Modeling Relative Effectiveness to Leverage Multiple Ranking Algorithms

Niranjan Balasubramanian and James Allan
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003
{niranjan,allan}@cs.umass.edu

ABSTRACT

In this work, we focus on modeling relative effectiveness of result sets to leverage multiple ranking algorithms. We use a relative effectiveness estimation technique (ReEff) that directly predicts the difference in effectiveness between a baseline ranking algorithm and other alternative ranking algorithms by using aggregates of ranker scores and retrieval features. Our ranker selection experiments on a large learning-to-rank data set shows that ReEff can provide substantial improvements over using a single fixed ranker – ReEff achieves more than 10% relative improvement on about 5% of the queries – and when using ranker and retrieval based features, modeling the relative effectiveness of rankers performs better than modeling their effectiveness independently. Further, compared to fusion, ranker selection yields different types of benefits and ranker selection using ReEff can further improve fusion for three fusion techniques.

1. INTRODUCTION

Many web search engines combine several features using learning to rank algorithms. Learning to rank algorithms (rankers) differ based on the features they use, the type of objective functions they target, as well as the optimization techniques they use to find parameter settings [11, 23, 16]. For example, RankSVM minimizes a pairwise objective function using convex optimization, whereas AdaRank [23] uses direct optimization for rank-based metrics such as MAP. Given these differences in the rankers, their relative performance can vary for different queries, even though their average performance can be similar. Ideally, a query-dependent selection of rankers will help improve performance over a fixed choice.

To this end, several query-dependent modifications have been proposed for selecting appropriate ranking functions for each query. While some focus on identifying the subset of training data to train query-specific ranking functions from, other focus on designing loss functions that are query-specific [5]. In recent work, Peng et al. [17] use the training data to identify nearest-neighbor queries, which are then used to estimate the expected performance of each ranker.

In contrast to these techniques, we focus on a direct approach for selecting the best ranker for each query. We use a relative effectiveness estimation technique (ReEff) that models the difference in effectiveness between the rankings. To represent each ranker, we use features derived from the ranker scores and the features used by the rankers themselves. Specifically, we use statistical aggregates such as mean and variance to characterize the distribution of the ranker scores. We also use aggregates of *retrieval features* and use the average similarity of the top K feature vectors of each ranking. Using these aggregate features, we learn a regression model that can predict the difference in effectiveness between a baseline ranker and the other rankers. The predicted difference is then used to select the best ranker for each query. Unlike independent estimation of the effectiveness of each ranker, this approach focuses on modeling a quantity that is closer to the end goal of selecting a ranker that performs better than a fixed baseline ranker.

Using ReEff we conduct ranker selection experiments on a large learning-to-rank data set and demonstrate that this approach yields substantial improvements (more than 10% relative improvement on about 5% of the queries) over using a single fixed ranker. We find that when using ranker based and retrieval based features, modeling relative effectiveness of rankers is better than modeling independent effectiveness.

We use ReEff to also improve *fusion* – the merging of rankings from multiple rankers. We find that selection yields different types of benefits when compared to simple fusion techniques: Selection has higher impact on a smaller subset of queries, whereas fusion has lower impact but on a larger subset of queries. To leverage the different benefits of selection and fusion, we conduct ranker fusion experiments, where we use query-dependent selection of rankers to augment fusion in three different ways: 1) Selective fusion, where we select fusion for some queries, while selecting individual rankings for others. 2) Selecting rankers, where we select the top few rankers to fuse for each query, and 3) Weighting rankers, where we weight each ranker based on its relative effectiveness with respect to the baseline ranker. Our experiments show that relative effectiveness estimation using ReEff can be effective for improving three state-of-the-art fusion techniques.

The main contributions of this work are in applying ReEff, a relative effectiveness modeling technique for selecting between different ranking algorithms, analyzing the impact of ranker selection using ReEff on a large web search data set, and demonstrating the utility of ranker selection for improving ranker fusion.

The remainder of the paper is organized as follows: Section 2 presents related work in query-dependent selection. Section 3 describes the ranker selection approach using ReEff and ways to improve fusion using query-dependent selection. Sections 4 and 5 present experimental results for ranker selection and fusion and Section 6 presents the analysis of ReEff's performance, followed

by conclusions in Section 7.

2. RELATED WORK

Several query-dependent application of retrieval techniques have been studied in various contexts including selective query expansion [7, 4], query-dependent learning [20, 5, 9] and query-dependent selection of ranking functions [18, 17].

Selective expansion techniques can be viewed as a selective application of a ranking function. For example, Amati et al. [4] use a measure of the difficulty of the initial ranking to predict whether expansion will lead to topic drift, whereas Townsend et al. [7] use relative entropy between the language models of the expanded and original ranked lists to detect if expansion causes topic drift. Soskin et al. [22] utilize query-drift and clarity based variants to improve the fusion of results retrieved from multiple query-expansion models. However, these techniques focus on measures that are designed to model aspects specific to query expansion and hence may not be well-suited for selecting between learning-to-rank algorithms for large scale web search.

Recently, query-dependent modifications to learning-to-rank algorithms have been proposed. Qin et al. [20] propose a modification to RankSVM that learns multiple hyperplanes (one for each top K rank), whose rankings are then combined to produce a single ranking. Bian et al. [5] develop query dependent loss functions which adjust the contribution of each query to the learning, based on the query's type: whether it is informational or navigational. Different from these approaches, instead of focusing on the training of the ranking algorithms, we focus on post-ranking modeling of the relative effectiveness of the rankings. Furthermore, the relative effectiveness modeling approach is not restricted to using a particular ranking algorithm.

Selecting similar queries has been proposed as a mechanism for query-dependent training and selection of rankers. He et al. [10] use query-based pre-retrieval features to identify a cluster of queries in the training set, and select the best retrieval model on the cluster. Another approach groups queries based on their difficulty by using features such as click entropy and learns different ranking functions for each group [24]. Plachouras et al. propose the use of two measures that characterize the broadness of query terms and show that it is effective for choosing between two types of evidence combination techniques for the topic distillation task [18]. [9] identify similar queries in the training set and use these neighbor queries to train different ranking functions using the same ranking algorithm.

Our approach is similar to that of learning-to-select (LTS) approach proposed by Peng et al. [17]. Using a baseline ranking algorithm such as BM25, they represent each ranker using the divergence of its scores with that of the baseline ranker's scores. The divergence scores are then used to find k-nearest neighbor queries in the training set in order to estimate the performance of each ranker.We use LTS as a key baseline for comparison with the proposed ranker selection technique.

