Learning to Rank TREC Disks 4 and 5 with RankLib

Ben Fox, Zizhuo Lin
Information Retrieval
University of California, Santa Barbara

December 12, 2018

1 Introduction

Ranking is a difficult problem in information retrieval tasks. In the document retrieval space, it involves retrieving relevant documents given a query. The order of the documents represents their degree of relevance for the particular query. This task is extremely prevalent throughout the web, from Google Search to Bing to DuckDuckGo to internal website search engines.

Learning to rank, on the other hand, is a machine learning method to re-rank search results based off of features within query-document pairs. Given a corpus of documents, queries associated with these documents, and annotated relevance labels (relevant being 1 and irrelevant being 0 in this project), learning to rank aims to create a supervised machine learning model to re-rank these query-document pairs to produce better relevancy results. These models are trained in an offline setting and evaluated with standard tools.

In this project, we use the TREC Disks 4 and 5 corpus of documents. This corpus is pre-processed using a variety of natural language processing (NLP) techniques. Further, a variety of features are extracted from these documents to train our learning to rank models. We use a total of one hundred fifty different queries matched with documents with assigned relevancy scores. Baseline ranking results were determined through Apache Lucene's ranking formula (which uses the BM25 metric). Training was done in RankLib [2] using AdaRank, Lambda MART, RankNet, and Random Forest. The results were evaluated and compared to the Lucene ranking of the same query-document pairs.

2 Techniques

2.1 Natural Language Processing and Data Pre-processing

The TREC Disks 4 and 5 data set combines a variety of Financial Times Limited, Congressional Records, Foreign Broadcast Information Service, Los Angeles Times, and Federal Register documents. Each document is comprised of a variety of information such as document headline (title), document text, country, publisher, dates, etc. In this method, we only extract the headline and the text for our re-ranking. It should be noted that the Lucene ranking algorithm includes all textual features, not just headline and text (this is discussed later). The documents were uploaded to python running pandas and the text for both document headline and document text were normalized (special characters, punctuation, and stop words were removed and all text was set to lowercase). Further, words were stemmed using the Porter2 stemmer. Stemming has been found to increase relevance in Arabic text [3]; however, stemming is largely question in the research community for improving search quality, but the general consensus is that in English, stemming reveals small improvements in information retrieval [4]. Queries were uploaded along with the binary relevancy scores and matched to the corresponding documents. The query text was normalized and stemmed using the same method.

2.2 Feature Extraction

A variety of features were extracted to train the machine learning algorithms for a total of fourteen features. First, document headline length, document headline existence (many documents did not have headlines), and document text length were found. Then using the query terms, BM25, term frequency (tf), inverse document frequency (IDF), and tf-IDF values were found for each document headline and document text. Further, query bigram and full query frequencies were found for the document text. Lastly, proximity features were included. Proximity features measure the distance between query terms in the document under the assumption that the closer the query terms in the document, the more relevant that document is to the query. This method has been shown to increase relevancy scores in the TREC8 dataset [5]. Both minimum distance and average distance were calculated, even though [5] showed that minimum distance produced better results. Using these features, and the relevancy labels as the response variable, machine learning models were trained.

2.3 Learning to Rank Techniques

All models were trained using five fold cross validation, and no hyper-parameter selection was performed. Features were normalized using Z-score normalization. Twenty percent of the data was used for validation and twenty percent was used for test. Further, each fold contained a specific set of queries, and there was no query overlap between folds.

2.3.1 Random Forest

Random forest is an ensemble learning method that trains multiple decision trees through bagging of samples. Additionally, features are sampled at random within these bagged samples, ensuring that the decision trees are train on a variety of features and not the most telling. With these decision trees combined, a relevance score can be determined for the test data. This method generalizes to unseen data and rarely overfits to the training data, so it was chosen as a viable method for the re-ranking.

2.3.2 AdaRank

AdaRank or adaptive ranking is inspired by the AdaBoost or adaptive boosting algorithm, which learns and combines a series of weak classifiers to produce output. In the document retrieval space, AdaBoost tries to minimize a loss function based on document pairs. Contrary to this, AdaRank attempts to optimize a loss function based on queries. AdaRank trains a series of weak classifiers and throughout training, assigns weights to the queries. Through a linear combination of the weak classifiers, AdaRank can produce a result [6]. AdaRank has been shown to produce (on four benchmark datasets) better results than BM25, RankBoost, and Ranking SVM [6]. Further, AdaRank's ease of implementation, high accuracy, and efficiency makes it a viable model.

2.3.3 Ranknet, LambdaRank, MART, and Lambda MART

Ranknet is the backbone to LambdaRank. It is any model that has a differentiable outcome, such as a neural net using a sigmoid function. The key difference between Ranknet and the other models mentioned is that it aims to minimize the number of incorrect orderings among pairs of results. It uses stochastic gradient descent to optimize a cost function; the cost function is typically the cross-entropy [1].

LambdaRank is derived from the Ranknet algorithm, except it was found that only the gradients with respect to the model score need to be calculated as opposed to the gradients to the cost function as in Ranknet. This is the main difference between Ranknet and LambdaRank.

Gradient boosting regression trees or MART is another ensemble learning method. This method learns multiple weak regression trees and attempts to minimize the gradient of the cost (squared regression error). In Lambda MART, MART and LambdaRank are combined to form a gradient boosted regression tree that uses the gradient method from LambdaRank. Lambda MART has been shown to produce better results than LambdaRank and Ranknet [1].

