Ranking Optimization with Constraints

Fangzhao Wu[†], Jun Xu[‡,*], Hang Li[‡], Xin Jiang[‡]
[†]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China
[‡]Noah's Ark Lab, Huawei Technologies Co. Ltd., Sha Tin, Hong Kong
wufangzhao@gmail.com, junxu@ict.ac.cn, {hangli.hl, jiang.xin}@huawei.com

ABSTRACT

This paper addresses the problem of post-processing of ranking in search, referred to as post ranking. Although important, no research seems to have been conducted on the problem, particularly with a principled approach, and in practice ad-hoc ways of performing the task are being adopted. This paper formalizes the problem as constrained optimization in which the constraints represent the post-processing rules and the objective function represents the trade-off between adherence to the original ranking and satisfaction of the rules. The optimization amounts to refining the original ranking result based on the rules. We further propose a specific probabilistic implementation of the general formalization on the basis of the Bradley-Terry model, which is theoretically sound, effective, and efficient. Our experimental results, using benchmark datasets and enterprise search dataset, show that the proposed method works much better than several baseline methods of utilizing rules.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

Keywords

Post Ranking; Ranking Optimization; Bradley-Terry Model

1. INTRODUCTION

Recent years have observed a significant progress in research and development on learning to rank, i.e., creation of ranking models in search using machine learning techniques. Now, it becomes a common practice to exploit the learning technologies to construct the basic ranking model of a search system. This paper is concerned with *post processing of ranking* , which we call post ranking. Post ranking is normally conducted at web search engines in ad-hoc manners. The paper aims to provide a principled approach to post ranking, which does not seem to have been seriously studied previously.

In practice, there are many situations in which one wants to further 'twist' the search results given by the basic ranking model, i.e., to conduct post ranking. Post ranking is widely adopted in practice, under the names of re-ranking, final ranking, etc. For example, the query is about a hot topic and one wants to boost a webpage about the topic from news channels to the top three positions, no matter how the ranking model does (note that it is usually hard to add such control into a learning to rank model). In another example, a web page is reported to be likely a spam page, and an immediate action is required to demote the position of the page, without change of the ranking model. (See more examples in Section 3.) Post ranking needs to be carried out not only from the viewpoint of enhancing search quality, but also due to operational, commercial, and even political reasons. There are other situations in which post ranking appears to be necessary, such as diversification of search result [6, 27, 7], context aware ranking [29], personalized ranking [25, 26, 23]. Therefore, post ranking is a necessary and important process for search.

Post ranking has the following characteristics. (1) It is usually query dependent, user dependent, or context dependent. (2) The effects of it may not be achieved by using the basic ranking model, because it is usually difficult, costly, or even impossible to implement in the basic ranking model. (3) It does not need learning or training.

The key challenge for post ranking lies in the difficulty of formalizing the problem in a theoretically sound, effective, and efficient way. The original search result might be very different from the rules of post processing, and the rules might also be contradictory to each other. Thus, it is not easy to incorporate the complicated controls into a single framework. Moreover, the process needs to be conducted online and thus must be very efficient.

This paper proposes formalizing ranking optimization as a *constrained optimization* problem. Given the ranking result of a query by the ranking model, we perform re-ranking on the result, by minimizing an objective function under a number of constraints, where the constraints represent the rules which we want to use for post processing, for example, boost a page to the top k position, and the objective function represents the trade-off between adherence to the original ranking and satisfaction of the constraints.

[*]Currently affiliated with Institute of Computing Technology, Chinese Academy of Sciences.

CIKM'14, November 03 - 07, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661895.

As the first study, we propose a method for ranking optimization. Our method adopts the Bradley-Terry model [2] for calculating the probability of a ranking list. It realizes the optimization problem as minimizing the negative log conditional probability of the original ranking list and the negative log conditional probability of the constraints given a Bradley-Terry model. The optimization problem has a simple form and is guaranteed to have a global optimal solution. Our method employs gradient decent to find the optimal solution, with linear order time complexity. It is thus a theoretically sound, effective, and efficient method.

We have conducted experiments using the LETOR benchmark datasets [14, 22] and a dataset from an enterprise search engine. We take as baselines several methods which adjust ranking results with heuristic rules. Experimental results indicate that our method consistently and significantly outperform the baseline methods in terms of NDCG on all datasets, indicating that it is better to employ our method in post ranking.

The contributions of the paper include (1) formalization of ranking optimization, (2) proposal of a method of ranking optimization, (3) empirical study of ranking optimization.

The rest of the paper is organized as follows. After an introduction to related work in Section 2, we describe the formulation of ranking optimization with constraints in Section 3. We describe the proposed method based on Bradley-Terry model in Section 4. Experimental results and discussions are given in Section 5. Section 6 concludes this paper and gives future work.

2. RELATED WORK

Construction of ranking model is one of the key problems in IR. Given a query, the ranking model assigns relevance scores to the retrieved documents and sorts the documents based on the scores, and thus it plays an important role in search. Traditionally, a ranking model is defined based on a small number of factors, e.g., term frequency, inversed document frequency, and document length. BM25 [24] and LMIR (Language Models for Information Retrieval) [21, 12] are such models. Recently, machine learning techniques are applied to construction of ranking model usually using a large number of features and a large amount labeled training data, referred to as learning to rank. Methods of learning to rank are categorized as 'pointwise' [18, 13], 'pairwise' [9, 8, 3, 4], and 'listwise' [5, 30, 31, 32, 28] methods, depending on the loss functions used. In this paper, we adopt the listwise method of LambdaMART [28] to train the basic ranking model.

As explained above, basic ranking is not enough, and post ranking is necessary in many cases. In practice, post ranking is conducted by using heuristic rules, and there has not been research on post ranking itself, as far as we know. There are several other ranking issues, which can be addressed through post ranking as well, such as search result diversification, personalized ranking, and context aware ranking.

In recent years, search result diversification arises as a hot topic in IR, in which the search system returns a list of documents which are not only relevant to the query but also cover many subtopics of the query. A common practice for diversification is to see diversification as a post ranking step after the ranking list based on relevance is created [6, 27, 7]. This is because it is usually hard to model relevance and diversity in a single framework.

Personalized search may also be realized in post ranking, for example, in what is called client side re-ranking [25, 26, 23]. Specifically, the ranking result by the basic ranking model is sent to the client side and re-ranking of the result based on the user's interest is conducted. One advantage of the approach is that re-ranking is carried out entirely on the client side and the privacy of user can be protected.

Xiang et al. [29] have proposed context aware ranking. For example, if the user has clicked a URL in the previous search in the same session, then it is very likely she will not click the same URL again when it appears in the result of the current search. That means that ideally the URL should be demoted in the current search result.

