Learning Rank Functionals: An Empirical Study

Truyen Tran, Dinh Phung, Svetha Venkatesh
Department of Computing, Curtin University, Australia

May 2011

Abstract

Ranking is a key aspect of many applications, such as information retrieval, question answering, ad placement and recommender systems. Learning to rank has the goal of estimating a ranking model automatically from training data. In practical settings, the task often reduces to estimating a rank functional of an object with respect to a query. In this paper, we investigate key issues in designing an effective learning to rank algorithm. These include data representation, the choice of rank functionals, the design of the loss function so that it is correlated with the rank metrics used in evaluation. For the loss function, we study three techniques: approximating the rank metric by a smooth function, decomposition of the loss into a weighted sum of element-wise losses and into a weighted sum of pairwise losses. We then present derivations of piecewise losses using the theory of high-order Markov chains and Markov random fields. In experiments, we evaluate these design aspects on two tasks: answer ranking in a Social Question Answering site, and Web Information Retrieval.

1 Introduction

Ranking is central to many applications in information retrieval, question answering, online ad placement and recommender systems. Typically, given a query (e.g. a set of keywords, a question, or an user), we need to return a ranked list of relevant objects (e.g. a set of documents, potential answers, or shopping items). Learning to rank (LTR) (e.g. see [12] for a recent survey) is a machine learning approach to automatically estimate a rank model from training data, and offers promise way to leverage a wide prior knowledge. This includes relevance to content, context or profile and object qualities such as well-structuredness and authority.

Most LTR algorithms aim at estimating a *rank functional* $f(q, o)$ which takes a query q and an object o and returns a real score, which is then used to rank objects with respect to the query. Typically object order reflects its relevance of the object to the query, that is, top ranked objects are considered most relevant. Mathematically, this is the function estimation problem, but in the new setting of ranking - where the goal is to output a sorted list of objects rather than to compute the function itself. This is also different from the traditional settings of regression or classification, where continuous or discrete labels are predicted.

This paper provides an empirical study to probe the construction of rank functionals, and studies the contribution of the factors relevant to the problem. Consider, for example, two application domains: answer ranking in a Social Question Answering (SQA) site, and Web Information Retrieval (WIR). In SQA, community members respond to the question asked by another member, but the quality of answers varies greatly, making answer ranking critical. Often, we have access to the actual questions and answers, and in fact, questions can be quite rich and contain deep linguistic structures. On the other hand, in Web document retrieval, queries are often short, but there are many clues to assess the quality of the related documents, for example, the link structures, the authority of the domain and the organisation of the page. It is safe to assume that there are probably hundreds of relevance indicators currently employed in current major Web search engines.

These differences suggest that the input representation for the rank functionals can vary significantly from domain to domain. When the query and objects are of different modalities, textual query versus image objects for example, it is reasonable to represent the query and objects separately. Likewise, in the case of SQA separate representation can be applied to questions and answers since they are generally rich in content and structure. On the other hand, for Web document retrieval, prior knowledge of relevance indicators has been well-studied for decades, and it can be useful to represent the query-object pair as a combined vector.

The second aspect is the choice of the rank functionals, which captures the relevancy of an object given a query. The specification of rank functionals must be based on on the input representation, but it is also critical that the functional space is rich enough to capture data variability. For the separate query-object representations, we study transformation methods to first project the query vector and the object vector onto the same subspace and then combine them. For the combined representation, we investigate the usefulness of quadratic functions together with overfitting controls.

Third, learning often minimises some loss function based on training data. It is therefore important that the loss functions reflect the rank metrics that will be used to evaluate the algorithm at testing. In particular, this involves placing more weight for high relevance scores and the first few objects at the top of the rank list. The main issue is that rank metrics are often computed based on the all objects of the query, and they are discrete in nature, making it difficult to optimise directly. To this end, we study different choices for making a loss function closely related to the rank errors while maintaining smoothness. In particular, we examine methods to approximate rank metrics, and investigate weighting schemes to combine piecewise rank losses, where the purpose of the weights are to emphasize the high rating and discount for the low rank.

Next, we study probabilistic approaches to derive piecewise rank losses. The first approach is based on the theory of high-order Markov chains. In particular, we investigate the utility of a weighted version of the Plackett-Luce model [13][15] and its reverse. The second relies on Markov random fields (MRFs), where we suggest the weighted version of the pseudo-likelihood, as well as propose a piecewise approximation to the intractable MRFs. We prove that this piecewise approximation provides an upper-bound on the log-loss of the original MRF.

Finally, we evaluate these design aspects in the domain of Social Question Answering (SQA) with an autism dataset retrieved from the Yahoo! QA site, and the domain of Web Information Retrieval (WIR) with the dataset from the Yahoo! LTR challenge [4]. In SQA, we apply the separate representation of features - one for questions and one for answers - as input for rank functionals. We investigate two nonlinear rank functionals against three losses: the multiclass logistic, the loss based on approximating the MRR metric, and the pairwise logistic loss. The results shown that it is better to use the loss specifically designed for the - the multiclass logistic loss in this case.

Differing from the SQA, the WIR data has pre-computed features in the combined representation. The hypothesis is that we can use the quadratic rank functionals to discover predictive feature conjunctions. We find that when regularisation and overfitting controls are properly installed, second-order features are indeed more predictive than linear counterparts. Another difference from the SQA data is that, the WIR data contain relevance ratings in a small numerical scale, leading to frequent occurrence of ties. This suggests the use of group-level rank functionals, where objects of the same rating are grouped into a mega-object. We evaluate several aggregation methods for computing the group-level rank functionals. The results indicate that max and geometric mean aggregations can be competitive, while maintaining modest computational requirements due to the small number of resultant groups. Experiments on the WIR data also show consistent results -weighting is critical for many piecewise loss functions. In particular, for the element-wise decomposition of rank losses, rank discount weighting is the most influential, while for the pairwise decomposition, rating difference combined with query length normalisation is the most effective method.

This paper is organised as follows. Section 2 presents a holistic picture of the problem of LTR, and the specification details of rank functionals as well as loss functions. In particular, we discuss the piecewise weighting schemes to approximate the loss functions. Section 3 follows by describing probabilistic approaches to derive piecewise losses. Design issues identified in the paper are then evaluated in Section 4, where we present the experiments on the Yahoo! QA data, and the Yahoo! LTR challenge data. Section 5 provides further discussion on these aspects, followed by related work in Section 6. Finally Section 7 concludes the paper.