The key difference of our approach is that instead of focusing on query-dependent training of ranking functions or using the training queries to estimate performance of the rankers, we directly model the relative effectiveness of the different rankings. In an earlier poster paper, we showed how this relative effectiveness approach can be used to select between two rankers on a small subset of the Letor 3.0 data set [1][1]. Using several additional variants of the ranker score based and retrieval-based features themselves, we show the relative effectiveness estimation can be used to effectively select between multiple rankers and also improve the effectiveness of fusion.

[1]Citation for poster anonymized for blind review.

3. RELATIVE EFFECTIVENESS

There are two broad approaches for leveraging multiple rankers: 1) Selection - Selecting a single ranking algorithm, and 2) Fusion - Combining the results from the available rankers and re-ranking them.

We argue that a query-dependent selection of rankers can improve the performance of both selection and fusion. First, since no single ranker performs the best for all queries, selecting the best ranker for each query can improve performance over using a fixed ranker for all queries. Second, query-dependent selection can improve fusion in multiple ways. For example, we find that for some queries fusion is beneficial, whereas for others selecting an individual ranker is better. Therefore, selectively applying fusion can improve over applying fusion for all queries.

Below we describe a relative effectiveness estimation technique (ReEff) for query-dependent selection of rankers.

3.1 ReEff

The main idea behind ReEff is to learn a regression model that can predict the difference in effectiveness between an alternate ranker and a baseline ranker. If there are multiple alternate rankers, then the alternate ranker with the highest positive predicted difference is selected. If no alternate ranker has a positive predicted difference, then we choose the baseline ranker itself. By modeling the relative difference in effectiveness, we learn to estimate the performance of the alternate rankers in relation to the baseline ranker. Thus, unlike independent estimation of effectiveness, modeling relative effectiveness allows us to capture dependencies between the performance of the rankers for each query.

Let $\mathbf{R} = \{B\} \cup \mathbf{A}$, be the set of m available rankers, where B is the baseline ranker and $\mathbf{A} = \{A_1, \cdots, A_{m-1}\}$ is the set of alternate rankers. Let $\mathbf{D}_R^q$ be the ranking produced by the ranker $R \in \mathbf{R}$ for query q. Also let $T(\mathbf{D}_R^q)$ denote the effectiveness of the ranking $\mathbf{D}_R^q$, as measured by a target effectiveness metric such as average precision. Then, for each query q, the ranker selection problem is to find a ranker, R^* that achieves the highest $T(\mathbf{D}_{R^*}^q)$.

$$R^* = \arg\max_{R \in \mathbf{R}} T(\mathbf{D}_R^q)$$

Since relevance information is not available for all queries, we need to estimate $T(\mathbf{D}_R^q)$ for unseen queries. Instead of learning a regression model to independently estimate $T(\mathbf{D}_R^q)$, we learn a regression model that estimates the relative effectiveness, $T_d(\mathbf{D}_R^q, \mathbf{D}_B^q) = T(\mathbf{D}_R^q) - T(\mathbf{D}_B^q)$. Given a set of training queries $\mathbb{Q}$ with relevance information and a set of regression functions $h_d : \mathbf{D} \times \mathbf{D} \rightarrow \mathbb{R}$ that approximate T_d, we learn a least squares regression function h_d^* as follows:

$$h_d^* = \arg\min_{h_d} \sqrt{\sum_{q \in \mathbb{Q}} \sum_{A_i \in \mathbf{A}} \left(h_d\left(\mathbf{D}_{A_i}^q, \mathbf{D}_B^q\right) - T_d\left(\mathbf{D}_{A_i}^q, \mathbf{D}_B^q\right)\right)^2}$$

Further, for all $q \in \mathbb{Q}$, we set $h_d^*(\mathbf{D}_B^q, \mathbf{D}_B^q) = 0$.

Then, for a given test query, q_t, we choose the ranking, R^* as:

$$R^* = \arg\max_{R \in \mathbf{R}} h_d^*(\mathbf{D}_R^{q_t}, \mathbf{D}_B^{q_t})$$

The maximization will select the baseline ranker B as R^*, if no alternate ranker A_i has $h_d^*(\mathbf{D}_{A_i}^{q_t}, \mathbf{D}_B^{q_t}) > 0$ for query q_t.

3.2 Features

To learn the regression function h_d^*, we first construct feature vectors that can represent the ranking of each both the baseline

Table 1: ReEff features: pos. refers to value of the feature each rank (e.g. the score at a given rank). hmean refers to harmonic mean and gmean refers to geometric mean. var, sd and cd refer to variance, standard deviation, and co-efficient of dispersion respectively.

Type	Feature Name	Description	Variants
Ranker Features	Alternate ranker id	Nominal feature that is identifies the type of alternate ranker.	None
	Scores of top k ranked documents	Un-normalized scores of the top k documents.	pos., min, max, mean, hmean, gmean, var, sd, cd, skew, and kurtosis
Retrieval Features	Features of top k ranked documents	Aggregates of the feature values of the top k ranked documents.	min, max, mean, hmean, gmean, var, sd, cd
	Average feature similarity	Average similarity of each document vector to the centroid of the top k document vectors	Un-normalized and L2 normalized.
	Overlapping documents.	Fraction of overlap in the top k ranked documents in the two rankings.	None

ranking, $\mathbf{D}_B^q$ and the alternate rankings, $\mathbf{D}_{A_i}^q$. Then, we use the difference of these feature vectors as the final regression feature vector.

We use two sources for constructing the feature vectors of each ranking. First, we use the scores assigned by the rankers as indicators of document relevance, which in aggregation can be effective for representing the quality of a ranking. Second, we use the features that are used by the rankers themselves to generate more aggregate features. The features used for ranking are also designed to reflect document relevance and are intimately related to the performance of each ranker. Table 1 lists the features that we use to represent each ranking.

For the ranker based features, we use the ranker id itself as a nominal and the scores of each top K document as individual features. We also use aggregates such as min, max, mean, variance, standard deviation and co-efficient of dispersion. Higher scores for top K documents and higher mean aggregates can indicate highly effective rankings, whereas higher variance can indicate poor effectiveness. Further, we use two additional variants of the mean, harmonic and geometric means and two higher order descriptive statistics, skew and kurtosis, to characterize the distribution of the ranker scores. Skew measures the symmetry around the mean (or a lack thereof) of the score distribution around the mean, whereas kurtosis measures the peakedness of the distribution.

For the retrieval-based features, we use the statistical aggregates for each retrieval feature and also use two additional measures, 1) *average feature similarity*, and 2) *fraction of overlapping documents*, which capture the intra-ranking similarity and inter-ranking similarity, respectively. To measure *average feature similarity* we treat each top ranked document as a n-dimensional feature vector and construct the centroid of the top k documents. Then, we compute the average of the distances of each top ranked document to this centroid[2]. The *faction of overlapping documents* is simply the ratio of the number of common documents in the two rankings and the number of top k documents considered.

