

# FRank: A Ranking Method with Fidelity Loss

Ming-Feng Tsai<sup>1\*</sup>, Tie-Yan Liu<sup>2</sup>, Tao Qin<sup>3</sup>, Hsin-Hsi Chen<sup>1</sup>, Wei-Ying Ma<sup>2</sup>

<sup>1</sup>Department of Computer  
Science and Information  
Engineering  
National Taiwan University  
Taipei 106, Taiwan  
mftsai@nlg.csie.ntu.edu.tw  
hhchen@ntu.edu.tw

<sup>2</sup>Microsoft Research Asia  
4F, Sigma Center, No. 49,  
Zhichun Road, Haidian  
District,  
Beijing 100080, China  
{tyliu, wyma}@microsoft.com

<sup>3</sup>Department of Electronic  
Engineering,  
Tsinghua University,  
Beijing 100084, China  
tsintao@gmail.com

## ABSTRACT

Ranking problem is becoming important in many fields, especially in information retrieval (IR). Many machine learning techniques have been proposed for ranking problem, such as RankSVM, RankBoost, and RankNet. Among them, RankNet, which is based on a probabilistic ranking framework, is leading to promising results and has been applied to a commercial Web search engine. In this paper we conduct further study on the probabilistic ranking framework and provide a novel loss function named fidelity loss for measuring loss of ranking. The fidelity loss not only inherits effective properties of the probabilistic ranking framework in RankNet, but possesses new properties that are helpful for ranking. This includes the fidelity loss obtaining zero for each document pair, and having a finite upper bound that is necessary for conducting query-level normalization. We also propose an algorithm named FRank based on a generalized additive model for the sake of minimizing the fidelity loss and learning an effective ranking function. We evaluated the proposed algorithm for two datasets: TREC dataset and real Web search dataset. The experimental results show that the proposed FRank algorithm outperforms other learning-based ranking methods on both conventional IR problem and Web searching.

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Experimentation, Theory

\*This work was conducted when the first author was visiting student at Microsoft Research Asia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.  
Copyright 2007 ACM 978-1-59593-597-7/07/0007 ...\$5.00.

## Keywords

FRank, Fidelity Loss, Learning to Rank

## 1. INTRODUCTION

In this information explosion era, accurately locating relevant information based on user information needs has become important. The IR problem can be formulated as a ranking problem: provided a query and a set of documents, an IR system should provide a ranked list of documents. Several methods have been proposed for solving the problem in the literature, such as Boolean model, vector space model, probabilistic model, and the language model; these methods can be regarded as empirical IR methods.

In addition to traditional IR approaches, machine learning techniques are becoming more widely used for the ranking problem of IR; the learning-based methods aim to use labeled data for learning an effective ranking function. There are several different ways to use learning-based methods for solving IR problems. One may regard IR as a binary classification problem in which documents are categorized as relevant or irrelevant. However, for real Web search engines, pages cannot simply be classified into two types relative to user information needs; they should have multiple grades, such as highly relevant, partially relevant, definitely irrelevant, and so on. Therefore, several learning-based methods, such as RankBoost [8], RankSVM [12], and RankNet [4], view the problem as a ranking problem. These methods mainly construct pairs between documents and use machine learning techniques for minimizing the number of mis-ordered pairs.

According to [4], the probabilistic ranking framework has many effective properties for ranking, such as pair-wise differentiable loss, and can better model multiple relevance levels; the proposed RankNet algorithm performs well in practice and has been used in a commercial Web search engine. However, the loss function in RankNet still has some problems: for instance, it has no real minimal loss and appropriate upper bound. These problems may cause the trained ranking function inaccurate and the training process biased by some hard pairs. Therefore, we conduct further studies on the probabilistic ranking framework, and propose a novel loss function named fidelity loss. The fidelity loss is inspired by the concept of Fidelity [14], which is widely used for measuring the difference between two states of a quantum

in the field of physics [2]. The fidelity loss function inherits those properties of the probabilistic ranking framework, and has several additional useful characteristics for ranking. For instance, the fidelity loss can achieve zero and is bounded between 0 and 1 for each document pair; this property is necessary for conducting query-level normalization.

For the sake of efficiently minimizing the fidelity loss function, we further propose a ranking algorithm based on a generalized additive model. Our ranking algorithm named Fidelity Rank (FRank) combines the probabilistic ranking framework with the generalized additive model. We evaluated the proposed FRank algorithm on both TREC dataset and a large-scale Web search dataset, which encompassed 19,600 queries collected from a commercial Web search engine. The experimental results show that the proposed approach outperforms other ranking methods on both conventional IR problem and Web searching, and that the improvements are statistically significant.

The remainder of this paper is organized as follows. We briefly review the previous research in Section 2. In Section 3, we revisit the probabilistic ranking framework and the fidelity measure, and describe the FRank algorithm. Section 4 reports experimental results and discusses the relationship between different methods. We conclude our paper and discuss future work in Section 5.

## 2. RELATED WORK

Many machine learning technologies [1, 4, 5, 6, 8, 12, 13] have been studied for IR. In [13], Nallapati treated IR as a binary classification problem that directly classifies a document as relevant or irrelevant with respect to the given query. However, in real Web searching, a page has multiple relevance levels, such as highly relevant, partially relevant, and definitely irrelevant. Therefore, some studies regard IR as a ranking problem: highly relevant web pages are ranked higher than partially relevant ones, and partially relevant ones are above irrelevant ones. In the literature, many learning-based ranking algorithms are proposed; typical examples include RankBoost, RankSVM, and RankNet.