2 Estimating Rank Functionals

In a typical setting, given a query q and a set of related objects $\mathbf{o}^{(q)} = (o_1, o_2, .., o_{N_q})$ we want to output a rank list of objects. Ideally, this would means estimating an optimal permutation $\boldsymbol{\pi}^{(q)} = (\pi_1, \pi_2, .., \pi_{N_q})$ so that $\pi_i < \pi_j$ whenever o_i is preferred to o_j, i.e.$o_i \succ o_j$. However, this is often impractical since the number of all possible permutations is $N_q!$. A more sensible strategy would be estimating a real *rank functional* $f(q, o_i; \mathbf{o}_{\neg i}^{(q)})$ so that $f(q, o_i; \mathbf{o}_{\neg i}^{(q)}) > f(q, o_j; \mathbf{o}_{\neg j}^{(q)})$ whenever $o_i \succ o_j$, where $\mathbf{o}_{\neg i}^{(q)}$ denotes all objects for query q except for o_i. This is efficient since the cost of a typical sorting algorithm is $N_q \log N_q$.

For simplicity, in this paper we drop the explicit dependency between o_i and other objects, and write the rank functional as $f(q, o_i)$. In training data, we are given for each query a labelling scheme $\mathbf{r}^{(q)}$. This labelling can be a rank list $\boldsymbol{\pi}^{(q)}$, but more often, it is a set of relevance scores, i.e. $\mathbf{r}^{(q)} = (r_1, r_2, ..., r_{N_q})$ where r_i is typically a small integer. Given a training data of D queries, the general way of estimating $f(q, o_i)$ is to minimise an regularised empirical risk functional

$$\mathcal{R}(f) = \frac{1}{D} \sum_{q=1}^{D} \ell(\mathbf{r}^{(q)}, \{f(q, o_i)\}_{i=1}^{N_q}) + \lambda \Omega(f) \tag{1}$$

with respect to f, where $\ell(\mathbf{r}^{(q)}, \{f(q, o_i)\}_{i=1}^{N_q})$ is the loss function, $\Omega(f)$ is a (convex) regularisation function and $\lambda > 0$ is the regularisation factor.

We now discuss details of rank functional $f(.)$ and rank loss $\ell(.)$.

Specifying Rank Functionals f

Depending on specific applications, for each query-object pair (q, o_i) we can represent the query and the object separately (e.g. as feature vectors $\mathbf{y}_q \in \mathbb{R}^{n_1}$ and $\mathbf{z}_o \in \mathbb{R}^{n_2}$), or combined into a single vector $\mathbf{x}_i^{(q)} \in \mathbb{R}^m$. For example, in domains where queries and objects are of different modalities, such as textual queries for image objects, separate representations will be particularly

MRR	$\frac{1}{D_{test}} \sum_{q=1}^{D_{test}} \frac{1}{\hat{\pi}_*}$
NDCG@T	$\frac{N@T}{N_{max}@T}$ where $N@T = \sum_{i=1}^{T} \frac{2^{r_i}-1}{\log_2(1+\hat{\pi}_i)}$
ERR	$\sum_i \frac{1}{\hat{\pi}_i} R(r_i) \prod_{j\|\hat{\pi}_j<\hat{\pi}_i} (1 - R(r_j))$ where $R(r_i) = \frac{2^{r_i}-1}{16}$

Table 1: Evaluation metrics. Here $\hat{\pi}_i$ is the predicted position of object o_i and $\hat{\pi}_*$ is the predicted position of the best object.

useful. On the other hand, in situations where rich domain knowledge can be utilised to obtain multiple relevance and quality indicators, the the combined representation may be of advantage.

Separate Representation

Since the two vectors $\mathbf{y}_q$ and $\mathbf{z}_o$ are from different spaces, it can be useful to apply transformations into the same space. For simplicity, let us focus on linear transformation operators $\mathbf{A} \in \mathbb{R}^{d\times n_1}, \mathbf{B} \in \mathbb{R}^{d\times n_2}$ to convert $\{\mathbf{y}_q, \mathbf{z}_o\}$ into $\{\mathbf{A}\mathbf{y}_q, \mathbf{B}\mathbf{z}_o\} \in \mathbb{R}^d$, respectively. Given the transformations, we can compute the degree of association using

$$f_1(\mathbf{y}_q, \mathbf{z}_o) = \boldsymbol{\sigma}(\mathbf{A}\mathbf{y}_q)'\mathbf{B}\mathbf{z}_o \tag{2}$$

where $\boldsymbol{\sigma}$ is an element-wise mapping. Typically we choose the sigmoid or tanh functions to introduce nonlinearity.

Another option is to use the distance metric

$$f_2(\mathbf{y}_q, \mathbf{z}_o) = -\frac{1}{\tau} \left\|\mathbf{A}\mathbf{y}_q - \mathbf{B}\mathbf{z}_o\right\|^2 \tag{3}$$

for some scaling factor $\tau > 0$. Thus, learning the rank functional f reduces to estimating the transformation matrices $\mathbf{A}$ and $\mathbf{B}$.

Combined Representation

We will focus on quadratic functions for the case of combined representation. A quadratic rank model is a function of pairwise feature conjunction/interaction of the form

$$f_3(q, o) = \alpha_0 + \mathbf{w}'\mathbf{x} + \mathbf{x}'\mathbf{C}\mathbf{x} \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the feature vector, and $\mathbf{w} \in \mathbb{R}^m$, $\mathbf{C} \in \mathbb{R}^{m\times m}$ are parameters. The hypothesis is that certain feature conjunction/interactions are likely to emphasize some relevancy/quality aspects captured in the feature engineering process. To ensure that such second-order features will benefit rather than harm the performance, we need to remove unlikely combinations. One way is to impose sparsity-boosting regularisation function $\Omega(f)$ but it is likely to complicate the optimisation process. In this study, we follow a simpler approach by pre-filtering unlikely second-order features by some criteria. In particular, we apply Pearson's correlation[1] between second-order features and relevance scores as a rough measure of quality. Then we filter out second-order features whose absolute correlation coefficient is less than a certain threshold $\rho \in (0, 1)$.

