Analysis of Adaptive Training for Learning to Rank in Information Retrieval

Saar Kuzi
skuzi2@illinois.edu
University of Illinois at Urbana-Champaign

Sahiti Labhishetty
sahitil2@illinois.edu
University of Illinois at Urbana-Champaign

Shubhra Kanti Karmaker Santu
santu@mit.edu
Massachusetts Institute of Technology

Prasad Pradip Joshi
prasad.pradip.joshi@gmail.com
Unbxd

ChengXiang Zhai
czhai@illinois.edu
University of Illinois at Urbana-Champaign

ABSTRACT

Learning to Rank is an important framework used in search engines to optimize the combination of multiple features in a single ranking function. In the existing work on learning to rank, such a ranking function is often trained on a large set of different queries to optimize the overall performance on all of them. However, the optimal parameters to combine those features are generally query-dependent, making such a strategy of "one size fits all" non-optimal. Some previous works have addressed this problem by suggesting a query-level adaptive training for learning to rank with promising results. However, previous work has not analyzed the reasons for the improvement. In this paper, we present a Best-Feature Calibration (BFC) strategy for analyzing learning to rank models and use this strategy to examine the benefit of query-level adaptive training. Our results show that the benefit of adaptive training mainly lies in the improvement of the robustness of learning to rank in cases where it does not perform as well as the best single feature.

ACM Reference format:
Saar Kuzi, Sahiti Labhishetty, Shubhra Kanti Karmaker Santu, Prasad Pradip Joshi, and ChengXiang Zhai. 2019. Analysis of Adaptive Training for Learning to Rank in Information Retrieval. In *Proceedings of The 28th ACM International Conference on Information and Knowledge Management, Beijing, China, November 3–7, 2019 (CIKM '19),* 4 pages.
https://doi.org/10.1145/3357384.3358159

1 INTRODUCTION

Learning to Rank (LTR) is an important general technique for optimizing search engine results. Its general idea is to use training data to learn a ranking function (model) by optimally combining a set of features to minimize the errors on the training data. The common practice of LTR is to train a single model on a single training set. However, such a strategy of optimizing search results uniformly for all the training queries is non-optimal for specific (test) queries because different queries often prefer a somewhat different way to combine the features. Indeed, queries are so diverse that each test query may potentially need a different version of the training data. A strong assumption made by most current work on LTR is that the optimal way to combine features is stable across different queries (which makes it possible to learn from one set of queries to optimize ranking for another set of different queries). However, this assumption is generally invalid. Take, for example, the combination of BM25 score with the PageRank score of a document. It is reasonable to believe that for some queries (e.g., informational queries), BM25 score is a more important feature and thus should have a higher weight than PageRank, but for others (e.g., navigational queries), PageRank may be much more important and thus deserves a higher weight. This simple example suggests that there is no single set of weights that would be optimal for all queries. The current way of training an LTR ranking function using a set of diverse queries would give us a model that is only optimal on average. A direct consequence of that is that even on the training set a trained ranking function would never achieve perfect performance no matter how much training data we have.

Query-specific features may help us address this problem by enabling dynamic weights on features, but this requires a massive amount of training data to allow a powerful learning algorithm to learn the correct way to combine those features in a query-specific way. Unfortunately, the search log data generally contain many training instances for head queries with generally little training data for tail queries. Such a skewed distribution of data means that only very few head queries can potentially benefit from a non-linear LTR method, while a large number of non-head queries would likely suffer in such a global training method where the objective function of optimization is inevitably biased toward head queries.

To address this problem, some previous work has studied how to adapt the training set to better serve each individual test query, including, e.g., training multiple models using subsets of the training data and attempting to select the best model for a test query [1, 4, 5], and using K-nearest neighbors for adaptive training [3]. The previous work has shown positive results from these strategies, showing that it is beneficial to train multiple models based on clusters of training queries, which can then be combined in a weighted manner at the testing time. Such a strategy appeared to be especially effective for difficult queries and using a large number of clusters was shown to be beneficial. However, previous work has not provided

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358159

an explanation of what benefit the adaptive training has achieved as compared to global training or why combining multiple models tends to work better than attempting to automatically select the right model for a test query. We speculate that one reason for this is because it is unclear how we can perform this kind of analysis.

