Learning to Rank from Bayesian Decision Inference

Jen-Wei Kuo*
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
rogerkuo@iis.sinica.edu.tw

Pu-Jen Cheng
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
pjcheng@csie.ntu.edu.tw

Hsin-Min Wang
Institute of Information Science
Academia Sinica
Taipei 115, Taiwan
whm@iis.sinica.edu.tw

ABSTRACT

Ranking is a key problem in many information retrieval (IR) applications, such as document retrieval and collaborative filtering. In this paper, we address the issue of learning to rank in document retrieval. Learning-based methods, such as RankNet, RankSVM, and RankBoost, try to create ranking functions automatically by using some training data. Recently, several learning to rank methods have been proposed to directly optimize the performance of IR applications in terms of various evaluation measures. They undoubtedly provide statistically significant improvements over conventional methods; however, from the viewpoint of decision-making, most of them do not minimize the *Bayes risk* of the IR system. In an attempt to fill this research gap, we propose a novel framework that directly optimizes the *Bayes risk* related to the ranking accuracy in terms of the IR evaluation measures. The results of experiments on the LETOR collections demonstrate that the framework outperforms several existing methods in most cases.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

Learning to Rank, Ranking function

1. INTRODUCTION

The rapid growth and popularity of the Web in the last decade has resulted in a huge number of information sources on the Internet, but it has made information retrieval (IR) more difficult for end users. Search engines have therefore become increasingly important in helping users accurately locate relevant content based on their information needs. IR can be formulated as a binary classification problem in which documents are categorized as relevant or irrelevant. However, in practice, the textual content should have multiple degrees of relevance to a query. Therefore, the IR problem can also be formulated as a ranking problem, which means that, given a query, the documents are sorted by the ranking function, and then the ranked list is returned to the user. Ranking functions influence both the quality of the search results and users' search experience directly. Research on ranking models has become a fundamental research topic. Many models and methods have been proposed to solve this problem, e.g., the Boolean model, vector space model [23], probabilistic model [22], and language modeling-based methods [19]. In empirical IR models, tuning parameters by using certain training data is a common practice; however, as ranking models become more sophisticated, parameter tuning becomes an increasingly challenging issue.

In the last decade, several human-judged relevance assessments have been made available for IR research. This makes it possible to incorporate many of the significant advances in machine learning into the design of ranking models. For this reason, many learning methods have been developed and applied to document retrieval and related fields. Basically, these methods transform the ranking problem into binary classification on pairs constructed between documents. In fact, methods like RankNet [3], RankSVM [7, 10], and RankBoost [6] typically minimize a loss function that is loosely related to the ranking accuracy in terms of the evaluation measures, such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) [9]. Therefore, a substantial amount of research effort has focused on constructing ranking functions by optimizing these evaluation measures directly. Related algorithms, such as SVM^{map} [28] and AdaRank [26], have proved effective for IR applications.

In this paper, we propose a learning to rank framework based on a Bayesian perspective. Under the framework, the Plackett-Luce Model is introduced as the probability model of permutations. Like the approach in [5], we transform the ranking scores to permutation probabilities such that the ranking function can be optimized indirectly from Bayesian decision inference. The optimal ranking function appears when the expected *Bayes risk* reaches the minimum. Accordingly, we call the proposed learning to rank

*The author is also with the Institute of Information Science, Academia Sinica, Taiwan.

CIKM 09, November 2–6, 2009, Hong Kong, China.

framework *Bayes*Rank. The framework is fairly general and provides flexibility for many applications, such as information retrieval, automatic summarization, and collaborative filtering. Under the framework, one can optimize the expected performance of a ranking function by adopting an arbitrary permutation loss related to desired metrics. Similar to other learning algorithms, *Bayes*Rank also minimizes an upper bounding function of the ranking error, which means that the ranking error can be iteratively reduced during the training process. For document retrieval, we design a learning algorithm for *Bayes*Rank with NDCG related permutation loss based on multi-layer perceptron neural networks. The results of experiments on the LETOR collections [13, 20], containing both TREC and OHSUMED benchmarks, demonstrate that, in most cases, *Bayes*Rank achieves consistent improvements over the compared ranking algorithms, namely AdaRank [26], ListNet [5], and SVM^{map} [28].

The remainder of this paper is organized as follows: In Section 2, we review previous works, and then formulate the ranking problem in terms of Bayes decision theory in Section 3. In Section 4, we describe the proposed learning framework and algorithm. In Section 5, we compare *Bayes*Rank with ListNet [5] and PermuRank [27] from a theoretical perspective. Section 6 contains the experiment results and a discussion of their implications. Then, in Section 7, we summarize our conclusions and indicate several directions for future research.

2. RELATED WORK

Information retrieval can be viewed as a ranking problem or a decision-making problem. In this section, we review previous works on information retrieval from these two aspects.

2.1 Ranking Aspect

IR problems, such as document retrieval, can be formulated as ranking problems, which can be solved by various popular models, such as the Boolean model, vector space model, probabilistic model, and language model. In recent years, many attempts have been made to utilize machine learning methods to solve IR problems. Learning to rank approach which tries to construct a ranking model using some training data, has been addressed in pointwise, pairwise, and listwise ways. The pointwise approach [17, 12] transforms the ranking problem into a regression or classification problem of a single document. The pairwise approach [24, 4, 30] defines a pairwise loss function and is concerned with classification of document pairs; typical methods include RanSVM [7, 10], RankBoost [6], and RankNet [3].