Freund et al. [8] used the Boosting approach for learning a ranking function and proposed RankBoost to solve the problem of combining preferences. Their method aims to minimize the weighted number of pairs of instances that are mis-ordered by the final ranking function. The algorithm can be summarized as follows. For each round  $t$ , RankBoost chooses  $\alpha_t$  and the weak learner  $h_t$  so as to minimize the pair-wise loss, which is defined by

$$Z_t = \sum_{(x_i, x_j)} D_t(x_i, x_j) \exp(\alpha_t(h_t(x_i) - h_t(x_j))),$$

where  $x_i$  is the instance ranked higher than  $x_j$ ,  $D_t(x_i, x_j)$  is the weight of pair  $(x_i, x_j)$ . After the optimal weak learner  $h_t$  is selected, the weight  $D_t(x_i, x_j)$  is adjusted with respect to  $\alpha_t$  and  $h_t$ . The rule of adjustment is to decrease the weight of pairs if  $h_t$  gives a correct ranking ( $h_t(x_i) > h_t(x_j)$ ) and increase otherwise. Finally, this algorithm outputs the ranking function by combining selected weak learners. Owing to the greedy search nature of Boosting, RankBoost can be implemented in parallel and thus scale up to a large-scale dataset.

Thorsten Joachims [12] proposed RankSVM algorithm, which uses Support Vector Machines for optimizing search

performance using click-through data. RankSVM aims to minimize the number of discordant pairs, which is similar to RankBoost, and to maximize the margin of pair. The margin maximization is equal to minimize the  $L_2$ -norm of hyperplane parameter  $\vec{w}$ . Given a query, if the ground truths assert that document  $d_i$  is more relevant than  $d_j$ , the constraint of RankSVM is  $\vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j)$ , where  $\Phi(q, d)$  is the feature vector calculated from document  $d$  relative to query  $q$ . Then, the ranking problem can be expressed as the following constrained optimization problem.

$$\begin{aligned} \min : V(\vec{w}, \vec{\xi}) &= \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \\ \text{s.t.} \quad &\begin{cases} \forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ \forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\ \forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \end{cases} \end{aligned}$$

From a theoretical perspective, RankSVM is well-founded in the structure risk minimization framework, and is verified in a controlled experiment. Moreover, the advantage of RankSVM is that it needs no human-labeled data, and can automatically learn the ranking function from click-through data.

Burges et al. [4] proposed a probabilistic ranking framework and used neural network for minimizing a pair-wise differentiable loss in a method named RankNet. Assume that the ranking model is  $f : R^d \mapsto R$ , so that the rank order of a set of samples is specified by the real value that the model takes. Therefore,  $f(d_i) > f(d_j)$  is taken to mean that the model asserts that  $d_i \succ d_j$ . Given a set of pairs of training samples  $(d_i, d_j)$  together with the target probability  $P_{ij}^*$  of  $d_i$  being ranked higher than  $d_j$ , a logistic function can be used for mapping the output of ranking function to a probability as follows:

$$P_{ij} = \frac{e^{o_{ij}}}{1 + e^{o_{ij}}},$$

where  $o_{ij} = f(d_i) - f(d_j)$ , and  $P_{ij} = P(d_i \succ d_j)$ . Then, cross entropy is adopted as a loss function for training, which can be represented as:

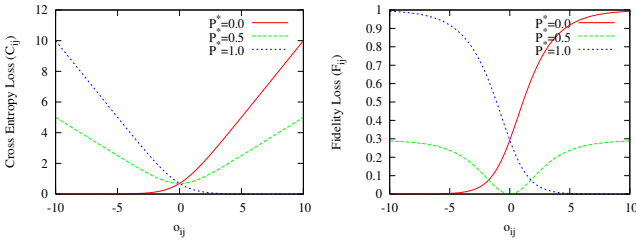
$$\begin{aligned} C_{ij} &\equiv C(o_{ij}) \\ &= -P_{ij}^* \log P_{ij} - (1 - P_{ij}^*) \log(1 - P_{ij}) \\ &= -P_{ij}^* o_{ij} + \log(1 + e^{o_{ij}}), \end{aligned}$$

where  $C_{ij}$  is the cross entropy loss of a pair  $(d_i, d_j)$ ,  $P_{ij}^*$  is the desired probability, and  $P_{ij}$  is the modeled probability.

RankNet uses back-propagation neural network for minimizing the criterion during training process, and then learn a ranking function. As compared to RankBoost and RankSVM, the loss function in RankNet is pair-wise differentiable, which can be regarded as an advantage in cases in which ground truths come from several annotators that may disagree. So, RankNet performs well in practice and has been successfully applied to a commercial search engine.

## 3. FIDELITY RANK

Inspired by the success of RankNet, we investigate the probabilistic ranking framework and propose a novel loss function named fidelity loss. In this section we briefly describe the probabilistic ranking framework, and discuss the fidelity loss and its useful properties. We also present a ranking algorithm named FRank for minimizing the loss and learning an effective ranking function.