The features we use are simple aggregates of features that are already used for ranking and can be computed efficiently. The average feature similarity computation requires two passes over the feature sets of the top ranked documents for all rankers – one pass to compute the centroid and the other to compute the distance of each feature vector to the centroid. Compared to scoring and sorting a large number of documents, which web search engines typically do, the time taken to compute the aggregates and the average feature similarity for the top few documents can be relatively small.

[2]This is akin to intra-cluster similarity used to measure the quality of a cluster of n-dimensional data points.

3.3 Leveraging ReEff for Ranker Fusion

In addition to using ReEff for selecting the best ranker for a given query, we also explore the use of ReEff for improving ranker fusion – combining the rankings from multiple rankers. ReEff provides estimates of the relative differences of the alternate rankers with the baseline ranker for each query. We utilize these estimated relative differences to improve fusion.

1. Selective Fusion - We target selective fusion – selecting queries for which fusing the rankings can be beneficial. Our initial analysis showed that for some queries, fusion is more effective than selection, whereas for others selection can be more effective. We model selective fusion as the task of choosing between the individual rankings and the fused ranking. The best ranking (individual or fused) is selected based on the estimated relative differences.

2. Selecting Rankers - Selecting the most effective rankers per query can help improve fusion performance. In many cases, we find that fusing fewer but more effective rankings is better than fusing all available rankings. We utilize the estimated relative effectiveness of the rankings to induce an ordering on all the available rankers and select the top k rankings to fuse for each query.

3. Weighting Rankers - Using weights that reflect the relative quality of the rankings that are being fused can also help improve fusion performance: intuitively, documents present in highly effective rankings should be preferred to documents from less effective rankings. However, since the effectiveness of the rankings is not known for all queries, we use the relative differences that ReEff estimates as weights indicating the effectiveness of the rankers. We compare this approach to MAPFuse [14], a recently proposed technique for utilizing the performance of the rankers on the training set as weights for combining rankings.

In the subsequent sections, we present empirical evaluation of ReEff for both ranker selection and ranker fusion.

4. RANKER SELECTION

We conduct ranker selection experiments to evaluate the utility of ReEff for selecting between rankers. We conduct these selection experiments on a large publicly available Microsoft Learning-to-Rank data set (MSLR-Web-10K) consisting of 10,000 queries. Each query-document pair in this dataset is represented by a set of 136 features, including low-level features such as covered query

Table 2: Ranking Algorithms and their mean average precision on the MSLR Web 10K dataset.

Name	MAP	Description
BM25	0.2561	Single feature ranker.[21]
RankBoost	0.2770	Boosting algorithm that minimizes discordant pairs in ranking [8]
AdaRank	0.2840	Boosting algorithm that directly optimizes for MAP [23].
L1-LogReg	0.3031	Logistic Regression with L1-constraints [2].
AFS	0.3053	Co-ordinate ascent-based algorithm with direct optimization for MAP [16].

term ratio, individual retrieval models such as TF-IDF [3], Okapi BM25 [21], and variants of Language Modeling [19] approaches that are applied to different fields such as URL, title, anchor and body and document quality features such as page rank[3].

We use a five-fold cross-validation approach for all our experiments. The training fold is used to train the ranking algorithms and the trained models are used to produce rankings on both the training and test set. We use five rankers listed in Table 2, including four competitive learning to rank methods and a single feature ranker BM25, which is the best individual feature. The table shows the MAP of the rankers on the entire data set and we see that the AFS based algorithm with direct optimization for MAP [16] performs the best and BM25, the single feature ranker, performs the worst.

We use four baseline methods for ranker selection for comparison with ReEff.

1. Prior-Based - Prior-Based selects a ranker using a random draw from a multinomial distribution which specifies the likelihood of each ranker being the best for a query. We estimate the parameters of this distribution from the training queries.

2. Best-on-Train - Instead of using the best individual ranker on the entire dataset, we use a stronger baseline, Best-on-Train. For each test fold, Best-on-Train selects the ranker that achieves the best average performance (in terms of MAP) on the corresponding training fold. We find that this leads to a better performance compared to only using AFS, the ranker with the best MAP on the entire dataset.

3. LTS [17] - LTS is a learning to select algorithm that utilizes the training data to estimate the performance of each ranker on queries that are similar to the test query. For each ranker R, LTS computes KL-divergences of the score distribution produced by R with respect to the score distributions produced by a baseline ranker such as BM25. This divergence feature is computed for the test query and the training queries. Then, LTS identifies from the training queries, the k-nearest neighbors (k-nn) for the test query using the divergence feature as the distance. The performance of R on this set of k-nn training queries is used as the expected performance of R on the test query. The process is repeated for all available rankers, and the ranker R^* which has the highest expected performance is selected as the best ranker for the test query.

   Peng et al. [17] use BM25 as their base ranker [4] but in our experiments since we use BM25 as one of the candidate rankers, we use TF-IDF as the base ranking function. We use a held-out set in the training data to estimate the two parameters for the LTS approach, the number of nearest neigbors (k) and the number of top ranked documents (n) used to compute the score distributions.

4. Indep. - Indep. learns a Random Forest [13] based regression model that can predict the performance of each ranker *independently*. For each query, the ranker with the highest predicted performance is selected as the best ranker. Indep. uses all the features listed in Table 1 except for the *overlapping documents* feature, which is a feature defined over two rankers. The features are extracted from the top 20 documents from each ranker.

5. ReEff - ReEff learns a Random Forest [13] based regression model that predicts the difference between a baseline ranker and the other alternate rankers. The alternate ranker with the highest positive difference is selected. If no candidate ranker has a positive predicted difference, then the baseline ranker is used. ReEff uses the Best-on-Train ranker as the baseline ranker. The features used to learn the regression are extracted from the top 20 documents for each ranker.

4.1 Results

We compare ranker selection results in terms of mean average precision (MAP) of the selected rankings and selection accuracy. To compare selection accuracy, we designate the ranker chosen by Best-on-Train as the baseline ranker[5], and treat the other rankers as alternate rankers. Then, we evaluate the selection methods on the number of queries for which they choose an alternate ranker, and the number of times the selected alternate ranker performed better, worse or the same when compared to the baseline ranker.

Table 3 shows the ranker selection results in terms of mean average precision (MAP) on 10,000 queries (top) and selection accuracy (bottom). The results show that ReEff outperforms all selection baselines. Below, we present a detailed analysis of the selection results.

