Robust Ranking Models via Risk-Sensitive Optimization

Lidan Wang
Dept. of Computer Science
University of Maryland
College Park, MD
lidan@cs.umd.edu

Paul N. Bennett
Microsoft Research
One Microsoft Way
Redmond, USA
pauben@microsoft.com

Kevyn Collins-Thompson
Microsoft Research
One Microsoft Way
Redmond, USA
kevynct@microsoft.com

ABSTRACT

Many techniques for improving search result quality have been proposed. Typically, these techniques increase average effectiveness by devising advanced ranking features and/or by developing sophisticated learning to rank algorithms. However, while these approaches typically improve average performance of search results relative to simple baselines, they often ignore the important issue of robustness. That is, although achieving an average gain overall, the new models often hurt performance on many queries. This limits their application in real-world retrieval scenarios. Given that robustness is an important measure that can negatively impact user satisfaction, we present a unified framework for jointly optimizing effectiveness and robustness. We propose an objective that captures the tradeoff between these two competing measures and demonstrate how we can jointly optimize for these two measures in a principled learning framework. Experiments indicate that ranking models learned this way significantly decreased the worst ranking failures while maintaining strong average effectiveness on par with current state-of-the-art models.

Categories and Subject Descriptors

H.3.3 [**Information Retrieval**]: Retrieval Models

Keywords

Re-ranking, robust algorithms, machine learning

1. INTRODUCTION

Many approaches for learning highly effective ranking models for search have been proposed in recent years [19, 5, 25]. Most commonly, they apply well-developed machine learning algorithms to construct ranking models from training data by optimizing a given IR metric. While the learned ranking models can be highly effective, these approaches have mostly ignored the important issue of robustness – in addition to performing well on average, the model should, with high probability, not perform very poorly for any individual query. Often effectiveness and robustness are competing forces that counteract each other: models optimized for effectiveness alone may not meet the strict robustness requirement for individual queries.

This paper introduces learning robust ranking models for search via risk-sensitive optimization, by exploiting and optimizing the tradeoffs between model effectiveness and robustness. At a basic level, this framework learns ranking models whose effectiveness and robustness can be explicitly controlled. While effectiveness and robustness can be thought of in terms of absolute performance, one can specialize these concepts to a case where we desire to perform well relative to a particular baseline – for example a personalized system vs. the non-personalized baseline, query expansion vs. no expansion, or a learned model vs. a retrieval formula. In this case, we can specialize the notions of effectiveness and robustness to say that we wish to have high average gain relative to the baseline (*i.e.*, positive change in performance) while at the say time incurring low risk relative to the baseline (*i.e.*, low probability of performing worse than the baseline for any particular query).

We develop a unified learning to rank framework for jointly optimizing both gain and risk by introducing a novel tradeoff metric that decomposes gain into reward (upside) and risk (downside) relative to a baseline. While this objective could be integrated into many learning models, we show how to extend and generalize a state-of-the-art algorithm LambdaMart [6] to optimize the new objective. We empirically demonstrate that models learned this way outperform both a selective personalization approach and standard learning to rank models in terms of achieving an optimal balance between gain and risk. Moreover, our results also provide important insights into why our learned models are more robust in terms of the learning convergence property of the new model as compared to standard effectiveness-centric models.

2. RELATED WORK

Multi-objective learning to rank has received much attention in recent years [31, 13]. This is because, in many practical settings, the performance of a ranking model is evaluated by multiple measures of interest (e.g., NDCG, MAP, freshness, efficiency). Multi-objective learning to rank trains ranking models to simultaneously optimize multiple IR measures. Svore *et al.* [31] use a standard web relevance measure, NDCG, as the primary optimization metric and a relevance derived from click-data is used as the secondary metric; the performance of the primary measure is maintained constant while the algorithm tries to improve the secondary measure. Dai *et al.* [13] develop a multi-objective learning to rank algorithm for freshness and relevance by extending a state-of-the-art divide and conquer ranking approach [3]. The key differences between these and our work is that we treat both risk and reward, which are potentially competing measures, as first-class metrics during optimization, unlike the "tiered" approach [31]; however, more importantly, our risk measure is inherently different from the previous multi-objective learning to rank metrics – instead of capturing just another facet of web search [31, 13], risk evaluates the fitness of the ranking model *with respect to* a baseline model under a specific metric. In other words, risk is orthogonal to the previous multi-objective metrics – risk is defined on top of these metrics with respect to a baseline model.

Although other risk-aware algorithms have been proposed for ad-hoc retrieval [35, 39] and query expansion [26, 10], our problem differs from them in three important dimensions. First, we focus on designing new learning to rank algorithms to create robust state-of-the-art ranking models using boosted regression trees, rather than focusing on risk-minimization for language models or simple term-based models [35, 39, 34]. Second, unlike [10, 26], the source of risk in our setting comes from the fact that users and queries are diverse, so that a single model cannot serve all users or queries well. Thus we design a new risk-sensitive algorithm that effectively leverages query-dependent features to build models that are highly robust and effective for individual queries. Finally, these previous problems represent particular strategies for ranking, hence their solutions cannot be trivially extended to the more general problem of learning robust ranking models for large-scale retrieval.

Risk can be indirectly addressed by building query-dependent models to better adapt to individual queries. This line of work resulted in recent papers that consider query-specific loss functions [4], where queries of different types (e.g., navigational, information) are optimized with different loss functions. Geng *et al.* [18] proposed a k-Nearest Neighbor based method which trains a query-dependent ranking function for each query based on its nearest neighbors in the training set. Bian *et al.* [3] proposed a clustering-based divide-and-conquer approach for building query-dependent ranking models. However, it is important to note that query-dependent models are *not* perfect – just like any other models, query-dependent models incur risk (*e.g.*, reduced performance for some queries and gains for other queries), and this fact has been largely ignored as well. In a sense, our proposed framework generalizes and complements this thread of work by learning risk-sensitive query-dependent models.

There has been a great deal of research devoted to developing effective retrieval models that have high average retrieval quality. This has given rise to a steady stream of techniques for effective ranked retrieval. Examples include learning to rank [19, 5, 25], learning to re-rank [22], numerous term proximity models [27, 8, 32], and search personalization [1, 36, 11]. However, unlike our work, they ignore the important issue of model robustness – while many queries see performance improvements, other queries are often hurt by the new and complex models as compared to simple alternatives such as BM25 [29] and language models for information retrieval [28].

Our work also has connections to query difficulty and performance prediction for Web search [38], as well as selective personalization [33]. These techniques mitigate risk in re-ranking by using a query performance prediction method to selectively re-rank queries whose baseline effectiveness is expected to be low, and avoid re-ranking for queries where the baseline effectiveness is already high. We note these techniques are robustness-centric, which may lead to overly reduced effectiveness due to unreliable decision in applying the model. In Section 5 we compare our approach to the use of selective re-ranking using performance variables.