[1]For two variables u, v, the Pearson's correlation is computed as $c(u,v) = \frac{\sum_i (u_i-\bar{u})(v_i-\bar{v})}{\sqrt{\sum_i (u_i-\bar{u})^2}\sqrt{\sum_i (v_i-\bar{v})^2}}$, where $\bar{u}, \bar{v}$ are the mean values of u and v, respectively.

	$f(\{o_j \vert r_j = \bar{r}\})$
Min	$\min\{f(o_j) \vert r_j = \bar{r}\}$
Max	$\max\{f(o_j) \vert r_j = \bar{r}\}$
Arithmetic Mean	$\frac{1}{K}\sum_j f(o_j)$
Geometric Mean	$\log\left(\frac{1}{K}\sum_j \exp f(o_j)\right)$

Table 2: Group-level rank functions.

Group-level Functions

In specifying f, one issue which is often overlooked is the the presence of ties. In practice, often the data is given in the form of relevance ratings which results in many ties. Thus, it may be useful to operate directly at the rating level, where ties are incorporated. The idea here is to treat a group of objects with the same rating as a mega-object. Like individual objects, a mega-object has a group-level rank function defined upon. This group-level function is an aggregation function, which combines individual rank functions in a sensible manner. For example, it is necessary to normalise the group-level function since the group size can vary greatly. Denote by $\bar{f}(\{o_j|r_j = \bar{r}\})$ the group-level rank function for the group whose relevance label is $\bar{r}$. Table 2 lists several candidate aggregation functions. Thus, assuming the relevance ratings are discrete with $|L|$ levels, there are at most $|L|$ groups. We note in passing that group-level function can be in conjunction with any individual functions.

Group-level rank functions can be useful for several reasons. First, they implicitly impose prior knowledge of ties. Second, for functions that perform averaging, we may achieve some kind of regularisation, making it more robust to noise. Third, often group-level functions can be computed in linear time and the overall complexity is also linear in number of objects per query. Therefore, we have computational savings, e.g. against pairwise losses, whose complexity is often quadratic in query size.

Specifying Rank Loss ℓ

The next important issue is the specification of the loss function ℓ. For clarity, let us make use of the notation $\ell(\mathbf{r}, \mathbf{o})$ to refer to the loss function and we drop the explicit mention of the query q when no confusion occurs. It is reasonable to assume that $\ell(\mathbf{r}, \mathbf{o})$ is highly correlated with the rank metrics used in evaluation, that is, a low loss should lead to high performance according to the rank metrics. Typically, rank metrics used in practice, such as MRR, NDCG [9] and ERR [5] (see Table 1), are functions of predicted positions of objects in the rank list. The predicted position is estimated as $\hat{\pi}_i = 1 + \sum_{j \neq i} \delta[f(o_j) > f(o_i)]$. This suggests that it may be useful for the loss function to incorporate pairwise comparisons among objects.

In this paper, we focus on two situations: when we need to emphasize on the singe best object, and when several objects are relatively useful.

Multiclass Logistic Loss

The first situation reduces to the multiclass categorisation problem, where a popular choice is the multiclass logistic loss

$$\ell_1(\mathbf{r}, \mathbf{o}) = -\log P(r^* = 1|\mathbf{o}) \tag{5}$$

	$\varphi(r_i, r_j, o_i, o_j)$ for $o_i \succ o_j$
Quadratic	$\{1 - (f(q, o_i) - f(q, o_j))\}^2$
Hinge [10]	$\max\{0, 1 - (f(q, o_i) - f(q, o_j))\}$
Exponential [8]	$\exp\{-(f(q, o_i) - f(q, o_j)\}$
Logistic [2]	$\log(1 + \exp\{-(f(q, o_i) - f(q, o_j)\})$

Table 3: Some pairwise losses.

where $P(r_i = 1|\mathbf{o}) = \exp\{f(q, o_i)\}/\sum_{j=1}^{N} \exp\{f(q, o_j)\}$, and $P(r^* = 1|\mathbf{o})$ denotes the probability of that best object is ranked first.

The second situation is more general, but also requires more sophisticated techniques to design the loss $\ell(\mathbf{r}, \mathbf{o})$. In this paper, we focus on three techniques: approximating the rank error, decomposition of loss into element-wise weighted sum and into pairwise weighted sum.

Smoothing Rank Errors

The first technique is to approximate the indicator $\delta[f(o_i) > f(o_j)]$ by a smooth function $\varrho(f(o_i), f(o_j)) \in [0, 1]$, and then optimise the resulting metrics directly

$$\ell_2(\mathbf{r}, \mathbf{o}) = 1 - M\left(\mathbf{r}, \{\varrho(f(o_i), f(o_j)\}_{ij|j>i}\right) \tag{6}$$

where $M(.)$ is rank metric. One popular choice of $\varrho(.)$ is the sigmoid function

$$\varrho(f(o_i), f(o_j)) = \frac{1}{1 + \exp\{-(f(o_i) - f(o_j))\}}. \tag{7}$$

The result is a smooth loss function with respect to f. The drawback is that the loss function is not flexible though complex. When the sigmoid function approaches the step function, it is hard to improve the performance any further.

Element-wise Decomposition

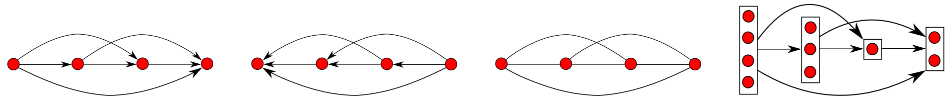
The second technique is to approximate the query-level loss by the following element-wise sum

$$\ell_3(\mathbf{r}, \mathbf{o}) = \sum_{i=1}^{N} W_i \omega(r_i, o_i; \mathbf{r}_{\neg i}, \mathbf{o}_{\neg i}) \tag{8}$$

where $\omega(r_i, o_i; \mathbf{r}_{\neg i}, \mathbf{o}_{\neg i})$ is a function of one variable r_i while fixing other labellings $\mathbf{r}_{\neg i}$, and $W_i \geq 0$ is a weighting parameter possibly depending on true position π_i of the object o_i. An example is perhaps in pointwise regression, where $\omega(r_i, o_i; \mathbf{r}_{\neg i}, \mathbf{o}_{\neg i}) = (r_i - f(o_i))^2$. Interestingly, it has been proven that with appropriate weighting scheme W_i, this approximation is indeed a good bound on the NDCG metric [7]. Another example is $\omega(r_i, o_i|\mathbf{r}_{\neg i}, \mathbf{o}_{\neg i}) = -\log P(r_i|\mathbf{r}_{\neg i}, \mathbf{o})$ as in the case of pseudo-likelihood approximation to the Markov random field, as we will present in the next section.