In this paper, we propose a novel Best-Feature Calibration (BFC) method for analyzing the robustness of LTR models. Using this method, we conduct an in-depth analysis of the benefit of query-level adaptive training and show that the main benefit of the adaptive training strategy is in reducing the performance degradation when LTR is ineffective (i.e., performing worse than the best single feature). Our study also reveals that LTR can perform worse than the best single feature on many queries if trained globally, suggesting that the robustness of LTR may be a serious concern in search engine applications as it may cause user dissatisfaction in many cases. We further propose an error decomposition framework for analyzing such a query-level adaptive training strategy and suggest an explanation of why it has been challenging to automatically predict the best model. Overall, our study enables us to better understand the benefit of adaptive training in LTR and suggests several promising future research directions on this topic.

2 CLUSTER-BASED ADAPTIVE TRAINING

The main idea explored in the previous work on query-level adaptive training is as follows: Training data are partitioned into clusters, a separate LTR model is trained on each cluster, and the final ranking is obtained by aggregating results from the models trained on each cluster. Such a strategy has been shown to improve over the baseline LTR trained with the entire training set. It is, however, unclear what exactly the benefit of such a fusion-based adaptive training strategy is as compared with the baseline LTR. In this paper, we attempt to answer this question by using the following approaches, which follow similar spirits to all the previous work.

Clustering approach: We represent each query q using a vector representation $\vec{q}$, which is computed using the mean of the feature vectors of the relevant documents with respect to q. Using this representation, we cluster the queries in the training set. (In this paper, we use K-means clustering with Euclidean distance.)

Next, we refine the resultant clusters using the performance profile of the queries in the different cluster models as follows. Given k clusters ($k = 5$ in our experiments), for each training query we compute evaluation measures for each of the k result lists, ranked by the different clusters. Formally, $\vec{q} = (m_1^1, ..., m_n^1, ..., m_1^k, ..., m_n^k)$; where m_j^i is the j'th evaluation measure of the query when the model of cluster i was used for ranking. The idea behind this representation is that queries that prefer similar LTR models (parameters) will likely have a similar representation. In our experiments, we use the following evaluation measures: *MAP*@100, *MRR*@100, $p@\{3, 5, 10\}$, and $ndcg@\{3, 5, 10\}$. Then, using the new query representation, we perform another round of clustering. The vector representation of queries can be further modified using the new clusters obtained and the process of clustering can be repeated several times. In this paper, we performed two such steps.

Selective Cluster: To select a cluster for a test query, we use supervised learning where our goal is to predict a cluster from a set of clusters for each query. To represent each query for classification, we first use a single strong feature (a feature which obtained the highest performance on average in the training set) to rank the result list of the query. Then, we use the average of the feature vectors of the top 10 documents in the result list to represent the query. Using the cluster assignments of the training queries as labels, we learn a Logistic Regression model to predict a cluster. We contrast the performance of this approach with an approach which selects a cluster for a query using the query performance with respect to the cluster model (serves as an upper bound). We denote this approach **Cluster Oracle** in the experimental results section.

Cluster Fusion: As also shown in the previous work, the automatic selection of the right cluster model is difficult, and combining multiple cluster models tends to work better. Here we present a general probabilistic fusion framework and use it to examine the performance of fusion. Our goal is to measure the probability that a document d is relevant to a query q, i.e., $p(R{=}1|d, q)$, where $R \in \{0, 1\}$ is a binary random variable indicating relevance. Given a set of clusters C, we can estimate this probability using the different models that are learned using the different clusters as follows: $p(R{=}1|d, q) = \sum_{c \in C} p(R{=}1|d, q, c) \cdot p(c|q)$. Where $p(R{=}1|d, q, c)$ is the probability of relevance of d to q based on cluster c and $p(c|q)$ is the probability that the model learned using cluster c is a good fit to query q. Intuitively, using $p(c|q)$, we incorporate our uncertainty in choosing the "correct" cluster for a query. We estimate $p(R{=}1|d, q, c)$ and $p(c|q)$ in our experiments as follows. $p(R{=}1|d, q, c)$ is estimated using the normalized reciprocal rank of d in the result list generated by the cluster c with respect to q. We estimate $p(c|q)$ using a linear interpolation as follows: $p(c|q) = 0.5 \cdot p(c|q) + 0.5 \cdot p(c)$; where $p(c|q)$ is the probability learned using supervision in our selective cluster approach and $p(c)$ is proportional to the size of the cluster (the number of training queries in the cluster). Our set of clusters C is a union of the clusters generated in each step of the clustering algorithm. Furthermore, we add to this set five clusters that are generated randomly.