It seems reasonable to assume that users tend to remember the singular, spectacular failures of a search engine rather than the many successful searches that preceded them. However, we are not aware of many user studies that have looked at the effect of IR robustness (or the lack thereof) on users' perceptions of system quality. One exception is a recent user study of recommender systems [23] that showed how optimizing the accuracy of a system is not enough: among other interesting findings, they found that high-variance recommendation algorithms can create a bad user experience. Informally, significant errors can have a large negative impact on the perceived quality of the system, even if the system frequently does well.

Finally, the term *robust ranking* has been used previously in the literature, but with somewhat different meaning. For example, Li *et al.* [24] call ranking functions 'robust' that account for volatility in ranker scores, and thus, document ordering, over time, by taking into account the distribution of ranking scores over results. Others have used 'robust ranking' to refer to ranking algorithms that are less vulnerable to spam or noise in the training data [2].

3. MOTIVATION

Before proceeding, it is important to identify sources of uncertainty that contribute to low robustness in ranking models. In most search settings and especially Web search, both users and information needs are highly diverse, thus, it is unlikely a common set of ranking features and the same model learned from these features can optimally work for all users and queries. For instance, it has been shown that: queries with low click entropies typically will not benefit from search personalization [1, 36, 11]; queries with low query clarity will not benefit from query expansion techniques as much [16]. Ignoring these facts during model training leads to a highly risky model. Although previous work has studied whether and when a learned model should be applied to a given query based on the aforementioned query-specific meta-features [33], the average effectiveness that results from the robust application of the ranking model is generally ignored – which means the robustness is typically improved but at an overly aggressive cost in reduced average effectiveness. We take a principled learning to rank approach that directly optimizes multiple requirements.

For instance, Figure 1 illustrates the improvements and degradations of a proposed model (tuned for average effectiveness) with respect to a baseline ranking across queries. The plots shows a high variance of the new results – the performance of a number of queries is improved (right side) at the expense of degraded results for other queries (left side). Directly optimizing for variance is a challenging task – variance is inherently a set-based measure defined on the distribution of results quality over queries, but typically, learning to rank algorithms are not directly applicable to optimizing

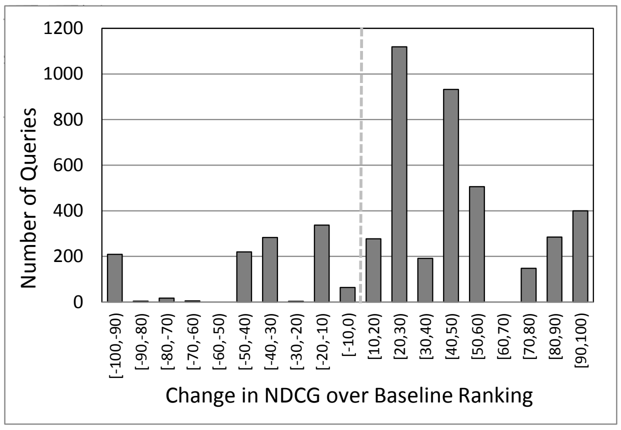


Figure 1: Typical helped-hurt histogram of re-ranking gains and losses compared to a given baseline ranking.

set-based measures. We instead adopt a simple approach suggested by this figure.

Informally, when compared to an alternative or baseline ranking, the probability of decreasing performance is related to the mass on the left-hand side of the figure. In fact the expectation of the left-hand side, which we call risk, is the amount we can expect to decrease retrieval quality on average if we pick a query at random. Likewise, the expectation of the right-hand side, which we call reward, is the amount we can expect to improve retrieval quality on average. By defining each average relative to the total number of queries rather than dividing reward by the number of improved queries and risk by the number of hurt queries, we control for issues of coverage. Typical methods optimize overall performance, $Reward - Risk + Baseline$, but since the baseline is fixed from the optimization point of view, this is equivalent to optimizing the difference, $Reward - Risk$. In order to control the risk or probability of harm, this suggests introducing a weight on the risk term that is user tunable. This is precisely what we do. In the next section, we formalize this intuition before demonstrating how to optimize the objective in practice.

4. A LEARNING TO RANK APPROACH FOR CREATING ROBUST MODELS

4.1 Problem setting

In this section, we present the formal setup for learning robust ranking models for Web search. Given a baseline ranking M_b we would like to construct a new ranking model M_m with high average effectiveness across queries, without degrading the performance of individual queries too much with respect to M_b. The exact instantiation of the baseline M_b depends on the specific application of our general framework. For personalization, M_b might be the initial non-personalized ranking (e.g., optimized for the average user). In a production environment, M_b might represent the results from an earlier release of the ranker. In learning to rank, M_b might be simpler models such as BM25 that are known to be effective on a particular subset of queries. Our framework is applicable to these and any other ranking-based applications where a risk/reward tradeoff may exist.

4.2 Measuring risk and reward

One way to measure risk is by measuring the variance of the new ranking model with respect to baseline results. We instead adopt a variant of this approach, by introducing a risk function $F_{RISK}(Q_T, M_b)$ to capture characteristics of the variance with respect to model M_b over a training set Q_T. We are interested in the downside risk of the new model, defined as the average reduction in effectiveness due to using the new model M_m compared to the baseline model M_b [1].

$$F_{RISK}(Q_T, M_b) = \frac{1}{N} \sum_{Q \in Q_T} \max\left[0, M_b(Q) - M_m(Q)\right] \quad (1)$$

where N denotes the total number of queries in the training set Q_T, and $M_b(Q)$ and $M_m(Q)$ denote the effectiveness of the baseline model and new model for a given query Q, respectively. We note that effectiveness can be measured by any commonly-used IR metrics (e.g. precision@k, recall, average precision, and NDCG@k). Thus, the downside risk accounts for the negative aspect of the retrieval performance across queries.