Pairwise Decomposition

The third technique is combining *piecewise loss*es, where each piece is a bivariate function



(a) (b) (c) (d)

Figure 1: Graphical illustration of query models. (a) Plackett-Luce, (b) Reverse Plackett-Luce, (c) Markov random field and (d) Group-level Plackett-Luce.

$$\ell_4(\mathbf{r}, \mathbf{o}) \quad = \quad \sum_{i=1}^{N} \sum_{j>i} V_{ij} \varphi(r_i, r_j, o_i, o_j) \tag{9}$$

where $V_{ij} \geq 0$ is a weighting factor for the pair $\{i, j\}$, possibly depending on the true positions $\{\pi_i, \pi_j\}$. This approximation is indeed well-studied in the LTR literature, where $\varphi(r_i, r_j, o_i, o_j)$ is often the pairwise loss (see Table 3 for popular losses). In fact, it has been shown that unweighted hinge, exponential and logistic losses are actually upper bounds of 1-NDCG [6].

This paper studies a number of important issues regarding the design of an effective LTR algorithm. First is the data representation of the pair query-object (q, o). Second is the choice of rank functionals $f(q, o)$ which capture the importance and relevancy of object o with respect to the query q. The effectiveness of the quadratic functional is studied in the experimental section. Third, the design of appropriate loss functions $\omega(.)$ and $\varphi(.)$ and specification the weights $\{W_i\}$ and $\{V_{ij}\}$ to combine those loss functions. It is likely that the weights should emphasize the high rating and discount for the low rank.

3 Deriving $\omega(.)$ and $\varphi(.)$ from Probabilistic Query Models

In this section, we present probabilistic approaches to derive the functions $\omega(.)$ of Eq.(8) and $\varphi(.)$ of Eq.(9). We start from specifying the query-level model distribution $P(\mathbf{r}|\mathbf{o})$. We consider two model representations: the *high-order Markov chain* and the *Markov random field*.

High-order Markov chains

For this model representation we work directly the permutation model of objects. Thus we will use the permutation notation $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_N)$ to denote the labelling scheme, where π_i is the position of the object in the list according to the permutation $\boldsymbol{\pi}$. Hence, according to $\boldsymbol{\pi}$, $\pi_i < \pi_j$ means that the object o_i is ranked higher than the object o_j, or equivalently $o_i \succ o_j$. In essence, the model aims at specifying the permutation distribution $P(\boldsymbol{\pi}|\mathbf{o})$.

For clarity, we drop the explicit reference to $\mathbf{o}$ and write $P(\boldsymbol{\pi})$ as well as $\ell(\boldsymbol{\pi})$ instead of $P(\boldsymbol{\pi}|\mathbf{o})$ and $\ell(\boldsymbol{\pi}, \mathbf{o})$, respectively. Let us start from the probabilistic theory that any joint distribution of N variables can be factorised according to the chain-rule as follows

$$P(\pi_1, \pi_2, ..., \pi_N) = P(\pi_1) \prod_{i=2}^{N} P(\pi_i | \boldsymbol{\pi}_{1:i-1}) \tag{10}$$

where $\boldsymbol{\pi}_{1:i-1}$ is a shorthand for $(\pi_1, \pi_2, .., \pi_{i-1})$. See Figure 1a for a graphical representation of this factorisation.

Informally in the context of LTR, this factorisation can be interpreted as follows: choose the first object in the list with probability of $P(\pi_1)$, and choose the second object with probability of $P(\pi_2|\boldsymbol{\pi}_1)$, and so on. The Luce's axioms of choice [13] assert that we should choose an object with probability proportional to its *worth*. Translated into our LTR problem, the worth of the object o_i can be defined[2] as $\phi(o_i) = \exp\{f(o_i)\}$. Any finally, according to Plackett [15], we can define the conditional probabilities as follows

$$\begin{aligned} P(\pi_1) &= \frac{\phi(o_1)}{\sum_{j=1}^{N}\phi(o_j)} \\ P(\pi_i|\boldsymbol{\pi}_{1:i-1}) &= \frac{\phi(o_i)}{\sum_{j=i}^{N}\phi(o_j)} \end{aligned}$$

This model is called the Plackett-Luce (PL) and has been studied in the context of LTR in [22] .

Weighted PL (WPL)

Typically, we learn the PL model by maximising the data likelihood, or equivalently, minimising the negative log-likelihood $-\log P(\boldsymbol{\pi})$. However, this does not necessarily lead to good performance since the likelihood and a typical rank metric are different. For example, rank metrics often put a great emphasis on the first few objects and effectively ignore the rest. This suggests a weighted log-loss

$$\ell_5(\boldsymbol{\pi}) = -W_1 \log P(\pi_1) - \sum_{i=2}^{N} W_i \log P(\pi_i|\boldsymbol{\pi}_{1:i-1}) \tag{11}$$

which is in the form of Eq.(8).

Reverse PL (RPL)

First, note that the factorisation in Eq 10 is general, and is not unique. In fact if we permute the indices of objects, the factorisation can still hold. Second, we are free to choose any realisation of the conditional distribution $P(\pi_i|\boldsymbol{\pi}_{1:i-1})$. For example, we can derive a *reverse* Plackett-Luce model as follows (see Figure 1b for a graphical representation)

$$Q(\pi_1, \pi_2, ..., \pi_N) = Q(\pi_N) \prod_{i=1}^{N-1} Q(\pi_i|\boldsymbol{\pi}_{i+1:N}) \tag{12}$$