Probabilistic models for ranking have been studied in statistics and related fields from many years ago. The most popular ones include Plackett-Luce model [20, 15], Mallows model [16], and Bradley-Terry model [2, 10]. Plackett-Luce model is a stagewise generative model, which decomposes the process of generating a permutation of n objects into n sequential stages. Mallows model is a distance-based model, which defines the probability of a permutation according to its distance to a centroid permutation. Bradley-Terry model calculates the probability of a permutation by pairwise comparisons. See [17] for a review on the topic. The Plackett-Luce model has been utilized in learning to rank [5], and the Bradley-Terry model is utilized in ranking optimization in this paper.

3. RANKING OPTIMIZATION WITH CONSTRAINTS

Let us describe the process of post ranking. Given a query, a ranking list of documents is first created by the basic ranking model, presumably built by learning to rank. Post ranking may be then conducted, depending on the query, user, or context, which means a refinement of the original ranking list, in which some documents are moved up and some are moved down. The original ranking list is created mainly from the viewpoint of relevance between query and documents. The refined ranking list is further created from the viewpoint of quality, diversity, personalization, contextualization, and so on.

The actions of post processing can be realized by using heuristic rules, which is a common approach in practice. A rule can be "if the query is in a list of terminologies, then always have the Wikipedia page of the terminology on the top one position". Another rule can be "if the documents are retrieved by both the original query and refined queries, then have at least one document retrieved by the original query ranked at the top three positions" (to reduce the risk of topic shift). Yet another rule can be "if the query is 'presidential election debates', then make all the top webpages of different rounds of the debates grouped together in the ranking list".

How to use rules for post ranking is not a trivial issue. First, the rules may not be hard rules and they only represent a guideline for refining the initial ranking list. For example, a rule may be "the document should be ranked at top three positions", in which no specific position is decided. Second, multiple rules might be applied at the same time, and the rules might be contradictory to each other. Third, different orders of applications of rules might yield different final ranking results and thus the order of applications needs also be considered. Finally, the rules usually only affect a

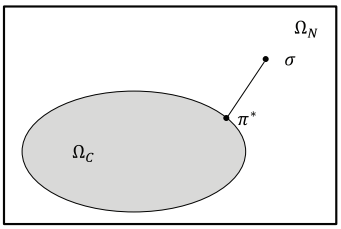


Figure 1: Illustration of ranking optimization with constraints.

small number of documents, it is important to make a balance between application of rules and preservation of the original ranking list.

In this paper, we formalize post ranking as constrained optimization, referred to as ranking optimization with constraints. The constraints represent the rules for post ranking, defined as functions over sets of permutations. The objective function represents the trade-off between adherence to the original ranking list and satisfaction of the constraints. The constraints are in fact soft constraints in the formulation. Post ranking is naturally performed by conducting the optimization problem. Therefore, the issues described above can all be naturally solved in the framework.

Suppose that σ denotes the original ranking list, $\mathcal{C}$ denotes the set of constraints, and π denotes the final ranking list of post ranking. The optimization can be written as follows

$$\min_{\pi \in \Omega_N} L(\sigma, \pi) + \lambda \cdot R(\mathcal{C}, \pi), \tag{1}$$

where L denotes agreement between σ and π, R denotes satisfication of $\mathcal{C}$ by π, λ denotes the tradeoff coefficient, and Ω_N denotes the set of all permutations (ranking lists) on the N documents in the current search. In the optimization process, we need to find the optimal final ranking π^*.

Suppose the set of constraints $\mathcal{C}$ is defined as $\mathcal{C} = \{c_i(\cdot)\}, c_i : \Omega_N \to \{0, 1\}$. If $c_i(\pi) = 1$, then permutation π violates constraint c_i, otherwise, $c_i(\pi) = 0$. The subset of permutations which do not violate the constraints are good candidates for the final ranking list.

We can define several types of constraints.

Top-k constraint: A document must be at top k positions.

Not-top-k constraint: A document cannot be at top k positions.

Clustering constraint: Two or more documents should be ranked together.

Diversity constraint: Two or more documents should not be ranked together.

Figure 1 illustrates the problem of ranking optimization with constraints (note that the figure is only for illustration purposes; the set of permutations forms a discrete set, not a Euclidean space.). Ω_N is the set of all possible permutations for N documents. $\Omega_{\mathcal{C}}$ is the subset of permutations satisfying the constraints in $\mathcal{C}$. σ is the original permutation given by the basic ranking model. In ranking optimization, we aim to find the optimal permutation (ranking list) π^* which is as close to σ as possible and in the meantime as within $\Omega_{\mathcal{C}}$ as possible.

4. OUR METHOD

In this section, we propose a method of ranking optimization on the basis of the Bradley-Terry model. The method only makes use of the top k constraint and the not top k constraint. We leave to future work the study of adding other constraints to the method.

4.1 Probabilistic Approach

We consider a probabilistic approach to ranking optimization. We assume that there exists a probabilistic ranking model M which gives rise to the ranking list π, by $\pi = \arg\max_\tau P(\tau|M)$ and we incorporate M into the optimization problem (1) to obtain

$$\min_{\pi \in \Omega_N, M} L(\sigma, \pi, M) + \lambda \cdot R(\mathcal{C}, \pi, M).$$

That is to say, we turn the optimization problem (1) with respect to π into an optimization problem with respect to both π and M. Let $L(\sigma, \pi, M) = -\log P(\sigma|M)$ and $R(\mathcal{C}, \pi, M) = -\log P(\mathcal{C}|M)$, where $P(\sigma|M)$ is the probability of generating the permutation σ given M and $P(\mathcal{C}|M)$ is the probability of generating all of the constraints in $\mathcal{C}$ given M. We first solve

$$\min_M -\log P(\sigma|M) - \lambda \cdot \log P(\mathcal{C}|M), \tag{2}$$

and then solve

$$\pi^* = \arg\max_{\pi \in \Omega_N} P(\pi|M). \tag{3}$$

where π^* denotes the optimal ranking list.

The interpretation of the method is as follows. Given the ranking list σ by the basic ranking model as well as the set of constraints $\mathcal{C}$, we want to first find a probability model M that can best explain the ranking list as well as the constraints (i.e., the product of the probabilities $P(\sigma|M)$ and $P(\mathcal{C}|M)$ is the largest, with a trade-off coefficient). After M is determined, we want to find the best ranking list given by M (i.e., the probability $P(\pi|M)$ is the largest).

In this paper, we choose the Bradley-Terry model for calculation of $P(\sigma|M)$ and $P(\mathcal{C}|M)$.