**Figure 1: Loss Function: (left) Cross Entropy Loss Function; (right) Fidelity Loss Function.**

### 3.1 Probabilistic Ranking Framework

According to [4], the mapping from the output of ranking function to a probability based on the probabilistic ranking framework has the following properties:

1. The ranking model puts consistency requirements on the  $P_{ij}$  (for example, if  $P(d_i \succ d_j) = 0.5$  and  $P(d_j \succ d_k) = 0.5$ , then  $P(d_i \succ d_k) = 0.5$ );
2. Any set of adjacency posteriors can uniquely identify a target posterior  $0 \leq P_{ij}^* \leq 1$  for each pair  $(d_i, d_j)$ ;
3. The expected strengthening (or weakening) of confidence in the ordering of a given pair can be held (for instance, if  $P(d_i \succ d_j) = 0.6$  and  $P(d_j \succ d_k) = 0.6$ , then  $P(d_i \succ d_k) > 0.6$ ).

When the outputs are mapped to probabilities, the general measure of probability distribution can be applied as the criterion for training, such as cross entropy, KL-divergence, and information radius. In RankNet [4], cross entropy is adopted for this purpose; the loss function of a pair can be represented by  $C_{ij} = -P_{ij}^* o_{ij} + \log(1 + e^{o_{ij}})$ , which is shown in Figure 1 (left).

The cross entropy loss function provides a principal way to make the samples have the same rank of ground truths. Previous works show that the loss function is effective and the corresponding RankNet algorithm performs well in practice. However, if we carefully investigate Figure 1 (left), we will find there are still some problems within this loss function.

- The cross entropy loss function cannot achieve the real minimal loss, zero, except for the pair with posterior probability is 0 or 1; this problem may cause the trained ranking function inaccurate.
- The cross entropy loss of a pair has no upper bound; in other words, this may cause the training procedure to become biased by some hard pairs.
- The query-level normalization is hard to define in cross entropy loss; this may also cause the whole model dominated by those queries with a large number of document pairs.

Considering these problems, it is meaningful to investigate the other measures of probability distribution that can provide better properties for ranking on the probabilistic ranking framework. This is a key point in this paper.

### 3.2 Fidelity Loss Function

After investigating many different measures of probability distributions such as KL-divergence and information radius, we find that an optimal candidate for the criterion of ranking problem is Fidelity [14]. Fidelity is originally a distance metric in physics to measure the difference between two states of a quantum. The fidelity of two probability distributions is defined by

$$F(p_x, q_x) \equiv \sum_x \sqrt{p_x q_x},$$

where  $\{p_x\}$  and  $\{q_x\}$  are the probability distributions. Note that when the distributions  $\{p_x\}$  and  $\{q_x\}$  are identical,  $F(p_x, q_x) = \sum_x p_x = 1$ . A better geometric understanding of the fidelity is that the fidelity is the inner product between vectors with components  $\sqrt{p_x}$  and  $\sqrt{q_x}$ , which lie on a unit sphere.

Fidelity is a useful quantity for mathematical purposes and is a clear physical interpretation. Moreover, it is a meaningful measure candidate for the loss function of probabilistic ranking framework. We adapt the fidelity measure so that the loss of a pair is measured by

$$F_{ij} \equiv F(o_{ij}) = 1 - (\sqrt{P_{ij}^* \cdot P_{ij}} + \sqrt{(1 - P_{ij}^*) \cdot (1 - P_{ij})}),$$

where  $P_{ij}^*$  is the given target value for the posterior probability and  $P_{ij}$  is the modeled probability; then, after the logistic function is adopted,  $F_{ij}$  becomes

$$F_{ij} = 1 - \left( \left[ P_{ij}^* \times \left( \frac{e^{o_{ij}}}{1 + e^{o_{ij}}} \right) \right]^{\frac{1}{2}} + \left[ (1 - P_{ij}^*) \times \left( \frac{1}{1 + e^{o_{ij}}} \right) \right]^{\frac{1}{2}} \right).$$

Figure 1 (right) plots  $F_{ij}$  as a function of  $o_{ij}$  for three values  $P_{ij}^* = \{0, 0.5, 1\}$ . If  $P_{ij}^* = 0.5$ , it means that there is no information available as to the relative rank of the two samples, then  $F_{ij}$  has its minimum at the origin. Note that the loss of this pair can obtain zero. This also gives us a principal way of training samples that we desire to have the same rank as the ground truth. Actually, the fidelity loss inherits all three properties of the probabilistic ranking framework in RankNet. In addition, it also has new properties that are helpful for ranking. We describe these properties in detail as follows.

1. The fidelity loss is capable of obtaining the real minimal loss for each desired probability  $P_{ij}^*$  of pairs. Unlike the cross entropy, the fidelity loss function has the zero loss for each document pair. Therefore, this new characteristic makes the trained ranking function more accurate.
2. The fidelity loss of a pair is bounded between 0 and 1. If the loss of a pair has no appropriate upper bound, the loss of some hard pairs would be too much when they continue to be placed in the wrong position. This may cause the performance deteriorates because the whole model is biased by these hard pairs. In this sense, fidelity loss is superior to cross entropy.
3. The fidelity loss of a query can easily be considered by means of dividing the number of pairs in the given query. This query-level normalization is more consistent with the evaluations of IR, such as MAP and NDCG, because they are mostly based on query level.