Similarly we defined reward in relative terms as the positive improvement in effectiveness over baseline model M_b, averaged across all queries in a training set Q_T:

$$F_{REWARD}(Q_T, M_b) = \frac{1}{N} \sum_{Q \in Q_T} \max\left[0, M_m(Q) - M_b(Q)\right] \quad (2)$$

where notations N, M_b, and M_m are defined as before. Thus, the reward function accounts for the positive aspects of the retrieval performance across queries. For any arbitrary baseline, M_b, the overall gain for training set Q_T can then be written:

$$F_{GAIN}(Q_T) = F_{REWARD}(Q_T, M_b) - F_{RISK}(Q_T, M_b). \quad (3)$$

Our goal is to automatically learn ranking models that better adapt to individual queries by optimally trading off between risk and reward. However, before we can learn such a well-balanced model, we must define a new metric that captures the tradeoff. Our objective function, which we call Risk-Reward tradeoff T, is defined as a weighted linear combination of gain (which includes risk and reward) and risk:

$$\begin{aligned} T_\alpha(Q_T, M_b) &= F_{GAIN}(Q_T) - \alpha \cdot F_{RISK}(Q_T, M_b) \quad (4) \\ &= F_{REWARD}(Q_T, M_b) - (1+\alpha) \cdot F_{RISK}(Q_T, M_b) \end{aligned}$$

where $\alpha \geq 0$ is a key *risk-sensitivity parameter* that controls the tradeoff between risk and reward. We chose this formulation so that the case $\alpha = 0$ corresponds to the standard learning-to-rank objective to maximize average F_{GAIN} alone. Note that T is not a single measure, but a family of measures, parameterized by the tradeoff parameter α. By varying α we can capture a full spectrum of risk/reward tradeoffs: from $\alpha = 0$ (standard default ranking) to larger values of α that lead to lower-risk models, or small values of α that lead to higher gain models.

The most common approach to optimizing multiple objectives [30] is to linearly combine objectives, as we have done with the risk and reward functions. While many other ways for combining metrics exist, such as by multiplication, division, geometric or harmonic means, optimizing an arbitrary

[1]For simplicity of notation, in the remaining of the paper M_b will refer to both the baseline model and the effectiveness of the baseline model. A similar convention is used for M_m.

optimization objective can be problematic in many learning algorithms. In contrast, linearly combined objectives are often easily optimized. For our purposes, we will extend the objective of the LambdaMart algorithm [37]. In Sec. 4.3, we prove that our tradeoff measure T satisfies a desirable consistency property [6].

4.3 Constructing robust ranking models

In this section, we describe how to learn models that directly optimize the proposed tradeoff metric T_α in the previous section. The basic assumption is that queries are different, thus requiring potentially different ranking functions to be individually effective. Previous query-dependent models [4, 18, 3] ignore risk. We devise learning to rank algorithms that directly optimize the tradeoff metric in the context of LambdaMart boosted regression trees framework [37].

4.3.1 Model and optimization algorithm

We choose to extend LambdaMart [37] for our ranking model. Boosted regression trees have been shown to be one of the best learning to rank models. In the recent Yahoo! Learning to Rank Challenge [9], boosting (and ensemble methods) have been a dominant technique used by winning entries. Boosted regression trees are a type of non-linear model that can capture the potentially complex interactions between various types of ranking features.

LambdaMart is derived from the tree-boosting optimization MART [17] and the listwise LambdaRank [7]. LambdaMart improves an effectiveness metric by fitting a sequence of boosted regression trees using an approximation to the derivative of a cost function by accumulating pairwise gradients weighted by the total change from the current ranking to a desired ranking.

We now present a brief overview on LambdaMart objective function as it serves as a foundation for deriving the learning algorithm for the tradeoff metric. The objective function of list-wise LambdaMart is based on the cross-entropy objective function used by RankNet [5]. Let r_i denote the relevance grade of a document D_i, so that $R_{ij} = 1$ if $r_i > r_j$, and $R_{ij} = -1$ when $r_j > r_i$. Let s_i denote the score assigned to D_i by the model. For documents D_i and D_j, the cross-entropy is expressed as follows:

$$C = \frac{1}{2}(1 - R_{ij})\beta(s_i - s_j) + \log\left(1 + e^{-\beta\cdot(s_i - s_j)}\right) \qquad (5)$$

where β is a shape parameter for the sigmoid function. This cost function assigns zero penalty if two documents are correctly ranked, and asymptotically, assigns linear penalty if they are ranked incorrectly. The derivatives of C with respect to score difference between s_i and s_j is:

$$\lambda_{ij} = \beta\left(\frac{1}{2}(1 - R_{ij}) - \frac{1}{1 + e^{\beta(s_i - s_j)}}\right) \qquad (6)$$

LambdaRank [7] and LambdaMart [37] modify this gradient defined on pairwise loss to optimize for list-wise IR measures. The new gradient λ_{ij}^{new} captures the desired change of document scores with respect to the IR metric after the documents have been sorted by their scores, in addition to capturing the original λ_{ij} value. The new gradient under the list-wise optimization is simply defined to be λ_{ij} multiplied by the absolute change in IR measure M due to swapping documents i and j:

$$\lambda_{ij}^{new} = \lambda_{ij}|\Delta M_{ij}| \qquad (7)$$

The gradient for each document D_i is obtained by summing λ_{ij} over all pairs that D_i participates in for query Q:

$$\lambda_i^{new} = \sum_{j \neq i} \lambda_{ij}|\Delta M_{ij}| \qquad (8)$$

The gradients λ_i^{new} defined on each document for each query are modeled and optimized by LambdaMart regression trees. In principle, LambdaMart can be extended to optimize any standard IR metric by simply replacing ΔM_{ij} in Eq. 8 by the corresponding change $\Delta M_{ij}^\star$ in the optimized metric $M^\star$. However, it is generally considered desirable for $M^\star$ to satisfy the *consistency property*: any pairwise swap between correctly ranked documents D_i and D_j must lead to decrease in $M^\star$, i.e., $\Delta M_{ij}^\star < 0$ if $r_i > r_j$. Generally consistency is desired since LambdaMart's approximation to the overall gradient is derived from the pairwise gradients.

Unlike standard IR metrics, the tradeoff measure is defined relative to a baseline ranking and it is a meta-measure – a combination of multiple measures. Despite its importance, little existing literature has provided insight into how to optimize meta-measures in a principled way. Next, we prove the consistency property of the tradeoff metric T and show how it can be optimized by boosted regression trees.

4.3.2 Consistency of Risk-Reward Tradeoff T_α

While LambdaMart is a list-wise optimization scheme and can optimize a wide range of IR measures – including NDCG, MAP, and MRR – it is often desirable for an objective measure to satisfy the *consistency property* [6] stated above: when swapping ranked positions of two documents, d_i and d_j, in a sorted document list where d_i is more relevant than d_j, and ranks before d_j, the objective should decrease. While most IR measures easily satisfy this property, in general an arbitrary objective measure M will not always satisfy the consistency property. For example, suppose measures A and B are NDCG@5 and NDCG@10, respectively, and we define a meta-measure $M = A - B$. It is easy to see that M does *not* satisfy the consistency property, because as a correct swap is made, the gain in NDCG@5 is offset by gain in NDCG@10, which may result in a negative change for the overall measure M, causing M to be inconsistent. In general, it is not straightforward to see whether a meta-measure, defined as a combination of different measures, can ensure the consistency property.

We now show that the tradeoff metric T, defined as a weighted linear combination of risk and reward in Eq. 4, satisfies the consistency property.