4.2 Using Bradley-Terry Model

The Bradley-Terry model represents the probability distribution of permutation of N documents (in general items) by making comparison among all pairs of documents. It assumes that the ranking model M is parameterized with a set of N scores $\Theta = (\theta_1, \cdots, \theta_N)$, each corresponding to a document. Furthermore, the parameters are assumed to be positive and sum to one, i.e., $\theta_i > 0$ for $i = 1, \cdots, N$ and $\sum_{i=1}^N \theta_i = 1$. In Bradley-Terry model, the probability of a preference pair (i, j) (document i be ranked higher than j) is defined as

$$p_{ij} = P\{(i, j)\} = \frac{\theta_i}{\theta_i + \theta_j}.$$

Given the permutation σ, the Bradley-Terry model defines the probability $P(\sigma|M)$ as being proportional to the product of probabilities of i ranked higher than j for all preference pairs (i, j):

$$P(\sigma|M) \propto \prod_{(i,j):\sigma(i)<\sigma(j)} p_{ij} = \prod_{(i,j):\sigma(i)<\sigma(j)} \frac{\theta_i}{\theta_i + \theta_j}.$$

Given the constraint set $\mathcal{C}$, the Bradley-Terry model defines the probability $P(\mathcal{C}|M)$ based on the preference pairs derived from $\mathcal{C}$:

$$P(\mathcal{C}|M) \propto \prod_{c \in \mathcal{C}} \prod_{(i,j) \in \mathcal{P}^c} p_{ij} = \prod_{c \in \mathcal{C}} \prod_{(i,j) \in \mathcal{P}^c} \frac{\theta_i}{\theta_i + \theta_j},$$

where $\mathcal{P}^c$ is the set of preference pairs derived from the constraint c.

The methods for deriving preference pairs depend on the types of constraints. Here, we give methods for the top-k constraint and not-top-k constraint. Given a top-k constraint c, the set of preference pairs $\mathcal{P}^c$ is defined as

$$\mathcal{P}^c = \{(i,j)|j : \sigma(j) > k\},$$

where i is the document to promote in constraint c. Similarly, Given a not-top-k constraint c, the set of preference pairs is

$$\mathcal{P}^c = \{(j,i)|j : \sigma(j) \leq k\},$$

where i is the document to demote in constraint c. We note that the top-k and not-top-k constraints can only be defined on the original ranking list σ, because only after the ranking list is given one can perform post ranking on it, i.e., impose constraints on it. Thus, the top-k and not-top-k constraints restrict the positions of a document in the original ranking σ.

Thus, the optimization in Equation (2) becomes

$$\begin{aligned} \min_{\Theta} f(\Theta) = -\sum_{(i,j):\sigma(i)<\sigma(j)} \log \frac{\theta_i}{\theta_i+\theta_j} - \sum_{c\in\mathcal{C}} \left(\rho^c \cdot \sum_{(i,j)\in\mathcal{P}^c} \log \frac{\theta_i}{\theta_i+\theta_j} \right) \\ \text{subject to} \forall i : \theta_i > 0, \sum_{i=1}^{N} \theta_i = 1, \end{aligned} \tag{4}$$

where $\rho^c > 0$ is the weight for constraint c. Note that parameter λ in Equation (2) has been merged to parameters ρ^c's in Equation (4).

The final ranking π^*, then, can be obtained via the maximization in Equation (3). With the use of Bradley-Terry model, it can be simplified to sorting of the documents in descending order of scores in Θ.

4.3 Optimization

It is easy to demonstrate that $f(\Theta) = f(\alpha \cdot \Theta)$ for any $\alpha > 0$ (note that $\log \frac{\theta_i}{\theta_i+\theta_j} = \log \frac{\alpha\theta_i}{\alpha\theta_i+\alpha\theta_j}$). Thus we can get rid of the constraint $\sum_{i=1}^{N} \theta_i = 1$ in Equation (4). This is because for any solution we can normalize the θ_i's by dividing them with $\sum_{i=1}^{N} \theta_i$. The normalized solution will satisfy the constraint and keep f unchanged.

Furthermore, the constraints of $\theta_i > 0$ can be discarded by replacing θ_i with e^{s_i} for $i = 1, \cdots, N$. Thus, the optimization problem in (4) becomes the following unconstrained optimization problem:

$$\begin{aligned} \min_{\mathcal{S}} f(\mathcal{S}) = \sum_{(i,j):\sigma(i)<\sigma(j)} (\log(e^{s_i} + e^{s_j}) - s_i) \\ + \sum_{c\in\mathcal{C}} \left(\rho^c \cdot \sum_{(i,j)\in\mathcal{P}^c} (\log(e^{s_i} + e^{s_j}) - s_i) \right), \end{aligned} \tag{5}$$

where $\mathcal{S} = \{s_1, s_2, \cdots, s_N\}$ is the set of parameters. The objective function $f(\mathcal{S})$ in Equation (5) is convex, as stated in Theorem 4.1.

THEOREM 4.1. *$f(\mathcal{S})$ is a convex function.*

Proof of the theorem can be found in Appendix. Theorem 4.1 indicates that the objective function f can be efficiently optimized with gradient descent. The gradient w.r.t. $\mathcal{S}$ can be written as $\frac{\partial f}{\partial \mathcal{S}} = \{\frac{df}{ds_1}, \cdots, \frac{df}{ds_N}\}$, where $\frac{df}{ds_i}$ is defined as

$$\begin{aligned} \frac{df}{ds_i} = \left(\sum_{j:\sigma(j)<\sigma(i)} \frac{e^{s_i}}{e^{s_i}+e^{s_j}} - \sum_{j:\sigma(i)<\sigma(j)} \frac{e^{s_j}}{e^{s_i}+e^{s_j}} \right) \\ + \sum_{c\in\mathcal{C}} \rho^c \left(\sum_{j:(j,i)\in\mathcal{P}^c} \frac{e^{s_i}}{e^{s_i}+e^{s_j}} - \sum_{j:(i,j)\in\mathcal{P}^c} \frac{e^{s_j}}{e^{s_i}+e^{s_j}} \right). \end{aligned} \tag{6}$$

Thus, the updating criterion for gradient descent is

$$\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} - \gamma^{(t)} \left. \frac{\partial f}{\partial \mathcal{S}} \right|_{\mathcal{S}=\mathcal{S}_{t-1}}, \tag{7}$$

where t is the iteration number, $\gamma^{(t)}$ is the optimal step size at the i-th iteration which is determined by backtracking. Algorithm 1 shows the pseudo code of the optimization algorithm.

The final document ranking π^*, then, is obtained by sorting with the scores in Θ returned in Algorithm 1.