4.1.1 Baselines.

Best-on-Train selects AFS as the best ranker for three folds and Logistic Regression for the other two folds. For each fold, the best performing ranker on the training queries also turns out to be the best on test queries. Note that Best-on-Train is a stronger baseline than the best individual ranker, as it performs better than only using AFS, the ranker with the best MAP (0.3053) on the entire data set.

Compared to this stable Best-on-Train baseline, Prior-Based random selection performs worse, which suggests that it is not trivial to select the best ranker for each query.

Using LTS for ranker selection also performs worse than the baseline Best-on-Train. LTS selects alternate rankers for a large number of the queries (for more than 63% of the queries) but mostly unsuccessfully – selection using LTS leads to poor performance in 48% of the queries, while only providing improvements for 42%.

Prior work [17] has shown that LTS can provide substantial improvements on the Letor learning to rank data sets, when selecting between three rankers. To validate our implementation of LTS, we conducted selection experiments on the Topic distillation subset of the Letor 3.0 data set [15]. On this smaller data set, but for the same set of rankers, LTS did achieve improvements in MAP over

[3]The data set and full list of the features are available at: `http://research.microsoft.com/en-us/projects/mslr`

[4]They do not use the base ranker as a candidate ranker available for selection but only use it to compute divergences.

[5]Best-on-Train may be different for different folds.

Table 3: Ranker selection results: Each fold comprises 2000 test queries. Bold-face indicates the best (non-oracle) performance in each column. * indicates statistically significant improvements over the Best-on-Train, determined using Fisher's randomization test ($p < 0.05$).

Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Best-on-Train	0.3056	0.3131	0.3027	0.3048	0.3111	0.3074
Prior-Based	0.2945	0.2918	0.2749	0.2922	0.3004	0.2908
LTS	0.3038	0.3023	0.3007	0.3034	0.3094	0.3039
Indep.	0.3041	0.3095	0.3013	0.3038	0.3118	0.3061
ReEff	**0.3131***	**0.3142**	**0.3071***	**0.3098***	**0.3189***	**0.3126***
Oracle	0.3658	0.3640	0.3604	0.3652	0.3727	0.3656

(a) Mean-average Precision for ranker selection.

Method	Alt.Queries	Better (%)	Worse (%)	Same (%)
Prior-Based	6969	2713 (38%)	4216 (62%)	0 (0%)
LTS	6323	2736 (43%)	3032 (48%)	555 (9%)
Indep.	6646	2855 (42%)	3037 (46%)	754 (12%)
ReEff	4506	2197 (49%)	1716 (38%)	513 (20%)

(b) Selection accuracy relative to the Best-on-Train baseline on the entire set of 10,000 queries. Better, Worse, and Same indicate the number of queries for which the method was better, worse or same compared to Best-on-Train.

the Best-on-Train[6]. However, on this larger web data set, despite conducting an exhaustive grid search over a wide-range of values for the k and n parameters (the number of nearest neighbors and the number of top-ranked documents respectively), LTS does not provide any improvements[7]. We find that the overall correlation between the expected performance of each ranker, determined as the average performance of the k-nearest neighbors, and the actual performance on the test query is very low (Pearson's ρ of 0.11 for LTS, compared to 0.27 for ReEff), which in part explains the overall poor selection performance.

Indep. also performs worse compared to Best-on-Train both in terms of MAP and in terms of the positive to negative impact. Indep. hurts more queries than it improves, as nearly 46% of the alternate ranker selection leads to worse performance, whereas less than 42% of alternate ranker selections leads to improvements. Despite having access to all the base set of features that ReEff utilizes, Indep. does not provide improvements over Best-on-Train, whereas ReEff performs consistently better. This shows that when using ranker based and retrieval based features directly modeling relative effectiveness is more useful than independently estimating the effectiveness of each ranker.

4.1.2 ReEff

ReEff provides consistent average improvements in MAP. ReEff performs the best on each individual fold and its improvements over Best-on-Train are statistically significant in all but one fold. ReEff achieves about 9% of the possible oracle improvements, the improvements that can be achieved with a selection oracle that selects the best ranker for every query (last row in Table 3). Even though the average improvements on the entire data set are small (about 0.0052 in absolute MAP), the actual gains achieved by ReEff are substantial improvements (0.0113 in absolute MAP) obtained on a smaller subset of queries, the subset for which ReEff selects alternate rankers. The average relative improvement within this subset of queries is about 3.75%.

Further, ReEff also performs better than the other baselines in terms of selection acuracy. Since ReEff selects alternate rankers for fewer queries (less than 50%), it provides better performance for fewer queries, compared to the other baselines. However, in 49% of the cases when ReEff selects an alternate ranker, it results in an improvement over the baseline, whereas only 38% of the times leads to a decrease in performance. We believe that this positive to negative impact ratio is a key strength of ReEff and we provide further analysis on this aspect of ReEff's performance in Section 6.

In summary, the ranker selection experiments show that using ReEff to select the best ranker for each query can outperform using a fixed ranker for all queries. In the next section, we explore the utility of ReEff for improving ranker fusion.

5. RANKER FUSION

We conduct ranker fusion experiments to demonstrate the utility of ReEff for improving fusion. We use three fusion techniques: 1) Reciprocal Rank - a rank based technique, 2) CombMNZ - a score based technique and 3) MapFuse - a rank based technique that utilizes past performance of rankers to perform weighted combination.

- *Reciprocal Rank* [6] (RR) uses the rank information of documents in each ranking to produce fused results. Reciprocal Rank is a competitive rank based fusion algorithm shown to achieve good fusion performance on the Letor datasets [15]. Given a set of rankers $\mathbf{R}$, the final Reciprocal Rank score of a document d for query q is computed as follows:

$$RR(q,d) = \sum_{R \in \mathbf{R}} \frac{1}{k + rank_R(q,d)}$$

  where, $rank_R(q,d)$ is the rank of document d in R's ranking for query q. k is a free parameter, which we set to 60 based on training set performance.

- *CombMNZ* [12] (CM) uses the sum of normalized scores assigned by the rankers, which is then weighted by the number of rankings in which the document was retrieved in the top k ranks.

[6]LTS improved MAP by 0.0053 over the Best-on-Train MAP of 0.2206, while ReEff gave an improvement of over 0.0109 in MAP.

[7]The range for k was [1,2,...5,10,15,20,..,50,100,200,...500,1000] and the range for n was [1,2,...,10,20,...50,100]. The best k was 500 for four folds, and 400 for the other and the best n was 5 for all five folds.

Table 4: Fusion versus Selection. Comparison of fusion techniques against selecting the best ranker using ReEff. Mean(ΔAP) denotes the mean of differences in AP between the baseline Best-on-Train and corresponding fusion technique. b,r,c and m superscripts indicate statistically significant improvements (determined using Fisher's randomization test with p < 0.05) over the Best-on-Train baseline, Reciprocal Rank (RR), CombMNZ (CM), and MapFuse(MF) methods respectively.

