Learning to Select Rankers

Niranjan Balasubramanian and James Allan
Department of Computer Science
University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003, USA
niranjan@cs.umass.edu, allan@cs.umass.edu

ABSTRACT

Combining evidence from multiple retrieval models has been widely studied in the context of of distributed search, metasearch and rank fusion. Much of the prior work has focused on combining retrieval scores (or the rankings) assigned by different retrieval models or ranking algorithms. In this work, we focus on the problem of choosing between retrieval models using performance estimation. We propose modeling the differences in retrieval performance directly by using *rank-time* features – features that are available to the ranking algorithms – and the retrieval scores assigned by the ranking algorithms. Our experimental results show that when choosing between two rankers, our approach yields significant improvements over the best individual ranker.

Categories and Subject Descriptors: H.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Algorithms, Experimentation, Theory

Keywords: Combining Searches, Learning to Rank, Metasearch

1. INTRODUCTION

Combining evidence from multiple sources has been studied in various contexts [2, 1, 4, 6]. The basic premise for combining evidence from multiple retrieval models is that there is no *single* model that performs the best on *all* queries. Several rank fusion [7] and rank aggregation [4] approaches have been proposed to re-rank documents based on retrieval scores (or rankings) obtained from individual rankers. However, most of these approaches either learn a fixed (query independent) set of weights that are used to combine document scores or utilize a voting scheme for combining the rankings.

Instead of learning to combine document scores in a query dependent manner, we consider the problem of selecting a ranker for a given query. We propose a simple framework that directly predicts the differences in effectiveness between the results of different retrieval models. In particular, we consider the web-search scenario, where a large number of features are often combined using sophisticated learning to rank algorithms (rankers). For the sake of simplicity, we assume that we have access to two different rankers that operate on the same set of features. We formally define the ranker selection problem as follows:

Problem Definition. Given two rankers, R_a and R_b, we choose one ranker to be the baseline ranker (say R_b) – either arbitrarily, or based on the prior knowledge about the average performance of the rankers. Then, for each query, the selection problem is to determine whether to use the baseline ranker R_b or the alternate ranker R_a.

This problem definition and the selection framework that we propose can be extended to the scenario where we have access to multiple alternate rankers.

Ranker Selection Framework. We propose a simple framework for directly predicting the difference between the performance of two rankers(R_b and R_a) in terms of average precision (AP). We use the retrieval scores and the features of the top ranked documents (referred as retrieval features henceforth) to train a regressor. In addition to being closely related to the performance of the rankers, these features are also easy to compute, compared to typical performance prediction measures such as Clarity [3].

As shown in Figure 1, for a given test query, we first rank documents using both rankers. Then, for each ranked list, we compute mean and variance of the scores and the standard deviation of the retrieval features to generate aggregate feature vectors. The difference between the two aggregate vectors is input to the regressor which predicts the difference in effectiveness (AP(R_a) - AP(R_b)). If the predicted difference is positive, then we select the alternate ranker, otherwise, we use the baseline ranker.

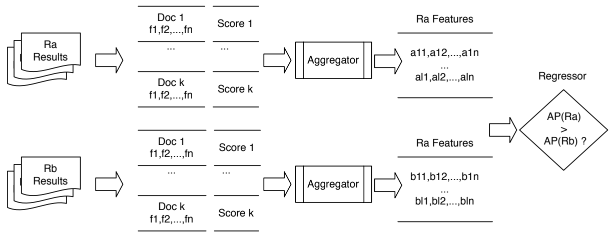


Figure 1: Ranker Selection Process

2. EXPERIMENTS

To evaluate our ranker selection approach, we use the LETOR 3.0 dataset [8] built on top of the TREC Gov2 collection. We conduct 5-fold cross validation experiments on a set of 225 queries created for the TREC named page finding tasks (NP 2003 and NP 2004). We use the results from three ranker baselines: RankBoost [5], Regression, and FRank [9]. To create features for the selection framework, we use the published test runs[1] for these rankers to obtain the document scores for top 10 ranking documents, and the list of 64 features that are available as part of LETOR. We use a non-linear Random Forest regression model for our experiments. We compare the rankers using mean-average precision (MAP).

[1]http://research.microsoft.com/en-us/um/beijing/projects/letor/letor3baseline.aspx

In terms of MAP, RankBoost is the best individual ranker, followed by FRank and Regression. Table 1 shows the potential for the use of query dependent ranker selection for named page finding. For example, RankBoost outperforms Regression on 90 queries, but performs worse on nearly half as many. Furthermore, we see that an oracle selection method can provide nearly 30% improvement over Regression, and nearly 15% improvement over FRank and 12% improvement over RankBoost.