3 BEST-FEATURE CALIBRATION

The potential benefit of adaptive training is to better customize a model for a test query, but it is unclear how we can analyze this benefit quantitatively. To address this challenge, we propose a novel Best-Feature Calibration (BFC) strategy to analyze the robustness of an LTR model. The basic idea is to compare the performance of LTR with that of the best-performing single feature. Intuitively, if LTR is effective, we should expect its performance to be better than that of the best single feature, whereas if LTR has been trained non-optimally for a query, the performance of LTR may not be better than that of the best single feature or even be worse. Thus, BFC can serve as a general way to measure the robustness of LTR. (In this paper, we chose this feature to be the one with the highest average performance in terms of *ndcg*@5 in the training set.)

4 EXPERIMENTAL SETUP

Data sets: We use two publicly available data sets for the evaluation. The first data set is the Microsoft LETOR 4.0 data set [6], which was built using the GOV2 collection. We use the query set of 1,700 queries from the Million Query track of TREC 2007 (MQ2007); 46 features are provided. The data set is split into five folds. We use the

Table 1: The performance of the adaptive approaches for three groups of queries (grouped based on the difference between their performance when using LTR and their performance when the single best feature is used). Statistically significant differences with the corresponding LTR baseline approach and the single best feature are marked with '$*$' and 's', respectively.

	LETOR4						Yahoo					
	Low (25%)		Medium (50%)		High (25%)		Low (25%)		Medium (50%)		High (25%)	
	MRR	*ndcg*@5	*MRR*	*ndcg*@5	*MRR*	*ndcg*@5	*MRR*	*ndcg*@5	*MRR*	*ndcg*@5	*MRR*	*ndcg*@5
Single Feature	.651	.617	.618	.301	.278	.322	.906	.735	.949	.678	.735	.482
LambdaMart	$.364_{s}$	$.367_{s}$	$.619_{s}$	$.316_{s}$	$.713_{s}$	$.647_{s}$	$.827_{s}$	$.606_{s}$	.949	$.718_{s}$	$.905_{s}$	$.769_{s}$
Cluster Oracle	$.752^{*}$	$.645^{*}$	$.682^{*}$	$.430^{*}$	$.885^{*}$	$.717^{*}$	$.927^{*}$	$.756^{*}$	$.961^{*}$	$.782^{*}$	$.941^{*}$	$.822^{*}$
Selective Cluster	$.479^{*}_{s}$	$.422^{*}_{s}$	$.568^{*}_{s}$	$.294^{*}$	$.584^{*}_{s}$	$.525^{*}_{s}$	$.854^{*}_{s}$	$.635^{*}_{s}$	$.941^{*}_{s}$	$.711^{*}_{s}$	$.871^{*}_{s}$	$.720^{*}_{s}$
Cluster Fusion	$.474^{*}_{s}$	$.456^{*}_{s}$	.611	$.326^{*}_{s}$	$.655^{*}_{s}$	$.572^{*}_{s}$	$.852^{*}_{s}$	$.635^{*}_{s}$	$.946^{*}_{s}$	$.717_{s}$	$.881^{*}_{s}$	$.737^{*}_{s}$
Single Feature	.580	.586	.653	.294	.278	.365	.895	.747	.942	.662	.761	.503
AdaRank	$.345_{s}$	$.383_{s}$	.653	$.298_{s}$	$.579_{s}$	$.611_{s}$	$.822_{s}$	$.573_{s}$	.942	$.673_{s}$	$.902_{s}$	$.732_{s}$
Cluster Oracle	$.674^{*}$	$.643^{*}$	$.708^{*}$	$.383^{*}$	$.727^{*}$	$.677^{*}$	$.919^{*}$	$.781^{*}$	$.957^{*}$	$.742^{*}$	$.936^{*}$	$.766^{*}$
Selective Cluster	$.398^{*}_{s}$	$.343^{*}_{s}$	$.537^{*}_{s}$	$.255^{*}_{s}$	$.457^{*}_{s}$	$.424^{*}_{s}$	$.863^{*}_{s}$	$.665^{*}_{s}$	$.935^{*}_{s}$	$.669^{*}_{s}$	$.852^{*}_{s}$	$.622^{*}_{s}$
Cluster Fusion	$.467^{*}_{s}$	$.466^{*}_{s}$	.647	.299	$.488^{*}_{s}$	$.521^{*}_{s}$	$.862^{*}_{s}$	$.665^{*}_{s}$	$.939^{*}_{s}$	$.671^{*}_{s}$	$.838^{*}_{s}$	$.635^{*}_{s}$