The listwise approach [2] has become increasingly popular in recent years. It attempts to solve the ranking problem by minimizing a listwise surrogate loss function. ListNet [5], an extension of RankNet, defines the loss function as the cross entropy between two parameterized probability distributions of permutations. RankCosine [21] and ListMLE [25] inherit a similar structure from ListNet, except for the surrogate loss functions.

However, minimizing the surrogate loss does not guarantee that the IR performance in terms of evaluation measures can also be optimized. Let us take the pairwise case as an example and consider the following scenario. For a given query, two ranking functions, f_A and f_B, are considered to rank 10 documents, two of which are judged as 'relevant' and the others are judged as 'irrelevant' by human. The ranked lists produced by these two ranking functions are shown in Table 1, where the relevant and irrelevant documents are denoted as '0' and '1' respectively. We observe that f_A incurs 50% PER (pair error rate), but it yields a better AP (average precision) than f_B, which only introduces 37.5% PER. The kind of mismatch occurs when the surrogate loss function is inconsistent with the evaluation metrics.

Table 1: An example of the inconsistency between PER (pairwise error rate) and AP (average precision)

	Ranked lists	PER	AP
f_A	1 0 0 0 0 0 0 0 0 1	8/16=50.0%	(1/1+2/10)/2=0.600
f_B	0 0 0 1 1 0 0 0 0 0	6/16=37.5%	(1/4+2/5)/2=0.325

For this reason, a branch of listwise methods, such as SVM^{map} [28], AdaRank [26], and PermuRank [27], tries to optimize the evaluation measures directly. Undoubtedly, they provide significant improvements over conventional methods; however, when we view IR as a decision-making problem, most of them do not minimize the *Bayes risk* of the system.

2.2 Decision-making Aspect

Information retrieval can be treated as a statistical decision-making problem [11, 29]. Given a user's query, the retrieval system faces a decision-making problem in that it must choose relevant documents from a hypothesized space and return a ranked list to the user. From the aspect, Zhai and Lafferty [29] proposed a risk minimization framework for information retrieval. However, they did not address the supervised learning scenario from the viewpoint of decision-making. Instead, they focused on developing retrieval methods for various retrieval cases, such as set-based retrieval, rank-based retrieval, and aspect retrieval. To the best of our knowledge, not much work has considered these two aspects jointly, especially for direct optimization of IR performance. In this paper, we propose a learning to rank framework that addresses both aspects.

3. RANKING PROBLEM

In document retrieval, documents related to a query are managed by a ranking model and presented to the user according to their relevance to the query. In practice, the ranking problem may be reduced to finding an appropriate scoring function that can evaluate individual documents. The ranking function sorts the documents in descending order of the assigned scores, and then forms a ranked list[1]. The notations used in this paper are summarized in Table 2. Suppose $D = \{d_1, d_2, ..., d_n\}$ is a set of n documents. In the retrieval stage, given a query q, the scoring function $\mathbf{g} \in \mathcal{G}$ evaluates every document in D, and then compiles a score list, say $\{y_1, y_2, ..., y_n\}$. The documents are sorted according to the scores and presented to the user. In the supervised learning stage, a set of training queries $Q = \{q_1, ..., q_m\}$ and a relevance mapping $\mathbf{r} \in \mathcal{G}$ are given. The relevance mapping $\mathbf{r}$, which can be regarded as a kind of scoring function in the function space $\mathcal{G}$, reflects the relevance judgments. The

[1]In this paper, ranked (document) list and permutation are identical.

Table 2: Summary of notations

Notation	Explanation
$d_i \in D$	i^{th} document in D
$q \in Q$	Query
$\mathbf{g} \in \mathcal{G}$	Scoring function
$\mathbf{r} \in \mathcal{G}$	Relevance mapping
$\pi \in \Pi_q$	Permutation for q
$\pi_q^* \in \Pi_q$	Perfect ranked list for q
$p(\pi; q)$	Conditional probability of π given q
$l(\pi, \pi_q^*)$	Permutation-level loss incurred by making a decision $\alpha(\pi; q)$
$R(\pi; q)$	Expected risk of selecting π for q

learning to rank approach tries to create the scoring function automatically from the training data, which include the query set Q and the relevance mapping function $\mathbf{r}$.

3.1 Formulation

Given a user's query q, a retrieval system attempts to make a decision $\alpha(\pi; q)$ that selects a ranked document list π from a set of possible permutations Π_q to return to the user. Note that we assume π is a random variable in the hypothesized permutation space Π_q with an unknown probability distribution $P_q(\pi)$. Let $l(\pi; \pi_q^*)$ be the permutation-level loss incurred by taking decision α when the perfect ranked list is $\pi_q^* = sort_{\mathbf{r}}\{d_1, \cdots, d_n\}$ in which the documents are sorted according to the relevance mapping $\mathbf{r}$. Generally, π_q^* should be a subset of Π_q rather than a unique perfect permutation. However, for ease of presentation, we let π_q^* be a perfect permutation hereafter.[2] In the retrieval stage, no explicit information about π_q^* is presented; i.e., any arbitrary permutation could be π_q^*. Therefore, we model the uncertainty by the conditional probability $p(\pi; q)$, which corresponds to the probability that π would be judged as the perfect permutation for query q. As a result, in the general framework of Bayesian decision theory, the expected risk of taking decision $\alpha(\pi; q)$ is given by

$$R(\pi; q) = \int_{\Pi_q} l(\pi, \pi') dp(\pi'; q). \quad (1)$$

The best decision $\bar{\alpha}$ can be selected by minimizing the expected risk as follows:

$$\bar{\alpha} = \arg\min_{\pi} \int_{\Pi_q} l(\pi, \pi') dp(\pi'; q). \quad (2)$$

