Learning to Rank from Noisy Data

Wenkui Ding, Department of Electronic Engineering, Tsinghua University
Xiubo Geng, Yahoo! Labs Beijing
Xu-Dong Zhang, Department of Electronic Engineering, Tsinghua University

Learning to Rank, which learns the ranking function from training data, has become an emerging research area in information retrieval and machine learning. Most existing work on learning to rank assumes that the training data is clean, which is, however, not always true. The ambiguity of query intent, the lack of domain knowledge, and the vague definition of relevance levels, all make it difficult for common annotators to give reliable relevance labels to some documents. As a result, the relevance labels in the training data of learning to rank usually contain noise. If we ignore this fact, the performance of learning-to-rank algorithms will be damaged.

In this paper we propose considering the labeling noise in the process of learning to rank and using a two-step approach to extend existing algorithms to handle noisy training data. In the first step, we estimate the degree of labeling noise for a training document. To this end, we assume that the majority of the relevance labels in the training data are reliable and use a graphical model to describe the generative process of a training query, the feature vectors of its associated documents, and the relevance labels of these documents. The parameters in the graphical model are learned by means of maximum likelihood estimation. Then the conditional probability of the relevance label given the feature vector of a document is computed. If the probability is large, we regard the degree of labeling noise for this document as small; otherwise, we regard the degree as large. In the second step, we extend existing learning-to-rank algorithms by incorporating the estimated degree of labeling noise into their loss functions. Specifically, we give larger weights to those training documents with smaller degrees of labeling noise and smaller weights to those with larger degrees of labeling noise. As examples, we demonstrate the extensions for McRank, RankSVM, RankBoost, and RankNet. Empirical results on benchmark datasets show that the proposed approach can effectively distinguish noisy documents from clean ones, and the extended learning-to-rank algorithms can achieve better performances than baselines.

CCS Concepts:
- Information systems → Learning to rank;

General Terms: Algorithms, Performance, Experimentation, Theory

Additional Key Words and Phrases: Learning to rank, noisy data, robust learning

ACM Reference Format:

1. INTRODUCTION

Ranking is the key task in many applications, such as information retrieval [Cao et al. 2006] and collaborative filtering [Harrington 2003]. In recent years, the approach that tries to improve ranking accuracy using machine learning technologies, has been intensively studied [Burges et al. 2005; Cao et al. 2007; Freund et al. 2003; Joachims 2002], and a new research area called learning to rank has gradually emerged.

In learning to rank, a training set is used to learn the ranking function, which contains sampled queries, their associated documents, and the relevance labels of these documents with respect to the queries. Most existing work on learning to rank has focused on developing new algorithms or adopting these algorithms to new applications. It is usually assumed in these works that the relevance labels in the training data are accurate and reliable. However, this is not always true. It is very likely that there is noise in the training data due to the difficulty of labeling in ranking, as depicted below.

Queries are usually short, and their intents might be quite ambiguous [Cronen-Townsend and Croft 2002]. For example, there are many websites relevant to the
A query “web search”, such as search service providers, industry overviews, research papers, and technical forums. It is difficult to decide which one is the most relevant to the query, and different people might have different opinions on the relevance of a given website. If we use the majority voting of multiple annotators to determine the relevance label of the website, then the label given by only one or two annotators (which is usually the real case due to the budget constraint) is highly possible to be unreliable. As a result, the training data may contain noise.

— Domain knowledge is often needed to make judgment for some queries. For example, for the query “solving RankSVM”, a document talking about “constrained quadratic programming” is highly relevant. However, annotators who are not familiar with RankSVM might think it irrelevant. These inexperienced annotators will introduce noise to the training data.

— The boundaries between different relevance labels are usually subtle and subjective. This is because the definition of a relevance label itself is not rigorous enough, and the difference between two labels cannot be quantitatively measured. Then it will become difficult for the annotators to clearly differentiate, for example, a “definitely relevant” document from a “partially relevant” document [Clarke et al. 2009]. If the annotator has to make a choice between these two, it is likely that his final decision will not be fully reliable. As a consequence, the training data will contain noise.

It has been found in [Xu et al. 2010] that many learning-to-rank algorithms are sensitive to the noise in the training data and their accuracy will be damaged if the noise is simply ignored. To the best of our knowledge, however, there is only limited work that considers the noise in the training data when performing learning to rank. Xu et al. [Xu et al. 2010] propose correcting noisy labels with click-through data. However, click-through data is not always available, which greatly limits the application scope of this approach. Carvalho et al. [Carvalho et al. 2008] propose using the sigmoid loss function instead of the hinge loss to reduce the influence of noisy documents in the training process. The problem of this work lies in that the new loss function does not really distinguish noisy documents from clean ones and thus will also reduce the impact of some correctly-labeled documents. Niu et al. [Niu et al. 2012] propose a top-k learning to rank framework where one motivation is to handle the difficulty in obtaining reliable relevance judgements. The labeling strategy, the ranking model as well as the evaluation measure all fall into the top-k ranking framework.

In this paper, we propose a novel approach to tackle the aforementioned problem. The idea is to identify the degree of labeling noise for each training document and incorporate such information into the training process. In particular, we propose a two-step approach to extend existing learning-to-rank algorithms to handle noisy training data.

In the first step, we estimate the degree of labeling noise of a training document. For this purpose, we assume that the majority of the relevance labels in the training data are reliable and use a graphical model to describe the generative process of a training query, the feature vectors of its associated documents, and the relevance labels of these documents. The parameters of this graph model are learned by means of maximum likelihood estimation. Then the conditional probability of the relevance label given the feature vector of a document is computed. If the probability is large, we regard the degree of labeling noise for this document as small; otherwise, we regard the degree as large. Note that the graphical model is query specific, i.e., each query has a different model. The reason of doing so is that the joint distribution of the label and the feature vector of a document may vary a lot across different queries. For example, the relevant documents to a popular query might have stronger features (e.g., higher
term frequencies, larger BM25 and PageRank scores) than the relevant documents to a rare query.

In the second step, we extend existing learning-to-rank algorithms by incorporating the estimated degree of labeling noise into their loss functions. Specifically, we give larger weights to those training documents with smaller degrees of labeling noise and smaller weights to those with larger degrees of labeling noise. In this way, the influence of the documents with noisy labels will be reduced in the learning process, and the extended algorithms can learn a more effective ranking function than the original ones when there is noise in the training data.