The tricky part is to define $Q(\pi_N)$ and $Q(\pi_i|\boldsymbol{\pi}_{i+1:N})$. Since π_N is interpreted as the most irrelevant object in the list, $Q(\pi_N)$ cannot be thought as the probability of choosing object according to its worth. We think of $Q(\pi_N)$ as the probability of *eliminating* the object instead. Thus, it is reasonable to assume that *the probability of an object being eliminated is inversely proportional to its worth.* This suggests the following specifications

$$\begin{aligned} Q(\pi_N) &= \frac{\phi^{-1}(o_N)}{\sum_{j=1}^{N}\phi^{-1}(o_j)} \\ Q(\pi_i|\boldsymbol{\pi}_{i+1:N}) &= \frac{\phi^{-1}(o_i)}{\sum_{j=1}^{i}\phi^{-1}(o_j)}. \end{aligned}$$

[2]Worth can be any positive function as long as it is monotonically increasing in f

This factorisation in fact has an interesting interpretation. First, we choose the last object to eliminated, then choose the second one, and so on. Note that, due to specific choices of conditional distributions, $P \neq Q$ in general. Again, position-based weighting can be applied

$$\ell_6(\boldsymbol{\pi}) = -W_N \log Q(\pi_N) - \sum_{i=1}^{N-1} W_i \log Q(\pi_i|\boldsymbol{\pi}_{i+1:N}), \tag{13}$$

i.e. this is an instance of Eq.(8).

Markov random fields

In this model representation, we consider the case where discrete ratings $\mathbf{r} = (r_1, r_2, ..., r_N)$ are given to corresponding objects $\mathbf{o} = (o_1, o_2, .., o_N)$, where each r_i is drawn from the same label set L. We treat the ratings as random variables and aim at modelling a joint distribution $P(\mathbf{r}|\mathbf{o})$. Again, for clarity, we will drop the explicit reference to $\mathbf{o}$ and write $P(\mathbf{r})$ as well as $\ell(\mathbf{r})$ instead of $P(\mathbf{r}|\mathbf{o})$ and $\ell(\mathbf{r}, \mathbf{o})$, respectively.

In particular, the model has the following form

$$\begin{aligned} P(\mathbf{r}) &= \frac{1}{Z} \prod_i \prod_{j>i} \psi(r_i, r_j) \text{ where} && (14) \\ \psi(r_i, r_j) &= \exp\{\gamma \sum_i \sum_{j>i} \text{sign}[r_i - r_j](f(o_i) - f(o_j))\} \\ Z &= \sum_{r_1, r_2 ..., r_N} \prod_i \prod_{j>i} \psi(r_i, r_j) && (15) \end{aligned}$$

$\gamma > 0$ is the scaling factor accounting for the variation in number of objects per query (e.g. $\gamma = \frac{2}{N(N-1)}$). The idea is that the model encourages the correlation between the relative rating order (encoded in $\text{sign}[r_i - r_j]$) and the relative ranking order (by $f(o_i) - f(o_j)$). In fact this idea is very similar to that of pairwise models, except we now consider all pairs simultaneously. The result is a fully connected Markov random field[3] (MRF, see Figure 1c for a graphical representation).

However, it is well-known that learning in MRFs is difficult since the likelihood can not be computed exactly in general cases. In practice, we often resort to approximate methods, such as MCMC sampling. Here we propose several approximation alternatives, where the smoothness of the approximated loss is retained, making optimisation easier.

Weighted Pseudo-Likelihood (WpLL)

The first approximation is to work on the local conditional distribution $P(r_i|\mathbf{r}_{\neg i})$ instead of $P(\mathbf{r})$ directly, where

$$P(r_i|\mathbf{r}_{\neg i}) = \frac{\prod_{j \neq i} \psi(r_i, r_j)}{\sum_{r_i'} \prod_{j \neq i} \psi(r_i', r_j)}.$$

We propose the following weighted pseudo-likelihood

[3] To be more precise, the model should be called conditional MRF, or conditional random fields since state variables are conditioned on the objects.

$$\ell_7(\mathbf{r}) \quad = \quad -\sum_i W_i \log P(r_i|\mathbf{r}_{\neg i}) \tag{16}$$

which has the form of Eq.(8).

Weighted Pairwise Upper-Bound (WUB)

We start by observing that *the log-loss* $\ell(\mathbf{r}) = -\log P(\mathbf{r})$ *can be upper-bounded by a sum of pairwise losses*

$$\ell(\mathbf{r}) \leq -\sum_i \sum_{j>i} \log Q(r_i, r_j) + C \tag{17}$$

for some constant C *and* $Q(r_i, r_j) \propto \psi(r_i, r_j)$.

The proof is presented in the appendix, but interested readers may start from the general Hölder's inequality and noting that the normalising constant in Eq.(14) $Z = \sum_{\mathbf{r}} \prod_i \prod_{j>i} \psi(r_i, r_j)$ has the sum-product form. Again, to account for the rank metrics, we propose to use the following weighted loss

$$\ell_8(\mathbf{r}) = -\sum_i \sum_{j>i} W_{ij} \log Q(r_i, r_j) \tag{18}$$

which has the form of Eq.(9).

4 Experimental Studies

In this section, we evaluate the design choices on two applications: *Social Question Answering* and *Web Information Retrieval*. The first case involves ranking answers offered to a question in a Social QA site whilst the second case aims at ranking documents related to a query.

Before proceeding into the experimental details, let us make a note on the implementation. Until now, we have not presented the details of the learning process where parameters are estimated by minimising the empirical risk $\mathcal{R}(f)$ in Eq.1. All of the models studied in this paper is continuous in the rank functional f, which is continuous in model parameters. Thus the empirical risk is also continuous in model parameters, where gradient-based optimisation routines can be applied for find the minimisers of $\mathcal{R}(f)$. Since the gradient details of mentioned models are not the main subject of this paper, we omit for clarity. For all models, however, are regularised by a simple Gaussian prior in parameters, and are trained using the L-BFGS, a limited memory Newton-like algorithm.

Answer Ranking in Social Question Answering

QA offers a different perspective from traditional IR as the question is often semantically rich, and the style in posing a question and an answer is different. From (Yahoo! QA[4]) we collect all questions relevant to the topic of *autism*. In the training data, an answer is chosen as the best one for the question. We use $36,113$ questions ($215,230$ answers) for training, $1,000$ questions ($6,342$ answers) for development, and 999 questions ($7,636$ answers) for testing.