The minimum expected risk is called the *Bayes risk*. In the supervised learning scenario, the ground-truth associated with each query q is presented. Hence, the *Bayes risk* of π_q^* over the training query set Q is given by

$$R = \sum_{q \in Q} p(q) R(\pi_q^*; q) = \sum_{q} p(q) \int_{\Pi_q} l(\pi_q^*, \pi') dp(\pi'; q). \quad (3)$$

R is the expected *Bayes risk* over Q. If we assume that the prior $p(q)$ is uniformly distributed and the permutation space Π_q is finite, the expected *Bayes risk* can be approximated by

[2]The assumption does not affect the correctness of the derivation.

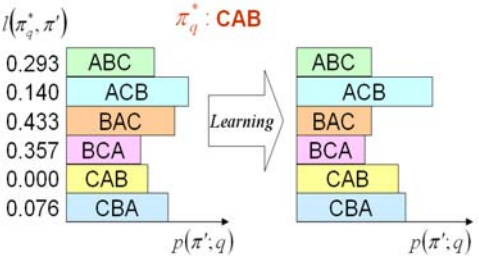

Figure 1: Changes in the probabilities of permutations with different permutation-level losses during the learning process.

$$R \approx \frac{1}{m} \sum_{q \in Q} \sum_{\pi' \in \Pi_q} l(\pi_q^*, \pi') p(\pi'; q), \quad (4)$$

where m is the number of queries in Q. Unlike many existing methods that embed the scoring function in the surrogate loss function, our approach tries to model the conditional probability $p(\pi; q)$ where the scoring function is embedded. This strategy suggests two advantages:

1. It is not necessary to formulate the ranking error as a surrogate loss function because of the non-differentiable nature.

2. The permutation-level loss can be directly related to the desired IR evaluation measures, such as MAP and NDCG.

In other words, the learning process tends to adjust the parameters in the scoring function such that a lower probability is assigned to the permutation with a higher loss, and the probability of that with a lower loss is increased. This leads, indirectly, to minimization of the objective function, i.e., the expected *Bayes risk*. Figure 1 illustrates the changes in the probabilities of permutations with different permutation-level losses during the learning process.

3.2 Permutation-level Loss

The permutation-level loss $l(\pi_q^*, \pi)$ is incurred by selecting π from Π_q for query q when the perfect ranked list is π_q^*; therefore, it can be directly related to an arbitrary IR evaluation metric that measures the distance between π and π_q^*. In general, we have to restrict the range of the loss, e.g., between zero and one, to bound the expected *Bayes risk* in order to prevent the model from being biased by some hard queries. To maximize the IR performance, the loss can be derived directly from the evaluation measures, i.e., $l(\pi_q^*, \pi) = 1 - E(\pi, \pi_q^*)$, where $E(\cdot, \cdot)$ can be MAP, P@n, or NDCG@n.

3.3 The Plackett-Luce Model

To model the ranked list appropriately, many probabilistic models have been proposed for modeling rank data, e.g., the Bradley-Terry-Luce model [1, 14], the Mallows model [15], and the Plackett-Luce model [18, 14]. Marden provided an excellent analytical review of the research on models for rank data [16], one of which is the Plackett-Luce model. It models a ranking as a sequential process and has been used

Algorithm 1 Learning Algorithm

Input:training queries $Q = \{q_i\}$,
  relevance mapping $\mathbf{r}$
Initialize parameters: λ, learning rate γ
repeat
  $R \leftarrow 0, \Delta_\lambda \leftarrow 0$
  for $i = 1, \cdots, m$ do
    for $j = 1, \cdots, n$ do //Precalculation
      Input q_i, d_j to neural networks,
        Evaluate $\exp(\mathbf{g}(d_j; q_i, \Lambda))$ and $\frac{\partial \mathbf{g}(d_j;q_i,\mathbf{\Lambda})}{\partial \lambda}$
    end for
    for π_1^k do
      for $j = 1, \cdots, n$ do
        Evaluate $\frac{\partial p_k(\pi_1^k;q_i,\mathbf{\Lambda})}{\partial \mathbf{g}(d_j;q_i,\mathbf{\Lambda})}$
        $\Delta_\lambda \leftarrow \Delta_\lambda + G_k(\pi_1^k, \pi_{q_i}^*) \frac{\partial p_k(\pi_1^k;q_i,\mathbf{\Lambda})}{\partial \mathbf{g}(d_j;q_i,\mathbf{\Lambda})} \frac{\partial \mathbf{g}(d_j;q_i,\mathbf{\Lambda})}{\partial \lambda}$
      end for
      $R \leftarrow R - p_k(\pi_1^k; q_i, \mathbf{\Lambda}) G_k(\pi_1^k, \pi_{q_i}^*)$
    end for
  end for
  Update $\lambda \leftarrow \lambda - \gamma \Delta_\lambda$
  Update γ
until R converges