One of the advantages of the proposed two-step approach is that the estimation of the labeling noise is decoupled from the training of the ranking function. As a result, the approach is applicable to many different learning-to-rank algorithms. As example, we demonstrate the extensions for McRank, RankSVM, RankBoost, and RankNet in this paper. Empirical results on benchmark datasets show that the proposed approach can effectively distinguish noisy documents from clean ones, and the extended learning-to-rank algorithms can achieve better performance than baselines.

The remainder of this paper is organized as follows. In Section 2, we introduce some related work. In Section 3, the details of our approach are described. The experimental results are reported in Section 4. Finally, Section 5 concludes the paper with a discussion on future work.

2. RELATED WORK

2.1. Learning to Rank

Learning to rank is a task as follows. In the training phase, a ranking function is learned from a training set that consists of a set of queries, their associated documents, and the relevance labels of these documents. In the test phase, given a query and its associated documents, the learned ranking function assigns each document a score, and then sorts the documents in descending order of their scores.

Learning to rank has gained great attention from the research community in recent years and many learning-to-rank algorithms have been proposed. These algorithms can be roughly categorized into three approaches [Liu 2009]:

— The pointwise approach defines the loss function based on a single document. For example, Li et al. propose reducing the ranking problem to a multi-class classification problem [Li et al. 2007a], and using the multi-class classification loss on each document as the loss function for ranking.

— The pairwise approach defines the loss function based on a pair of documents. For example, Burges et al. reduce ranking to a pairwise classification, and apply the cross entropy loss to learn the ranking function [Burges et al. 2005]. The same strategy has also been used in [Joachims 2002] and [Freund et al. 2003], where different loss functions are used for pairwise classification (which are the hinge loss and the exponential loss respectively in these two works).

— The listwise approach defines the loss function based on all the documents associated with each query. Example algorithms include ListNet [Cao et al. 2007] and SVM-MAP [Yue et al. 2007]. For a more comprehensive review on these algorithms, please refer to [Liu 2009].

2.2. Current Progress on Learning to Rank

In many practical settings, the performance of a ranking model is evaluated by multiple measures of interest. Multi-objective learning to rank trains ranking models to simultaneously optimize multiple IR measures. Svore et al. [Järvelin and Kekäläinen 2000] use a standard web relevance measure, NDCG, as the primary optimization
metric and a relevance derived from click-data is used as the secondary metric; the performance of the primary measure is maintained constant while the algorithm tries to improve the secondary measure. A robust ranking model is proposed in [Wang et al. 2012] which considers not only the average ranking performance (effectiveness) but also robustness.

Learning to rank is also widely applied to different domains and specific properties are exploited to train the model better. Severyn et al. [Severyn and Moschitti 2012] apply learning to rank to Question Answering (QA) systems and design algorithms that exploit structural relationships between a question and their candidate answer passages to learn a re-ranking model. Ozertem et al. [Ozertem et al. 2012] propose a machine learning framework for ranking query suggestions. The model is trained on co-occurrences mined from the search logs, with novel utility and relevance models, and the machine learning step is done without any labeled data by human judges. Hong et al. [Hong et al. 2012] discuss how to build effective systems for ranking social updates from a unique perspective of LinkedIn and address the problem by using learning to rank.

Some other works deal with data by using semi-supervised learning, transductive learning or transfer learning. Semi-supervised learning algorithms builds a ranking function on the basis of a labeled training set and an unlabeled training set. A extended version of RankBoost algorithm is proposed in [Duh and Kirchhoff 2008]. In a first stage, the algorithm loops over the labeled set and assigns, for each labeled training example, the same relevance judgment to the most similar examples from the unlabeled training set. The algorithm then minimizes the exponential loss function over the labeled training set and the tentatively labeled part of the unlabeled data. Some studies have addressed the use of unlabeled data to learn a ranking function in a transductive setting [Joachims et al. 2003]. In this case, one is given sample points from a labeled training set and an unlabeled test set, and the goal is to build a prediction function which orders only unlabeled examples from the test set.

In this paper, we focus on the data aspect of learning to rank. However, we do not follow the way of semi-supervised learning or transfer learning. We try to model the noise of labeling and integrate it into our training process.

2.3. Learning from Noisy Data

As far as we know, there are only three works that consider noise in the training data when performing learning to rank. First, in [Xu et al. 2010], click-through log is used to identify the noisy training documents and correct their labels. Since click-through data is not available in many scenarios, the application scope of this approach is limited. Second, to reduce the influence of noisy training documents, the sigmoid loss is used to replace the hinge loss of RankSVM in [Carvalho et al. 2008]. However, this approach does not really distinguish noisy documents from clean ones and the impact of clean documents is also mistakenly reduced. Third, a noise-tolerant ranking model based on conditional random fields is proposed in [Geng et al. 2012]. However, the loss function in the model is highly related to the loss function in RankNet [Burges et al. 2005] and model cannot be used to improve other existing learning-to-rank algorithms.

In parallel, there has been some work on learning from noisy data for classification. Lawrence and Schölkopf [Lawrence and Schölkopf 2001] propose a probability model to handle noise in classification. They assume that the noise is generated from a noise process, the data is generated from a Gaussian distribution, and the parameters can be estimated using likelihood maximization. Li et al. [Li et al. 2007b] extend the Gaussian distribution to a mixture of Gaussians to handle complicated data. However, it is non-trivial to extend these approaches to learning to rank. This is because in these works, the component of noise detection and the component of classification function
learning are highly coupled with each other. One can hardly replace classification function learning with ranking function learning, and apply these approaches to learning to rank in a straightforward manner.

In [Brodley and Friedl 1999], the authors construct several candidate classifiers from the original training data, and then use the majority voting of these classifiers to identify and remove noisy instances. After removing these instances, a new classifier is learned from the new training data. This approach can be extended to learning to rank, since the noise removal and classifier learning is decoupled. The disadvantage of this approach, however, lies in that the generalization ability of the learned model will be hurt since the size of the training data is reduced after noise removal (in contrast, all the training data is used in the other works introduced above, such as [Carvalho et al. 2008], [Lawrence and Schölkopf 2001], [Li et al. 2007b], and [Xu et al. 2010]).