Algorithm 1 Ranking Optimization Algorithm

Require: Initial ranking σ, constraints $\mathcal{C}$, and shrinkage rate $0 < \alpha < 1$
1: $\mathcal{S}^{(0)} \leftarrow$ random values
2: $t \leftarrow 1$
3: **repeat**
4: $\quad \nabla\mathcal{S} = \left.\frac{\partial f}{\partial \mathcal{S}}\right|_{\mathcal{S}=\mathcal{S}^{(t-1)}}$ {Equation (6)}
5: $\quad \gamma \leftarrow 1$
$\quad$ {search optimal step size using backtracking}
6: $\quad$ **while** $f(\mathcal{S}^{(t-1)} - \gamma\nabla\mathcal{S}) > f(\mathcal{S}^{(t-1)}) - \frac{\gamma}{2}\| \nabla\mathcal{S} \|^2$ **do**
7: $\quad\quad \gamma \leftarrow \alpha\gamma$
8: $\quad$ **end while**
9: $\quad \mathcal{S}^{(t)} \leftarrow \mathcal{S}^{(t-1)} - \gamma\nabla\mathcal{S}$ {Equation (7)}
10: $\quad t \leftarrow t + 1$
11: **until** convergence
12: **return** $\Theta = \{\frac{e^{s_1}}{Z}, \cdots, \frac{e^{s_N}}{Z}\}$, where $Z = \sum_{n=1}^{N} e^{s_n}$

4.4 Analysis

4.4.1 Convergence

We analyze the convergency of Algorithm 1 and have the following theorem:

THEOREM 4.2. *Algorithm 1 converges in finite steps and the convergence rate is $O(\frac{1}{\epsilon})$, where $\epsilon > 0$ is the tolerance.*

Proof of the theorem can be found in Appendix. Theorem 4.2 implies that Algorithm 1 can return an optimal ranking model in a reasonably short time, which makes it possible to apply the algorithm online.

4.4.2 Intuitive Explanation

The gradient in Equation (6) has an intuitive explanation. The gradient consists of two parts, one based on the original ranking list σ and the other based on the constraint set $\mathcal{C}$. Both parts contribute to the gradient and are calculated on the basis of preference pairs.

Given a preference pair (i,j), there will be a force that pushes the preferred document i upward, by subtracting a positive term $\frac{e^{s_j}}{e^{s_i}+e^{s_j}} = \frac{1}{e^{s_i-s_j}+1}$ to the gradient $\frac{df}{ds_i}$ (note that in gradient descent s_i is updated with negative gradient). The term also indicates that the force is related to the

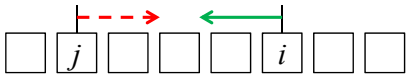


Figure 2: Intuitive explanation of the gradient. Given a preference pair (i, j), document i will be pushed upward (green solid arrow) and document j will be pushed downward (red dashed arrow). The strengths of the forces are identical.

Table 1: Statistics of datasets.

dataset	# queries	#documents	#relevance levels
MQ2007	1692	69623	3
MQ2008	784	15211	3
OHSUMED	106	16140	3
.Gov	50	49058	2
Enterprise	183	5464	3

difference between the scores. A small $s_i - s_j$, which means the current scores s_i and s_j do not agree with the preference pair, leads to a strong force to promote document i. In the same time, the un-preferred document j will be pushed downward, by adding the same term to gradient $\frac{df}{ds_j}$. Figure 2 illustrates the forces that respectively push the document i and document j upward and downward.

Given all of the preference pairs derived from σ and from $\mathcal{C}$, the overall forces that promote (or demote) a document i are jointly determined by all the preference pairs related to document i.

5. EXPERIMENTS

We conducted experiments to test the performances of our method for post ranking (ranking optimization).

5.1 Experiment Setting

We know of no existence of public data available for post ranking. As approximation, we used relevance datasets for the experiments. That is, we assume that we know that some documents are relevant (i.e., should be ranked high) and boost the ranks of the documents in post ranking.

We made use of the following subsets of the LETOR benchmark dataset: MQ2007, MQ2008, OHSUMED and .Gov, as well as a dataset of enterprise search, denoted as Enterprise. Table 1 gives the statistics of the four LETOR datasets [14, 22] and the Enterprise dataset.

The Enterprise dataset consists of 183 queries; each query is associated with about 30 documents. In total, the dataset contains 5464 query-document pairs. Each query-document pair is assigned with a label representing relevance at three levels: Good, Fair, or Bad. The dataset is split into training data (130 queries) and test data (53 queries).

The ranking models were trained using LambdaMART [28], which is a state-of-the-art method in learning to rank. The standard features in LETOR datasets were utilized and we also defined 18 features for Enterprise, including BM25 [24] and word level edit distance[1], etc.

Two types of constraints were tested in our experiments: the top-k constraints and not-top-k constraints. For each test query, we generated one top-k ($k = 1, 3, 5$) constraint and one not-top-k ($k = 5, 10$) constraint based on the labels of documents with respect to the query[2]. Specifically, we sorted the documents according to their labels (to obtain a perfect ranking) and randomly selected one document i from the top k positions. Then we created a constraint c which states that the selected document i should be ranked to top k positions in the final ranking. Similarly, we also created a not-top-k constraint by randomly select a document j from positions after k in the perfect ranking.

For simplicity, we assumed that all top-k (and all not-top-k) constraints are equally important and take the values of ρ^t (and ρ^n). Thus, the ranking optimization of Equation (5) becomes

$$\begin{aligned}\min_{\mathcal{S}} f(\mathcal{S}) = &\sum_{(i,j):\sigma(i)<\sigma(j)} (\log(e^{s_i} + e^{s_j}) - s_i) \\ &+\rho^t \sum_{(i,j)\in\mathcal{P}^t} (\log(e^{s_i} + e^{s_j}) - s_i) \\ &+\rho^n \sum_{(i,j)\in\mathcal{P}^n} (\log(e^{s_i} + e^{s_j}) - s_i)\,.\end{aligned} \tag{8}$$

As for baseline methods, we use the following four heuristics for modifying the original ranking:

Radical For the top-k constraint, Radical always ranks the selected document to top one position. For the not-top-k constraint, it always ranks the selected document to the bottom position of the list.

Moderate For the top-k constraint, Moderate always ranks the selected document to the middle of the top k positions. For the not-top-k constraint, it always ranks the selected document to the middle of the remaining list after k.

Conservative For the top-k constraint, Conservative always ranks the document to the position of k. For the not-top-k constraint, Conservative always ranks the document to the position of $k + 1$.

Proportional The above three baselines do not consider the position of the document in the original list. Proportional promotes the document selected by the top-k constraint to the position of $\lceil k \times \frac{pos}{N} \rceil$, where pos is the rank of the document in the original ranking list and $\lceil \cdot \rceil$ is the ceiling function. Therefore, the document will be ranked higher if its original position is also high. Similarly, the baseline demotes the document selected by the not-top-k constraint to the position of $\lceil k + pos(1 - \frac{k}{N}) \rceil$.

