Programming Assignment 4

SUNetID: pchase, chuang39

Traditionally, the ranking model is created without training. A typical approach is based on probabilistic theory like BM25 model. However, nowadays learning to rank becomes a main trend in document retrieval to automatically construct the ranking model. In this project, we explore two different machine learning approaches and apply both linear and nonlinear regression models. A variety of signals are also examined to optimize the final score.

Task 1 - Pointwise Approach and Linear Regression

Learning to rank is a supervised learning process and thus consists of two phases: training and testing. The training set is comprised of queries and documents. Each query is associated with a number of documents. The reference results (relevant) of each document regarding to the query is provided for training.

Pointwise approach is essentially an ordinal classification which each query-document pair in the training data has a ordinal score. We use linear regression to predict the score. The linear regression function is as f(x) = wx+b. We rank them according to the difference between score and label. For task 1, we use tf-idf to calculate the similarity for different field. They are used to train and test the model. The NDCG score of training set and dev set are listed as below.

Dataset	NDCG
Train	0.842
Dev	0.852

Task 2 - Pairwise Approach and Ranking SVM

For task 2, we used the same tf-idf features as task 1, however instead of using a linear regression to score each query-document pair, we used a classifier that would say which document is better for a pair of documents in the candidate set.

First, to train the classifier we had to convert the pointwise training features for each document from task 1 into pairwise features with a classification. To do this, we first standardized the features over the entire training set, so that each features had a mean of 0 and a standard deviation of 1. Then we went through each sorted list of documents and made ordered pairs from the documents. For each of the ordered pairs we took the difference in the features and then added the correct classification of the difference. When choosing the ordered pairs, we made all of the ordered pairs and then had a boolean that would flip each time to decide if we wanted to have (A-B, +1) or (B-A, -1) for A > B. This was to ensure that we had roughly the same number of positive and negative examples.

To rank documents using the classifier we overrode the compareTo() method to to the difference in the feature vectors for the two Instances and then send the difference to the classifier and return the result. We then sorted using this compareTo() method to get the final ordering.

We tested both a linear SVM and a non-linear SVM with a RBF kernal. The results for the different kernals are below. In addition, we tuned each of the parameters for the RBF kernal using grid search over the suggested values in the notes. While tuning the parameters we made sure the performance on the training and testing set was similar to make sure we weren't overfitting the training data. The final parameters are shown below.

C = 0.25
Lambda = 0.25

So we can see that the pairwise approach gives results comparable to the pointwise approach. In addition, there is a slight benefit to using a non-linear SVM like one with an RBF kernal.

Model	**Train Dataset Score**	**Dev Dataset Score**
Linear SVM	0.8403	0.8462
SVM with RBF kernal	0.8534	0.8475

Task 3 - More Features and Analysis

In this task, we explore several new signals to investigate their improvement on ranking score. Following three kinds of signal are applied mainly. The details of each signal are described below and after that, results are presented for analysis.

1. BM25F feature: We used the BM25 feature (without pagerank) directly from PA3 as a single feature. This feature is useful, as it not only takes into account the term frequencies for each field, but it normalizes these by the average length of that field and the inverse document frequency.
2. Pagerank: We also included a feature based on pagerank. We saw that including the log(pagerank + lambda) was better than just the raw pagerank score in PA3, so we did the same here. We used Lambda = 3.0 from PA3.
3. Smallest window: For a given query, the smallest window $w_{q,d}$ is defined to be the smallest sequence of tokens in document d such that all of the terms in the query q for are present in that sequence. As we have validated in PA3, smallest window is an useful boost factor for relevance ranking.

In addition, there are two flaws we can improve. First, using raw window size as signal is misleading. If the query length is long (e.g 4) and even exactly the same word is found in some doc, the minimum window size is 4. However, for the short query with only 2 terms, window size can be lower than 4 like 3 even though the pattern doesn't match the query. Therefore, a better metrics is to use the ratio of (query length)/(smallest window size).

Second, we also notice that there exist many pdf url in our data set. In our training set, most cases pdf URLs rank in lower order finally. We add a new signal to indicate if url is linked to a pdf or not.

We use the pairwise approach and linear SVM model as the baseline. It achieves an NDCG of 0.8467 for dev dataset. Ablation test is performed to check the influence of new signal. From the experiment, we can find that BM25F and Pagerank provide significant increase on NDCG score. The reason why those two signals are so useful and effective is that for BM25, it takes into account the term frequencies, the average length of each field, and the inverse document frequencies, which improves ranking as seen in PA3. For pagerank, it is a straightforward metrics to reflect the importance of web page. Using those two signals for training and testing can provide better model. With those two signals, we can achieve score 0.8611 for dev dataset. The effect of smallest window and pdf detection is not as obvious as former two signals. Smallest window provides minor boost on accuracy. Result of pdf detection makes it look like a noise in statistics for the model, since for some case like pdf detection alone, it shows negative impact. But for some case, it has minor improvement. Although it is unexpected. we think the cause is that pdf detection is a deterministic factor in model or the sample size for training/testing is not sufficient to reflect the correctness of the signal.

With all bm25 and pagerank enabled, we obtains the highest score 0.8611 for dataset. But one thing should be noticed is that the score of training dataset is not the highest. Also, the result of (BM25F+Pagerank+Smallest Window) seems as good as (BM25F+Pagerank). It has the highest score for dev set.

Test ID	**Signal(s)**	**Train Dataset Score**	**Dev Dataset Score**
1	Base	0.8458	0.8467
2	BM25F+Pagerank+Smallest Window+PDF detection	0.8595	0.860
3	BM25F+Pagerank+Smallest Window	0.8676	0.8602
4	BM25F+Pagerank+PDF detection	0.86	0.8599
5	BM25F+Smallest Window+PDF detection	0.852	0.8559
6	Pagerank+Smallest Window+PDF detection	0.8600	0.8548
7	BM25F+Pagerank	0.863	0.8611
8	BM25F	0.8511	0.8565
9	Pagerank	0.8557	0.8521
10	Smallest window	0.8521	0.8493

11	PDF detection	0.8406	0.8462

For extra work, we explore some advanced feature provided by SVM for pointwise approach. We create the SVR kernel for SVM as following. It has some improvement on final result.
svm.setSVMType(new SelectedTag(LibSVM.SVMTYPE_NU_SVR, LibSVM.TAGS_SVMTYPE));

Test ID	**Signal(s)**	**Train Dataset Score**	**Dev Dataset Score**
1	Pointwise+Linear Regression	0.8423	0.8520
2	Pointwise+SVR	0.8472	0.8528