Given the limitations of the aforementioned work, it is necessary to develop a more effective and more widely applicable approach to handle the noisy training data in learning to rank. This is just the motivation of our work.

3. OUR APPROACH

As aforementioned, the performance of many learning-to-rank algorithms will get hurt when there is noise in the training data [Xu et al. 2010]. To tackle this problem, we propose a two-step approach to learn a robust ranking function from noisy data. In the first step, we estimate the degree of labeling noise of each training document. In the second step, we extend existing learning-to-rank algorithms to deal with noisy data by incorporating the estimated degree of labeling noise into their loss functions.

Before introducing the details of these two steps, we first give some notations in Table I. We assume that a larger relevance label corresponds to better relevance. That is, if $y_{qi} > y_{qj}$, then the $i$-th document is more relevant than the $j$-th document for query $q$.

### 3.1. Step I: Estimation of Labeling Noise

To effectively estimate the labeling noise, we need to make an assumption about the data. The assumption is that "the majority of the relevance labels in the training data are reliable". This assumption is intuitive and reasonable in the following sense. If the majority of the training data is noisy, then the useful information contained in the training data will be overwhelmed by noise. As a result, it is very difficult for any machine learning algorithm to obtain effective ranking functions from the data. In fact, the same assumption has also been made by previous work that learns classification and regression functions from noisy data [Angluin and Laird 1988].

With the above assumption, we first learn $P(x_{qi}^q, y_{qi})$, the joint probability of the feature vector and the relevance label of the $i$-th document for query $q$. Specifically, we use a graphical model to describe the generative process of $(x_{qi}^q, y_{qi})$. The graphical model is shown in Figure 1. The parameters of the model can be estimated using all the associated documents of query $q$, by means of maximum likelihood estimation. Then for
each individual document, we fit its feature vector and label to the joint distribution, and compute the conditional probability \( P(y^q_i|x^q_i) \). If the value of \( P(y^q_i|x^q_i) \) is large, we regard the relevance label \( y^q_i \) as reliable. Otherwise, if the value of \( P(y^q_i|x^q_i) \) is very small, it is likely that the relevance label \( y^q_i \) is noisy.

In the following subsections, we will provide more details about the graphical model and the parameter estimation.

3.1.1. The Graphical Model. To model the generative process of a query and its associated documents, we make the following assumptions.

— The relevance label of an individual document is independent of those of other documents. For example, for query “WSDM”, the relevance labels of webpages “www.wsdm-conference.org/2014/” and “sigir.org/sigir2014/” are independent of each other.

— The features of a document are generated in isolation. For example, we assume that the features “BM25” and “PageRank” of a document are independent of each other. Note that this assumption is made for simplification and ease of computation, just as in the Naive Bayesian classifier [Zhang 2004].

— Since we have no prior knowledge about the underlying distribution of each feature, we take a non-parametric approach to estimate it. In particular, we quantize the original feature variable \( x^q_{i,f} \) into a discrete variable \( \tilde{x}^q_{i,f} \in \{1, \ldots, b\} \) as follows,

\[
\tilde{x}^q_{i,f} = k, \quad \text{if} \quad x^q_{i,f} \geq f^q_{\text{min}} + \frac{k - 1}{b} (f^q_{\text{max}} - f^q_{\text{min}}) \quad (1)
\]

\[
\text{and} \quad x^q_{i,f} < f^q_{\text{min}} + \frac{k}{b} (f^q_{\text{max}} - f^q_{\text{min}}),
\]

where \( f^q_{\text{max}}, f^q_{\text{min}} \) is the maximum and minimum values of the \( f \)-th feature for query \( q \). We assume the quantized variable \( \tilde{x}^q_{i,f} \) to follow the Multinomial distribution.

With the above assumptions, our model describes the generative process of a document with respect to a query in the following fashion.

(1) Choose a relevance label \( y^q_i \sim \text{Multinomial}(\alpha^q) \).

(2) For each of the features,

— Choose the feature value \( \tilde{x}^q_{i,f} \sim \text{Multinomial}(\beta^q_{y^q_i, f,k}) \).

Here \( \alpha^q \) is a \( K \)-dimensional vector and its \( r \)-th element \( \alpha^q_r \) corresponds to the probability of generating a document with the \( r \)-th label for query \( q \). \( \beta^q_{r,f,k} \) is a \( b \)-dimensional vector and its \( k \)-th element \( \beta^q_{r,f,k} \) represents the probability of generating value \( k \) for feature \( f \) of a document with label \( r \) for query \( q \).

Then we have

\[
P(y^q_i) = \prod_r (\alpha^q)^{f(y^q_i=r)}, \quad (2)
\]

\[
P(\tilde{x}^q_{i,f}|y^q_i) = \prod_k (\beta^q_{y^q_i,f,k})^{I(\tilde{x}^q_{i,f}=k)}, \quad (3)
\]

and

\[
P(\tilde{x}^q_i, y^q_i) = P(y^q_i) \prod_f P(\tilde{x}^q_{i,f}|y^q_i). \quad (4)
\]

Remark. Note that the graphical model as introduced above is similar to the probabilistic ranking model for information retrieval proposed in [Baeza-Yates and Ribeiro-Neto 1999]. The model proposed in [Baeza-Yates and Ribeiro-Neto 1999] has been
proven to be effective in document retrieval. However, its application scope is limited because it needs relevance feedback for any query under investigation, which is not always available in real retrieval scenarios. In contrast, in our case, the graphical model is only applied to training queries, the documents of which are always labeled.