As evaluation measures, Normalized Discounted Cumulative Gain (NDCG) [11] at positions 1, 3, and 5 were used.

5.2 Experimental Results

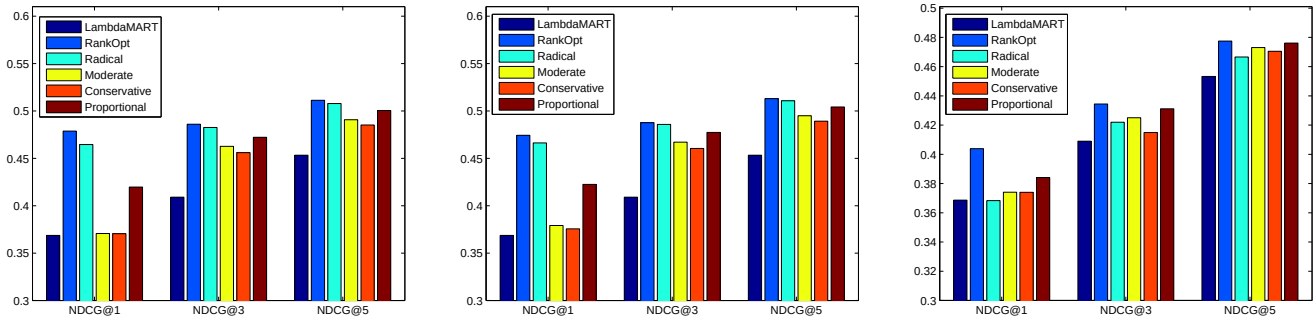
5.2.1 *Results on LETOR*

In all the four datasets in LETOR, the queries and associated documents were split to 5 subsets, and 5-fold cross validations were conducted. The performances reported here are the averages over 5 trials.

For each dataset, we used the training data to learn the basic ranking model, the validation data to tune the parameters, and the test data to perform ranking optimization. There are two parameters ρ^t and ρ^n tuned with the validate

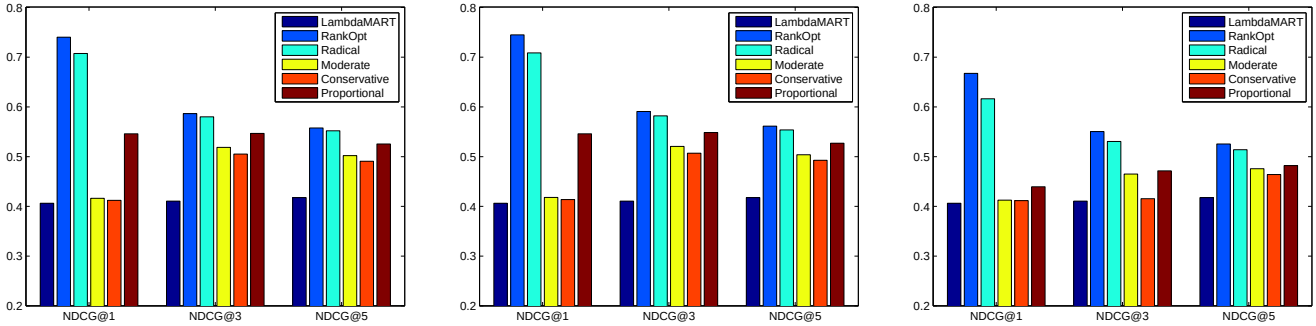
[1] http://en.wikipedia.org/wiki/Edit_distance

[2] Ranking optimization can be conducted or not conducted depending on queries. Here for experimentation purpose, ranking optimization is assumed to be carried out for all queries.



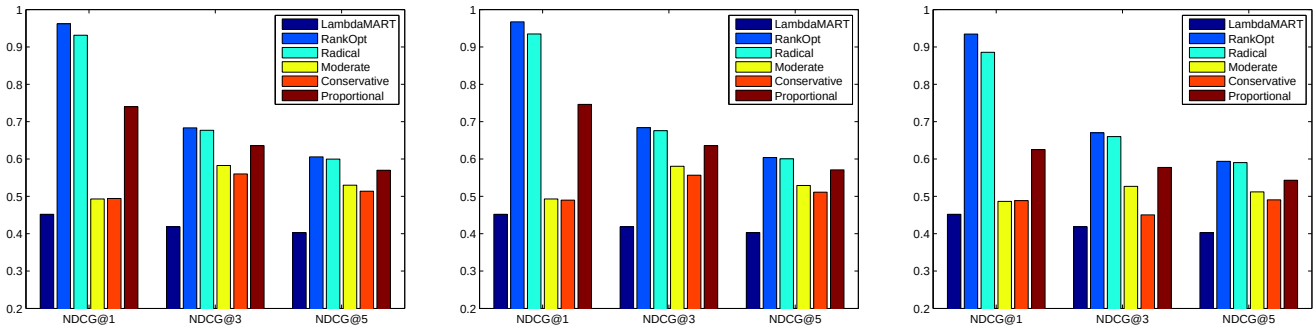
(a) top-3, not-top-5, ρ^t=10, ρ^n=0 (b) top-3, not-top-10, ρ^t=10, ρ^n=10 (c) top-5, not-top-10, ρ^t=10, ρ^n=10

Figure 3: Ranking accuracies on MQ2008 dataset.



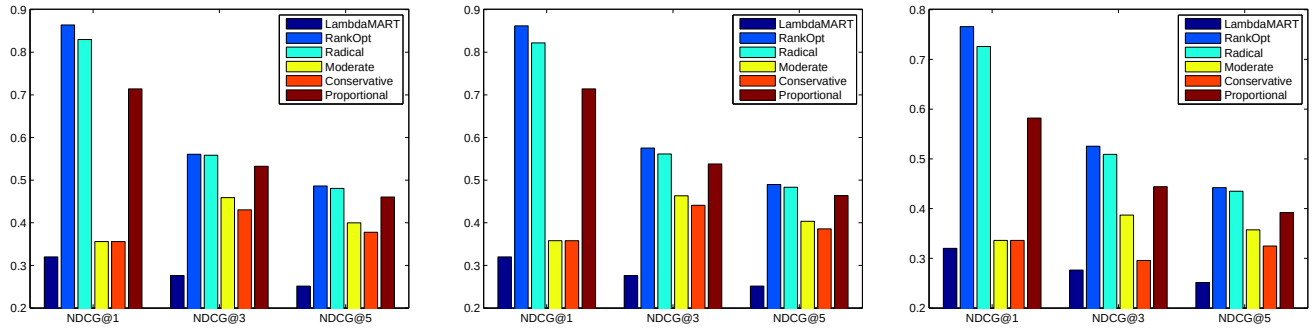
(a) top-3, not-top-5, ρ^t=100, ρ^n=10 (b) top-3, not-top-10, ρ^t=100, ρ^n=10 (c) top-5, not-top-10, ρ^t=100, ρ^n=10

Figure 4: Ranking accuracies on MQ2007 dataset.