The fidelity loss of all queries, therefore, can be defined as follows:

$$\sum_q \frac{1}{|\#q|} \sum_{ij} F_{ij},$$

where  $|\#q|$  is the number of pairs for query  $q$  and  $F_{ij}$  is the fidelity loss of a pair. In this way, each query contributes equally to the total loss in the training process. Therefore, the model is not dominated by those queries with a large number of document pairs.

Although the third property of query-level fidelity loss looks natural, the similar extension for the previous methods is non-trivial. Each query is not guaranteed to contribute equally to the total loss since their loss function do not have an appropriate upper bound. With the new properties, we regard our proposed fidelity loss as a more suitable measure of ranking loss. In the next subsection, we discuss how to efficiently minimize this loss function so as to learn an effective ranking function.

### 3.3 Algorithm Derivation

Considering the efficiency and scalability for large-scale datasets, we choose a generalized additive model, which is similar to Boosting, for minimizing the fidelity loss. The primary consideration is that the additive model has great potential to implement in parallel since the examination of features is independent in the training process.

In the additive model, the final ranking function is defined as:

$$H(x) = \sum_t \alpha_t h_t(x),$$

where  $h_t(x)$  is a weak learner and  $\alpha_t$  is the combination coefficient of the weak learner, and  $t$  is the index of iterations. Consider the ranking function after the  $k$ -th iteration,

$$H_k(x) = \sum_{t=1}^k \alpha_t h_t(x) \quad \text{or} \quad H_k(x) = H_{k-1}(x) + \alpha_k h_k(x).$$

Then, the fidelity loss of  $H_k(x)$  over all queries is

$$J(H_k) = \sum_q \frac{1}{|\#q|} \sum_{ij} F_{ij}^{(k)},$$

where

$$F_{ij}^{(k)} = 1 - \left[ P_{ij}^* \times \left( \frac{e^{H_k(x_i) - H_k(x_j)}}{1 + e^{H_k(x_i) - H_k(x_j)}} \right) \right]^{\frac{1}{2}} - \left[ (1 - P_{ij}^*) \times \left( \frac{1}{1 + e^{H_k(x_i) - H_k(x_j)}} \right) \right]^{\frac{1}{2}}.$$

Given  $H_k(x) = H_{k-1}(x) + \alpha_k h_k(x)$  and denoted

$$D(i, j) = \frac{1}{|\#q|}, \quad (1)$$

the fidelity loss  $J(H_k)$  can be formulated as

$$\sum_{ij} D(i, j) \times \left( \begin{aligned} &1 - \left[ P_{ij}^* \times \left( \frac{e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}}{1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}} \right) \right]^{\frac{1}{2}} \\ &- \left[ (1 - P_{ij}^*) \times \left( \frac{1}{1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}} \right) \right]^{\frac{1}{2}} \end{aligned} \right), \quad (2)$$

where  $H_{k-1}^{i,j} \triangleq H_{k-1}(x_i) - H_{k-1}(x_j)$  and  $h_k^{i,j} \triangleq h_k(x_i) - h_k(x_j)$ .

Setting the derivative of Equation (2) with respect to  $\alpha_k$  to zero, we have the equation as follows:

$$\sum_{ij} D(i, j) \times \left( \begin{aligned} &-\frac{1}{2} \left[ P_{ij}^* \times \left( \frac{e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}}{1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}} \right) \right]^{\frac{-1}{2}} \\ &\times \left[ P_{ij}^* \times \frac{h_k^{i,j} \cdot e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}}{(1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}})^2} \right] \\ &-\frac{1}{2} \left[ (1 - P_{ij}^*) \times \left( \frac{1}{1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}} \right) \right]^{\frac{-1}{2}} \\ &\times \left[ (1 - P_{ij}^*) \times \frac{-h_k^{i,j} \cdot e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}}}{(1 + e^{H_{k-1}^{i,j} + \alpha_k h_k^{i,j}})^2} \right] \end{aligned} \right) = 0.$$

For a general weak learner  $h_k(x)$ , solving the close form of  $\alpha_k$  from the above equation is quite difficult. For simplicity, we adopt the same choice of weak learners as in [8], which are binary weak learners, to construct the final ranking function. When a binary weak learner is introduced, the above equation can be simplified because  $h_k^{i,j}$  only takes values -1, 0, and 1. Therefore, the equation can be expressed by

$$\begin{aligned} &\sum_{h_k^{i,j}=1} D(i, j) \left( \frac{(P_{ij}^* e^{H_{k-1}^{i,j}})^{\frac{1}{2}} e^{-\alpha_k}}{(e^{-\alpha_k} + e^{H_{k-1}^{i,j}})^{\frac{3}{2}}} - \frac{e^{H_{k-1}^{i,j}} (1 - P_{ij}^*)^{\frac{1}{2}} e^{-\alpha_k}}{(e^{-\alpha_k} (e^{-\alpha_k} + e^{H_{k-1}^{i,j}})^3)^{\frac{1}{2}}} \right) \\ &= \sum_{h_k^{i,j}=-1} D(i, j) \left( \frac{(P_{ij}^* e^{H_{k-1}^{i,j}})^{\frac{1}{2}} e^{\alpha_k}}{(e^{\alpha_k} + e^{H_{k-1}^{i,j}})^{\frac{3}{2}}} - \frac{e^{H_{k-1}^{i,j}} (1 - P_{ij}^*)^{\frac{1}{2}} e^{\alpha_k}}{(e^{\alpha_k} (e^{\alpha_k} + e^{H_{k-1}^{i,j}})^3)^{\frac{1}{2}}} \right). \end{aligned}$$