3.1.2. Parameter Estimation. We estimate the parameters \( (\alpha^q, \beta^q) \) by likelihood maximization, i.e.,

\[
(\hat{\alpha}^q, \hat{\beta}^q) = \arg \max_{\alpha^q, \beta^q} \prod_i P(\tilde{x}^q_{i,f}, y^q_i).
\] (5)

With some derivation, one can obtain

\[
\hat{\alpha}^q_r = \frac{\sum_{i=1}^{m_q} I(y^q_i = r)}{m_q},
\] (6)

\[
\hat{\beta}^q_{r,f,k} = \frac{\sum_{i=1}^{m_q} I(y^q_i = r)I(\tilde{x}^q_{i,f} = k)}{\sum_{i=1}^{m_q} I(y^q_i = r)}.
\] (7)

Note that the above model is query specific, since the parameters \( \alpha^q \) and \( \beta^q \) are query dependent, as indicated by the superscript. The motivation of using such a query-specific model is that the joint distribution of the label and the feature vector of a document may vary a lot across different queries. For example, the relevant documents to a popular query might have stronger features (e.g., higher term frequencies, larger BM25 and PageRank scores) than the relevant documents to a rare query. However, there is also a disadvantage with this approach. That is, we can only use documents associated with query \( q \) to estimate its parameters. If the query has only a small number of labeled documents, the estimation of the parameters may be inaccurate. To tackle this problem, we propose smoothing the estimation of each query specific model using a collection-wise background model. This idea has been used in many models in information retrieval, such as language models [Zhai and Lafferty 2004] and Okapi BM25 [Robertson and Walker 1994]. Here, we propose using the Jelinek-Mercer smoothing [Jelinek 1980] which involves a linear combination of the query-specific parameters and the collection-wise parameters. That is,
(1) we estimate the global parameters $\alpha$ and $\beta$ of the collection-wise background model using the documents of all the training queries:

$$\hat{\alpha}_r = \frac{\sum_q \sum_{i=1}^{m_q} I(y^q_i = r)}{\sum_q m_q},$$

(8)

$$\hat{\beta}_{r,f,k} = \frac{\sum_q \sum_{i=1}^{m_q} I(y^q_i = r) I(\tilde{x}^q_i,f = k)}{\sum_q \sum_{i=1}^{m_q} I(y^q_i = r)}.$$  

(9)

(2) we then smooth the query specific parameters with the global parameters:

$$\tilde{\alpha}^q_i = \lambda \hat{\alpha}^q_i + (1 - \lambda) \hat{\alpha}_r,$$

(10)

$$\tilde{\beta}^q_{r,f,k} = \lambda \hat{\beta}^q_{r,f,k} + (1 - \lambda) \hat{\beta}_{r,f,k},$$

(11)

where $0 \leq \lambda \leq 1$ is a parameter to trade off the two items. When $\lambda = 0$, we only use the collection-wise parameters, which means that documents of all queries follow the same distribution. When $\lambda = 1$, we only consider query-specific parameters, which means that we do not use smoothing at all.

### 3.1.3. Estimation of Labeling Noise

With the estimated parameters $(\alpha^q, \beta^q)$, we can compute the conditional probability $P(y^q_i|\tilde{x}^q_i)$ as follows,

$$P(y^q_i|\tilde{x}^q_i) = \frac{P(y^q_i) \prod_f P(\tilde{x}^q_{i,f}|y^q_i)}{\sum_y P(y) \prod_f P(\tilde{x}^q_{i,f}|y)},$$

(12)

where $P(y^q_i)$ and $P(\tilde{x}^q_{i,f}|y^q_i)$ are computed according to Eqn. (2) and Eqn. (3), and $y$ enumerates all possible relevance labels. To avoid the multiplications in Eqn. (12) and calculate $P(y^q_i|\tilde{x}^q_i)$ accurately, we transform it as follows:

$$P(y^q_i|\tilde{x}^q_i) = \left( \sum_y \exp \left( \ln P(y) - \ln P(y^q_i) + \sum_f \ln P(\tilde{x}^q_{i,f}|y) - \sum_f \ln P(\tilde{x}^q_{i,f}|y^q_i) \right) \right)^{-1}.$$

The value of $P(y^q_i|\tilde{x}^q_i)$ can be used to indicate the degree of labeling noise of the $i$-th document. If the value of $P(y^q_i|\tilde{x}^q_i)$ is small, it is quite possible that the label of the document is noisy, and we should reduce the influence of the document in the training process.

### 3.2. Step II: Extension of Learning-to-Rank Algorithms

Given the estimated probability $P(y^q_i|\tilde{x}^q_i)$, we can extend existing learning-to-rank algorithms to handle noisy training data. The basic idea is to modify the loss function of an algorithm according to $P(y^q_i|\tilde{x}^q_i)$. If $P(y^q_i|\tilde{x}^q_i)$ of a document is large, we will assign a large weight to the document in the loss function; otherwise, we will give it a small weight. This idea is simple yet general. It can be used to extend most pointwise and pairwise learning-to-rank algorithms. In this subsection, we will take McRank, RankSVM, RankBoost, and RankNet as examples for illustration. We leave the extension of listwise learning-to-rank algorithms to our future work.

#### 3.2.1. Extension of McRank

McRank is a pointwise learning-to-rank algorithm. It reduces the ranking problem to a multi-class classification problem, and learns several classification functions $\{f_k\}_{k=0}^{K-1}$ by minimizing the following loss function,

$$L = \sum_q \sum_i \sum_r - \log \frac{\exp(f_r(x^q_i))}{\sum_{k=0}^{K-1} \exp(f_k(x^q_i))} I(y^q_i = r),$$

(13)
where \( f_k(.) \) is the classification function for the \( k \)-th label, and \( \frac{\exp(f_r(x^q_i))}{\sum_{k=0}^{K-1} \exp(f_k(x^q_i))} \) is the probability of the \( i \)-th document of query \( q \) with the \( r \)-th relevance label.

McRank defines its loss based on individual documents, and thus the extension is quite straightforward. We can extend McRank by setting the corresponding conditional probability of a document as its weight in the loss function, i.e.,

\[
L = \sum_q \sum_i P(y^q_i | \tilde{x}^q_i) \sum_r -\log \frac{\exp(f_r(x^q_i))}{\sum_{k=0}^{K-1} \exp(f_k(x^q_i))} I(y^q_i = r).
\]

We denote the extended algorithm as McRank-W for ease of reference.

### 3.2.2. Extension of RankSVM

Let \( P^q = \{(i,j)|y^q_i > y^q_j\} \) be the set of document pairs associated with query \( q \). The loss functions of RankSVM can be written as follows,

\[
L = \frac{1}{2} \|\omega\|^2 + C \sum_q \sum_{(i,j) \in P^q} (1 - f_\omega(x^q_i) + f_\omega(x^q_j))_+
\]

where \( f_\omega(.) \) is the ranking function, \( \omega \) is its parameter and \( (1 - f_\omega(x^q_i) + f_\omega(x^q_j))_+ = \max\{1 - f_\omega(x^q_i) + f_\omega(x^q_j), 0\} \) is the hinge loss function used in RankSVM.