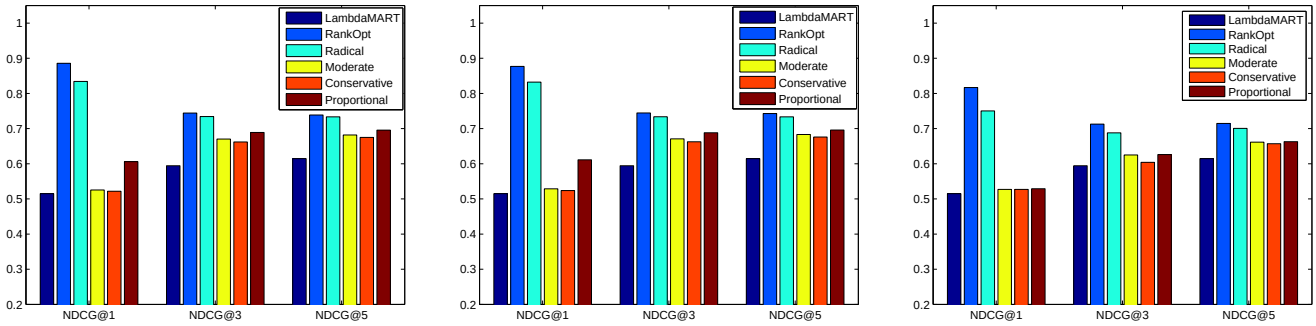
(a) top-3, not-top-5, ρ^t=200, ρ^n=0 (b) top-3, not-top-10, ρ^t=200, ρ^n=10 (c) top-5, not-top-10, ρ^t=200, ρ^n=30

Figure 5: Ranking accuracies on OHSUMED dataset.



(a) top-3, not-top-5, ρ^t=50, ρ^n=50 (b) top-3, not-top-10, ρ^t=50, ρ^n=50 (c) top-5, not-top-10, ρ^t=50, ρ^n=50

Figure 6: Ranking accuracies on .Gov dataset.



(a) top-3, not-top-5, ρ^t=60, ρ^n=40 (b) top-3, not-top-10, ρ^t=60, ρ^n=40 (c) top-5, not-top-10, ρ^t=60, ρ^n=40

Figure 7: Ranking accuracies on Enterprise dataset.

Table 2: Average time (in milliseconds) of ranking optimization in setting of (top-5, not-top-10).

	MQ2008	MQ2007	OHSUMED	.Gov	Enterprise
time	4.24	6.85	134.53	70.06	6.45

set. The performances on the test sets are those based on the best performing parameters.

We tested all the six combinations of top-k constraint ($k = 1, 3, 5$) and not-top-k constraint ($k = 5, 10$) on all of the four datasets. Figures 3, 4, 5, and 6 report the experimental results on MQ2008, MQ2007, OHSUMED, and .Gov, respectively. Our ranking optimization method is denoted as 'RankOpt' in the figures. The best performing parameters of ρ^t and ρ^n are also shown. The performances of the combinations (top-3, not-top-5), (top-3, not-top-10), and (top-5, not-top-10) are reported.

From the results, we can see that our method outperforms the baselines as well as the basic ranking model of LambdaMART on all datasets. The experimental results for (top-3, not-top-5) and (top-3, not-top-10) are very similar. This is because all the methods focus on the top of ranking list. The changes of k on the not-top-k constraint have less impact on the final ranking list. Our method and the baselines perform equally well, when k is 1 for the top-k constraint. This is because the top-1 constraint means promoting the document to the top one position, and our method can function as a hard rule in such case and produce the same ranking result as the baselines (we omit the result because of space limitation). We will have discussions on the effect of different k values in Section 5.3.

We conducted significant tests (t-test) on the improvements of our method over all the baselines. The results indicate that all the improvements are statistically significant (p-value < 0.05), except (top-5, not-top-10) over 'Proportional' on MQ2008 in terms of NDCG@5, (top-3, not-top-5) and (top-3, not-top-10) over 'Radical' on OHSUMED and .Gov in terms of NDCG@3 and NDCG@5. Note that when top-1 constraint is adopted, our method performs equally well with the baselines.

Table 2 reports average running time of ranking optimization by our method (with top-5 and not-top-10 constraints) in milliseconds on a Laptop PC with 2.4GHZ CPU and 4GB memory. We can see that our method runs very fast, even on an unpowerful machine. We also observed that for most queries the algorithm converges within 10 iterations. Similar experimental results were also observed in other experiments. The results empirically verify the conclusion of Theorem 4.2.

5.2.2 *Results on Enterprise*

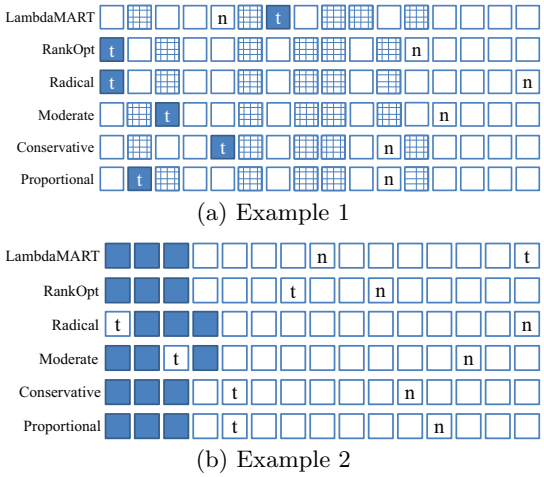
In the experiment, we utilized the training data to learn the basic ranking model and to tune the parameters ρ^t and ρ^n. The test data was utilized to perform ranking optimization. We tested all the six combinations of top-k constraint ($k = 1, 3, 5$) and not-top-k constraint ($k = 5, 10$) on the Enterprise dataset. The performances of the combinations (top-3, not-top-5), (top-3, not-top-10), and (top-5, not-top-10) are reported in Figure 8. Our ranking optimization method is denoted as 'RankOpt' in the figures. From the results, we can see that our method outperforms the baselines as well as the basic ranking model of LambdaMART on all datasets. The experimental results for (top-3, not-top-5) and (top-3, not-top-10) are very similar, as in the experiments on LETOR datasets. We conducted t-test on the improvements of our method over the baselines in terms of NDCG@1, NDCG@3 and NDCG@5. The improvements are statistically significant. Again, our method and the baselines perform equally well when the top-1 constraint is adopted. We omit the result because of space limitation.