THEOREM 1. *The tradeoff metric T in Eq. 4 satisfies the consistency property.*

PROOF. Let M_m and M_b be the effectiveness of the LambdaMart model and baseline model respectively. After swapping documents d_i and d_j, denote the resulting change in tradeoff by Δ_T, change in effectiveness by Δ_M, change in reward by Δ_γ and change in risk by Δ_σ. Let Rel(d) be the relevance of document d. To show T is consistent we show that swapping documents d_i and d_j, where d_i is more relevant than d_j and ranks before d_j, results in a decrease of T or equivalently $\Delta_T < 0$. We consider two scenarios as new trees are added to the current LambdaMart ensemble: 1) $M_m \leq M_b$ and 2) $M_m > M_b$. We show T is consistent in both cases.

Scenario A: $M_m \leq M_b$

Case A1) Swap d_i and d_j, where $\mathrm{Rel}(d_i) > \mathrm{Rel}(d_j)$ and $i < j$. In this case $\Delta_\gamma = 0$ and $\Delta_\sigma = \Delta_M < 0$. Thus $\Delta_T = (1+\alpha) \cdot \Delta_M < 0$ as desired.

Case A2) Swap d_i and d_j, where $\mathrm{Rel}(d_i) > \mathrm{Rel}(d_j)$ and $i > j$. Then there are two subcases:

$$\begin{aligned} &\text{If } M_b > M_m + \Delta_M \text{ then} && \Delta_\gamma = 0,\ \Delta_\sigma = \Delta_M \quad \text{and} \\ & && \Delta_T = (1+\alpha) \cdot \Delta_M > 0. \\ &\text{If } M_b \leq M_m + \Delta_M \text{ then} && \Delta_\gamma = M_m + \Delta_M - M_b \\ &\quad \Delta_\sigma = M_b - M_m \quad \text{and} && \Delta_T = \alpha \cdot (M_b - M_m) + \Delta_M. \end{aligned}$$

Then $\Delta_T > 0$, as desired.

Scenario B: $M_m > M_b$

Case B1) Swap d_i and d_j, where $\mathrm{Rel}(d_i) > \mathrm{Rel}(d_j)$ and $i < j$. Then there are two subcases:

$$\begin{aligned} &\text{If } M_b > M_m - |\Delta_M| \text{ then } \Delta_\sigma = -M_b + (M_m - |\Delta_M|) < 0 \\ &\ \Delta_\gamma = M_b - M_m < 0 \text{ and } \Delta_T = \alpha \cdot (M_m - M_b) - (1+\alpha) \cdot |\Delta_M| \\ &\text{If } M_b \leq M_m - |\Delta_M| \text{ then } \Delta_\sigma = 0 \quad \text{and} \quad \Delta_T = \Delta_M. \end{aligned}$$

Then $\Delta_T < 0$, as desired.

Case B2) Swap d_i and d_j, where $\mathrm{Rel}(d_i) > \mathrm{Rel}(d_j)$ and $i > j$. There is no risk and $\Delta_M > 0$, so $\Delta_T > 0$ as desired.

Thus, in all cases the tradeoff measure T satisfies the required consistency property. □

We note that NDCG, MRR, and MAP (Mean Average Precision) have all been shown to satisfy consistency [14], and thus Theorem 1 applies generally to risk-reward tradeoff measures based on these widely-used IR measures, as well as any others that have been proven consistent.

4.3.3 Risk-sensitive LambdaMart optimization

The details of the proof for Theorem 1 above also tell us how to compute the Δ_T's used in the LambdaMart gradients for optimizing T. We observe that by optimizing the tradeoff metric, the LambdaMart puts more emphasis on the "risky" queries, by modifying the gradient of such queries so that incorrect results for risky queries hurt the tradeoff metric more, and the algorithm learns to optimally utilize query-dependent features in order to be more risk-averse. To illustrate this point, we see that when the current model effectiveness is less than the baseline's and a correct swap is made (i.e., Case A2), the optimizer assigns a larger positive change to the tradeoff if the swap erases the performance difference between the model and baseline (i.e., $\Delta_T = \alpha \cdot (M_b - M_m) + \Delta_M > (1+\alpha) \cdot (M_b - M_m)$); otherwise, it assigns a *smaller* positive tradeoff change to the query ($\Delta_T = (1+\alpha) \cdot \Delta_M < (1+\alpha) \cdot (M_b - M_m)$). As another example, in the case where the current model has a better effectiveness than the baseline and when two documents are incorrectly swapped (i.e., Case B1), the algorithm assigns a non-zero risk to the query if the swap causes the model to degrade below the baseline effectiveness; otherwise, it assigns no penalty. Such risk-sensitive optimization is in sharp contrast to the standard LambdaMart algorithm which treats all queries identically, without paying extra attention to high-risk queries. Thus, our risk-sensitive framework generalizes standard learning-to-rank algorithms like LambdaMart that only optimize for average effectiveness gain without accounting for risk.

5. EXPERIMENTS

We now show the benefits of augmenting the standard average-gain maximization ranking objective with our risk-sensitive objective. We emphasize that our key learning goal is not to maximize NDCG gains alone, or to minimize variance alone, but *to achieve the best possible joint tradeoff between change in risk and change in reward.* The ability to control such tradeoffs can be important in operational decisions, and in giving a better picture of the achievable performance space of a ranking algorithm. We also analyze the effect of the risk-aversion parameter α on the convergence rate of the learning algorithm and on characteristics of risk-averse rankings.

5.1 Datasets

We report results using two datasets, one public and one proprietary. These datasets were chosen to illustrate different applications of risk-sensitive ranking for different types of queries and Web applications. For clarity, we will use "baselines" to refer to the models used to relativize loss in our risk-reward optimization. The model without any risk sensitivity (i.e., $\alpha = 0$), we will call the *gain-only model.*

5.1.1 MSLR learning-to-rank dataset

We use MSLR-WEB10K, a widely-used publicly released dataset from Microsoft Research[2]. MSLR is a sampling of 10,000 Web search queries and their results from a major commercial search engine that includes graded relevance judgments for supervised learning-to-rank experiments. This dataset is pre-partitioned into five folds, each with train, test, and validation subsets. We use and report results using these same folds in our experiments. We report NDCG@1 and NDCG@10 since both measures are of importance in Web search evaluation. Since our optimization is performed relative to a baseline, we have taken the ranking obtained from sorting by BM25 using the `BM25.whole.document` feature as a baseline. Thus, on this dataset, the goal is to make a more robust tradeoff relative to a well-motivated IR formula for ranking.

5.1.2 Location personalization dataset

We use a proprietary dataset for learning to personalize Web results by user location, with details and baseline results as described in [1]. That work introduced a methodology for automatically learning a location distribution to associate with a search result with the goal of promoting results when the user's search location matches the search result's location distribution. The data consists of logs from a competitive top commercial search engine and labels based on a single binary *last satisfied click* by the user per query impression as an implicit relevance judgment. We follow the same methodology as [1] and report average change across test folds in MRR relative to the search engine's original ranking for the last satisfied click. Thus, on this dataset, the goal is to learn both how to personalize and when to personalize simultaneously.