Since the hinge loss function is defined on document pairs, to extend RankSVM, we need to compute the probability of labeling noise for each document pair. Actually this is not difficult, as shown below.

\[
P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j) = \sum_r \sum_{i<r} P(y^q_i = r | \tilde{x}^q_i) P(y^q_j = t | \tilde{x}^q_j).
\]

With the above probability, we can extend the original loss functions as follows,

\[
L = \frac{1}{2} \|\omega\|^2 + C \sum_q \sum_{(i,j) \in P^q} P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j) (1 - f_\omega(x^q_i) + f_\omega(x^q_j))_+.
\]

As an optimization problem, it is usual to transform the original primal problem to its dual problem. By applying standard techniques in optimization, we obtain the following dual problem for a linear ranking function \( f_\omega(x) = w^T x \):

\[
\begin{aligned}
\max & \sum_q \sum_{(i,j) \in P^q} \alpha^q_{i,j} - \frac{1}{2} \sum_q \sum_{(i_1,j_1) \in P^q} \sum_{(i_2,j_2) \in P^q} \alpha^q_{i_1,j_1} \alpha^q_{i_2,j_2} (x^q_{i_1} - x^q_{j_1})(x^q_{i_2} - x^q_{j_2}) \\
\text{s.t.} & 0 \leq \alpha^q_{i,j} \leq CP(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j).
\end{aligned}
\]

This quadratic programming problem can be optimized by various existing methods such as sequential minimal optimization method [Platt et al. 1998] and interior point methods [Vanderbei 1999].

We denote the extended algorithm as RankSVM-W for ease of reference.

### 3.2.3. Extension of RankBoost

The loss functions of RankBoost can be written as follows,

\[
L = \sum_q \sum_{(i,j) \in P^q} \exp(-f_\omega(x^q_i) + f_\omega(x^q_j)),
\]

where an exponential loss function is used.
With the estimated probability of labeling noise for each document pair, we can extend the original loss functions as follows,

$$L = \sum_q \sum_{(i,j) \in P^q} P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j) \exp(-f_\omega(x^q_i) + f_\omega(x^q_j)).$$

This formulation can be integrated into existing implementations of RankBoost with very little cost. Remember that in RankBoost, a probability distribution over the training pairs is maintained in every step and it is initialized as a uniform distribution at the beginning of the algorithm. To minimize the weighted exponential loss function, we set the probability of each training pair $P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j)$ proportional to its estimated probability of labeling noise and update the probability distribution just as the algorithm proposed in [Freund et al. 2003]. By doing so, it can be proved that the weighted exponential loss function is minimized.

We denote the extended algorithm as RankBoost-W for ease of reference.

### 3.2.4. Extension of RankNet

The loss functions of RankNet can be written as follows,

$$L = \sum_q \sum_{(i,j) \in P^q} \log(1 + \exp(-f_\omega(x^q_i) + f_\omega(x^q_j))),$$

where a cross entropy function is used.

We can extend the original loss functions as follows,

$$L = \sum_q \sum_{(i,j) \in P^q} P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j) \log(1 + \exp(-f_\omega(x^q_i) + f_\omega(x^q_j))).$$

As a neural network based model, it is common to use gradient descent method to optimize it. When a linear ranking function is used, i.e. $f_\omega(x) = w^T(x)$, the derivative of the loss function with respect to the parameter $\omega$ can be calculated by

$$\frac{\partial L}{\partial \omega} = \sum_q \sum_{(i,j) \in P^q} P(y^q_i > y^q_j | \tilde{x}^q_i, \tilde{x}^q_j) \frac{1 + \exp(\omega(x^q_i - x^q_j))}{1 + \exp(\omega(x^q_i - x^q_j))}(x^q_j - x^q_i).$$

The parameter of the model is updated according to:

$$\omega_{t+1} = \omega_t - \delta_t \frac{\partial L}{\partial \omega},$$

where $\omega_t$ is the parameter at the $t$-th step and $\delta_t$ is the learning rate at the $t$-th step.

We denote the extended algorithm as RankNet-W for ease of reference.

### 4. EXPERIMENTS

In this section, we first introduce experimental settings, includes datasets, baselines, parameter tuning and evaluation metrics. We then compare the ranking accuracy of different algorithm on the used datasets. We also discuss the ranking performance under different noise levels and the effect of smoothing.

#### 4.1. Experimental Settings

##### 4.1.1. Datasets

We conducted experiments on four datasets: TD2003 and TD2004 in the LETOR benchmark collection [Liu et al. 2007], MSLR-WEB10K in Microsoft Learning to Rank Datasets and SET2 in the Yahoo! Learning to Rank Challenge Datasets.

TD2003 and TD2004 contain 50 and 75 queries respectively. For each query, there are about 1000 associated documents. Each document is labeled as relevant or irrelevant. There are 64 features extracted for each document, including term frequency,
BM25, LMIr, PageRank, etc. The datasets are partitioned for five-fold cross validation, and we will report the averaged results.

To examine whether the proposed approach can effectively deal with noisy training data, we add noise to the original training set in TD2003 and TD2004. Specifically, the label of each document in the training set is flipped with a probability of 10%, while the validation and test set remain unchanged. Therefore, one can view the modified training set as noisy, while the validation and test set as clean. Note that this experimental setting can well reflect the cases in real applications. For example, in commercial search engines, usually tens of thousands of training queries are used to learn the ranking function. Due to the budget constraint, each training query is labeled by only one or two annotators. In contrast, the validation set (which is used to determine whether a ranking function is effective enough to release) is relatively small and each query in it is usually labeled by multiple annotators. In this case, the training set is also noisier than the validation set.