We also tested the running time of our method of ranking optimization, in the setting of top-5 and not-top-10 constraints. The last column of Table 2 reports average time of our method in milliseconds. We can see that our method runs very fast. Similar experimental results were obtained for other combinations of top-k and not-top-k constraints.

5.3 Discussions

We first investigated the reasons that our method of ranking optimization outperforms the baseline methods. We found that in general our method can really work better than the baselines to make a good comprise between using the original ranking and using the constraints.

Here, we use the result of MQ2008 dataset with regard to two queries to illustrate why our method is superior to the baselines. Figure 8(a) shows the original ranking by LambdaMART and the final rankings by different methods with regard to one query. The empty blocks, grid blocks, and filled blocks represent not relevant documents, partially relevant documents, and relevant documents, respectively. The documents selected by top-k constraints and not-top-k constraints are marked with 't' and 'n', respectively. From the results, we can see that our method of ranking optimiza-



(a) Example 1

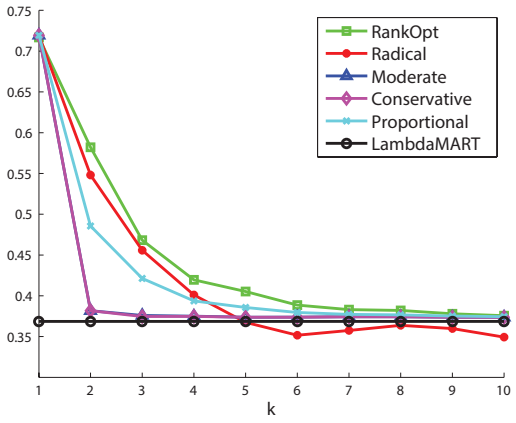
(b) Example 2

Figure 8: Example rankings from MQ2008.

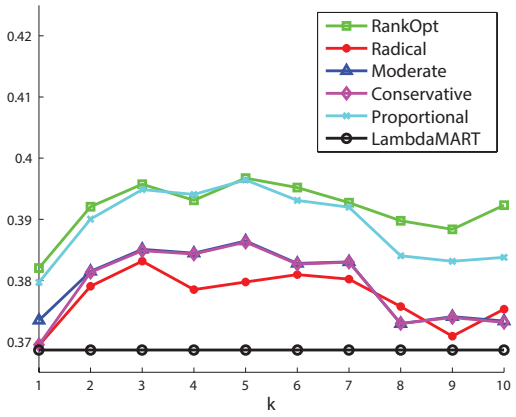
tion can really promote the relevant document (marked with 't') and demote the not relevant documents (marked with 'n'). In this case, our method outperforms the baselines of Moderate, Conservative, and Proportional. The results with regard to the other query are reported in Figure 8(b), which is a noisy case in which the document selected by the top-k constraint is actually not relevant (noises may also exist in the rules in practice). Our method of ranking optimization also takes into account the position of the document in the original ranking list given by LambdaMART and thus does not promote it too much. In other words, our method can make a good trade-off between the constraints and the original ranking. On the other hand, the baselines of Radical and Moderate do not have such consideration and cannot rank the document satisfactorily. Therefore, our method is more capable for post ranking than the baselines.

We further conducted experiments to see the impact of different k values on different methods. For the top-k constraint, larger k means a softer control on the final ranking list. When $k = 1$, the constraint becomes a hard rule. From the results reported in Figure 9(a), we can see that our method of ranking optimization always works better than or as well as the baselines for all k values. When k gets close to one, our method will perform similarly as the baselines; when k gets larger, the improvements of our method over the baselines will also be larger. The results indicate that our method is more robust than the baselines. We also note that the baseline of Radical hurts the basic ranking results when k gets large. This is because Radical is very sensitive to the noise in the constraints (rules). The results indicate that it is risky to directly apply Radical to post ranking though it outperforms the other baselines in most of the experiments.

On the other hand, for the not-top-k constraint, smaller k means a softer control on the final ranking list. When $k = N-1$, the constraint becomes a hard rule. From the results reported in Figure 9(b), we can see that our method of ranking optimization always works better than or as well as the baselines for all k values. When k gets close to one, our method will perform similarly as the baselines, because the not-top-k constraint has a soft control on the final ranking list. When k gets larger, the improvements of our method over the baselines will also become larger. There is a peak



(a) Top-k constraint



(b) Not-top-k constraint

Figure 9: Performances of ranking optimization with respect to different k values in terms of NDCG@1.

for performance of our method around $k = 5$. The performance will drop after $k = 5$. This is because the not-top-k constraint impacts more on the tail of the ranking list, and a larger k will have less impact on the top of the ranking list, which is more important in ranking evaluation.

We also investigated the influence of different types of constraints. Specifically, we tested performances of our method of rank optimization without the constraints (LambdaMART only), with top-5 constraint only, with not-top-10 constraint only, and with both types of constraints. Figure 10 reports the results. From the figure we can see that the two types of constraints can individually improve the ranking performances if they are adopted. The performances can be further improved if they are used simultaneously. The results indicate that our method of ranking optimization can leverage multiple types of constraints within one framework. The top-5 constraint outperforms the not-top-10 constraint, because the evaluation measure of NDCG emphasizes the importance of top ranking, which is also the focus of the top-k-constraint.

Finally, we evaluated how sensitive our method of ranking optimization is to the parameter settings. In the experiment, two parameters ρ_t and ρ_n were tested, which are weights of the top-k constraint and not-top-K constraint, respectively.

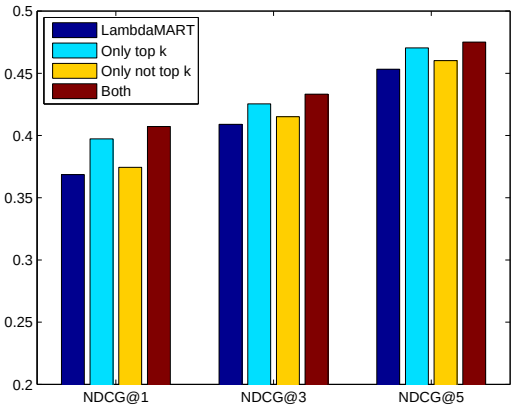
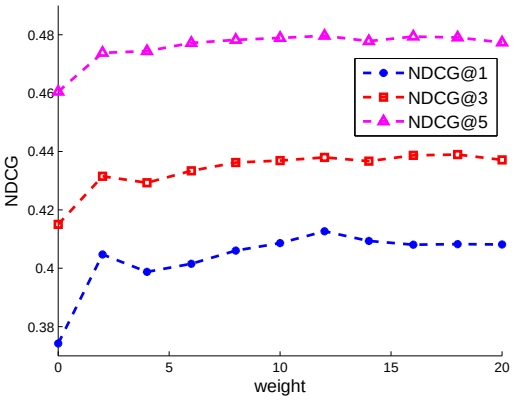
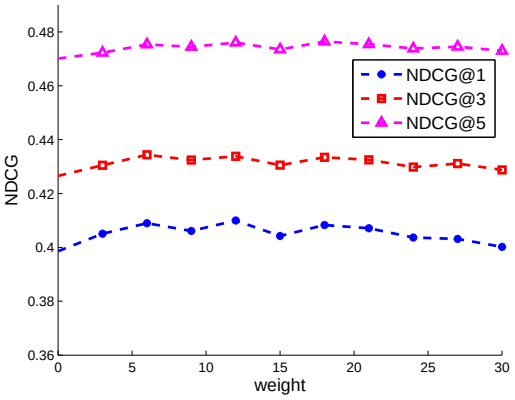