[4]http://answers.yahoo.com

Rank func.	Multi.logit.(ℓ_1)	MRR Opt.(ℓ_2)	Pair.logit.(ℓ_3)
f_1(Eq.2)	0.5474	0.4944	0.5360
f_2(Eq.3)	0.5381	0.4662	0.4697

Table 4: MRR scores of rank functionals for answer ranking, where $d = 10$, $n_1 = n_2 = 3,000$. Multiclass logistic loss is defined in Eq.(5), pairwise logistic loss is listed in Table 3, and the MRR metric is in Table 1.

We use words as features and employ the separate representation of input and the rank functionals developed in Eqs.(2) and (3). We construct the vocabularies for questions and answers separately. The vocabularies are constructed by selecting the top $3,000$ frequent non-stop words in the training questions, and training answers, respectively.

Table 4 reports the MRR scores of the two rank functionals when used in conjunction with three loss functions: the multiclass logistic, the pairwise logistic and the direct optimisation of the MRR metric. Note that since this data has only one correct answer per query, it is essentially an instance of the multiclass categorisation problem. In all combinations, the embedding space has the dimensionality of $d = 10$. For this problem, it appears that the multiclass logistic method performs best, and optimising the metric directly does not necessarily lead to improvement.

Web Information Retrieval

In this application, we employ data from the Yahoo! learning to rank challenge [4]. The data contains the groundtruth relevance scores (from 0 for irrelevant to 4 for perfectly relevant) of $473,134$ documents returned from $19,944$ queries. We use a subset of $1,568$ queries ($47,314$ documents) for training, and another subset of $1,520$ queries ($47,313$ documents) for testing. There are 519 pre-computed unique features for each query-document pair. Thus, this falls into the category of combined feature representation. We first normalise the features across the whole training set to have mean 0 and standard deviation 1 - this appears to consistently improve the performance over raw features. When there is not enough space, we report the ERR score only since it was required for the challenge.

Evaluation of Rank Functionals

Quadratic rank functionals. We evaluate the quadratic rank functionals f_3 (Eq.4) with weighted pairwise logistic loss (see Table 3, ℓ_4, Eq.9), where where the weight is $\left|2^{r_i-1} - 2^{r_j-1}\right| / N_q$ for the pair $\{o_i, o_j\}$. Table 5 reports the performance of this setting with respect to the feature selection threshold ρ. It can be seen that quadratic rank models can improve over linear models given appropriate overfitting control (since the number of parameters is much larger than the linear case). However, the improvement is not free - the time and space complexity generally scale linearly with the number of parameters. Evaluation with other losses shows the same pattern.

Group-level rank functionals. Table 6 reports the results of using group-level rank functionals in comparison with standard linear ones, where the loss is the standard Plackett-Luce. It can be seen that the group-level rank functionals (except for the min-aggregation type) are competitive in the ERR scores.

ρ	No. feats	N@1	N@5	N@10	ERR
	519	*0.6941*	*0.6638*	*0.6848*	*0.4979*
0.00	134,477	0.6792	0.6480	0.6742	0.4889
0.05	53,394	0.6979	0.6630	0.6862	0.4989
0.10	26,318	0.7020	0.6651	0.6887	0.5019
0.15	14,425	0.7090	0.6666	0.6897	**0.5048**
0.20	7,114	**0.7112**	**0.6716**	**0.6931**	0.5038
0.25	2,983	0.6942	0.6636	0.6877	0.4991
0.30	1,294	0.6926	0.6519	0.6730	0.4960

Table 5: Performance of weighted pairwise logistic loss (ℓ_4) with second-order features w.r.t. the selection threshold ρ. For comparison, the first row includes the result with first-order features.

Aggregation type	N@1	N@5	N@10	ERR
	0.6796	*0.6518*	*0.6777*	*0.4845*
Min	0.6027	0.6098	0.6430	0.4561
Max	0.6875	0.6555	0.6796	0.4961
Arithmetic Mean	0.6773	0.6465	0.6727	0.4892
Geometric Mean	0.6855	0.6539	0.6784	0.4918

Table 6: Results of group-level rank functionals (see Table 2 for definitions), applied for unweighted Plackett-Luce loss (ℓ_5 with all unity weights). For comparison, the top row includes the result of the standard object-level rank functional.

Evaluation of Rank Losses

We now present the results for various rank losses, all trained on linear rank functionals for simplicity.

Metrics optimisation. Table 7 reports the results of approximating rank metrics of NDCG and ERR using soft step functions in Eq.(7). The MRR is not included since it assumes that there is only one best object per query. It can be seen that the performance of rank metrics optimisation is comparable to standard pairwise models (see Table 3).

Weighting schemes. The importance of appropriate weighting scheme for position-wise decomposition of loss function is reported in Table 8. For loss functions based on the WPL and the WpLL, weighting is critical to achieve high quality. For the RPL, however, the role of weight-

	N@1	N@5	N@10	ERR
Pairwise Quadratic	0.6608	0.6462	0.6712	0.4789
Pairwise Logistic	0.6617	0.6473	0.6717	0.4801
Pairwise Hinge	0.6573	0.6462	0.6718	0.4787
NDCG Optimisation	0.6676	0.6439	0.6712	0.4871
ERR Optimisation	0.6683	0.6355	0.6606	0.4871

Table 7: Approximate metrics optimisation using ℓ_2 (see Table 1 for metric definitions). N@K is a short hand for NDCG score for the top K results.

Weight $W_i^{(q)}$	WPL	RPL	WpLL
1	*0.4845*	*0.4980*	*0.4882*
r_i	0.4955	0.4944	0.4980
$\sqrt{r_i}$	0.4915	0.4957	0.4979
$\frac{2^{r_i-1}}{2^{\lvert L\rvert-1}}$	0.4993	0.4876	0.4999
$\frac{1}{\pi_i}$	**0.5017**	0.4989	**0.5013**
$\frac{1}{\log(1+\pi_i)}$	0.4977	0.4998	0.4979

Table 8: ERR scores w.r.t. weighting schemes in ℓ_3 (Eq.8), and π_i is the position of the object o_i in the ranked list.