With further derivations and relaxations, we eventually obtain

$$\alpha_k = \frac{1}{2} \ln \frac{\sum_{h_k^{i,j}=1} W_{i,j}}{\sum_{h_k^{i,j}=-1} W_{i,j}}, \quad (3)$$

where

$$W_k^{i,j} = D(i, j) \left( \frac{(P_{ij}^* e^{H_{k-1}^{i,j}})^{\frac{1}{2}} - e^{H_{k-1}^{i,j}} (1 - P_{ij}^*)^{\frac{1}{2}}}{(1 + e^{H_{k-1}^{i,j}})^{\frac{3}{2}}} \right). \quad (4)$$

Consequently, the algorithm operates in this way. In the  $k$ -th iteration, a new weak learner  $h_k(x)$  with the minimal fidelity loss  $J(H_k)$  is obtained according to the current pair weight  $W_k^{i,j}$ , and then it is combined with the previous ones using the combination coefficient  $\alpha_k$ . In a stepwise manner, we eventually can obtain the final ranking function  $H(x)$ . We name this algorithm Fidelity Rank (FRank), for which details are summarized in Algorithm 1.

## 4. EXPERIMENTS

### 4.1 Evaluation Measures

Precision is an IR performance measure that quantifies the fraction of retrieved documents which are known to be relevant.  $P@n$  is capable of measuring the accuracy within top  $n$  results of the returned rank list for a query.

$$P@n = \frac{\# \text{ of relevant docs in top } n \text{ results}}{n}$$

In general, the present Web search engines display 10 returned documents for each page and many users only browse the results in the first page. Therefore, we use precision within ten returned documents for evaluating the performance of each ranking algorithm.

---

**Algorithm 1** Learning Algorithm of FRank

---

**Input:** Pairs over all training queries and weak learner candidates  $h_c(x)$ ,  $1 \leq c \leq m$ , where  $m$  is the number of features

**Initialization:** Pair weight by Eq. (1) and the number of weak learners  $k$

**for**  $t = 1$  to  $k$  **do**

**for**  $c = 1$  to  $m$  **do**

    Calculate  $\alpha_{t,c}$  by Eq. (3)

    Calculate fidelity loss over all queries by Eq. (2)

**end for**

$h_t(x) \leftarrow h_{t,c}(x)$  with the minimal fidelity loss

$\alpha_t \leftarrow$  the corresponding  $\alpha_{t,c}$

  Pair weight is updated by Eq. (4)

**end for**

**Output:**  $H(x) = \sum_{t=1}^k \alpha_t h_t(x)$

---

In addition, MAP [17, 18] is also considered as evaluation metric for evaluating ranking methods. Given a query  $q_i$ , its average precision (AP) is calculated as:

$$AP_i = \frac{\sum_{j=1}^N (P(j) \times pos(j))}{\# \text{ of relevant docs in } q_i}$$

where  $N$  is the number of documents retrieved,  $P(j)$  is the precision value at position  $j$ , and  $pos(j)$  is a binary function to indicate whether the document at position  $j$  is relevant. Then, we obtain MAP by use of averaging the AP values of all the queries.

Considering that the web search dataset has multiple rating grades, we also use the normalized discount cumulative gain (NDCG) [9, 10] to evaluate the performance of ranking algorithms. For a given query, the NDCG value for a ranked document at position  $i$  is computed as follows:

1. Compute the gain  $2^{r(j)} - 1$  of each document  $j$ ;
2. Discount the gain of each document  $j$  by its ranked position, which can be expressed by  $\frac{(2^{r(j)} - 1)}{\log(1+j)}$ ;
3. Cumulate the discounted gain for document  $j$  at position  $i$ , which can be express by  $\sum_{j=1}^i \frac{(2^{r(j)} - 1)}{\log(1+j)}$ ;
4. Normalize the discounted cumulative gain for document  $j$  at position  $i$  using  $n_i \sum_{j=1}^i \frac{(2^{r(j)} - 1)}{\log(1+j)}$ ,

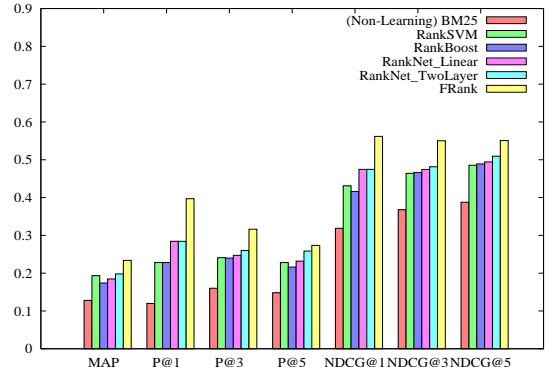
where  $r(j)$  is the rating of the  $j$ -th document in the ordered list, and the normalization constant  $n_i$  is chosen so that a perfect ordering gets NDCG value 1. We use NDCG within ten returned documents to evaluate the performance in the following experiments.

## 4.2 Comparison Methods

Three learning-based ranking algorithms are compared in this study: RankBoost [8], RankNet [4], and RankSVM [12]. For RankBoost, we implemented binary weak learner whose output is 0 and 1. For RankNet, we used linear and two-layer perceptrons, and those are denoted as RankNet\_Linear and RankNet\_TwoLayer. For RankSVM, we directly used the binary code of SVMlight [11]. In addition, non-learning based IR model, BM25 [18], was utilized as a baseline for comparison with conventional IR approach.