successfully in ListNet [5]. Cao et al. have introduced an increasing and strictly positive function to transform ranking scores into probabilities [5]. In line with their work, we adopt the following form of the Plackett-Luce model in our framework:

$$p(\pi; q, \mathbf{g}) = \prod_{i=1}^{n} \frac{\exp(\mathbf{g}(\pi(i); q))}{\sum_{j=i}^{n} \exp(\mathbf{g}(\pi(j); q))}, \tag{5}$$

where i and j are the rank indices and $\pi(i)$ denotes the document with rank i in π. The permutation probability is estimated through the scoring function $\mathbf{g}$. In [5], the authors clarified an important property for this form of Plackett-Luce Model. Given a scoring function, the ranked list of the documents sorted based on the scores has the highest permutation probability, while the list of documents sorted in the inverse order has the lowest permutation probability. The property implies that choosing the ranked list with the highest probability is equivalent to the way a typical ranking function selects a list.

4. ALGORITHM

In this paper, we take the multi-layer perceptron neural networks as an example of the scoring function in the ranking model and design the permutation-level loss as the opposite of *NDCG@k* score, i.e.,

$$l(\pi_q^*, \pi) = -G_k(\pi, \pi_q^*), \tag{6}$$

where

$$G_k(\pi, \pi_q^*) = \left(\sum_{i=1}^{k} \frac{2^{\mathbf{r}(\pi_q^*(i))}}{\log(1+i)}\right)^{-1} \left(\sum_{j=1}^{k} \frac{2^{\mathbf{r}(\pi(j))}}{\log(1+j)}\right). \tag{7}$$

Then, the objective function to be minimized becomes

$$R(\mathbf{\Lambda}) = -\frac{1}{m} \sum_{q \in Q} \sum_{\pi \in \Pi_q} G_k(\pi, \pi_q^*) p(\pi; q, \mathbf{\Lambda}), \tag{8}$$

where $\mathbf{\Lambda}$ is the set of parameters in the neural networks. For each permutation π, we divide the document list into two sublists: π_1^k and π_{k+1}^n, as follows:

$$\begin{aligned} \pi &= < \underbrace{\pi(1), \cdots, \pi(k)}_{\pi_1^k}, \underbrace{\pi(k+1), \cdots, \pi(n)}_{\pi_{k+1}^n} > \\ &= < \pi_1^k, \pi_{k+1}^n > . \end{aligned} \tag{9}$$

Because of the nature of the position-dependent loss, permutations with the same π_1^k incur the same loss $-G_k$, i.e., $G_k(\pi, \pi_q^*) = G_k(\pi_1^k, \pi_q^*)$. This makes it possible to sum the probabilities of those permutations directly, and evaluate Equation (8) by only considering the top k documents, i.e., π_1^k. Therefore, Equation (8) can be re-written as

$$R(\mathbf{\Lambda}) = -\frac{1}{m} \sum_{q \in Q} \sum_{\pi_1^k} G_k(\pi_1^k, \pi_q^*) p_k(\pi_1^k; q, \mathbf{\Lambda}), \tag{10}$$

where

$$\begin{aligned} p_k(\pi_1^k; q, \mathbf{\Lambda}) &= \sum_{\pi_{k+1}^n} p(< \pi_1^k, \pi_{k+1}^n >; q, \mathbf{\Lambda}) \\ &= \prod_{i=1}^{k} \frac{\exp(\mathbf{g}(\pi_1^k(i); q, \mathbf{\Lambda}))}{\sum_{j=i}^{k} \exp(\mathbf{g}(\pi_1^k(j); q, \mathbf{\Lambda})) + T_{\mathbf{g},q}^k} \end{aligned} \tag{11}$$

and

$$T_{\mathbf{g},q}^k = \sum_{d' \notin \pi_i^k} \exp(\mathbf{g}(d'; q, \mathbf{\Lambda})). \tag{12}$$

where $p_k(\pi_1^k; q, \mathbf{\Lambda})$ is identical to the *top k probability* defined in [5]. Taking the derivative of Equation (10) with respect to the parameter λ yields

$$\frac{\partial R}{\partial \lambda} = -\frac{1}{m} \sum_{q \in Q} \sum_{\pi_1^k} \sum_{d} G_k(\pi_1^k, \pi_q^*) \frac{\partial p_k(\pi_1^k; q, \mathbf{\Lambda})}{\partial \mathbf{g}(d; q, \mathbf{\Lambda})} \frac{\partial \mathbf{g}(d; q, \mathbf{\Lambda})}{\partial \lambda}, \tag{13}$$