Method	MAP	Mean(ΔAP)	Better	Worse	RI
Best-on-Train	0.3074	+0.0000	0	0	NA
RR	0.3017	-0.0058	4062	4874	-0.080
CM	0.3108	+0.0031^{b}	4637	4292	+0.034
MF	0.3052	-0.0022	4229	4706	-0.048
ReEff	0.3126	+0.0052^{b,r,c,m}	2196	1718	+0.048

The final score is computed as follows:

$$CM(q,d) = |M| \sum_{R \in \mathbf{R}} nscore_R(q,d)$$

where, $M = \{R \in \mathbf{R} | rank_R(q,d) \leq k\}$ and $nscore_R(q,d)$ is the min-max normalized score assigned to document d by R for query q. We set k to 1000 based on training set performance.

- *MAPFuse* [14] (MF) uses the performance of each ranker on the training set of queries to produce the final score. The MAPFuse score for each document is computed as follows:

$$MF(q,d) = \sum_{R \in \mathbf{R}} \frac{MAP_R(q)}{rank_R(q,d)}$$

where, $MAP_R(q)$ is the mean-average precision of the ranking produced by ranker R for query q.

5.1 Fusion versus Selection

Table 4 shows results for fusion using all rankers and for selecting the best ranker for each query (selection) . For each method, we tabulate 1) the mean of differences in AP with respect to the Best-on-Train baseline, denoted as Mean(ΔAP), 2) the number of queries for which the technique was better than Best-on-Train, 3) the number of queries for which the technique was worse, and 4) the Robustness Index, defined as RI $=$ (# Better $-$ # Worse)$/n$, where n is the total number of queries.

Using ReEff to select the best ranker for each query is better than fusing the results from all five rankers. In fact, only one fusion technique, CM, provides additional average improvements over the Best-on-Train baseline. Also, selection performance is slightly better in terms of robustness measured by RI. Even though CM provides improvements for a larger proportion of queries compared to selection (46% versus 22%), it also degrades performance for a larger proportion (42% versus 17%). The trend is similar for the other two fusion techniques as well.

Thus, we find that fusion and selection provide different benefits and we use ReEff to augment fusion to leverage these benefits.

5.2 Using ReEff to Improve Fusion

We use ReEff to augment the fusion techniques in three ways.

- *Selective Fusion* (Selective {RR, CM, MF}) - First, we use ReEff to select between the individual rankings and the fused ranking. The fused ranking is generated by combining all the available individual rankings. In this setting, the baseline ranker is the Best-on-Train ranker, which is chosen from the set that includes both the individual rankers as well as the fusion ranker.

- *Ranker Selection* (S+RR, S+CM, S+MF) - Second, we use ReEff to select the top K rankers for fusion. We report the performance of fusing top 2 to top 5 rankers using each of the three fusion techniques.

- *Ranker Weighting* (W+RR, W+CM,W+MF) - Third, we use ReEff to assign weights to the rankings produced by each ranker. First, we use ReEff to obtain predicted differences of the alternate rankers with respect to the baseline ranker. Then, we normalize the predicted differences using a min-max normalization to avoid negative weights[8]. Finally, the document scores for each ranker are multiplied by the corresponding normalized weight and these weighted document scores are used by the fusion techniques to produce the final fused ranking. We report the performance of weighting in conjunction with ranker selection[9].

  The final score of a document for the fusion techniques are computed as follows:

  1. Weighted Reciprocal Rank (W+RR)

$$W{+}RR(q,d) = \sum_{R \in \mathbf{R}} \frac{w_R}{k + rank_R(q,d)}$$

  2. Weighted CombMNZ (W+CM)

$$W{+}CM(q,d) = \sum_{R \in \mathbf{R}} w_R \times nscore_R(q,d)$$

  3. Weighted MAPFuse (W+MF)

$$W{+}MF(q,d) = \sum_{R \in \mathbf{R}} \frac{w_R}{rank_R(q,d)}$$

  where, $rank_R(q,d)$ is the rank of the document in R's ranking, $nscore_R(q,d)$ is the min-max normalized score of document d assigned by R and w_R is the normalized ReEff weight for R.

5.3 Results

5.3.1 Selective Fusion

Table 5 shows the results for 1) fusion – fusing the results of all rankings using RR, CM, and MF, 2) ranker selection – selecting the best individual ranking for each query, shown as (ReEff), and 3) selective fusion – selecting between the individual rankings and the fused ranking (Selective RR, Selective CM and Selective MF).

For all three fusion techniques, selective fusion performs better than fusion. All improvements of selective fusion over the corresponding fusion methods are statistically significant. Furthermore, selective fusion also performs better than ranker selection and the improvements over ranker selection are statistically significant for RR and CM.

[8]The baseline ranker's predicted difference is set to zero.

[9]The impact of weighting alone can be compared when fusing all available rankers.

Table 5: Selective Fusion: Results of selective fusion. Mean(ΔAP) denotes the mean of differences in AP between the baseline Best-on-Trainand corresponding fusion technique. b,r,c,m, and e superscripts indicate statistically significant improvements (determined using Fisher's randomization test with p < 0.05) over the Best-on-Train baseline, Reciprocal Rank (RR), CombMNZ (CM), MapFuse(MF), and ReEff methods respectively. Bold-face entry indicates the best MAP.

Method	MAP	Mean(ΔAP)	Better	Worse	RI
Best-on-Train	0.3074	+0.0000	0	0	NA
RR	0.3017	-0.0058	4062	4874	-0.080
CM	0.3108	+0.0031^{b}	4637	4292	+0.034
MF	0.3052	-0.0022	4229	4706	-0.048
ReEff	0.3126	+0.0052^{b,c}	2196	1718	+0.048
Selective RR	0.3141	+0.0060^{b,r,e}	2676	2076	+0.060
Selective CM	**0.3156**	+0.0081^{b,c,e}	3550	2654	+0.090
Selective MF	0.3132	+0.0058^{b,m}	3082	2494	+0.059

Table 6: Selecting and Weighting Rankers for Fusion: Mean average precision (MAP) results for selecting and weighting rankers using ReEff. Bold-face entries indicate the best performing method for each column. * indicates statistically significant improvements of the S+ or W+ methods over the corresponding top K ranker fusion (B+ methods). ** indicates statistically significant improvements of the weighted fusion (W+ methods) over the corresponding selection of rankers (S+ methods) using ReEff. All statistical significances are determined using Fisher's randomization test (p-value < 0.05).