To evaluate our approach on large scale datasets, we also used Microsoft Learning to Rank Datasets and Yahoo! Learning to Rank Challenge Datasets. Microsoft Learning to Rank Datasets [Qin et al. 2010] is a standard large scale dataset in learning to rank community. We conducted our experiments on MSLR-WEBl0K which includes 10,000 queries. Each query-document pair is represented as a 136-dimensional feature vector. The dataset is partitioned into five folders and we will report the averaged results over all the five folders. Yahoo! Learning to Rank Challenge Dataset [Chapelle and Chang 2011] is another standard large scale dataset for learning to rank. SET2 is a dataset in Yahoo! Learning to Rank Challenge and it is from an Asian country. The dataset includes a training set with 1266 queries, a validation set with 1266 queries and a test set with 3798 queries. There are 596 features extracted for each query-document pair.

4.1.2. Baselines. We implemented four learning-to-rank algorithms, McRank, RankSVM, RankBoost and RankNet as baselines. We then extended them to McRank-W, RankSVM-W, RankBoost-W, and RankNet-W, as described in previous sections.

In addition, we implemented RankSVM-Meta, as a baseline for RankSVM-W. RankSVM-Meta is an algorithm proposed in [Carvalho et al. 2008]. The main idea is to use a sigmoid loss to replace the hinge loss in RankSVM. Besides, to avoid being tracked into a poor local optimum, it uses the ranking function learned by RankSVM as the starting point in the learning process.

We also implemented the approach proposed in [Brodley and Friedl 1999] as another baseline. Specifically, we first divide the training set into ten parts, and train a classifier using every nine parts. In this way, we can get ten classifiers. Then we test each document with the ten classifiers, and remove a document if the majority voting of the ten classifiers is different from its original label. After removing noisy documents in this way, we apply McRank, RankSVM, RankBoost, and RankNet to learn ranking functions from the new training set. We denote the corresponding methods as McRank-Rmv, RankSVM-Rmv, RankBoost-Rmv, and RankNet-Rmv respectively.

As another natural baseline, we reassign the label $y$ that maximize the conditional probability $p(y|\tilde{x}_i^q)$ to each document $\tilde{x}_i^q$ in the training data and apply existing learning-to-rank algorithms on the relabeled training data. We denote the corresponding methods as McRank-Rel, RankSVM-Rel, RankBoost-Rel, and RankNet-Rel respectively.

For ease of comparison, all the above algorithms (including our proposed ones) used linear ranking functions.

4.1.3. Parameter Tuning. There is a hyper parameter $C$ in RankSVM [Joachims 2002], RankSVM-W and RankSVM-Rmv. We tried a set of values, $\{0.0001, 0.0002, 0.0005,$
0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5}, and selected the best value using the validation set. For RankSVM-Meta, we set its hyper parameter \( \sigma = 0.1 \) as suggested in [Carvalho et al. 2008].

In our proposed algorithms, we need to quantize the original feature variables. We set the parameter \( b = 10 \) in Eqn. (1). For the smoothing parameter \( \lambda \) in Eqn. (10) and Eqn. (11), we tried a set of values, \( \{0, 0.2, 0.4, 0.6, 0.8, 1\} \), and selected the best one using the validation set.

4.1.4. Evaluation Metrics. We use NDCG [Järvelin and Kekäläinen 2002] and MAP [Baeza-Yates et al. 1999] to evaluate the ranking accuracy, which are both popular measures used in the literature of information retrieval. NDCG@\( k \) is defined as below,

\[
\text{NDCG}@k \triangleq Z_k \sum_{j=1}^{k} \begin{cases} 2^{r(j)} - 1, & j = 1, 2 \\ \frac{2^{r(j)} - 1}{\log(j)}, & j > 2 \end{cases}, \tag{25}
\]

where \( j \) is the position in the document list, \( r(j) \) is the label of the \( j \)-th document in the list (e.g. \( r(j) = 1 \) if the \( j \)-th document is relevant and \( r(j) = 0 \) otherwise), and \( Z_k \) is chosen so that NDCG at each position of the perfect ranking list equals one.

Average Precision (AP) of a query is defined as follows,

\[
\text{AP} = \frac{1}{n_1} \sum_{k=1}^{n} \sum_{i \leq k} I(r(i) = 1) \cdot \frac{1}{k}, \tag{26}
\]

where \( I(\cdot) \) is the indicator function, \( n_1 \) is the number of relevant documents, \( n \) is the total number of documents. MAP is the average AP over all the queries.

4.2. Analyze the Noise on Datasets

To analyze the noise level of different datasets, we plot the distribution of the estimated probabilities \( P(y_q^i | x_q^i) \) in Table II. We can see that a large portion of estimated probability falls into the range \( 0 - 0.1 \) and \( 0.9 - 1 \). Comparing the two large scale datasets used in experiments, Microsoft Learning to Rank Datasets exhibit more noise than Yahoo! Learning to Rank Challenge Dataset: more documents in range \( 0 - 0.1 \) and less documents in range \( 0.9 - 1 \). We also calculate the average log-likelihood by the following formula:

\[
L(D) = -\frac{1}{|D|} \sum_{(q,i) \in D} \log P(y_q^i | x_q^i). \tag{27}
\]

The result also verifies that Microsoft Learning to Rank Datasets are more noisy than Yahoo! Learning to Rank Challenge Dataset: this number is 1.275 for Yahoo! Learning to Rank Challenge Dataset and 1.781 for Microsoft Learning to Rank Datasets (averaged over 5 folders).

We show an example from Microsoft Learning to Rank Datasets. Query with ID “10” has three distinct documents with label “4”. By applying the proposed probabilistic model, we find that one document is very noisy \( (P(y_q^i | x_q^i) \approx 1.3 \times 10^{-4}) \) while the other two have no noise at all \( (P(y_q^i | x_q^i) \approx 0.99) \). It turns out that several features are very strong to distinguish the noisy document from the clean ones. For example, the 13-th feature assigns a probability 0.67 to \( P(x_{1,13}^q | y_q^i = 3) \) and 0.25 to \( P(x_{1,13}^q | y_q^i = 4) \) for the noisy document, while the probabilities are 0.067 and 0.45 respectively for both clean
Table II. Distributions of Estimated Probabilities