We next define some key evaluation terms used in our experiments.

[2]`research.microsoft.com/en-us/projects/mslr/`

5.2 Evaluation measures

A *retrieval quality* measure for a ranked result list is simply a standard statistic that measures some aspect of relevance. The retrieval quality measures we use to report results on the Web data sets in our study are Non-Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR).

Given a baseline ranking, the *reward* of using a retrieval algorithm over a set of queries is the total *positive changes* in retrieval quality compared to the baseline ranking divided by the total number of queries (Eq. 2). The *risk* of using a retrieval algorithm is the total negative changes (losses) in retrieval quality compared to the baseline ranking divided by the total number of queries (Eq. 1). Visually, risk corresponds to the total mass on the left of the vertical zero line in Figure 1 normalized by the total number of queries, while reward is the total mass on the right half normalized by the total number of queries. We call the *overall gain* – or simply gain – of an algorithm the average combined gain over *all* queries, so that $Gain = Reward - Risk$. There is typically a *risk-reward* tradeoff for retrieval algorithms [10].

Taken together, the $(risk, gain)$ tradeoff pair provides a more complete picture of a ranker's performance profile than simply reporting average NDCG gain. Often, algorithms that appear similar in NDCG gain can differ greatly in their risk-sensitivity, for example. This is evident in some of our results later in Sec. 5.6.

5.3 Query Performance Selective Strategies

Query-dependent features such as click entropy [15], query clarity [21], query length [12], and the maximum of BM25 scores [20] have been shown to be highly indicative of individual query performance and can be used to differentiate performance between different kinds of queries. An alternative approach to attempting to learn a model is simply to selectively apply a learned model by leveraging the insights from query performance prediction. We provide two such selective methods for comparison points. For the MSLR dataset, we use insights regarding the maximum BM25 score from [20] and use the max of the `BM25.whole.document` feature to predict when the baseline (the ranking obtained from the same feature) will perform well. We can then simply sweep out thresholds by using the gain-only learned model when the max BM25 score is low and the BM25 ranking otherwise. We then vary this threshold in an analogy to varying the α parameter. We refer to this as *maxBM25*.

For the location dataset, we also present a selection strategy. For each search result in the location dataset, the entropy of the result's location distribution measures how location-specific a single search result is. When the minimum of this feature over all search results is low, it means that at least one search result has a very location-specific meaning, and therefore, location personalization is more likely to succeed. We again vary the threshold using the personalized gain only model when the minimum is low and using the baseline original ranking of the search engine otherwise. We refer to this as *minEntropy* below.

5.4 Learning methodology

To ensure all methods have the same information available, both the gain-only models and the risk-sensitive models also have the features (max BM25 for MSLR and minimum entropy for location) that are used by the selective methods. In the case of MSLR, in order to ensure strong gain-only model performance, we performed sweeps on LambdaMart parameters using the validation sets. On the MSLR data, for $\alpha = 0$ (i.e. standard optimization), for each fold we used performance on the validation data to do a grid search for parameter settings. The parameter ranges we experimented with were: minimum documents in leaf $m \in 500, 1000, 1500$; number of leaves $l \in 10, 25, 50$; number of trees in ensemble $t \in 50, 100, 200, 400, 800$ and learning rate $r \in 0.025, 0.05, 0.07$. For three folds the optimal values were $m = 1000$, $l = 50$, $t = 800$ and $r = 0.075$, and for the two remaining folds, the values were identical except $m = 500$. For the models learned using $\alpha > 0$, we used these same parameters to keep other variables constant. For consistency of comparison we used the same parameter setting for the Location dataset experiments as in [1].

5.5 Summary statistics

Tables 1 and 2 show summary retrieval quality and robustness statistics for ranking on the MSLR and Location datasets. The gain over baseline effectiveness is broken down into its constituent Risk and Reward components. (Recall that $Gain = Reward - Risk$.) Unless otherwise noted, other summary stats are based on the top 10 results. The Wins/Losses row shows the counts of queries that contributed to the Reward vs Risk respectively.[3] The 'Loss > 20%' statistic counts queries whose relative NDCG change over baseline NDCG was worse than -20%. Thus, smaller Loss numbers are better.

Risk, reward and wins/losses are relative to the baseline ranking. The NDCG@10 score for the BM25 baseline was 30.869. The ΔMRR score for the Location minEntropy baseline is zero, since we report relative gains on that dataset for proprietary reasons.

As expected, there is a clear trade-off between overall retrieval effectiveness and amount of risk reduction for both collections. As the risk-aversion parameter α is increased, the Wins/Losses row shows consistent reduction in losses, with a small decline in overall effectiveness (NDCG or ΔMRR). Note in particular that for the Location data, the chance of a query being hurt by more than 20% loss in ΔMRR compared to the baseline vanishes almost to zero, while still retaining significant overall effectiveness gains.

5.6 Risk-reward tradeoff

Finding a re-ranking strategy that reduces variance compared to the baseline is easy - simply avoid re-ranking and always use the baseline ranking. In this case, the ΔMRR will be zero, with zero risk. However, this obviously loses all the benefits of the re-ranking for queries that are actually helped by re-ranking. Our goal is to find optimal tradeoff strategies that *dominate* other possible strategies for trading risk for reward.

Figure 2 shows our risk-sensitive objective in LambdaMart indeed dominates alternative effective strategies for selective re-ranking, as measured by the tradeoff curve for risk vs. gain for each collection. This tradeoff curve for the selective re-ranking strategy is produced as a function of the feature threshold (the minEntropy feature for Location queries, and maxBM25 feature for MSLR queries). As the feature threshold is lowered, more and more queries have the feature

[3]We ignore queries with zero NDCG@10 difference from the baseline, with gains over baseline being a 'Win', and losses being a 'Loss'.

Collection/Statistics		All features			
		$\alpha = 0$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
MSLR ($N = 10000$)	NDCG@1	46.166	45.835	44.615	43.658
	NDCG@10	47.272	47.221	46.312	45.540
	Risk	2.239	1.974	1.722	1.540
	Reward	18.642	18.327	17.165	16.282
	Wins/Losses	7512/1972	7630/1845	7613/1838	7645/1776
	Loss > 20%	740	684	630	573

Table 1: Summary of generalization performance over test folds for the MSLR dataset.

