0.850 | 0.8499 | 0.8492 | 0.8318 |

| header weight | 1 | **2** | **3** | 4 |
|---|---|---|---|---|
| NDCG | 0.8484 | 0.8489 | 0.8492 | 0.849 |

| anchor weight | 1 | 1.5 | 2 | **2.5** |
|---|---|---|---|---|
| NDCG | 0.849 | 0.8494 | 0.8491 | 0.849 |

The final parameters and final results are shown as below.

task1_W_url = 9, task1_W_title = 10, task1_W_body = 1, task1_W_header = 3, task1_W_anchor = 2.5.

The above parameters for the weights on the different fields indicate that the terms in the url and the title are particularly important, and if one of these matches the query, then the relevance score of the document should increase a lot. Intuitively, this makes sense because the title and url will have often have words that are most indicative of the subject of the document. For example, if the query is “stanford semester calendar”, we should rank the page with the keywords in title higher than the page with keywords in body.

| Dataset | NDCG |
|---|---|
| Train | 0.8492 |
| Dev | 0.8515 |

*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).*

To improve the ranking results, there are several kinds of optimization we can apply. One approach is to enhance the relevance calculation with more signals. For example, one signal we can add is user feedback. If we have large amount of visitors, assume two pages A and B, page A has more clicks than page B though they have the same relevance regarding to certain query, we should rank page A higher since it has more clicks. In addition, we could have a static quality score associated with a web page. A high quality score would mean the page has reputable, well-written information, while a low quality score would mean the article is spam. We could train this classifier from data collected by showing crowdworkers pages and having them classify them. Third, we could also improve the ranking, by augmenting the query or rewriting the query. For example, we could expand the query with related words. Fourth, we could use various techniques to improve the tf vectors, like stemming, n-grams and stopword removal. Last, some advanced smoothing approach can be used for term vector normalization. Instead of fixed smoothing factor, the factor can be dynamic generated according to some metrics of web pages. For example, if page lengths are much different, we should use larger factor for smoothing.

## Task 2 - BM25F

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

First, for the weights for the different fields, we used the values from task 1 to start with to decrease the number of parameters to tune. Then for each one of the remaining parameters we followed this process. First, initialize all parameters to be 1. Then for each parameter increase and decrease it by 0.5 and move in the direction that improves performance on the training set. Continue to move in the direction that improves performance until the performance decreases and then go back to the best value. If the performance decreases as you change it in both directions from 1, leave the parameter at 1. This is a greedy strategy in that we optimized each parameter by itself, but it performs pretty well. After tuning each of the parameters we also checked that the performance on the testing set was the same as on the training set to make sure we weren't overfitting the training data.

The final parameters can be seen at the end of the section. We found that decreasing B_url and B_anchor increased the performance, which again illustrates the importance of these fields. This is because the weight of B is inversely proportional to the effect that the field will have on the weight, so a lower B means the field is more important. We found that varying the other B parameters did not affect performance much. The parameter that had a very large effect on performance was K_1. As we increased K_1 from 1 it continued to improve performance until it reached 200, and the performance improved dramatically from 0.82 to 0.85 and above. This is because K_1 = 1 places very little value on the ftf's and as you increase it the text features begin to have more effect with improves the performance. We also experimented with varying lambda and lambdaPrime, but neither of these affected performance very much, so we left them both at 1.

*In BM25F, why did you select a particular $V_j$ function?*

We first started with V_j as log function following the advice in the notes. We experimented with saturation and sigmoid functions as well, but neither improved performance. In addition, the log function is a natural function to apply to pagerank scores because there are often many low pagerank scores in a network and just a few high ones. Applying the log to the pagerank scores smooths out this distribution and makes the the pagerank a more useful feature for ranking.

Final parameters and scores are shown as below.

task2_W_url = 7, task2_W_title = 7, task2_W_body = 1, task2_W_header = 1, task2_W_anchor = 2, task2_B_url = 0.1, task2_B_title = 1, task2_B_body = 1, task2_B_header = 1, task2_B_anchor = 0.5, task2_K_1 = 200, task2_lamba = 1, task2_lambaPrime = 1, task2_V_j = log

| Dataset | NDCG |
|---|---|
| Train | 0.8631 |
| Dev | 0.8658 |

## Task 3 - Smallest Window

*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 use smallest window to improve the relevance score in ranking. If $w_{q,d}$ is equal to |Q|, the max boost B is achieved; Otherwise, according to the minimum window between q and d, partial B is returned. The larger $w_{q,d}$ , the less boost we get. Also, the decrease of boost should be exponential to the $w_{q,d}$ in the sense if words are scattered in large window size, the boost should also be minimum. The formula to calculate the boost is shown as below.

$$Boost = \left(\frac{|Q|}{w_{q,d}}\right)^{boostmod}$$

The function obtains more accurate boost since the window size reflects the relevance in different fields. Previously, we give more weight to title and url which raise two issues. One is that it treats all title and url equally. One extreme example is that for query “A B”, the title with “A B” should be ranked higher than “B A”. Secondly, it overweights the importance of url and title. With smallest window, we notice that some lower value of url and title weight can achieve better result. For the training data set, we achieved a 0.003 lift in NDCG over the original cosine similarity model by augmenting it with the smallest window boost. We found that increasing B increase the NDCG, as illustrated below. According to our experiment on training and dev data set, B with value around 1.5-2 achieves relatively good result. We set boostmod to 2 empirically. Final parameters and results are listed as below.

| B | 1 | 1.5 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| NDCG | 0.850 | 0.8523 | 0.8529 | 0.8527 | 0.8524 |

task3_W_url = 8, task3_W_title = 8, task3_W_body = 1, task3_W_header = 2.5, task3_W_anchor = 1.5, task3_B = 1.5

| Dataset | NDCG |
|---|---|
| Train | 0.8532 |
| Dev | 0.8569 |