<table>
<thead>
<tr>
<th></th>
<th>0-0.1</th>
<th>0.1-0.2</th>
<th>0.2-0.3</th>
<th>0.3-0.4</th>
<th>0.4-0.5</th>
<th>0.5-0.6</th>
<th>0.6-0.7</th>
<th>0.7-0.8</th>
<th>0.8-0.9</th>
<th>0.9-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTRC</td>
<td>0.0717</td>
<td>0.0014</td>
<td>0.0008</td>
<td>0.0009</td>
<td>0.0007</td>
<td>0.0012</td>
<td>0.0011</td>
<td>0.0013</td>
<td>0.9194</td>
<td></td>
</tr>
<tr>
<td>MSLR fold1</td>
<td>0.2805</td>
<td>0.0125</td>
<td>0.0103</td>
<td>0.0094</td>
<td>0.0098</td>
<td>0.0108</td>
<td>0.0131</td>
<td>0.0201</td>
<td>0.6147</td>
<td></td>
</tr>
<tr>
<td>MSLR fold2</td>
<td>0.2805</td>
<td>0.0124</td>
<td>0.0102</td>
<td>0.0094</td>
<td>0.0098</td>
<td>0.0108</td>
<td>0.0131</td>
<td>0.0201</td>
<td>0.6137</td>
<td></td>
</tr>
<tr>
<td>MSLR fold3</td>
<td>0.2819</td>
<td>0.0126</td>
<td>0.0106</td>
<td>0.0095</td>
<td>0.0109</td>
<td>0.0131</td>
<td>0.0200</td>
<td>0.6128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSLR fold4</td>
<td>0.2810</td>
<td>0.0126</td>
<td>0.0105</td>
<td>0.0094</td>
<td>0.0098</td>
<td>0.0108</td>
<td>0.0131</td>
<td>0.0200</td>
<td>0.6137</td>
<td></td>
</tr>
<tr>
<td>MSLR fold5</td>
<td>0.2811</td>
<td>0.0125</td>
<td>0.0106</td>
<td>0.0094</td>
<td>0.0096</td>
<td>0.0106</td>
<td>0.0130</td>
<td>0.0200</td>
<td>0.6149</td>
<td></td>
</tr>
</tbody>
</table>

Table III. Strong features for query with id “10” in Microsoft Learning to Rank Datasets
13, 26, 30, 48, 50, 51, 53, 58, 63, 65, 71, 75, 76, 80, 86, 90, 108, 111, 113, 115, 116, 120, 123

Fig. 2. Comparison Between McRank-W and Baselines

documents. We list all the strong features \( P(\tilde{x}_{13}^q | y_i^q = 3) > 2 P(\tilde{x}_{13}^q | y_i^q = 4) \) for this example in Table III.

As a comparison, we also study the Euclidean distance in this example. The Euclidean distances between each pair of documents which are 1.46, 1.49 and 1.43 respectively. Given that the average distance between a pair of document for this query is 1.38, these documents are not far from each other in the feature space. So Euclidean distance (as well as many other distance measures) cannot effectively detect noisy documents. There are two reasons: first, all the features are treated equally in Euclidean distance but only some of the feature are strong according to Table III (23 out of 136 features in total); second, the feature value is not as effective as the conditional probability calculated from the value. As a conclusion, our approach can effectively distinguish the noisy document from the clean ones which is highly non-trivial.

4.3. Comparisons of Ranking Accuracy on TD2003 and TD2004

The ranking accuracies of McRank-W, RankSVM-W, RankNet-W, RankBoost-W and the baselines on TD2003 and TD2004 are shown in Figures

— The extended algorithms (i.e., McRank-W, RankSVM-W, RankNet-W, and RankBoost-W) outperform the original algorithms (i.e., McRank, RankSVM, RankNet, and RankBoost) for almost all the measures and on both TD2003 and TD2004 datasets. The only exception is for McRank in terms of MAP. Our extended

---

\(^2\)Since the features in Microsoft Learning to Rank Datasets are not disclosed, we only list the feature IDs here.
method leads to a minor drop of the MAP value on TD2003. In all other cases, our proposed methods achieve better performances. For example, RankBoost-W outperforms RankBoost by 10% and 25% in terms of MAP and NDCG@10 on TD2003. The possible explanation to the improvement is that our proposed approach can effectively identify noisy documents and reduce their influences on the training process. This can be verified by Figure 6. In this figure, we plot the distribution of the estimated probabilities $P(y^*_q|x^*_q)$ for “correctly-labeled” documents (whose labels are not flipped) and “noisy” documents (whose labels are flipped) of the five folds.
in TD2003. From the figure we can find that the estimated probabilities of most correctly-labeled documents are very close to one while those of most noisy documents are close to zero. Since the estimated noisy documents will be assigned small weights in training, their negative influences will be greatly reduced.

— The performance of RankSVM-Rmv is consistently worse than that of RankSVM for all the measures and on both datasets. We also have similar observations on McRank-Rmv. For RankBoost-Rmv and RankNet-Rmv, they outperform the original algorithms (RankBoost and RankNet) in some cases but underperform the original algorithms in even more cases. The explanation to these observations is that removing noisy documents will reduce the size of the training set, and thus affect the effectiveness of the learned ranking function. As a result, it is not guaranteed that the learned ranking function will have better performance than the ranking function learned from noisy but larger training data.

— RankSVM-Meta performs consistently worse than RankSVM. For example, RankSVM-Meta is worse than RankSVM by 7%, and 10% in terms of MAP and NDCG@10 respectively on TD2003. This verifies our discussions in Section 2. That is, although the sigmoid loss can help reduce the influence of noisy documents, it also reduces that of correctly-labeled documents (which should be emphasized, however).

— The extended algorithm RankSVM-W consistently outperforms RankSVM-Rel. For example, RankSVM-W outperforms RankSVM-Rel by 4% and 5% in terms of MAP and NDCG@10 respectively on TD2003. Since the two algorithm share the same probabilistic noise model, it demonstrates our solution that incorporates the probabilistic noisy model into existing learning-to-rank algorithms is not trivial. Furthermore, the performance of RankSVM-Rel is worse than RankSVM for some measures such as NDCG@10 on TD2004. Similar observations and conclusion can be made for other algorithms.

4.4. Comparisons of Ranking Accuracy on Large Scale Datasets

The ranking accuracies of McRank-W, RankSVM-W, RankNet-W, RankBoost-W and the baselines are shown in Figures 7-10. From the table, we have the following observations.