2.4 Evaluation

There are a variety of existing ranking quality methods for information retrieval such as mean reciprocal rank, mean average precision, expected reciprocal rank, and normalized discounted cumulative gain (NDCG) [1]. In this project, we used the NDCG evaluation measure to evaluate the quality of our document retrievals. NDCG is defined from the discounted cumulative gain function (DCG):

$$DCG@T = \sum_{i=1}^{T} \frac{2^{l_i} - 1}{log(1+i)}$$

Here, T is the truncation level or the "how many documents do we care about" number. l_i is the relevancy label of the ith document. NDCG is DCG normalized by the maximum DCG possible for a given query. NDCG is thus:

$$NDCG@T = \frac{DCG@T}{maxDCG@T}$$

and falls between zero and one. The above learning models were evaluated using the normalized discounted cumulative gain at ten measure (NDCG@10).

3 Results

The RankLib library [2] was used to train the five different models. Z-score normalization was applied to the features, and the testing NDCG@10 scores were found. The baseline was set by training on only the BM25 values produced by Lucene's ranking function. The NDCG@10 results are displayed in table 1.

The best model was Ranknet with an average NDCG@10 measure of 0.4851. Ranknet slightly outperformed Ranknet base. Ranknet base and AdaRank base tied for second highest NDCG@10 score with an average of 0.4847. Following, Lambda MART outperformed Lambda MART base with an average NDCG@10 score of 0.4735 compared to 0.4641, respectively. Random forest produced an averaged NDCG@10 of 0.4640, but had

Fold	AdaRank Base	AdaRank	Lambda MART Base	Lambda MART	Ranknet Base	Ranknet	Random Forest
1	0.4379	0.4338	0.4425	0.4363	0.4379	0.4379	0.4371
2	0.4958	0.4324	0.4552	0.4877	0.4958	0.4958	0.4945
3	0.4829	0.4766	0.4875	0.4935	0.4829	0.4851	0.492
4	0.4506	0.3605	0.4119	0.424	0.4506	0.4506	0.4284
5	0.5562	0.2981	0.5234	0.526	0.5562	0.5562	0.4678
Average	0.48468	0.40028	0.4641	0.4735	0.48468	0.48512	0.46396

Table 1: Learning Models NDCG@10 Values

no baseline to compare it to because random forest does not support one feature learning. AdaRank performed significantly worse than AdaRank base with an average NDCG@10 score of 0.4003 and decrease of 17% compared to AdaRank base.

4 Discussion and Challenges

As seen in the results, our additional features improved the relevance scores for Lambda MART and RankNet compared to baseline values. However, relevance scores decreased in AdaRank compared to baseline. Random Forest could not train using only one feature, so baseline is unavailable for that model, but it can be seen that the results of Random Forest were slightly less than other methods revealing that the other methods are likely better choices in a document re-ranking task.

The large decrease in the AdaRank score compared to the AdaRank base is interesting. In [6], they found increased relevancy results with few features for AdaRank, including term frequency, IDF, document length, BM25, and combinations of these. They did four fold cross validation and were able to outperform other ranking methods. The main difference was that they had a scale of relevancy labels (for a total of three labels) as opposed to our binary labels and performed a hyper-parameter search. After reviewing the top three AdaRank feature weights for each fold as seen in table 2, it was found that the features were weighted significantly different over every fold, potentially attributing to the decreased NDCG@10.

Another potential contributor to the decreased AdaRank NDCG@10 values could be related to our features being derived from only two textual features within the documents (document text and document headline), while Lucene's BM25 values were derived using the entirety of the documents' textual features. This could also be affecting the other models. Thus, the small improvements seen in the other models' NDCG@10 values are likely more significant than they seem because these models are trained with less text and less information than the baseline values.

Fold	Imp Feat 1	Imp Feat 2	Imp Feat 3
1	BM25: 18.96	Bigramfreq: 6.71	IDF: 2.37
2	HasHL: 9.90	BM25: 8.43	Bigramfreq: 4.55
3	BM25: 8.18	HasHL: 3.83	Bigramfreq: 2.08
4	HasHL: 15.13	BM25: 11.33	Bigramfreq: 4.29
5	Bigramfreq: 2.75	HasHL: 1.37	tfIDF: 1.19

Table 2: AdaRank Fold Top Three Feature Scores

5 Conclusion

Overall, we successfully learned a variety of machine learning models to re-rank documents given a set of query-document pairs and their relevancy scores. The best model for the Trec Disks 4 and 5 data set training on one hundred fifty queries with fourteen extracted features was found to be the Ranknet algorithm. The most improved algorithm compared to its baseline was the Lambda MART algorithm. AdaRank performed poorly compared to its baseline values likely due to a feature weight, binary relevance labels, and/or fewer textual features to train on problem. Random forest produced decent NDCG@10 values; however, all of the other models performed better.

To improve this study in the future, researchers should look into the differences in training on multi-class relevancy scores as opposed to binary class relevancy scores and if there is value in labeling query-document pairs with more than two classes. Additionally, the feature weight disparities across different folds and queries should be examined further to see if these varied are contributing to diminished NDCG values in the AdaRank model. Lastly, hyper-parameter selection should be done for all models to find the best parameters for the re-ranking task at hand.

References

[1] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[2] V Dang. Ranklib-a library of ranking algorithms, 2013.

[3] Leah S Larkey, Lisa Ballesteros, and Margaret E Connell. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–282. ACM, 2002.

[4] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[5] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 295–302. ACM, 2007.

[6] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.