**Table 1: The extracted features of TREC dataset**

Feature Name	Number of Features
BM25 [18]	1
MSRA1000 [19]	1
PageRank [15]	1
HostRank [20]	1
Relevance Propagation [16]	10



**Figure 2: Experimental Results on TREC Dataset**

## 4.3 Experiment on TREC Dataset

TREC dataset is crawled from the .gov domain in early 2002, and has been used as the data collection of Web Track since TREC 2002. There are totally 1,053,110 pages with 11,164,829 hyperlinks in it. When conducting the experiments on this corpus, we used the topic distillation task in the Web Track of TREC 2003 [7] as query set, which has 50 queries in total. The ground truths of this task are provided by the TREC committee with binary judgment: relevant and irrelevant. The number of relevant pages for each query spans from 1 to 86. There are 14 features extracted from TREC dataset for training, as listed in Table 1.

Four-fold cross validation is conducted in this experiment owing to the size of TREC dataset. The experimental results are plotted in Figure 2 in which the values are averaged scores over the four trials. The proposed algorithm, FRank, is a relatively effective ranking algorithm, as observed from a comparison with other ranking algorithms in Figure 2. All learning-based methods outperform BM25; our proposed FRank algorithm even obtains about 40% improvement over non-learning based method, BM25, in MAP. This would indicate that learning to rank is a promising direction in IR.

The proposed FRank algorithm gains more improvement at the top position of retrieved results, especially precision at position 1, as indicated from Figure 2. This result indicates that the proposed method is also suitable for Web searching, which emphasizes the top position of result. Therefore, an experiment on real Web search is conducted in the following subsection.

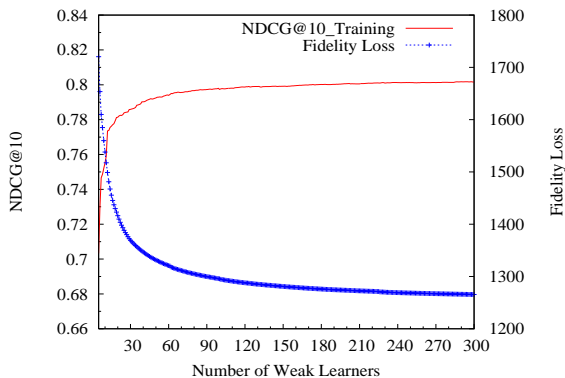
## 4.4 Experiment on Web Search Dataset

### 4.4.1 Web Search Dataset

The dataset consists of 19,600 queries of more than 3,300,000 web pages within a commercial Internet search engine. These web pages are partially labeled with ratings from 1 (irrele-

**Table 2: Details of Web Search Datasets**

	Number of queries	Number of docs
Training dataset	12,000	385,293
Validation dataset	3,800	663,457
Testing dataset	3,800	693,180

**Figure 3: Training Process of FRank on Web Search Training Dataset**

vant) to 5 (highly relevant) by human annotators. Because the dataset is too large to label completely, the unlabelled pages are given the rating 0. For a given query, a page is represented by query-dependent (e.g., term frequency) and query-independent (e.g., PageRank) features extracted from the page itself and the preprocessed indices. The total number of features is 619. Considering that these features reflect the business secrets of that search engine company, we would not describe them in detail. However, this does not influence our experimental study since all the methods under investigation operate on the same feature set.

For ease of experimental study, we divide the dataset into three sets: 12,000 queries for training, 3,800 queries for validation, and 3,800 queries for testing. Some unlabeled pages would be relevant, so that performance descends if we use the whole unlabeled pages for training. Therefore, 20 unlabeled documents are randomly selected as the documents with rating 0 for training. So, our training dataset has 385,293 documents; it totally contains 2,635,976 pairs for training. Table 2 summarizes the details of the training, validation, and testing datasets.

#### 4.4.2 Training Process of FRank

The training process is shown in Figure 3, in which the number of weak learners starts from 5. The proposed FRank algorithm indeed decreases the fidelity loss with the increasing number of weak learners, and at the same time it increases the value of NDCG@10, as illustrated in Figure 3. These values change dramatically in top selected 30 weak learners, and then they vary smoothly in the following weak learners.

#### 4.4.3 Performance Comparisons

Only three experimental results of FRank, RankBoost, and RankNet on Web Search dataset are reported, since RankSVM runs out of memory in this dataset. After training, the validation dataset is used for selecting the best parameter setting for each method, such as the number of

weak learners of FRank and RankBoost, and the number of epochs of RankNet. This procedure is taken to guarantee the better generalization of these ranking methods. Here, we used NDCG@10 as the criterion for selecting the parameter setting.

Figure 4 plots the NDCG@10 curves of FRank, RankBoost, and RankNet on validation data. FRank outperforms RankBoost for each weak learner, as demonstrated in Figure 4 (left) where the number of weak learners starts from 10. RankBoost performs worse when the number of weak learners is few than 20; however, when the number of weak learners is over 20, RankBoost eventually obtains better performance. This is because the power of representation of RankBoost is quite limited with few weak learners. Accordingly, we set the number of weak learners as 224 for FRank and 271 for RankBoost, as observed from the middle of flat tail in Figure 4 (left).