Weight $V_{ij}^{(q)}$	Quad.	Logist.	Hinge	WUB
1	*0.4789*	*0.4801*	*0.4787*	*0.4829*
$\frac{1}{N_q}$	0.4881	0.4885	0.4864	0.4912
$\lvert r_i - r_j\rvert$	0.4789	0.4834	0.4772	0.4854
$\frac{\lvert r_i - r_j\rvert}{N_q}$	0.4883	0.4909	0.4885	0.4918
$\frac{(R_i-R_j)(\eta_i-\eta_j)}{\text{NDCG}_{Max}}$	0.4928	0.4962	0.4935	0.4976
$(R_i - R_j)(\eta_i - \eta_j)$	0.4916	0.4964	0.4917	0.4950
$R_i - R_j$	0.4903	0.4936	0.4903	0.4939
$\frac{R_i-R_j}{N_q}$	**0.4952**	**0.4979**	**0.4953**	**0.4977**

Table 9: ERR scores w.r.t. weighting schemes in (ℓ_4, Eq.9), where $R_i = \frac{2^{r_i}-1}{2^{\lvert L\rvert-1}}$, $\eta_i = \frac{1}{\log(1+\pi_i)}$, and π_i is the position of the object o_i in the ranked list. Note that in the pairwise approximation to MRF, the weight should not include the ratings since they have been accounted for in the log-likelihood.

ing generally does not help to improve performance. This is interesting since it is probable that the reverse Plackett-Luce (RPL) puts more emphasis of removing the bad objects (see Eq.(12)), which in effect, increases the chance of the good objects at the top.

Table 9 reports the performance of models with weighted pairwise decomposition. Again, it clearly demonstrates that with careful weighting schemes we can greatly improve the quality of models.

5 Discussion

In this paper, we aim at taking a holistic view when designing a robust learning to rank (LTR) algorithm. We consider domains of Social Question Answering (SQA) and Web Information Retrieval (WIR). This section presents some more elaboration on a variety of design aspects.

In SQA, we have chosen the separate representation of features, and investigated two nonlinear rank functionals f_1 and f_2 against three losses: the multiclass logistic ℓ_1, the loss ℓ_2 based on approximating the MRR metric, and the pairwise logistic loss ℓ_3. The results indicate that it is better to use the loss specifically designed to the task. More specifically, when the task is to predict the single best object, the multiclass logistic loss should work better than more generic rank losses. We note in passing that we do not aim at outperforming state-of-the-art results in the

QA literature, which has a history of several decades, but rather show that simple representation such as words can be useful for this complicated task.

In the WIR task, on the other hand, combined features are pre-computed, and are not revealed to the public. However, it is plausible that they contain relevance assessment accordingly multiple criteria. Although the feature details are hidden, the quadratic rank functionals can be useful to detect which feature conjunctions are predictive. Our evaluation confirms that second-order features are indeed useful, provided that we have effective method to control overfitting. In particular, we employ regularisation and feature filtering for the task. This result, together with those reported for the Yahoo! LTR challenge [4], suggests that the space of rank functionals is a critical factor in achieving high performance LTR algorithms.

Different from SQA, the WIR data contain relevance ratings in a small numerical scale, which suggests the presence of ties. This calls for group-level rank functionals, where objects of the same rating are grouped into a mega-object. We have evaluated several aggregation methods for computing the group-level rank functionals. The results indicate that max and geometric mean aggregations can be competitive in term of the ERR score. The main benefit is that learning can operate directly on the level of groups, and since the number of groups is often much smaller than the number of objects per query, computational saving can be attained.

Like in the SQA task, we also evaluated the direct optimisation of approximation of rank metrics, which has been suggested by several recent studies [16] [19] [21]. However, our empirical results show that this is not necessarily the case. We conjecture that due to the approximation of the step function, the resulting functions are often highly complex, and possibly non-convex, making it difficult for optimisation. This observation is shared in [21].

The strong message we obtained from the experiments with the WIR data is that for many piecewise loss functions ($\ell_3$in Eq.8 and ℓ_4 in Eq.9), weighting is an important factor. This is expected since piecewise loss functions often operate locally with one or two free variables, while rank metrics are often a function of the whole query. In particular, for the element-wise decomposition of ℓ_3, rank discount weighting is the most influential (Table 8). For the pairwise decomposition of ℓ_4, rating difference combined with query length normalisation is the most effective method (Table 9).

6 Related Work

Several rank functionals other than the simple linear ones have been suggested in the literature, notably the neural net [2], kernels [10][14], regression trees [11] and bilinear [1]. The neural net rank functionals are often non-convex, and their discriminative power has not been clearly documented. The kernels, on the other hand, can be expensive for large-scale data, since most kernel-based algorithms scale super-linearly in number of training documents. Regression trees are interesting due to their flexibility in function approximation. Finally, bilinear functions are the linear combination of the query and object feature vectors - thus it falls into the group of separate feature representation. Our work contributes to this line of representation by introducing several non-linear functions.

Instance weighting has been used in LTR models in several places [2] [7] [6]. In [2], the piecewise approximation in Eq.(9) is used where the function $\varphi(.)$ is the log loss of the logistic model, and the weight $W_{ij} = \exp\{r_i\}/(\exp\{r_i\}+\exp\{r_j\})$. In [7], element-wise decomposition in Eq.(8) is suggested to obtain a bound on the NDCG score, but the details of W_i are not reported. In [6], NDCG-based weights are introduced for pairwise and Plackett-Luce models. We extend this by investigating a wider range of weighting schemes on several new models.

Query-level models have been advocated in several places [23] [12] [20]. In particular, in [23], the standard Plackett-Luce model is employed, but weighting is not considered. In [20], a Markov random field is suggested, and the learning involves MCMC sampling, which leads to non-smooth risk functionals and is hard to judge the convergence. Our weighted pseudo-likelihood and weighted piecewise approximation are, on the other hand, smooth.

There have been a number of recent studies on how to optimise rank metrics directly [17][24][21][3][6]. There are two approaches: one is based on approximating the rank metrics [16] [19] [21], and the other on bounding the rank errors [7][6]. We contribute further to the first approach by approximating the ERR and MRR metrics.

The issue of ranking with ties in the context of LTR has received little attention [25][18]. In [25], ties are considered among pairs of objects, but not the entire group of objects. On the other hand, [18] studies ties in groups, but their algorithm is only efficient for a particular case of group-level rank functionals (geometric mean - see Table 2).

7 Conclusion

In this paper, we have investigated choices when designing learning to rank algorithm to perform well against evaluation criteria. We evaluated design aspects on two tasks: answer ranking in a Social Question Answering site, and Web Information Retrieval. Among others, we have found that representing and selecting features, choosing a (cost-sensitive) loss function, handling ties, and weighting data instances are important to achieve high performance algorithms.