Collection/Statistics		All features			
		$\alpha = 0$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
Location ($N = 541958$)	ΔMRR	1.451	1.144	0.784	0.555
	Risk	0.4725	0.193	0.080	0.051
	Reward	1.923	1.337	0.864	0.606
	Wins/Losses	43616/24499	32253/14952	23598/8041	19818/8090
	Losses > 20%	2539	718	266	14

Table 2: Summary of generalization performance over test folds for the Location dataset.

value above the threshold and are re-ranked according to the baseline ranking. The tradeoff curve for the risk-sensitive re-ranking strategy is a function of α. At $\alpha = 0$, the risk-sensitive ranking coincides with the gain-only ranking. As α is increased, the learned model exhibits a stronger preference for the baseline, which acts like an increasing 'soft' query selection for the baseline ranking. The dotted Break Even line shows the break-even point where the tradeoff objective $T = 0$, and any result above the dashed Proportional Reduction line has the desirable property that it reduces risk proportionally more than it reduces reward. The "near-optimal hull" curve represents an envelope of achievable tradeoffs that results from incorporating an automatic stopping strategy for α, details of which are given in Sec. 5.7.

For both collections it is notable that the selective threshold re-ranking method performs significantly better than the break-even point. Additionally, since it lies above the Proportional Reduction line, it reduces risk proportionally more than it reduces gain at almost all points of the curve. This is most evident at high values of the respective threshold features, when the 'best' queries amenable to that method are most likely to be selected. At much lower threshold values, it is clear that most queries do not benefit from re-ranking with the baseline feature, giving performance closer to break-even. Furthermore, while other selective personalization approaches could be taken, it is clear based on performance that the selective threshold method here provides an informative comparison point.

In turn, for the Location data, the risk-gain tradeoff achieved by risk-sensitive ranking completely dominates that of selective threshold re-ranking. For any given risk level, risk-sensitive ranking achieves significantly higher gain, and conversely, for any given level of gain, it achieves that gain at much lower risk than the selective method. The results also show that the near-optimal hull dominates the selective method for both collections.

5.7 Guidance for setting the α parameter

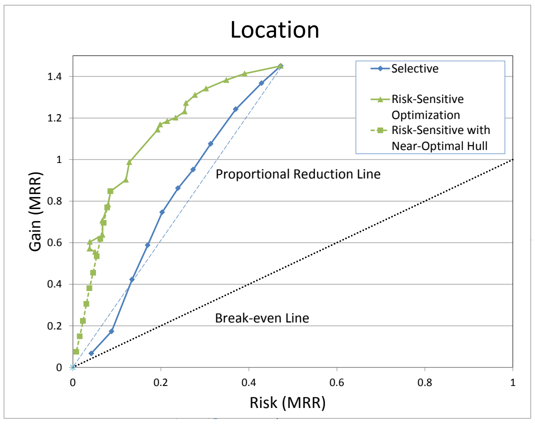
A natural question is how to set α, or where one should stop trying new α's. While α is a user-set parameter to control risk, the question of where to stop can be approximated. For any model, the risk can always be reduced by flipping a coin with probability p and applying the method given "heads" and the baseline given "tails". Varying p naturally both reduces the gain and the risk (e.g., $p{=}0.20$ yields 20% risk and 20% reward). Since we can apply the same logic to the gain only model, this leads to natural guidance in terms of the gain-to-risk ratio of the gain only model (GOM): when $\alpha = Gain_{GOM}/Risk_{GOM}$ the original gain only model would yield a tradeoff objective $T{=}0$. Furthermore, randomly reducing the gain only model for greater values of α than this will continue to ensure the tradeoff objective remains non-negative. Thus, solutions with $T > 0$ in the segment $\alpha \in [0, Gain_{GOM}/Risk_{GOM}]$ are interesting non-trivial tradeoffs relative to the original gain-only model.

To find the size of this segment of interest, we used performance over the training data. For MSLR, this yielded a value of $\alpha{=}27.9{\pm}1.3$ and for Location $\alpha{=}4.0{\pm}0.04$. Then starting from the maximal nearest computed α points (30 and 4 respectively) in the segment, we simulate the random reduction from there toward the origin and call this the "Near-Optimal Hull" line on the curves. This gives a more realistic picture of achievable tradeoffs.

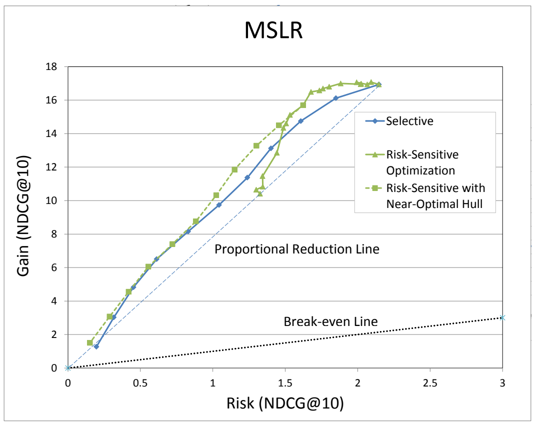
5.8 Convergence rates

We discussed in Sec. 4.3.3 how adding our risk-sensitive objective to LambdaMart results in modifying the LambdaMart gradient in such a way that its magnitude is amplified as α increases, accelerating the learning rate for queries where baseline effectiveness is already high. Thus, we hypothesized that on average across all training queries, we would see faster convergence rates at high values of α compared to low values of α.

Figure 4 demonstrates that this phenomenon actually occurs on both collections, comparing the ranker convergence when $\alpha = 0$ to setting $\alpha = 5$. The x-axis indicates number of iterations; the y-axis shows the overall NDCG@10 achieved at any given iteration (up to a maximum of 100 iterations), averaged across all validation folds. Although the algorithm ultimately converged to very similar NDCG@10

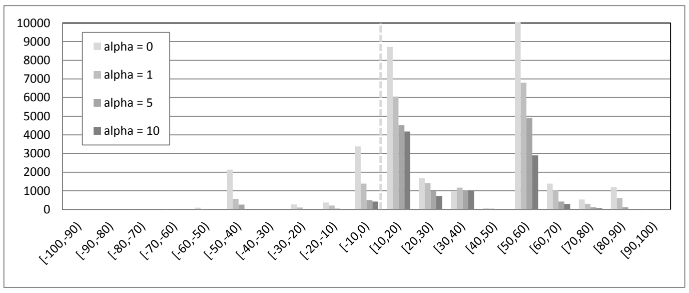


(a) Location: risk vs. gain

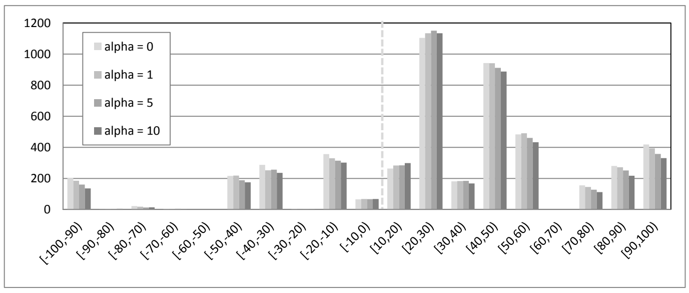


(b) MSLR: risk vs. gain

Figure 2: Risk/gain tradeoffs achieved by risk-sensitive LambdaMart learning vs. selective re-ranking for Location dataset (left), and MSLR dataset (right).