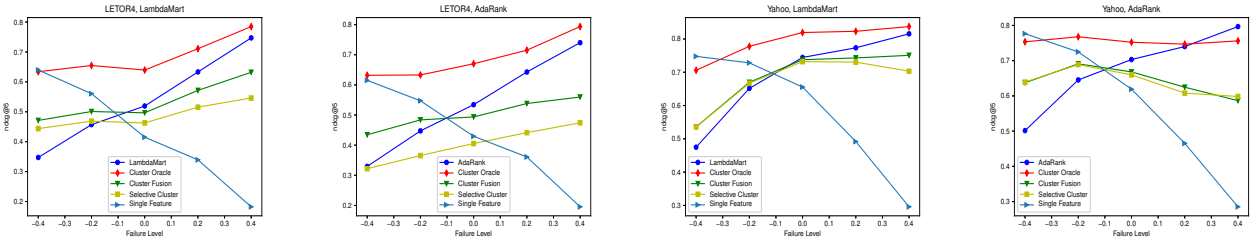

Figure 1: Performance of the different groups of queries for different levels of failure of the LTR approach with respect to using the single strong feature.

same folds and perform 5-fold cross-validation. The second data set is the Yahoo Learning to Rank Challenge dataset [2]. This data set was built based on the query log of the Yahoo search engine and contains 29,921 queries and 519 features. The data set is originally split into a training, test, and validation set and we use the same partition in our experiments. (We do not perform cross-validation here to make our results comparable to previous works.)

LTR algorithms and evaluation metrics: We use two LTR algorithms, AdaRank [8] and LambdaMart [7], for evaluation. The RankLib library is used to that end (sourceforge.net/p/lemur/wiki/RankLib). For both algorithms, we use *ndcg*@5 as the measure for optimization based on preliminary experiments to obtain a strong baseline; all free parameters are set to default values. We report the performance of the approaches using *MRR*@100 and *ndcg*@5. Statistically significant differences of performance are determined using the two-tailed paired t-test at a 95% confidence level.

5 EXPERIMENTAL RESULTS

Our main results are shown in Table 1, where we compare both LTR baselines (trained on all data), with Best Single Feature, Cluster Oracle, Selective Cluster, and Cluster Fusion on both data sets. Using the BFC strategy, we show the results by breaking down the queries into three groups based on how well LTR performed as compared with the best single feature. The low 25% bracket and high 25% bracket include queries where LTR has the largest deficit and largest surplus, respectively, as compared with the best feature. From the table, we can make several interesting observations:

1) Both LambdaMart and AdaRank can perform significantly worse than the best feature for a large number of queries on both data sets, strongly suggesting that LTR is indeed not robust and cannot learn optimal combinations of features when trained on all data.
2) Cluster Oracle consistently outperforms both the LTR baseline and the best single feature, suggesting great potential for customizing the training data to improve accuracy; these are cases where training using fewer (but more relevant) training data is clearly better than training with all the data.
3) The overall accuracy of Selective Cluster is not good, suggesting that it is hard to select the right cluster model. However, it is interesting to note that it performs well on the low 25% queries. Since those are the queries where LTR performed worse than the best single feature, the results suggest that Selective Cluster can reduce the degradation, likely because there is less bias for such queries when trained with a cluster than when trained with all the data.
4) Cluster Fusion is generally better than Selective Cluster, suggesting that the fusion strategy is indeed effective and robust, as also consistently reported in the previous work. However, our decomposed results reveal that the improvement is largely due to the reduction of performance degradation on the queries where LTR performed worse than the best single feature. Indeed, it is worse than LTR on queries in the high 25% bracket, likely due to the use of smaller amounts of data when each model is trained. This provides a clear answer to the question about the benefit of adaptive training.

The benefit of adaptive training in improving the robustness of LTR on queries where it performed worse than the best single feature can be seen even more clearly from Figure 1, where

we plot the *ndcg*@5 values of multiple methods over a spectrum of queries with different amount of degradation of LTR as compared with the best single feature. Specifically, the X-axis shows *ndcg*@5(*LTR*)-*ndcg*@5(*SingleFeature*), and the Y-axis shows the average performance of a method on queries for which this difference is similar[1]. From this figure, we can easily see that the slopes of the curves of the adaptive training methods are flatter than both the LTR curve and the best single feature curve, explaining the benefit of adaptive training in improving the robustness of LTR.

One possible reason for this benefit is that those queries where LTR performed worse than the best single feature are "minority" ("outlier") queries that prefer a very different way to combine features than most queries in the training data, and they suffered substantially from using a model optimized for the average of all the training data. However, by using clustering, we increase the diversity of the models, ensuring that there exists at least one model that can much better represent such a query than the globally trained model. While Selective Cluster cannot select the right model accurately in general, it is still generally beneficial for those outlier queries because even if it ends up choosing the second best or even third best cluster, the performance may still be better than the globally trained model. This explains why Selective Cluster tends to show more benefit for the low 25% queries.