In Figure 4 (right), RankNet\_TwoLayer is observed to be more effective than RankNet\_Linear; however, its performance seems sensitive to the validation dataset. The performance of RankNet\_TwoLayer dropped when the number of epochs was more than 10; this would be the problem of overfitting. In contrast, ranking algorithms, such as FRank and RankBoost, based on the generalized additive model are more robust against the problem of overfitting; this robustness is also an essential characteristic of the generalized additive model. We set the number of epochs as 25 for RankNet\_Linear and 9 for RankNet\_TwoLayer, as observed from Figure 4 (right).

After setting the parameter of models, we use the settings to examine the performance of the ranking algorithms on the testing dataset. Standard IR model [3, 17, 18], i.e., BM25, is also conducted for a comparison of learning-based and non-learning based methods. In addition, an empirical approach of using linear combination of BM25 and PageRank is also performed, where the parameter is 0.8 for BM25 and 0.2 for PageRank (i.e.,  $0.8 \times \text{BM25} + 0.2 \times \text{PageRank}$ ); this parameter is obtained after tuning.

Figure 5 plots NDCG values from 1 to 10 of ranking algorithms on the testing dataset. Simply using conventional IR model is inadequate for Web searching, as indicated from a comparison of BM25 with those learning-based methods. This is because the learning-based methods are capable of leveraging various features in a large-scale Web search dataset. The proposed FRank algorithm is relatively more effective than the other learned-based ranking algorithms from NDCG@1 to NDCG@10; RankNet\_TwoLayer also performs well on this large-scale dataset. These results indicate that the probabilistic ranking framework is a suitable framework for learning to rank. Although both FRank and RankNet are based on the framework, FRank is capable of ranking more accurately than RankNet; this is consistent with the discussion about the superiority of fidelity loss in Section 3.2. Moreover, the FRank algorithm performs well at the top position of ranking lists, as observed from a comparison of NDCG@1 with RankNet\_TwoLayer in Figure 5.

A significance test for FRank and RankNet\_TwoLayer with a confidence level of 98% is performed for verifying whether the improvement is statistically significant. The corresponding  $p$ -values are 0.0114 for NDCG@1, 0.007 for NDCG@5, and 0.0056 for NDCG@10. This result indicates that, as to Web searching, FRank is significantly better than RankNet, and also significantly outperforms other ranking algorithms.

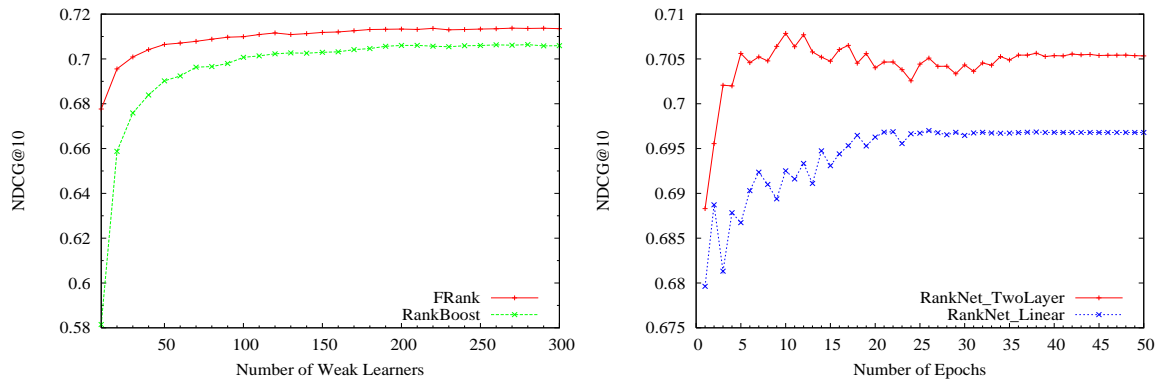


Figure 4: Experimental Results on Web Search Validation Dataset (left) FRank and RankBoost, (right) RankNet with Linear and Two-Layer Perceptrons

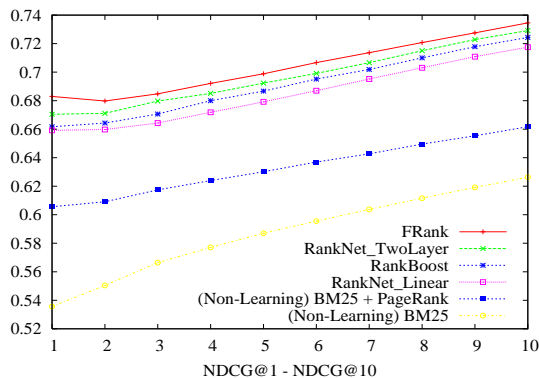


Figure 5: Experimental Results of Ranking Algorithms on Web Search Testing Dataset

#### 4.4.4 Experiment on Different Size of Training Dataset

Considering RankSVM ran out of memory on the Web search training dataset, we further conducted several experiments on smaller-scale training datasets to provide more insight about RankSVM. In addition, investigating how the number of training queries affects performance is also essential for Web searching. For this purpose we separately trained these referenced ranking algorithms on 1,000, 2,000, 4,000, 8,000, and 12,000 queries. However, RankSVM ran out of memory with 8,000 queries in these experiments. This is mainly because RankSVM has to construct as many constraints as the number of document pairs, and then the number of variables in the dual problem becomes enormous when training on large-scale dataset. Linear kernel is used for those case that RankSVM can operate upon, and the parameter  $c$  of RankSVM is tuned on the validation dataset, which is similar to the procedure in Section 4.4.3.

