FRank: A Ranking Method with Fidelity Loss

Ming-Feng TSAI[1], Tie-Yan LIU[2], Tao QIN[3], Hsin-Hsi CHEN[1], Wei-Ying MA[2]

[1]Dept. Computer Science and Information Engineering, National Taiwan University, Taiwan 106, ROC

[2]Microsoft Research Asia, No.49 Zhichun Road, Haidian District, Beijing 100080, P.R. China

[3]Dept. Electronic Engineering, Tsinghua University, Beijing, 100084, P.R. China

[1]{mftsai, hhchen}@nlg.csie.ntu.edu.tw

[2]{tyliu, wyma}@microsoft.com

[3]qinshitao99@mails.tsinghua.edu.cn

ABSTRACT

Ranking problem is becoming increasingly important in many applications, especially in information retrieval. Many machine learning technologies have been proposed to solve this problem, such as RankSVM, RankBoost and RankNet. Among them, RankNet, which is based on the probabilistic ranking framework, is leading to promising results and has been applied to commercial Web search engine. Inspired by the success of RankNet, we conduct further study on the probabilistic ranking framework in this paper, and propose a novel loss function named Fidelity to measure loss of ranking. This new loss function not only inherits effective properties of the loss function used in RankNet, it also possesses several new properties that are helpful for ranking. This includes Fidelity loss obtaining zero for each desired document pair, and having a finite upper bound that is necessary for conducting query-level normalization. In these aspects, the Fidelity loss is more consistent with widely-used query-level evaluation criteria for information retrieval. To efficiently minimize Fidelity loss and learn an effective ranking function, we propose an algorithm named FRank based on a generalized additive model. We provide experiments and significance test on a large-scale dataset collected from a commercial Web search engine showing that the proposed FRank algorithm provides significantly better results than RankSVM, RankBoost, and RankNet.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithms

Keywords

Learning to Rank

1. INTRODUCTION

In this information explosion era it is becoming more and more important to accurately retrieve relevant information based on user information needs. The information retrieval (IR) problem is usually formulated as a ranking problem: provided a query and a set of documents, an IR system should provide an output of relevant documents ranked higher than irrelevant ones based on the query. Several methods were proposed to solve this problem in the literature, such as Boolean model, vector space model, probabilistic model, and the language model. These methods can be classified as empirical IR methods. In addition, many machine learning methods have become more widely used over the past several years. The learning-based methods aims to use the labeled dataset to learn the underlying ranking function of IR.

There are several different ways to use learning-based methods to solve IR problems. Indeed, one may regard IR as a binary classification problem in which documents are categorized as relevant or irrelevant. However, for real-world search engines, documents 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. In contrast, several methods view IR as a ranking problem, such as RankBoost, RankSVM, and RankNet. Unlike the classification method, ranking algorithms construct pairs between documents and use machine learning technique to minimize the number of misordered pairs. Specifically, RankSVM is an ordinal regression approach to minimize the number of discordant document pairs. RankBoost attempts to directly solve the preference learning problem, rather than solving an ordinal regression problem. RankNet models the ordinal relationship between two documents by means of probability, and adopts a cross entropy to measure the distance between two probability distributions.

According to [7], the probabilistic ranking framework has many optimal properties, and can better model multiple relevance levels. As a result, RankNet leads to efficient retrieval performance and is used in a commercial search engine. Inspired by the success of RankNet, we conduct further study on probabilistic ranking framework and propose a novel loss function named Fidelity loss. This new loss function is inspired by the concept of Fidelity [21], which is widely in use to measure the difference between two states of a quantum in the field of physics [4]. Moreover, this new loss function inherits those properties of the probabilistic ranking framework, and has several additional usable properties. For instance, Fidelity loss can achieve zero for each desired pair probability, whereas cross entropy does not. Moreover, Fidelity pair loss is bounded between 0 and 1, whereas cross entropy has no upper bound. These new properties make Fidelity more feasible for defining an optimal ranking function in information retrieval.

To learn the underlying ranking function, we further propose a generalized additive model to minimize this Fidelity loss. As a result, our new learning method named Fidelity Rank (FRank) combines the generalized additive model with the probabilistic ranking framework. Experimental results show the algorithm performs well on a large-scale dataset, which encompassed 19,600 queries collected from a commercial internet search engine. We also perform a t-test with the confidence level of 98%. The corresponding results from our t-test show the improvements are statistically significant.

The remainder of this paper is organized as follows. We 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 plans in Section 5.

2. RELATED WORK

How to improve IR accuracy is attracting great attention, and in recent years, many machine learning technologies [7][8][10][11][15][19][20] have been studied to learn underlying ranking functions. In [20], Nallapati treated IR as a binary classification problem that directly classifies a document as relevant or irrelevant with respect to given queries. However, in real-world web searches, 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 ranking algorithms are proposed; typical examples include RankBoost, RankSVM, and RankNet.