where

$$\frac{\partial p_k(\pi_1^k; q, \mathbf{\Lambda})}{\partial \mathbf{g}(d; q, \mathbf{\Lambda})} = p_k(\pi_1^k; q, \mathbf{\Lambda}) \times \begin{cases} 1 - \sum_{i=1}^{rank(d)} \frac{\exp(\mathbf{g}(d;q,\mathbf{\Lambda}))}{\sum_{j=i}^{k} \exp(\mathbf{g}(\pi_1^k(j);q,\mathbf{\Lambda})) + T_{\mathbf{g},q}^k} & d \in \pi_1^k \\ - \sum_{i=1}^{k} \frac{\exp(\mathbf{g}(d;q,\mathbf{\Lambda}))}{\sum_{j=i}^{k} \exp(\mathbf{g}(\pi_1^k(j);q,\mathbf{\Lambda})) + T_{\mathbf{g},q}^k} & d \notin \pi_1^k, \end{cases} \tag{14}$$

where $rank(d)$ denotes the rank of document d in the ranked list π_1^k. The gradient of $\mathbf{g}(d; q, \mathbf{\Lambda})$ with respect to λ, $\frac{\partial \mathbf{g}(d;q,\mathbf{\Lambda})}{\partial \lambda}$, can be found in [3]. Thus, λ is updated using the gradient descent with a positive learning rate γ:

$$\lambda \leftarrow \lambda - \gamma \frac{\partial R}{\partial \lambda}. \tag{15}$$

In the estimation of the gradient in Equation (14), the exponentiation operation incurs a high computational overhead; therefore, we evaluate $\exp(\mathbf{g}(d; q, \mathbf{\Lambda}))$ beforehand and then update λ using the batch gradient descent algorithm. The learning algorithm is detailed in **Algorithm 1**.

5. THEORETICAL ANALYSIS

In this section, we provide a proof of the correctness of *Bayes*Rank and discuss its relation to ListNet [5] and PermuRank [27].

5.1 Error Bound

THEOREM 1. *Let $\tilde{\pi}_q$ be the ranked document list that possesses the maximal probability $p(\tilde{\pi}_q; q, \mathbf{g})$ for a training query $q \in Q$. Then, the bound holds on the ranking error*

$$\sum_q l(\pi_q^*, \tilde{\pi}_q) \leq \kappa \cdot R,$$

where $\kappa = \max_q \#_{\Pi_q}$, and $\#_{\Pi_q}$ is the size of the permutation space for query q.

A proof of Theorem 1 is given in the Appendix. Since κ is a fixed constant during the training process, the theorem implies that minimizing the expected *Bayes risk* R will lead to a continuous reduction of the upper bound of the ranking error. Xu et al. [27] classify the methods that directly optimize IR evaluation measures into three categories in terms of loss function optimization. Our method belongs to the first category, which minimizes the upper bound of the basic loss function defined according to the IR evaluation measures.

5.2 Relation to ListNet

*Bayes*Rank bears some resemblance to ListNet [5], which models the ranking error as a surrogate function based on the cross entropy. It is assumed that there is uncertainty in the prediction of ranked lists using the ranking function. In contrast, *Bayes*Rank focuses on modeling the conditional permutation probability so as to minimize the expected *Bayes risk* from the decision-making aspect. We now show that, in some cases, the loss function of ListNet is the upper bound of the expected risk.
For ListNet, the loss function for query q is defined as

$$l_q^{ListNet} = -\sum_{\pi \in \Pi_q} p(\pi; q, \mathbf{g}) \log p(\pi; q, \mathbf{r}) \qquad (16)$$

where $p(\cdot; q)$ represents the ranking probability, which can be defined as the *top k probability* provided by

$$p_k(\pi; q, \mathbf{g}) = \prod_{i=1}^{k} \frac{\exp\left(g(\pi(i); q)\right)}{\sum_{j=i}^{n} \exp\left(g(\pi(j); q)\right)}$$

As the result, we have the following theorem:

THEOREM 2. *Equation (16) is an upper bound of expected risk $R(\pi_q^*; q)$ in the case that $l(\pi_q^*, \pi)$ is evaluated as $1 - p_k(\pi; q, \mathbf{r})$.*

It is straightforward to verify that Theorem 2 holds by applying Jensen's inequality. ListNet can be viewed as maximizing the performance in terms of $p_k(\pi_q^*; q, \mathbf{r})$. From this perspective, *Bayes*Rank provides a tighter bound for optimizing such a measure.

5.3 Relation to PermuRank

To minimize the ranking error $l(\pi_q^*, \tilde{\pi}_q)$ of $\tilde{\pi}_q$, which is the permutation selected for q by the ranking model, Xu et al. introduced two types of bounds for direct optimization methods [27]. The *type one* bound, optimized by AdaRank [26], is defined directly on the IR measures; while the *type two* bound is defined with the pairs comprised of a perfect permutation and an imperfect permutation. PermuRank is

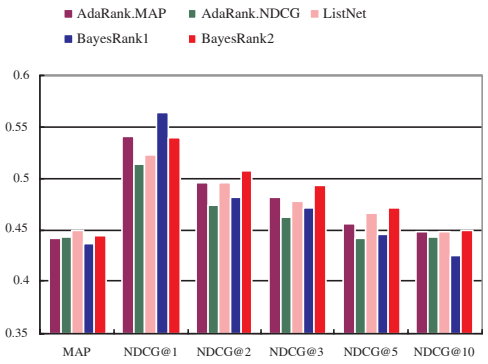

Figure 2: Ranking accuracy of various methods on OHSUMED (LETOR 2.0).

a generalized algorithm that minimizes the *type two* bound, which is derived from a loss function [27] as follows:

$$\max_{\pi \in \Pi_q} l(\pi_q^*, \pi)[[\mathcal{F}(\pi_q^*; q) \leq \mathcal{F}(\pi; q)]] \qquad (17)$$

where $\mathcal{F}(\pi; q)$ evaluates permutation π and $[[\cdot]]$ is one when the condition is satisfied; otherwise, it is zero.