Figure 6 plots the results of NDCG@10 for each ranking algorithm trained on different numbers of queries; Table 3 also summaries these results. We have several observations as follows.

- The FRank algorithm performs effectively in small training dataset since it introduces query-level normalization in the fidelity loss function. However, the performances of other ranking algorithms are relatively

Table 3: NDCG@10 of Ranking Algorithms on the Different Size of Training Dataset

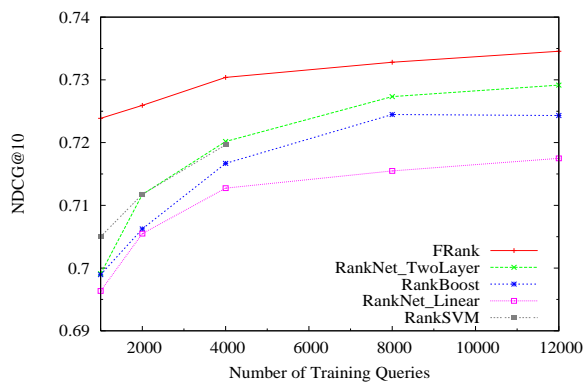
	Number of Training Queries				
	1,000	2,000	4,000	8,000	12,000
FRank	0.723	0.725	0.730	0.732	0.734
RankNet TwoLayer	0.699	0.711	.720	0.727	0.729
RankNet Linear	0.696	0.705	.712	0.715	0.717
RankBoost	0.698	0.706	0.716	0.724	0.723
RankSVM	0.704	0.711	0.719	None	None

worse when the number of training queries is 1,000; this is because the models are statistically incomplete with few training queries.

- Using more training data does not guarantee improvement on performance. As illustrated in Figure 6, performance increases when the number of training queries is added. However, when the number of queries is more than 8,000, the performance only slightly improves, and sometimes even decreases (e.g., RankBoost).
- RankSVM performs as well as RankNet\_TwoLayer when the number of training queries is small. Therefore, we consider that if the scalability issue of RankSVM can be fixed, it may be an effective candidate for learning to rank in Web searching.
- The probabilistic ranking framework performs well when the amount of training data is large. This is because more pair-wise information is capable of making the trained ranking function more accurate.
- FRank outperformed other ranking algorithms for all cases, and it was also more stable with respect to the number of training queries. These results suggest that FRank is more suitable for learning to rank in Web searching.

## 5. CONCLUSIONS

This paper presents an approach for learning to rank with the goal of improving the accuracy of conventional IR and Web searching. On the basis of the probabilistic ranking framework, we propose a novel loss function named fidelity



**Figure 6: Results of Ranking Algorithms on Different Size of Web Search Testing Dataset**

loss to measure the loss of ranking, and accordingly present a ranking algorithm named FRank based on a generalized additive model. Experiments with significance test show that the FRank algorithm performs well in practice, even for conventional TREC dataset and real Web search dataset. Several issues remain for future work:

- For theoretical aspects, we hope to investigate how to prove the generalization bound based on the probabilistic ranking framework.
- Considering that many approaches can be applied to minimize the fidelity loss, we would like to study whether it is more effective to combine the fidelity loss with other machine learning techniques, such as kernel methods.
- On scalability issues, we plan to implement a parallel version of FRank that can handle even larger training datasets.

## 6. ACKNOWLEDGMENTS

Research of this paper was partially supported by National Science Council, Taiwan, under the contract NSC95-2752-E-001-001-PAE.

## 7. REFERENCES

- [1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, 2006.
- [2] N. Birrell and P. Davies. *Quantum fields in curved space*. Cambridge University Press New York, 1982.
- [3] A. Bookstein. Outline of a general probabilistic retrieval model. *Journal of Documentation*, 39(2):63–72, 1983.
- [4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
- [5] Y. Cao, J. Xu, T. Liu, H. Li, Y. Huang, and H. Hon. Adapting ranking SVM to document retrieval. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2006.
- [6] K. Crammer and Y. Singer. Pranking with ranking. *Advances in Neural Information Processing Systems*, 14:641–647, 2002.
- [7] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 Web Track. *Proceedings of TREC*, 2003, 2003.
- [8] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, 4(6):933–969, 2004.
- [9] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, 2000.
- [10] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
- [11] T. Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola, 1999.
- [12] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.
- [13] R. Nallapati. Discriminative models for information retrieval. *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 64–71, 2004.
- [14] M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.
- [15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1998.
- [16] T. Qin, T. Liu, X. Zhang, Z. Chen, and W. Ma. A study of relevance propagation for web search. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 408–415, 2005.
- [17] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.
- [18] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, 1994.
- [19] R. Song, J. Wen, S. Shi, G. Xin, T. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, and W. Ma. Microsoft Research Asia at Web Track and Terabyte Track of TREC 2004. *Proceedings of the Thirteenth Text REtrieval Conference Proceedings (TREC-2004)*, 2004.
- [20] G. Xue, Q. Yang, H. Zeng, Y. Yu, and Z. Chen. Exploiting the hierarchical structure for link analysis. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2005.