Method	Top 1	Top 2	Top 3	Top 4	All
B+RR	0.3074	0.3099	0.3098	0.3052	0.3017
B+CM	0.3074	0.3179	0.3180	0.3126	0.3108
B+MF	0.3074	0.3153	0.3150	0.3112	0.3061
S+RR	**0.3126***	0.3130*	0.3115*	0.3077*	0.3017
S+CM	**0.3126***	0.3187	0.3195*	0.3154*	0.3108
S+MF	**0.3126***	0.3164*	0.3168*	0.3135*	0.3061
W+RR	**0.3126**	0.3133	0.3123**	0.3100**	0.3067**
W+CM	**0.3126***	**0.3189**	**0.3202****	**0.3174****	**0.3153****
W+MF	**0.3126***	0.3167*	0.3173*	0.3156**	0.3112**

Selective fusion combines the merits of both fusion and selection. For example, when using CM, selective fusion increases the number of queries with a positive impact by about 13% compared to selection, while also increasing the number of queries with negative impact by about 9%. This trade-off leads to an overall improvement in MAP, and also provides substantial improvements in overall robustness as shown by the RI values. These results suggest that selective fusion can help to combine the benefits of both fusion and selection.

5.3.2 Selecting Rankers for Fusion

Table 6 shows the performance of selecting the top k rankers for fusion. The rows for the B+ and S+ methods in Table 6 show the results of using Best-on-Train and ReEff respectively for selecting the top k rankers. The Top 1 column corresponds to the case of selecting a single ranker for each query, whereas the Top 5 column corresponds to the case of using all the five rankers for fusion. The B+ entries for Top 1 show the baseline performance for Best-on-Train whereas the S+ and W+ entries show the performance of selecting using ReEff.

For all three fusion techniques, the performance of fusing the rankings of the top few rankers is better than fusing all rankings. As shown by the B+ rows in the table, when using Best-on-Train to select the top few rankers, using just the top 2 or 3 rankers provides the best performance.

Using ReEff to select rankers yields substantial improvements over using Best-on-Train to select rankers. All corresponding improvements, except for selecting the top 2 rankers for CM fusion, are statistically significant. Further, the best performance with selection is achieved by S+CM for Top 3 i.e., when selecting the top 3 rankers for CM (MAP 0.3195) . This setting is also significantly better than the best performance that can be achieved by selecting rankers using Best-on-Train – when using the top 2 rankers (MAP 0.3179). These results suggest that ranker selection using ReEff can further improve fusion performance.

5.3.3 Weighting Rankers for Fusion

The W+ rows in Table 6 show the results for using the relative difference weights in conjunction with ranker selection. Using ReEff for weighting rankers yields further improvements in all cases. When fusing the top 2 rankers, weighting the selected rankers does not yield substantial improvements. However, when fusing the top 3, 4 and 5 rankers, weighting provides substantial additional improvements for all fusion techniques. RR and CM do not use any weights on the rankers, and by introducing some weighting on the rankers through ReEff we obtain additional improvements. However, it is worth noting that MF already uses weights on the rankers and replacing these static weights with query-dependent weights from ReEff provides substantial additional improvements. This shows that ReEff yields reliable query-dependent weights that can be used for improving fusion.

In summary, we find that fusion and selection provide different types of benefits and using ReEff we can improve the combination of multiple rankers through selective fusion, ranker selection and ranker weighting.

6. ANALYSIS

We analyze the performance of ReEff to identify the features that

Table 7: Feature Importance for ReEff randomForest regression: Top ranked features sorted by the mean decrease in node purity, which is a measure of importance of the feature in the regression.

Feature Type	Aggregate Type	Importance	Feature Type	Aggregate Type	Importance
Ranker scores	standard deviation	1.66	Ranker scores	geometric mean	0.83
Ranker scores	skew	1.59	Ranker scores	arithmetic mean	0.80
Ranker scores	variance	1.57	BM25 body	max	0.71
Query-URL click count	geometric mean	1.42	BM25 whole document	max	0.67
Ranker scores	co-effic. of dispersion	1.37	Page Rank	max	0.67
Query-URL click count	mean	1.35	Ranker scores	harmonic mean	0.64
Ranker scores	kurtosis	1.29	...	...	...
Query-URL click count	variance	1.23	LMIR.ABS whole document	variance	0.63
Average feature similarity	individual	1.18	LMIR.JM whole document	variance	0.62
Ranker scores	position 20	1.08	URL dwell time	geometric mean	0.61
Query-URL click count	standard deviation	1.06	Document length	max	0.61
Ranker score	positions 1:19	1.02-0.87	URL length	co-effic. of dispersion	0.60

are most important for selection and also to better understand the types of improvements that ReEff provides.

6.1 Feature Importance

Table 7 shows the list of the most important features for selection, where importance is determined as the normalized reduction in the random forest regression error. The most important features include a mix of the ranker based and retrieval based features – the ranker scores and their aggregates, aggregates of the retrieval features such as click-based and BM25 features, as well as the average feature similarity measure. For ranker scores the most important aggregates are those that characterize the variance in the ranker scores and the two second order aggregates skew and kurtosis which characterize the shape of the score distributions. The most important retrieval based features correspond to aggregates of the query-url click count, which is a strong indicator of document relevance. The average feature similarity feature, which measures similarity of the top ranked documents is also one of the top 10 important features. Individual retrieval model scores such as BM25, language modeling scores, which are strong indicators of relevance also turn out to be important features for modeling relative effectiveness.

Overall, the ranker scores appear to be the most important set of features, since there are more aggregates of ranker based features in the top ranks than retrieval score based features. To better understand the impact of ranker scores versus retrieval-based features, we also conduct selection experiments with each individual group of features.

Table 8 shows the selection performance of the two groups on a single fold (Fold 1) of data. We compare the feature groups in terms of MAP, improvements over Best-on-Train, shown as ΔAP, and the robustness index (RI). Using ranker score based features alone does not yield any improvements over the baseline and has poor robustness. Using retrieval based features alone provides good improvements over the baseline. Despite the poor performance when used alone, ranker score based features add substantial improvements when combined with the retrieval based features. These results suggest that the selection performance of ReEff depends on both retrieval based features as well as ranker score based features.

6.2 Impact of Rankers

Table 9 shows the selection performance as the available alternate rankers is increased. As shown by entries in the oracle column, as the number of rankers is increased, the potential for selection increases, with most potential delivered by the top 2 rankers, while adding more rankers leads to smaller diminishing increases.

Table 8: Selection performance of different feature groups on a single fold (Fold 1). Ranker and Retrieval rows indicate performance of ranker score based, and retrieval-based features respectively.