In contrast, the expected *Bayes risk* can also be extended to a generalized form of the upper bound of the ranking error.

THEOREM 3. *For all $\pi \in \Pi_q$, the following bound holds on the ranking error $l(\pi_q^*, \tilde{\pi}_q)$:*

$$l(\pi_q^*, \tilde{\pi}_q) \leq \max_{\pi \in \Pi_q} l(\pi_q^*, \pi) \frac{\sum_{\pi'} \exp\left(\mathcal{F}(\pi; q)\right)}{\sum_{\pi'} \exp\left(\mathcal{F}(\pi'; q)\right)}. \qquad (18)$$

A proof of Theorem 3 is given in the Appendix. It implies that *Bayes*Rank does not try to minimize the two types of bounds defined by Xu et al. Instead, it adopts a new type of upper bounding function, as shown in Equation (18). As a result, in future research, it will be possible to develop new ranking models of soundness based on the new type of bound.

6. EXPERIMENTS

6.1 Data Collections

LETOR (LEarning TO Rank) [13] is a benchmark collection constructed for learning to rank research. The second version (LETOR 2.0) has been widely used to evaluate various ranking algorithms; however, the provider acknowledged that there are some issues with LETOR 2.0 [20]. For example, some low-level information is missing, and the sampling of documents associated with each query is somehow biased. To make LETOR more reliable, the provider improved it in three ways and released LETOR 3.0 in December, 2008. We conducted our experiments on LETOR 2.0 and LETOR 3.0, both of which contain two datasets: OHSUMED and .Gov.

OHSUMED [8] is a subset of MEDLINE, a database of medical publications. There are totally 106 queries, each of which has about 152 documents on average for feature extraction. In contrast to LETOR 2.0, each query-document

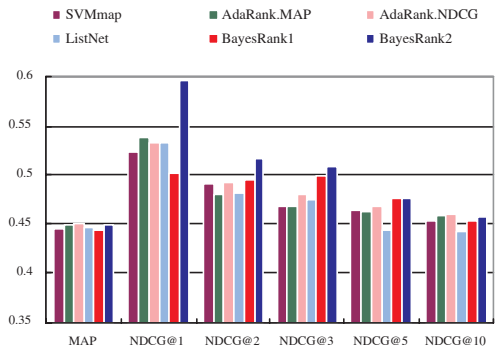

Figure 3: Ranking accuracy of various methods on OHSUMED (LETOR 3.0).

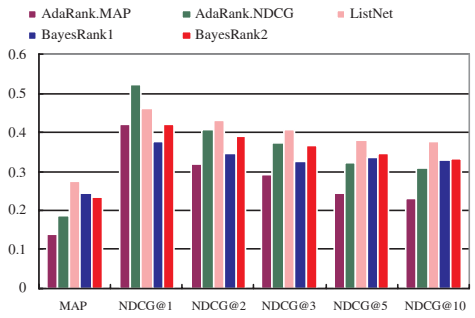

Figure 4: Ranking accuracy of various methods on TD2003 (LETOR 2.0).

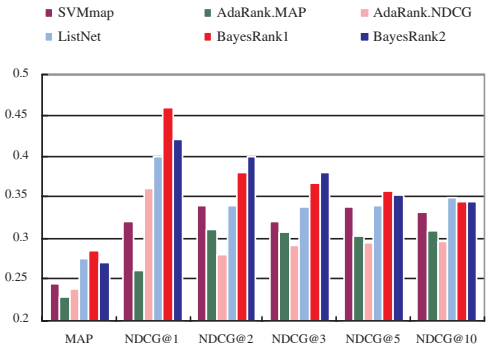

Figure 5: Ranking accuracy of various methods on TD2003 (LETOR 3.0).

pair in LETOR 3.0 has 45 features. The .Gov dataset was crawled in early 2002 and has been used as the data collection for TREC Web Track, which involves three research tasks: topic distillation (td), homepage finding (hp), and named page finding (np). The dataset contains 125 queries in total. In LETOR 3.0, for each query-document pair, 64 features are extracted for learning and testing.

The whole collection was created as a set of document-query pairs, each represented as a feature vector and a corresponding relevance judgment. In the TREC collections, each example is labeled as relevant or irrelevant. For OHSUMED examples, there are three possible labels: relevant, possibly relevant, and irrelevant. All datasets are partitioned for 5-fold cross-validation. In each trial, three of the subsets are used for training, one for validation, and the other for testing the performance of the trained model. The score reported is the average of the five folds.