(a) Location: $\alpha = 0, 1, 5, 10$



(b) MSLR: $\alpha = 0, 1, 5, 10$

Figure 3: Helped-hurt histograms, showing how variance around baseline (dotted line) shrinks as the risk-aversion parameter α is increased from $\alpha = 0$ (lightest bars) to $\alpha = 10$ (darkest bars), for Location dataset (left), and MSLR dataset (right). Each histogram bucket corresponds to a different 10% range of loss/gain in effectiveness compared to that collection's baseline, and contains four bars, corresponding to the number of queries helped or hurt for the four risk-reward tradeoff settings $\alpha = 0, 1, 5, 10$. For clearer scaling the large spike of queries for the [0, 10) bucket has been omitted in both charts.

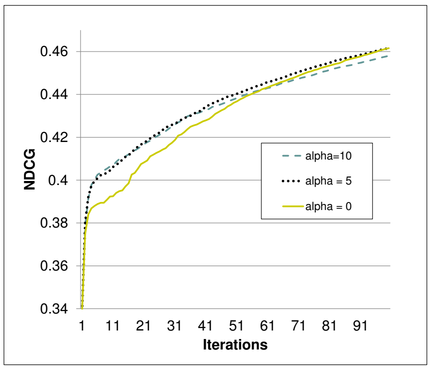
for both settings, the $\alpha = 5$ runs converged faster for all folds on both collections. In the case of the Location data, the speed improvement was dramatic – learning with $\alpha = 5$ took an average of 10.2 iterations to achieve 95% of the final convergence effectiveness, compared to 41.5 with $\alpha = 0$. The improvement was smaller but still consistent across folds for the MSLR data: for example, learning with $\alpha = 5$ reached NDCG@10 gain of 0.42 in an average of 25.4 iterations, compared with 33.4 iterations for $\alpha = 0$.

5.9 Model consistency across risk levels

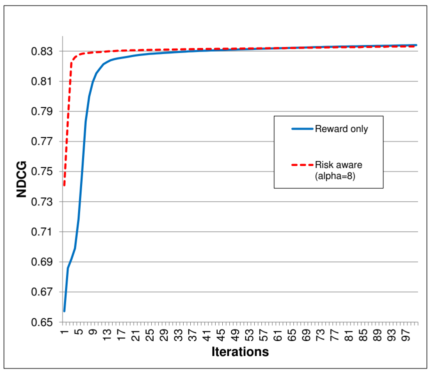
We now examine how rankings change with respect to the baseline results as the risk-aversion parameter α is increased. To measure change between rankings, we use Kendall's tau distance, a standard distance between two ranked lists that corresponds to the (normalized) number of pairwise swaps required to move from one ranking to another. A large Kendall distance indicates low similarity to the baseline ranking, and conversely a small Kendall distance indicates a ranking that is very similar to the baseline ranking.

For illustration purposes, we binned Kendall distance values into 10 bins from 0 to 9. Queries whose re-ranking is the same or very similar to the baseline ranking are put in Bin 0, while queries with re-rankings that are very different from the baseline are put into Bin 9, or an intermediate bin according to the Kendall distance.

Figure 5 shows, for each value of α, a distribution over Kendall bins using the location personalization dataset. It shows that as the permitted risk increases – alpha changes from high to low – more and more queries start to deviate significantly from the baseline. For example, the lower-risk model learned with $\alpha = 9$ has 85% of the queries staying close to the baseline (bin 0), while the higher-risk model learned with $\alpha = 1$ has only 42% of the queries close to the baseline ranking, with the remaining queries dispersed over much more diverse rankings with much larger Kendall distances from the baseline. This behavior can be explained in terms of risk/reward tradeoff. The higher-risk models have the incentive to deviate significantly from the baseline in exchange for more drastic improvement in effectiveness (which is val-



(a)



(b)

Figure 4: Risk-sensitive model (high α) consistently converges faster than reward-only objective ($\alpha = 0$), for both the MSLR dataset (top) and Location dataset (bottom). Each iteration point is averaged over all validation folds.

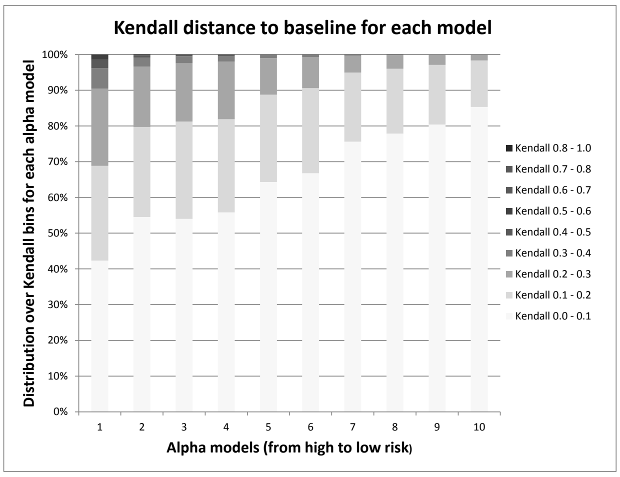


Figure 5: Distributions of Kendall distances to baseline ranking, for increasing values of the risk parameter α (location dataset). For high-risk models ($\alpha = 1$), only 42% of queries have re-ranked results that remain very close to the baseline ranking (Bin 0), compared to low-risk ($\alpha = 10$) models having 90% of queries with results very close to the baseline.

ued more by the tradeoff metric under a low α value); while low-risk models (high α value) tend to stay close to baseline, since risk is aggressively penalized by the tradeoff metric under large α and there is no incentive to significantly deviate from baseline (even if that means a higher gain) due to the potentially large penalty incurred from it. This fact is also confirmed by results shown in Table 2, where the higher-risk model trades off more robustness for effectiveness.

Moreover, in models that heavily penalize risk, the re-ranked results for a particular query may deviate significantly from baseline, but only under the condition that the reward (NDCG gain) acquired as a result of the deviation must be high enough to outweigh the increased risk. We can see this behavior in the risk-sensitive model $\alpha = 9$. Although a majority of the queries are in Bin 0 (near baseline), the overall NDCG gain is moderate and the overall risk is the lowest among all models. Also interesting is that as Kendall distance increases, the safe model tends to have the largest reward among other models under same kendall distance. For safe model ($\alpha = 9$), reward is 0.00371 under kendall bin 2, which is the highest among other models in the same Kendall bin 2. Thus, one potential application of this type of cross-model analysis might be to use the consistency across different models to produce the best reranking model for each query.