How can we explain why Cluster Fusion works clearly better than Selective Cluster? One possible explanation is that taking a (possibly weighted) average over all the models in fusion avoids dominance by any single model (which can always be risky for some outlier queries). Indeed, after combining multiple models, the top-ranked document in each model is almost guaranteed to show at a relatively high-ranked position. For example, if we use a round-robin algorithm to take the top-ranked documents from each of the k models to generate a final ranking, then the top-ranked document of every model would have a rank at least as high as the k-th rank. So, if one of the k models represents a query so well that the top-ranked document is relevant, then the Reciprocal Rank for this query cannot be lower than $1/k$. Thus, unless a query cannot be represented well by any of the k cluster models we considered, the performance of the query cannot be too bad. This analysis also helps explaining why previous work has found that adaptive training tends to work better when using more clusters because with more clusters, the chance of having at least one cluster representing every test query well would be higher. Of course, the disadvantage of using too many clusters is that each cluster has fewer training data points, which may potentially hurt the performance. This suggests that a very important question that should be further studied in the future is: What kind of queries should be grouped together for training? This is the same question as: What kind of queries can be expected to prefer the same way of combining features?

This question prompts us to propose a general theoretical framework for analyzing the errors in adaptive training, where we would decompose the errors in adaptive training into two types: 1) errors due to biased LTR model trained on the wrong training data (model training error), and 2) errors due to inaccurate selection of a cluster model (model selection error). The observed error on a test query can thus be modeled as the sum of the model training error (MTE) and the model selection error (MSE). Informally, this is to say an error may be due to the lack of a model representing a test query well or the inability of selecting the right model. This framework can partially explain why Selective Cluster has not worked well. This can be due to the fact that it is hard to minimize both MTE and MSE. This is because in order to minimize MTE, ideally, the query clusters should be formed by using the ground truth, but if we do that, it would be impossible to represent a test query in the same way (due to lack of ground truth for the test query), which inevitably would cause a high MSE. Alternatively, we could sacrifice MTE by using a query representation that we can realistically compute for test queries as well, and this may help reduce the MSE, but it is unclear whether the benefit on MSE could compensate for the loss on MTE. The MTE-MSE tradeoff is an interesting research topic that clearly needs more research in the future.

[1]More precisely, a single point in the graph corresponds to the average performance of queries for which this difference is greater/equal than the corresponding value in the X-axis and smaller than the adjacent value in the X-axis.

6 CONCLUSIONS

In this paper, we have done an in-depth empirical analysis of the benefit and challenge of adaptive training for LTR. We proposed a novel Best-Feature Calibration (BFC) strategy for analysis of LTR and used it to analyze the clustering-based adaptive training approach. Our study reveals that LTR, when trained on the entire data, can substantially perform worse than the best single feature on many queries, suggesting that the robustness of LTR is a serious concern in search engines as it can potentially negatively impact the satisfaction of users. Our analysis further shows that the main benefit of the clustering-based adaptive training is in reducing the performance reduction of LTR when it performed worse than the best single feature, and thus is a beneficial strategy that can be adopted in search engines to improve the robustness of LTR. Finally, we further provided an explanation of why fusion of clusters tends to be more robust and proposed an error decomposition framework for analyzing the errors of this approach.

Acknowledgments. This material is based upon work supported by Unbxd and the National Science Foundation under Grant No. 1801652.

REFERENCES

[1] Jiang Bian, Xin Li, Fan Li, Zhaohui Zheng, and Hongyuan Zha. 2010. Ranking Specialization for Web Search: A Divide-and-conquer Approach by Using Topical RankSVM. In *Proceedings of WWW*. ACM, New York, NY, USA, 131–140.

[2] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.

[3] Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. 2008. Query Dependent Ranking Using K-nearest Neighbor. In *Proceedings of SIGIR*. ACM, New York, NY, USA, 115–122.

[4] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2017. Query-level Ranker Specialization. In *Proceedings of the 1st International Workshop on LEARning Next gEneration Rankers (LEARNER)*.

[5] Hsuan-Yu Lin, Chi-Hsin Yu, and Hsin-Hsi Chen. 2011. Query-dependent rank aggregation with local models. In *Asia Information Retrieval Symposium*. Springer, 1–12.

[6] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[7] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.

[8] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of SIGIR*. ACM, 391–398.