Figure 10: Performances of ranking optimization with different constraint types.



(a) ρ^t



(b) ρ^n

Figure 11: Performances of ranking optimization with respect to different parameter settings.

We changed one parameter and fixed the other to its optimal value. Figure 11(a) and Figure 11(b) show the performances of ranking optimization of our method in terms of NDCG at the positions of 1, 3, and 5. From the results, we can see that our method is not sensitive to the parameter settings, and thus is quite robust.

6. CONCLUSION

In this paper, we have studied the problem of post processing of ranking in search, which we call post ranking. Practices on post ranking in reality tend to be heuristic and we have, perhaps for the first time, formalized the problem as an optimization problem. In the formulation, we represent the post processing rules as constraints and manage to minimize the objective function denoting the trade-off between the original ranking and the constraints. As a result, one can perform post ranking through solving the optimization issue.

We have also given a specific probabilistic implementation of the optimization formulation. Bartley-Terry model is employed to calculating the probability of ranking list. The objective function is defined based on the conditional probability of the original ranking and the conditional probability of the constraints given the model. The optimization amounts to minimization of the sum of the negative log probabilities.

We have compared the performances of our method with several baselines in experiments using a number of datasets including benchmark datasets. The baseline methods represent practical methods of using rules. The results show that it is always better to employ our method in post ranking than the baselines.

There are still many open questions with regard to ranking optimization. We plan to conduct more research on the problem in the future. The open questions include (1) whether there exists a more general framework for ranking optimization, (2) how to define and incorporate other types of constraints into the framework, (3) how to naturally add diversification of results, etc. into the framework, (4) whether there are more effective and efficient methods for the task.

7. REFERENCES

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, USA, 2004.

[2] R. A. Bradley and M. E. Terry. The rank analysis of incomplete block designs — I. The method of paired comparisons. *Biometrika*, 39:324–345, 1952.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, 2005.

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, SIGIR '06, pages 186–193, 2006.

[5] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

[6] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, SIGIR '98, pages 335–336, 1998.

[7] Z. Dou, S. Hu, K. Chen, R. Song, and J.-R. Wen. Multi-dimensional search result diversification. In

Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pages 475–484, 2011.

[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, Dec. 2003.

[9] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA, 2000. MIT Press.

[10] R. Hunter. Mm algorithms for generalized bradley-terry models. *The Annals of Statistics*, 32:2004, 2004.

[11] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[12] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference*, SIGIR '01, pages 111–119, 2001.

[13] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 2007.

[14] T.-Y. 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.

[15] R. D. Luce. *Individual Choice Behavior: A theoretical analysis*. Wiley, 1959.

[16] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1-2):114–130, June 1957.

[17] J. I. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

[18] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference*, SIGIR '04, pages 64–71, 2004.

[19] Y. Nesterov and I. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer, 2004.

[20] R. L. Placket. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, SIGIR '98, pages 275–281, 1998.

[22] T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13(4):346–374, Aug. 2010.

[23] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, SIGIR '06, pages 691–692, 2006.

[24] S. Robertson and D. A. Hull. The trec-9 filtering track final report. pages 25–40, 2000.

[25] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on World Wide Web*, pages 675–684, 2004.

[26] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th Annual International ACM SIGIR Conference*, SIGIR '05, pages 449–456, 2005.

[27] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Yahia. Efficient computation of diverse query results. In *IEEE 24th International Conference on Data Engineering*, pages 228–236, 2008.

[28] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, June 2010.

[29] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *Proceedings of the 33rd International ACM SIGIR Conference*, SIGIR '10, pages 451–458, 2010.

[30] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, SIGIR '07, pages 391–398, 2007.

[31] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st Annual International ACM SIGIR Conference*, SIGIR '08, pages 107–114, 2008.

[32] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, SIGIR '07, pages 271–278, 2007.

Appendix

Proof of Theorem 4.1.

PROOF. The objective function f can be decomposed as $f = f_a - f_b$, where $f_a = \sum_{(i,j)\in\mathcal{P}} a_{(i,j)} \cdot \log(e^{s_i} + e^{s_j})$ and $f_b = \sum_{i=1}^{N} b_i \cdot s_i$. Here $a_{(i,j)} \geq 0$ for all (i,j) pairs in $\mathcal{P}$, which is the union of all preference pairs. $b_i \geq 0$ for $i = 1, \cdots, N$.

e^{s_i} is a log-convex function w.r.t s_i. Thus, $\log(e^{s_i} + e^{s_j})$ is also convex for any $(i,j) \in \mathcal{P}$, because log-convexity can be persevered under addition (cf., [1]). Hence, f_a is convex because $a_{(i,j)} \geq 0$ holds for all $(i,j) \in \mathcal{P}$.

Finally, we conclude that $f = f_a - f_b$ is convex because f_b is linear. □

Proof of Theorem 4.2.

PROOF. Since $s_i < \log(e^{s_i} + e^{s_j})$ holds for any preference pair (i,j), obviously the objective function f in Equation (5) has a lower bound of 0. Also, according to Theorem 4.1, f is convex. Thus, there exits one and only one optimal solution for the problem of Equation (5), denoted as $\mathcal{S}^*$.

Since gradient descent and backtracking are adopted for the optimization, the following inequality holds (cf., [19]):

$$f(\mathcal{S}^{(t)}) - f(\mathcal{S}^*) \leq \frac{\|\mathcal{S}^{(0)} - \mathcal{S}^*\|^2}{2\eta_{min} t},$$

where t is the number of iterations, $\eta_{min} = \min\{1, \alpha/L\}$, L is Lipschitz constant for f, and α is the shrinkage rate. Thus, Algorithm 1 can converge within $O(\frac{1}{\epsilon})$ to reach the tolerance ϵ. □