Freund *et al*. [11] used the Boosting approach for learning a ranking function and propose RankBoost to solve the problem of combining preferences. Their method aims to minimize the weighted number of pairs of instances that are misordered by the final ranking function. The algorithm can be summarized as follows. For each round *t*, RankBoost chooses α_t and the weak learner h_t so as to minimize the pair-wise loss, which is defined by

$$Z_t = \sum_{x_1 x_0} D_t(x_0, x_1) \exp(\alpha_t (h_t(x_0) - h_t(x_1))),$$

where x_0 is the instance ranked higher than x_1, $D_t(x_0, x_1)$ is the weight of pair (x_0, x_1). After the optimal weak learner h_t is selected, the weight $D_t(x_0, x_1)$ is adjusted with respect to α_t and h_t. The rule of adjustment is to decrease the weight of pairs if h_t gives a correct ranking ($h_t(x_1) > h_t(x_0)$) 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 large datasets.

Thorsten Joachims [19] proposed RankSVM algorithm, which uses Support Vector Machines (SVM) to learn retrieval function from click-through data. RankSVM aims to minimize the number of discordant pairs, which is similar to RankBoost, and to maximize the margin (this is equal to minimize L-2 norm of the hyperplane parameter $\vec{\omega}$). Given a query, if the ground truths assert that document d_1 is more relevant than d_2, the constraint of RankSVM is $\vec{\omega}\,\Phi(q, d_1) > \vec{\omega}\,\Phi(q, d_2)$, where $\Phi(q, d_i)$ is the feature vector calculated from document d_i relative to query *q*. Then, the whole RankSVM algorithm can be expressed as the following constrained optimization problem.

$$\min : V(\vec{\omega}, \vec{\xi}) = \frac{1}{2}\vec{\omega} \cdot \vec{\omega} + C \sum \xi_{i,j,k}$$

$$s.t. \begin{cases} \forall (d_i, d_j) \in r_1^* : \vec{\omega}\,\Phi(q_1, d_i) \ge \vec{\omega}\,\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ \forall (d_i, d_j) \in r_n^* : \vec{\omega}\,\Phi(q_n, d_i) \ge \vec{\omega}\,\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\ \forall i \forall j \forall k : \ \xi_{i,j,k} \ge 0 \end{cases}$$

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 does not need human-labeled data, and can automatically learn the ranking function from click-through data.

Burges *et al*. [7] proposed a pair-wise differentiable loss function based on their probabilistic ranking framework and used neural networks to optimize this criterion 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 testing samples is specified by the real value that the model takes. Therefore, $f(d_1) > f(d_2)$ is taken to mean that the model asserts that $d_1 \succ d_2$. Given a set of pairs of training samples [d_i, d_j] together with the target probability P_{ij}^* of sample *i* being ranked higher than sample *j*, a logistic function [3] can be used to map the output of the 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 the 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 (*i*, *j*), P_{ij}^* is the desired probability, and P_{ij} is the modeled probability.

RankNet uses back-prop equation to optimize the above criterion. 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. Therefore, RankNet performs well in practice and successfully applies when used on a commercial search engine.

3. FIDELITY RANK

Inspired by the success of RankNet, we investigate probabilistic ranking framework [7] in this paper and propose a novel loss function named Fidelity loss. In this section we discuss probabilistic ranking framework and the motivation of proposing a new loss function. We also describe the fidelity loss function and its properties, and derive the FRank algorithm based on the additive model for minimizing the fidelity loss function. This algorithm successfully combines probabilistic ranking framework with Boosting for learning an effective ranking function in information retrieval.

3.1 Probabilistic Ranking Framework

As mentioned in Section 2, in the probabilistic ranking framework, the map from outputs to probabilities is modeled by logistic function. According to [7], this framework has the following properties:

(1) The 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 $0 \le P_{ij}^* \le 1$ for every pairs of sample 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 [7], cross entropy is adopted for this purpose, and a pair-wise differentiable loss function is proposed. 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(a).

The pair-wise 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 is successful. However, if we carefully investigate Figure 1(a), we will find there are still some problems with this loss function.

(1) Cross entropy loss function cannot achieve the real minimal loss, zero, expect for the posterior is 0 and 1. This may make the corresponding learning algorithm inaccurate.

(2) The penalization is too great when a pair is in the wrong position. In other words, there is no upper bound for the loss of a pair, which may cause the training procedure to become biased by some hard pairs.

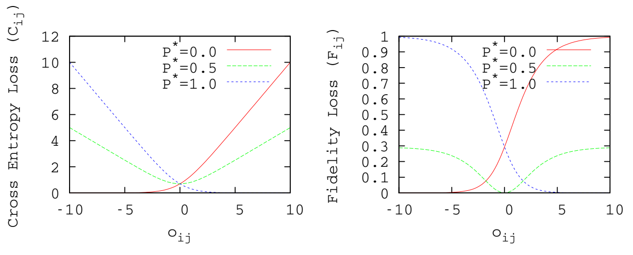
Considering these problems, it is meaningful to investigate the other measures of probability distribution that can provide better properties for probabilistic ranking. This is a key point in this paper.