LTR is a fast growing field with many established techniques, and thus it is of practical importance to have a clear picture where nuts and bolts are identified. This paper is aimed as a step towards that goal. However, there are theoretical issues that needed to be addressed. First, this is mathematically a function estimation problem, where we still do not have a good understanding of the generalization properties. Second, it appears that the structure of the rank functional space is a key to the success of rank algorithms, and thus there is room for more investigation into data partitioning and nonparametric settings.

A Proof of The Upper-bound in WUB

Recall from Eq.(14) that $Z = \sum_{\mathbf{r}} \prod_i \prod_{j>i} \psi(r_i, r_j)$. Let us define an extended function $\Psi_{ij}(\mathbf{r}) = \psi(r_i, r_j)$ for all realisations of $\mathbf{r}_{\neg ij}$, thus

$$Z = \sum_{\mathbf{r}} \prod_{ij|j>i} \Psi_{ij}(\mathbf{r})$$

According to the general Hölder's inequality

$$Z \leq \prod_{ij|j>i} \left(\sum_{\mathbf{r}} \Psi_{ij}(\mathbf{r})^{q_{ij}} \right)^{\frac{1}{q_{ij}}} \tag{19}$$

for any $q_{ij} > 0$ subject to $\sum_{ij|j>i} 1/q_{ij} = 1$. Further, notice that $\Psi_{ij}(\mathbf{r})$ indeed depends only on (r_i, r_j), thus

$$\begin{aligned} \sum_{\mathbf{r}} \Psi_{ij}(\mathbf{r})^{q_{ij}} &= \sum_{\mathbf{r}_{\neg ij}} \sum_{r_i} \sum_{r_j} \Psi_{ij}(\mathbf{r})^{q_{ij}} \\ &= |L|^{N-2} \sum_{r_i} \sum_{r_j} \psi(r_i, r_j)^{q_{ij}} \end{aligned} \tag{20}$$

where $|L|$ is the size of the label set. The factor $|L|^{N-2}$ comes from the fact that there are $|L|^{N-2}$ ways of enumerating a set of $N-2$ discrete variables in $\mathbf{r}_{\neg ij}$, each of size $|L|$.

For any integer q_{ij}, we have

$$\sum_{r_i} \sum_{r_j} \psi(r_i, r_j)^{q_{ij}} \leq \left(\sum_{r_i} \sum_{r_j} \psi(r_i, r_j) \right)^{q_{ij}} \tag{21}$$

since $\psi > 0$ and the LHS is a part in the expansion of the RHS.

Substituting (21) into (20) and then into (19), we obtain

$$\begin{aligned} Z &\leq \prod_{ij|j>i} \left(|L|^{\frac{N-2}{q_{ij}}} Z_{ij} \right) \quad \text{or equivalently,} \\ \log Z &\leq \sum_{ij|j>i} \log Z_{ij} + C \end{aligned}$$

where $Z_{ij} = \sum_{r_i} \sum_{r_j} \psi(r_i, r_j)$, and

$$C = \sum_{ij|j>i} \log |L|^{\frac{N-2}{q_{ij}}} = (N-2) \log |L|$$

In the last equation, we have eliminated q_{ij} by using the fact that $\sum_{ij|j>i} 1/q_{ij} = 1$.

Recall that $P(\mathbf{r}) = \prod_{ij|j>i} \psi(r_i, r_j)/Z$, thus

$$\begin{aligned} \ell(\mathbf{r}) &= -\log P(\mathbf{r}) \\ &= \log Z - \sum_{ij|j>i} \log \psi(r_i, r_j) \\ &\leq \sum_{ij|j>i} \left(\log Z_{ij} - \log \psi(r_i, r_j) \right) + C \\ &= -\sum_{ij|j>i} \log Q(r_i, r_j) + C \end{aligned}$$

where we have used $Q(r_i, r_j) = \psi(r_i, r_j)/Z_{ij}$. This completes the proof.

References

[1] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information retrieval*, 13(3):291–314, 2010.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of ICML*, page 96, 2005.

[3] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 88–96. ACM, 2008.

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *JMLR Workshop and Conference Proceedings*, volume 14, pages 1–24, 2011.

[5] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*, pages 621–630. ACM, 2009.

[6] W. Chen, T.Y. Liu, Y. Lan, Z. Ma, and H. Li. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22:315–323.

[7] D. Cossock and T. Zhang. Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

[8] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4(6):933–969, 2004.

[9] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):446, 2002.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of SIGKDD*, pages 133–142. ACM New York, NY, USA, 2002.

[11] P. Li, C. Burges, Q. Wu, JC Platt, D. Koller, Y. Singer, and S. Roweis. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems*, 2007.

[12] T.Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[13] R.D. Luce. *Individual choice behavior*. Wiley New York, 1959.

[14] A. Moschitti and S. Quarteroni. Linguistic kernels for answer re-ranking in question answering systems. *Information Processing & Management*, 2010.

[15] R.L. Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[16] T. Qin, T.Y. Liu, and H. Li. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13(4):375–397, 2010.

[17] M. Taylor, J. Guiver, S. Robertson, and T. Minka. SoftRank: optimizing non-smooth rank metrics. In *Proceedings of the international conference on Web search and web data mining*, pages 77–86. ACM, 2008.

[18] T. Truyen, D.Q Phung, and S. Venkatesh. Probabilistic models over ordered partitions with applications in document ranking and collaborative filtering. In *Proc. of SIAM Conference on Data Mining (SDM)*, Mesa, Arizona, USA, 2011. SIAM.

[19] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to Rank by Optimizing NDCG Measure. In *NIPS*, 2009.

[20] M.N. Volkovs and R.S. Zemel. BoltzRank: learning to maximize expected ranking gain. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[21] M. Wu, Y. Chang, Z. Zheng, and H. Zha. Smoothing DCG for learning to rank: A novel approach using smoothed hinge functions. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1923–1926. ACM, 2009.

[22] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[23] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. of ICML*, pages 1192–1199, 2008.

[24] 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 on Research and development in information retrieval*, pages 107–114. ACM, 2008.

[25] K. Zhou, G.R. Xue, H. Zha, and Y. Yu. Learning to rank with ties. In *Proc. of SIGIR*, pages 275–282, 2008.