Group	MAP	Mean(ΔAP)	Better	Worse	RI
Ranker	0.3061	0.0005	609	590	0.01
Retrieval	0.3096	0.0037	412	353	0.03
All	0.3131	0.0075	494	405	0.05

Accordingly, we see that most gains are achieved by selecting between the top 2 rankers (more than 80%), and adding more rankers provides a smaller additional increase (less than 20% of the total gains). Importantly, we see that ReEff utilizes all available rankers to deliver improvements over the baseline ranker.

Table 9: Ranker Selection results on a single fold (Fold 1) while increasing number of rankers for ReEff.

Rankers	**ReEff**	**Oracle**
1	0.3056	0.3056
2	0.3117	0.3405
3	0.3121	0.3512
4	0.3121	0.3592
5	0.3131	0.3658

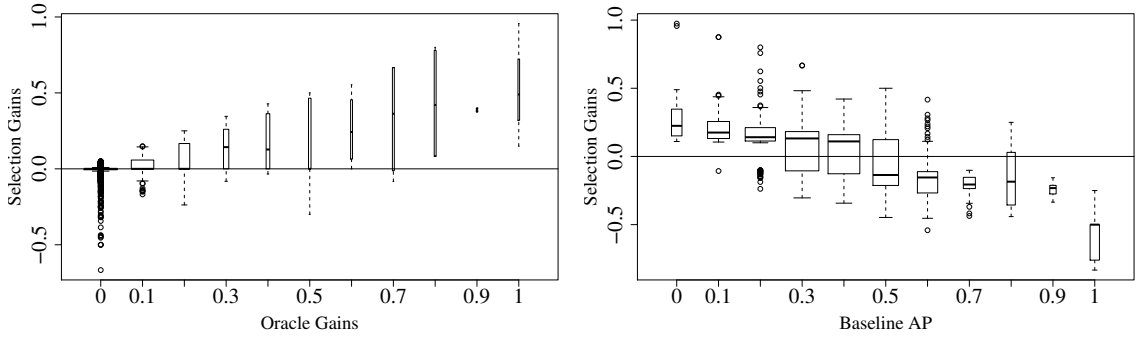
In addition to using the best individual ranker (Best-on-Train) as the baseline ranker, we also conduct experiments using other rankers as the baseline rankers on a single fold (Fold 1). Table 10 shows the performance of using other rankers as the baseline ranker for ReEff. In all cases, we find that selection improves over the corresponding baseline. More importantly, selection always improves over the best individual ranker Logistic Regression for this fold. This shows that while using the best individual ranker as the baseline yields substantial improvements, selection performance is not entirely due to the choice of the baseline ranker alone.

Table 10: Selection using other rankers as baselines on Fold 1.

Baseline	MAP(Baseline)	MAP(ReEff)
AdaRank	0.3018	0.3106*
BM25	0.2548	0.3113*
AFS	0.3005	0.3099*
RankBoost	0.2790	0.3118*
LogReg	0.3056	0.3131*

6.3 Distribution of Selection Gains

Figure 1a(a) plots the selection gains against the oracle gains – gains that can be achieved if we have a perfect selection tech-



(a) Oracle Performance versus Selection Gains

(b) Baseline Performance versus Selection Gains.

Figure 1: Distribution of Selection Gains using ReEff.

nique. Whenever there is high potential, ReEff provides gains in most cases and most of the errors in selection happen in cases where the potential is low. This shows that ReEff is effective at modeling large positive differences more effectively.

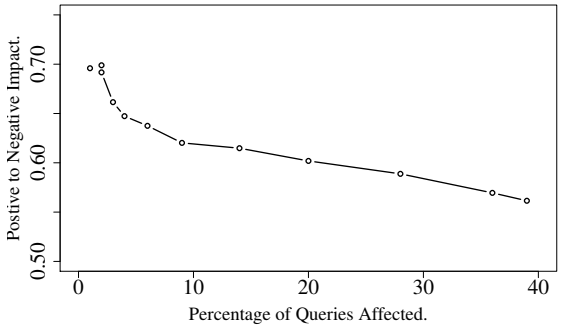
Figure 1(b) shows the distribution of large differences in AP achieved by using ReEff against the performance of the baseline ranker, Best-on-Train. The boxplot shows the distribution of differences in AP that are greater than 0.1 (we have 1875 such instances using ReEff). Large positive differences are obtained for queries whose performance on the baseline ranker is below 0.4 in MAP and large negative differences are obtained for queries whose baseline performance is above 0.4 in MAP. This is in part because the potential for gains are higher when the baseline ranker performs poorly. This suggests that selection is useful for improving the performance for queries whose baseline performance is poor.

6.4 Quality of Impact

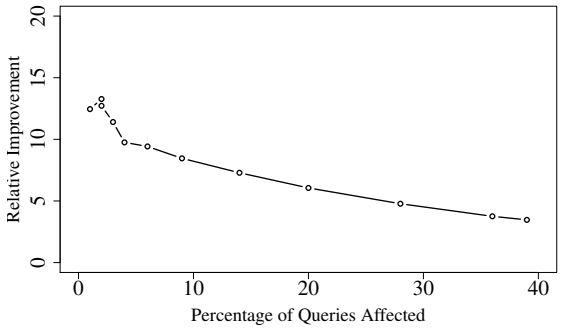
Next, we analyze the impact of controlling the number of queries affected by ranker selection. To this end, we conduct selection experiments where we impose a threshold on the predicted difference – i.e., we select an alternate ranker only if the predicted relative difference exceeds the specified threshold. When the threshold is set to zero, it is equivalent to the selection results we report in Section 4. However, as we set the threshold to increasing positive values, we select alternate rankers for fewer queries and consequently ranker selection affects fewer queries. For the purposes of this analysis, we experiment with thresholds between 0.0 and 0.05 with increments of 0.005.

Figure 2(a) shows the effect of thresholding on the percentage of positive improvements on the subset of queries with a non-zero impact. When fewer queries are affected the ratio of positive impact increases. Reducing the affected queries from 40% to 10% leads to only a small improvement in positive impact of about 5%, but further reductions in the percentage of affected queries leads to dramatic improvements in positive impact. When only affecting around 3% of the queries, selection can have a positive impact of about 70%. Figure 2(b) shows similar trends of the relative improvements over the Best-on-Train on the set of queries that are affected. When reducing the percentage of queries affected, we see higher relative improvements leading up to nearly 13% increase on a subset of 5% of the total queries. These results show the potential for calibrating ranker selection using ReEff to achieve a desired trade-off in percentage queries affected versus quality of impact.