6.2 Experiment Setup

We compared *Bayes*Rank with four popular listwise ranking algorithms, namely, AdaRank.MAP, AdaRank.NDCG [26], SVM^{map} [28], and ListNet [5]. The evaluation tools used in the experiments and the results of the baseline ranking algorithms are all available on the LETOR website[3]. The neural networks used as the scoring function for *Bayes*Rank have only one hidden layer, and the number of neurons in the hidden layer is tuned on the validation sets. The experiment results of *Bayes*Rank using NDCG@1 and NDCG@2 as training measures are denoted as *Bayes*Rank1 and *Bayes*Rank2 respectively.

6.3 Experiment Results

We use the abbreviations "L2" and "L3" to denote LETOR 2.0 and LETOR 3.0 respectively.

6.3.1 *Experiments on the OHSUMED Dataset*

Figures 2 and 3 show the results for the OHSUMED dataset. From Figure 2, we observe that all methods perform similarly. If we focus on the NDCG@1 measure, *Bayes*Rank1 outperforms the other methods on this dataset. However, surprisingly it performs worse than the other methods on L3. On the other hand, *Bayes*Rank2 achieves notable improvements consistently over the baseline methods. Note that there is no significant difference between the MAP measures of these methods.

6.3.2 *Experiments on the TD2003 Dataset*

Figures 4 and 5 show the results for the TREC2003 dataset. We observe that ListNet is the best method on L2, almost outperforming all the other algorithms except AdaRank.NDCG at the very top position. However, on L3, *Bayes*Rank yields a promising performance across every position compared to the other methods.

6.3.3 *Experiments on the TD2004 Dataset*

Figures 6 and 7 show the results for the TREC2004 dataset. Clearly, *Bayes*Rank2 achieves the best performance in this experiment. In terms of MAP, *Bayes*Rank obtains relative improvements of 4% and 14% over ListNet on L2 and L3 respectively. We also performed a significance test (*t-test*) on the improvements of *Bayes*Rank2 over the baseline algorithms on L2. As shown in Table 3, *Bayes*Rank2 achieves significant improvements. On L3, *Bayes*Rank1 performs as well as *Bayes*Rank2.

6.4 Discussion

The learning curve of *Bayes*Rank in terms of the expected NDCG and pairwise loss is shown in Figure 8. We observe that the pairwise loss is reversely correlated with the expected NDCG, which means that we can also reduce the

[3]http://research.microsoft.com/en-us/um/beijing/projects/letor/index.html

Table 3: The p-value of the *t-test* on the improvements of *Bayes*Rank over the baseline methods on TD2004

	MAP	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	NDCG@10
AdaRank.MAP	**0.023468**	0.187320	**0.009056**	**0.023454**	**0.010537**	**0.022297**	**0.007674**
AdaRank.NDCG	**0.000485**	0.058655	**0.001628**	**0.007598**	**0.003891**	**0.004865**	**0.000571**
ListNet	0.120390	0.310160	**0.029130**	0.082094	**0.040034**	**0.049598**	0.189950

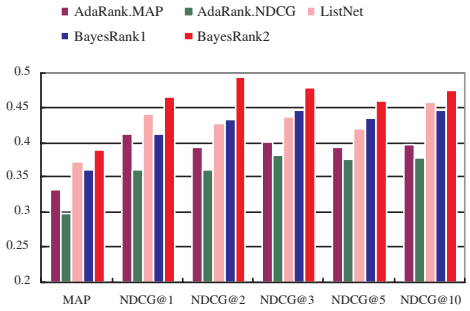

Figure 6: Ranking accuracy of various methods on TD2004 (LETOR 2.0).

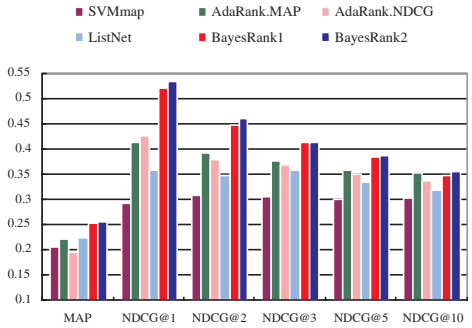

Figure 7: Ranking accuracy of various methods on TD2004 (LETOR 3.0).

pairwise loss effectively as the number of training iterations increases. The experiment results demonstrate that, in most cases, the proposed *Bayes*Rank framework is more effective than the compared listwise ranking algorithms, especially on the newly released LETOR 3.0 collection. The results also indicate that *Bayes*Rank1 is not as effective as *Bayes*Rank2. The observation implies that as the truncation level of NDCG increases, more information about this metric becomes available for learning. However, for preventing the over-fitting problem, the regularization might become an important issue as the truncation level increases.

7. CONCLUSIONS AND FUTURE WORK

We have proposed a learning framework, called *Bayes*Rank, for learning to rank from the Bayesian decision inference. The framework tries to minimize the expected *Bayes risk* over the training set and can be regarded as a direct optimization method for evaluation measures when the loss function is related to IR metrics. Experiment results show that *Bayes*Rank yields consistent improvements over baseline methods in most cases. Our contribution in this work is threefold: First, we propose a novel learning to rank framework from the Bayesian decision inference. The framework is fairly general and an arbitrary ranking model and loss function can be adopted; thus, it can be applied to other ranking problems. Second, we take the multi-layer perceptron neural networks as the ranking function and develop a listwise learning algorithm based on minimization of the expected *Bayes risk*. The effectiveness of the algorithm with the NDCG-based permutation-level loss is verified on the LETOR collections. Finally, we compare *Bayes*Rank with ListNet and PermuRank, and provide a new type of upper bound of the ranking error. As a result, in future research, it will be possible to develop new ranking models of soundness based on the proposed bounding function.