Table 1: Potential for Improvement using Ranker Selection. R_b – Baseline ranker (RankBoost), R_a – Alternate ranker. Worse and Better indicate the number of queries for which R_a is worse than R_b in terms of MAP and vice versa. RankBoost has a MAP of 0.6596

R_a	MAP(R_a)	Worse	Better	Oracle Selection
Regression	0.5476	90	42	0.7096
FRank	0.6429	62	45	0.7316

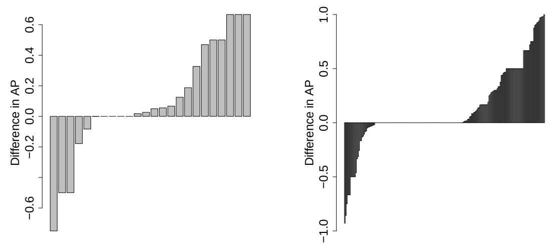
We conduct two sets of selection experiments one with RankBoost as the baseline ranker, and the other with RankBoost as the alternate ranker. Even though the rankers train on differences in performance, the distribution of the positive and negative differences change for each setting, thereby leading to different behavior in terms of the achieved improvements.

Table 2: Ranker Selection effectiveness on a set of 225 name page finding queries on the Gov2 collection. R_b – Baseline ranker, R_a – Alternate ranker. $MAP(R_s)$ indicates the MAP achieved with ranker selection. Underline indicates best MAP. * indicates significant improvements over Regression/FRank when using a paired t-test with $p < 0.05$

R_b	R_a	MAP(R_b)	MAP(R_a)	MAP(R_s)
Regression	RankBoost	0.5476	0.6596	0.6623*
FRank	RankBoost	0.6429	0.6596	0.6591*
RankBoost	Regression	0.6596	0.5476	$\underline{0.6722^*}$
RankBoost	FRank	0.6596	0.6429	0.6607*

Results of the two selection experiments are tabulated in Table 2. When using RankBoost as the alternate ranker, selection yields improvements over both Regression and FRank. This is in part because RankBoost performs better than both these algorithms for most queries. However, selection does not provide substantial improvements over RankBoost, the best individual ranker. On the other hand, when using RankBoost as the baseline ranker and Regression as the alternate ranker, we obtain substantial improvements using selection. Interestingly, even though FRank has a higher MAP compared to Regression, using FRank as the alternate ranker yields smaller improvements. This suggests that effectiveness of the selection also depends on the type of ranking algorithm used, in addition to the performance of the ranker itself.

The distribution of gains achieved for ranker selection between RankBoost and Regression is shown in Figures 2 (a) and (b). In both cases, we see that for a large fraction of the queries, choosing the alternate ranker results in gains, and very few cases result in losses. When using RankBoost as the baseline ranker, selection uses Regression for a small number of queries (28), and provides gains for subset (14), but the choice results in fewer losses (4). However, when using RankBoost as the alternate ranker, selection uses RankBoost for a large number of queries (198), out of which 83 queries result in gains and 29 result in losses. This suggests that while ranker selection yields substantial gains, it can also benefit from limiting losses due to poor selection. For example, thresholding on the predicted differences can reduce the number of queries for which the alternate ranker is queried.



(a) R_b: RankBoost (b) R_b: Regression

Figure 2: Distribution of Ranker Selection Gains: (a) When using RankBoost as the baseline and (b) When using RankBoost as the alternate ranker

3. CONCLUSIONS

In this paper, we proposed a simple learning approach for query-dependent selection of rankers. Our selection framework utilizes rank-time features – features that are available to the ranking algorithms during ranking. For selecting between two rankers, our experimental results show that a simple regression model that directly predicts differences in effectiveness, can achieve substantial improvements over the best individual ranker. As part of future work, we plan to investigate selection between multiple rankers using more sophisticated features for performance prediction.

4. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-0910884. Any opinions, findings and conclusions or recommendations expressed here are the authors' and do not necessarily reflect those of the sponsor.

5. REFERENCES

[1] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 173–181, 1994.

[2] W. B. Croft. Incorporating different search models into one document retrieval system. *SIGIR Forum*, 16(1):40–45, 1981.

[3] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR '02: 25th Annual ACM SIGIR Conference Proceedings*, pages 299–306, 2002.

[4] M. Farah and D. Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 591–598, 2007.

[5] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4, 2003.

[6] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 180–188, 1995.

[7] D. Lillis, F. Toolan, R. Collier, and J. Dunnion. Probfuse: a probabilistic approach to data fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146, 2006.

[8] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007.

[9] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, 2007.