6.5 Fusion Analysis



(a) Positive Impact Ratio

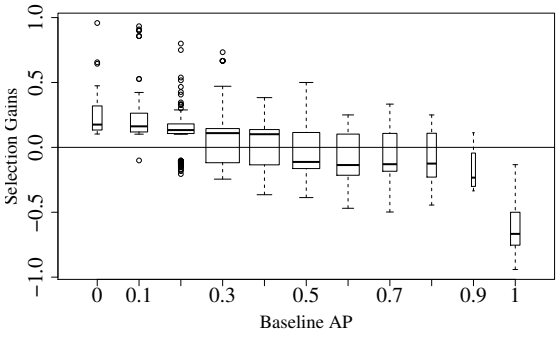


(b) Relative Improvement on Affected Queries.

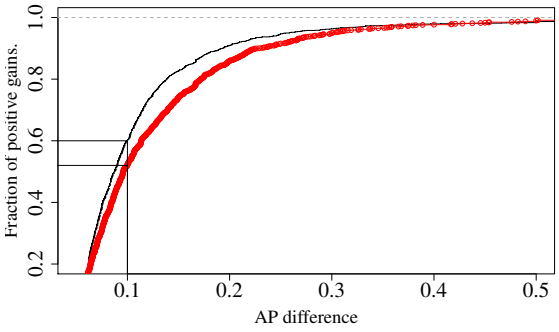
Figure 2: Trade-offs in Selection Impact versus Percentage Queries Affected.

Figure 3(a) shows the distribution of large differences in AP achieved by using CM fusion using all five rankers plotted against the performance of the baseline ranker, Best-on-Train. The boxplot shows the distribution of absolute differences in AP that are greater than 0.1. Similar to the performance of selection, fusion also yields improvements for queries with poor baseline performance, and degrades performance of queries with higher baseline performance. However, when compared to selection (in Figure 1(b)), fusion yields improvements for more number of queries with higher baseline performance.

Figure 3(b) shows the distribution of positive improvements over Best-on-Train for fusion and selection. The upper curve represents the cumulative density function (CDF) for fusion's improvements



(a) Distribution of fusion gains versus Best-on-Train AP.



(b) Positive improvements of fusion and selection.

Figure 3: Distribution of fusion improvements.

over Best-on-Train and the lower curve represents the CDF for selection. The CDF's show that selection typically provides larger improvements compared to fusion. For example, more than 50% of selection's improvements are more than 0.10 in MAP, whereas only 40% of fusion's improvements are greater than 0.10 in MAP. However, as we look at larger improvements, for example for improvements of over 0.3 in MAP there is no clear trend. This is in part because there are fewer queries for which such large improvements are obtained either through fusion or through selection. In conjunction with the results from Table 4, the distribution of these large improvements show that fusion and selection provide different types of improvements. Fusion provides smaller improvements over a large subset of queries, whereas selection provides larger improvements over a smaller subset.

In summary, ReEff utilizes both ranker-based and retrieval-based features to select between rankers and provides substantial gains when there is large potential and also provides a thresholding mechanism that can be used to control the quality of its impact.

7. CONCLUSIONS

There has been a profusion of learning-to-rank algorithms for improving web search. Leveraging these multiple ranking algorithms can yield substantial improvements. In particular, a query-dependent selection of ranking algorithms has high potential for improving retrieval performance over a single fixed choice. In this work, we showed that by modeling relative differences in effectiveness between rankers, we can enable query-dependent selection of ranking algorithms. Our experiments on a large scale web search data set using five rankers to demonstrate the utility of relative effectiveness estimation. We find that modeling relative differences is better than modeling effectiveness of each ranker individually, when using ranker based and retrieval score based aggregates. Our analysis shows that fusion and selection provide different kinds of benefits for leveraging multiple rankers. Using the estimated relative differences we combined the benefits of fusion and selection through selective fusion, ranker selection and ranker weighting to further improve performance.

As part of future work, we will explore the utility of ReEff for different ranking functions generated by using different subsets of features and training data. Furthermore, we will investigate the utility of ReEff for improving other fusion approaches.

8. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor. The authors would like to thank Van Dang for providing the learning-to-rank library, RankLib.

9. REFERENCES

[1] Details suppressed as required by double-blind reviewing.
[2] Regression on letor. `http://research.microsoft.com/en-us/um/beijing/projects/letor/Baselines/Regression.html`.
[3] A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.
[4] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. *Lecture notes in computer science*, 2997:127–137, 2004.
[5] J. Bian, T.-Y. Liu, T. Qin, and H. Zha. Ranking with query-dependent loss for web search. In *Proceedings of WSDM*, pages 141–150, 2010.
[6] G. Cormack, C. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of ACM SIGIR*, pages 758–759, 2009.
[7] S. Cronen-Townsend, Y. Zhou, and W. Croft. A framework for selective query expansion. In *Proc. of CIKM*, pages 236–237, 2004.
[8] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.
[9] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *Proceedings of ACM SIGIR*, pages 115–122, 2008.
[10] B. He and I. Ounis. A query-based pre-retrieval model selection approach to information retrieval. In *Proceedings of RIAO*, 2004.
[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD*, pages 133–142, 2002.
[12] J. H. Lee. Analyses of multiple evidence combination. In *Proceedings of ACM SIGIR*, pages 267–276, 1997.
[13] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.
[14] D. Lillis, L. Zhang, F. Toolan, R. W. Collier, D. Leonard, and J. Dunnion. Estimating probabilities for effective data fusion. In *Proceedings of ACM SIGIR*, SIGIR '10, pages 347–354, 2010.
[15] 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.
[16] D. Metzler. Automatic feature selection in the markov random field model for information retrieval. In *Proceedings of CIKM*, 2007.
[17] J. Peng, C. Macdonald, and I. Ounis. Learning to select a ranking function. In *Proceedings of ECIR*, 2009.
[18] V. Plachouras, I. Ounis, and F. Cacheda. Selective combination of evidence for topic distillation using document. In *Proceedings of RIAO*, pages 610–622, 2004.
[19] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of ACM SIGIRl*, pages 275–281, 1998.
[20] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proceedings of ACM SIGIR*, pages 279–286, 2007.

[21] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. of TREC-3*, pages 109–126, 1994.

[22] N. Soskin, O. Kurland, and C. Domshlak. Navigating in the dark: Modeling uncertainty in ad hoc retrieval using multiple relevance models. In *Proceedings of ICTIR*, pages 79–91, 2009.

[23] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of ACM SIGIR*, page 398, 2007.

[24] Z. A. Zhu, W. Chen, T. Wan, C. Zhu, G. Wang, and Z. Chen. To divide and conquer search ranking by learning query difficulty. In *Proceedings of CIKM*, pages 1883–1886, 2009.