When considering non-positional dependent permutation-level losses, we may face a problem with the enormous of the hypothesized space of permutations. This computational issue exists in most listwise algorithms. In [27], the authors proposed keeping a small pool of permutations for training. Based on their technique, we can consider optimizing the MAP measure directly and evaluating the performance on the ad-hoc retrieval task with longer queries.

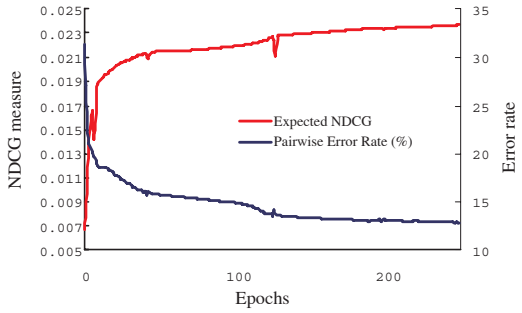

Figure 8: Expected NDCG vs. Pairwise loss

8. ACKNOWLEDGMENTS

This work was supported in part by Taiwan e-Learning and Digital Archives Program (TELDAP) sponsored by the National Science Council of Taiwan under Grant: NSC98-2631-001-013.

9. REFERENCES

[1] R. Bradley and M. Terry. Rank analysis of incomplete block designs. *Biometrika*, 39(3/4):324–345, 1952.

[2] C. Burges, R. Ragno, and Q. Le. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference*. MIT Press, 2007.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2006. ACM.

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

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[7] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[8] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[9] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, New York, NY, USA, 2000. ACM Press.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[11] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001. ACM.

[12] P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems 20*.

[13] T. Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[14] R. Luce. Individual Choice Behavior: A Theoretical Analysis. *New York*, 1959.

[15] C. MALLOWS. NON-NULL RANKING MODELS. I. *Biometrika*, 44(1-2):114–130, 1957.

[16] J. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall/CRC, 1995.

[17] R. Nallapati. Discriminative models for information retrieval. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, New York, NY, USA, 2004. ACM.

[18] R. Plackett. The analysis of permutations. *Applied Statistics*, 24(2):193–202, 1975.

[19] J. M. Ponte and B. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM Press.

[20] T. Qin, T. Liu, J. Xu, and H. Li. How to make letor more useful and reliable. In *Proceedings of SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[21] T. Qin, X. Zhang, M. Tsai, D. Wang, T. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing and Management*, 44(2):838–855, 2008.

[22] S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[23] G. Salton, editor. *Automatic text processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[24] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, New York, NY, USA, 2007. ACM.

[25] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, New York, NY, USA, 2008. ACM.

[26] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, USA, 2007. ACM.

[27] J. Xu, T. Y. Liu, M. Lu, H. Li, and W. Y. Ma. Directly optimizing evaluation measures in learning to rank. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114, New York, NY, USA, 2008. ACM.

[28] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, New York, NY, USA, 2007. ACM.

[29] C. Zhai and J. Lafferty. A risk minimization framework for information retrieval. *Information Processing and Management*, 42(1):31–55, 2006.

[30] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its

application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*.

APPENDIX

Proof of Theorem 1.

PROOF. $\tilde{\pi}_q$ is the ranked document list that possesses the maximal probability $p(\tilde{\pi}_q; q, \mathbf{g})$, which implies that

$$\tilde{\pi}_q = \arg\max_{\pi} p(\pi; q, \mathbf{g})$$

Since $l(\pi_q^*, \tilde{\pi}_q) \in [0, 1]$, the expected risk is bounded by

$$\frac{l(\pi_q^*, \pi_q^*)}{\#_{\Pi_q}} \leq R(\pi_q^*; q)$$

Therefore, $l(\pi_q^*, \pi_q^*)$ is upper bounded by $\#_{\Pi_q} \cdot R(\pi_q^*; q)$, and we obtain

$$\sum_q l(\pi_q^*, \pi_q^*) \leq \sum_q \#_{\Pi_q} \cdot R(\pi_q^*; q) \leq \kappa \cdot R$$

where

$$\kappa = \max_q \#_{\Pi_q}$$

□

Proof of Theorem 3.

PROOF. Since an exponential function is monotonically increasing and for all $\pi \in \Pi_q$,$\mathcal{F}(\pi; q) \leq \mathcal{F}(\pi^*; q)$, we have

$$\begin{aligned} &\forall \pi \in \Pi_q, \\ l(\pi_q^*, \tilde{\pi}_q) &\leq l(\pi_q^*, \tilde{\pi}_q) \frac{\sum_{\pi'} \exp\left(\mathcal{F}(\pi^*; q)\right)}{\sum_{\pi'} \exp\left(\mathcal{F}(\pi'; q)\right)} \\ &\leq \max_{\pi \in \Pi_q} l(\pi_q^*, \pi) \frac{\sum_{\pi'} \exp\left(\mathcal{F}(\pi; q)\right)}{\sum_{\pi'} \exp\left(\mathcal{F}(\pi'; q)\right)} \end{aligned}$$

□