6. DISCUSSION AND FUTURE WORK

In this study we gave a definition of 'risk' in terms of the size of the downside tail of a distribution over a specific performance measure: NDCG. However, our framework is not limited to that choice and can address a much broader family of scenarios.

First, the risk and reward parts of the ranking objective may address other performance aspects of the results ranking, as long as the combined measure is consistent.

Second, we can look beyond strictly performance-related definitions of risk to include other properties of result sets. For example, the *churn* of a set of results for a given query refers to undesirable and unexpected volatility in a given query's results after re-ranking, even (or especially) when the rankings of the best relevant documents - and thus overall performance - might be unchanged or only minimally reduced. All things being equal, given two rankings with the same retrieval performance, we may prefer the one that gives less churn compared to a baseline ranking. Thus, the 'risk' of ranking might be defined using a ranking dissimilarity measure such as Kendall's tau compared to the baseline, instead of the strictly performance-based difference in NDCG we used here.

Third, another assumption we made was that reducing downside variance was a desirable goal. In general, this is true. However, there may be some retrieval tasks where it is beneficial to have an objective that tries to *increase upside variance*, even if this leads to an increased chance of large errors. Typically this will be in cases where a greater diversity of retrieval hypotheses is desirable as part of a larger precision-oriented system. For example, a search engine, acting as one source among several that provides candidate documents to a question-answering application, may be more likely to find high-reward documents if it does more aggressive promotion from deep in the original ranking.

Finally, we note that our use of an objective that minimizes negative outcomes with respect to a baseline is a special case of regret-based learning. In particular, we could generalize our risk-reward objective to minimize the average loss over the worst $X\%$ tail of negative outcomes, instead of all negative outcomes, and this could be viewed as a minimax regret objective - with connections to the widely-used Conditional Value-at-Risk objective from financial optimization. Beyond ranking, we believe this family of robust objectives deserves wider consideration in IR, with potential applicability to tasks such as query rewriting, federated resource selection, meta-search and others, where risk-reward scenarios exist and finding the highest-quality operational tradeoffs is critical.

7. CONCLUSIONS

We presented a general approach for improving the robustness of learned rankers by performing risk-sensitive optimization of ranking models. Our optimization adds a novel additional ranking objective that minimizes downside risk relative to a given baseline, in addition to the standard objective factor that maximizes average effectiveness across queries. Our approach can be viewed as a way to control the robustness/effectiveness tradeoff in learning-to-rank approaches. Thus, it generalizes the standard learning to rank approaches as special cases that do not consider robustness. Our robustness metric is orthogonal to previous multi-objective learning to rank metrics for capturing different facets for search, since robustness is defined *using* standard facet-based and IR performance measures *with respect to* a baseline model. We proved the theoretical consistency of the proposed tradeoff measure and designed a principled learning to rank approach, using boosted regression trees, to optimize it. Our approach can be potentially applied to a wide range of applications in learning-to-rank by plugging in different baseline models. Furthermore, by performing an extensive empirical evaluation using both public and proprietary datasets we demonstrated that our robust models achieve significant benefits in learning rate, robustness, and reliability over existing methods.

8. REFERENCES

[1] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *SIGIR*, pages 135–144, 2011.

[2] R. Bhattacharjee and A. Goel. Algorithms and incentives for robust ranking. In *SODA 2007*.

[3] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha. Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm. In *WWW*, pages 131–140, 2010.

[4] J. Bian, T.-Y. Liu, T. Qin, and H. Zha. Ranking with query-dependent loss for web search. In *WSDM*, pages 141–150, 2010.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[6] C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, Microsoft Research, 2010.

[7] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to Rank with Nonsmooth Cost Functions. In *NIPS*, pages 193–200. MIT Press, 2006.

[8] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR*, pages 621–622, 2006.

[9] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - Proceedings Track*, 14:1–24, 2011.

[10] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *CIKM 2009*, pages 837–846.

[11] K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *CIKM*, pages 403–412, 2011.

[12] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR*, pages 299–306.

[13] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *SIGIR*, pages 95–104, 2011.

[14] P. Donmez, K. M. Svore, and C. J. C. Burges. On the local optimality of lambdarank. In *SIGIR*, pages 460–467, 2009.

[15] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW*, pages 581–590, 2007.

[16] E. N. Efthimiadis. Chapter: Query expansion. *Annual Review of Information Science and Technology*, 31:121–187, 1996.

[17] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[18] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *SIGIR*, pages 115–122, 2008.

[19] F. Gey. Inferring probability of relevance using the method of logistic regression. In *Proc. 17th SIGIR Conference*, 1994.

[20] Q. Guo, R. W. White, S. Dumais, J. Wang, and B. Anderson. Predicting query performance using query, result, and user interaction features. In *RIAO*, 2010.

[21] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *CIKM*, pages 1419–1420, 2008.

[22] C. Kang, X. Wang, J. Chen, C. Liao, Y. Chang, B. L. Tseng, and Z. Zheng. Learning to re-rank web search results with multiple pairwise features. In *WSDM'11*, pages 735–744, 2011.

[23] B. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *J. of User Modeling and User-Adapted Interaction (UMUAI)*, 22, 2011.

[24] F. Li, X. Li, S. Ji, and Z. Zheng. Comparing both relevance and robustness in selection of web ranking functions. In *SIGIR '09*, pages 648–649, 2009.

[25] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3), 2009.

[26] Y. Lv, C. Zhai, and W. Chen. A boosting approach to improving pseudo-relevance feedback. In *SIGIR*, pages 165–174, 2011.

[27] D. Metzler and W. B. Croft. A Markov Random Field model for term dependencies. In *Proc. 28th SIGIR Conference*, pages 472–479, 2005.

[28] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. 21st SIGIR Conference*, pages 275–281, 1998.

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

[30] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley, 1986.

[31] K. M. Svore, M. N. Volkovs, and C. J. Burges. Learning to rank with multiple objective functions. In *WWW*, pages 367–376, 2011.

[32] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proc. 30th SIGIR Conference*, 2007.

[33] J. Teevan, S. T. Dumais, and E. Horvitz. Potential for personalization. *ACM TOCHI*, 17(1), 2010.

[34] E. M. Voorhees. The trec robust retrieval track. *SIGIR Forum*, 39(1):11–20, June 2005.

[35] J. Wang and J. Zhu. Portfolio theory of information retrieval. In *SIGIR*. ACM, 2009.

[36] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *CIKM*, pages 1009–1018, 2010.

[37] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13:254–270, June 2010.

[38] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Research and Development in Information Retrieval*, pages 543–550, 2007.

[39] J. Zhu, J. Wang, I. J. Cox, and M. J. Taylor. Risky business: modeling and exploiting uncertainty in information retrieval. In *SIGIR*. ACM, 2009.