— All the extended algorithms (i.e., McRank-W, RankSVM-W, RankNet-W, and RankBoost-W) outperform the original algorithms (i.e., McRank, RankSVM, RankNet, and RankBoost) for almost all the measures and on both Microsoft Learning to Rank dataset and Yahoo! Learning to Rank Challenge dataset. In all most all cases, our proposed algorithms achieve a large improvement over the corresponding baseline. For example, McRank-W outperforms McRank by 0.64% and 1.04% in terms of MAP and NDCG@2 on Microsoft Learning to Rank dataset. We also conducted significant test on the obtained result and it shows that most improvement are significant.

— Because of the large scale nature of the datasets, all the baselines exhibit similar performance and are consistently worse than all the extended algorithms that considers noise in the training data.

4.5. Results w.r.t. Different Noise Levels

To understand how different approaches perform with respect to different noise levels, we plot the performances of RankSVM-W, RankSVM, RankSVM-Rmv, and RankSVM-
Fig. 6. Distributions of Estimated Probabilities on TD2003

Fig. 7. Comparison Between McRank-W and Baselines on Large Scale Datasets

Meta w.r.t. the noise levels of 0.05, 0.1, 0.15, 0.2, 0.25, and 0.3 in Fig. 11-12. We

As in Section 4.1.1, a noise level $\gamma$ means we randomly flip the label of a document with a probability $\gamma$.
discuss the observations obtained from the experiments on TD2003 as below. Similar observation can be obtained for TD2004.

— Roughly speaking, with more noise introduced, the performances of all the algorithms drop. This verifies the argument that noise in the training data will hurt the accuracy of learning to rank.

— RankSVM-W always outperforms RankSVM no matter how much noise is introduced. For example, RankSVM-W is better than RankSVM by 2%, 1.5%, 17%, 20%,
70%, 30% in terms of NDCG@10 when the noise level is 0.05, 0.1, 0.15, 0.2, 0.25, and 0.3. This verifies the effectiveness of our proposed approach. Furthermore, with the increase of the noise level, the improvement of RankSVM-W over RankSVM becomes larger. This shows that when the noise level is higher, it becomes more critical to identify noisy documents and reduce their influences on the learning process.

— When the noise level is low, the performance of RankSVM-Rmv is much worse than that of RankSVM. However, when the noise level is relatively high, its performance is comparable (or better than) that of RankSVM. The explanation to this observation is related to the two effects of removing noisy documents. On one hand, removing noisy documents will make the training data cleaner and more effective for learning. On the other hand, removing noisy documents will reduce the size of the training set and hurt the generalization ability of the learning algorithm. The observation shows that when there is little noise, the first effect dominates while with more noise the second effect becomes more dominative.

— The performance of RankSVM-Meta is relatively stable with respect to different noise levels. This is in accordance with the discussions on the algorithm in Section 2. That is, using the sigmoid loss can resist the effect of noise. However, since it also reduces the effect of clean documents, the performance of RankSVM-Meta is in general worse than RankSVM when the noise level is low.

### 4.6. Effects of Smoothing

In this subsection, we study the effect of the smoothing parameter $\lambda$ in Eqn. (10) and Eqn. (11). When $\lambda = 1$, we estimate the parameters only using the documents associated with a given query. And when $\lambda = 0$, all the queries share the same parameter. We plot the performance of RankSVM-W with respect to the $\lambda$ values of 0, 0.2, 0.4, 0.6, 0.8, and 1 in Figure 13. From the figure, we can find that with the increase of $\lambda$, the performance of RankSVM-W first increases and then decreases. The best performance is achieved when $\lambda$ equals 0.6. We have also observed similar phenomena for other algorithms. Due to space restrictions, we have not plotted their figures.

The above observations are in accordance with our previous discussions in Section 3.1.2. That is, using query specific parameters can better distinguish different levels of labeling noise for different queries, and leveraging the smooth factor estimated from all the queries to some extent can solve the problem of lacking instances for parameter estimation of each query.
5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a two-step approach to learning to rank from noisy data. In the first step, we build a graphical model to estimate the degree of labeling noise of each training document. In the second step, we extend existing learning-to-rank algorithms using the estimated noise information to reduce the influence of the noisy documents in the training process. Empirical results show that our approach is more effective in handling noisy data than previous algorithms.

In future, we plan to study the following issues.

— We have conducted experiments on the LETOR datasets in this paper. We will conduct more experiments on larger datasets to further validate the effectiveness of our proposal.
— We have used a simple smoothing strategy in the proposed approach in Section 3.1.2. We will investigate other advanced smoothing strategies [Zhai and Lafferty 2004].
— We have shown how to extend the pointwise and pairwise learning-to-rank algorithms in this paper. We will study how to extend the listwise algorithms using our approach, which appears to be non-trivial.
REFERENCES
T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. 2007. Letor: Benchmark dataset for research on learning to
rank for information retrieval. In ACM SIGIR conference on Research and development in information
retrieval workshop. 3–10.

Shuzi Niu, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2012. Top-k learning to rank: Labeling, ranking and
evaluation. In Proceedings of the 35th international ACM SIGIR conference on Research and develop-
ment in information retrieval. ACM, 751–760.

Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a ma-
chine learning framework for ranking query suggestions. In Proceedings of the 35th international ACM
SIGIR conference on Research and development in information retrieval. ACM, 25–34.

John Platt and others. 1998. Sequential minimal optimization: A fast algorithm for training support vector


Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model
for probabilistic weighted retrieval. In Proceedings of the 17th annual international ACM SIGIR confer-

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer
re-ranking. In Proceedings of the 35th international ACM SIGIR conference on Research and develop-
ment in information retrieval. ACM, 741–750.

and software 11, 1-4 (1999), 451–484.

Lidan Wang, Paul N Bennett, and Kevyn Collins-Thompson. 2012. Robust ranking models via risk-sensitive
optimization. In Proceedings of the 35th international ACM SIGIR conference on Research and develop-
ment in information retrieval. ACM, 761–770.

J. Xu, C. Chen, G. Xu, and H. Li. 2010. Improving Quality of Training Data for Learning to Rank Using
Click-Through Data. In Proceedings of the third ACM international conference on Web search and data
mining. ACM.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A support vector method for
optimizing average precision. In Proceedings of the 30th annual international ACM SIGIR conference
on Research and development in information retrieval. ACM, 271–278.

C. Zhai and J. Lafferty. 2004. A study of smoothing methods for language models applied to information