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 [21]. 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 just the inner product between vectors with components $\sqrt{p_x}$ and $\sqrt{q_x}$, which lie on a unit sphere.



(a) (b)

Figure 1. Loss Function: (a) Cross Entropy Loss Function, (b) Fidelity Loss Function

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

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

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(b) plots F_{ij} as a function of o_{ij} for three values P^* = {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 possesses all three properties of cross entropy in RankNet. In addition, it also has some other properties that are helpful for ranking. We elaborate on these properties in detail as follows.

(1) Fidelity loss achieves the real minimal loss for each desired probability P^* of pairs.

Unlike the cross entropy in [4], the fidelity loss function has the zero loss for each pair. This property makes the algorithm more accurate in the training phrase.

(2) Fidelity loss of a pair is bounded between 0 and 1.

If the loss of a wrong pair does not have an appropriate upper bound, some hard pairs lose too much when still not be placed in the correct position. Therefore, their performance deteriorates, since the whole model is dominated by these hard pairs. In this sense, fidelity loss is superior to cross entropy.

(3) It is easy to define the fidelity loss of a query.

The loss functions of RankSVM, RankBoost, and RankNet are all based on pair levels; however, the evaluation of IR is based on query level, such as mean average precision (MAP) [24][25], and normalized discounted cumulative gain (NDCG) [16][17]. This inconsistency causes some queries to be neglected in the training process. However, since the fidelity loss of a pair is between 0 and 1, the loss of a query can be easily considered by means of dividing the number of pairs in the given query, which can be represented 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 this definition of query-level fidelity loss looks natural, the similar extension to the query-level loss for previous ranking methods such as RankBoost and RankNet is non-trivial. The major reason is that their loss functions do not have an appropriate upper bound. Therefore, even after being normalized by the number of document pairs per query, we still cannot guarantee that every query contributes equally to the total loss.

With the three 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 adopt a generalized additive model similar to the boosting approach. The primary consideration is that the additive model has great potential to implement in parallel since the evaluation 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 α_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,

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

The fidelity loss of $H_k(x)$ over all queries is

$$\begin{aligned} J(H_k) &= \sum_q \frac{1}{|\#_q|} \sum_{ij} F_{ij}^{(k)} \\ &= \sum_q \frac{1}{|\#_q|} \sum_{ij} \left(1 - \left[P_{ij}^* \times \left(\frac{e^{H_k(x_i)-H_k(x_j)}}{1+e^{H_k(x_i)-H_k(x_j)}} \right) \right]^{\frac{1}{2}} - \left[(1-P_{ij}^*) \times \left(\frac{1}{1+e^{H_k(x_i)-H_k(x_j)}} \right) \right]^{\frac{1}{2}} \right) \end{aligned}$$

Given $H_{k-1}(x)$ and $h_k(x)$, the criterion can be re-written as:

$$\begin{aligned} &J(H_k) \\ &= \sum_q \frac{1}{|\#_q|} \sum_{ij} \left(1 - \left[P_{ij}^* \times \left(\frac{e^{H_{k-1}^{i,j}+\alpha_k h_k^{i,j}}}{1+e^{H_{k-1}^{i,j}+\alpha_k h_k^{i,j}}} \right) \right]^{\frac{1}{2}} - \left[(1-P_{ij}^*) \times \left(\frac{1}{1+e^{H_{k-1}^{i,j}+\alpha_k h_k^{i,j}}} \right) \right]^{\frac{1}{2}} \right) \end{aligned}$$

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

Then, we denoted

$$D(i,j) = \frac{1}{|\#_q|}, \tag{1}$$

the fidelity loss over all queries can be formulated as

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

Setting the derivative of Equation (2) with respect to α_k to zero, we have the expression as follows:

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

Note that for a general weak learner $h_k(x)$, solving a close form for α_k from the above equation is quite difficult. For simplicity, we adopt the same choice of weak learners as in [11], 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

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

With some further derivations and relaxations, we eventually obtain

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

where

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

Consequently, the algorithm operates in this way. Using Equation (3), we train a new weak learner $h_k(x)$ according to the current pair weight $W_{i,j}$, and combine this weak learner with the previous ones using the combination coefficient α_k. In a stepwise manner, we eventually can obtain the final ranking function. We name this algorithm Fidelity Rank (FRank), for which details are summarized in Figure 2.

```
Algorithm: FRank
Given: pair over all training queries, and weak learner candidates
    $h_c(x)$, $i$=1,2, …
Initialization:
    Initialize pair weight by Eq. (1)
For $t=1,2, …, k$
  (a)For each weak learner candidate $h_c(x)$
        (a.1) Calculate the optimal $\alpha_{t,c}$ by Eq. (3)
        (a.2) Calculate the loss over all queries by Eq. (2)
  (b) Choose the weak learner $h_{t,c}(x)$ with the minimal loss as the
      weak learner $h_t(x)$ in this round
  (c) Choose the corresponding $\alpha_{t,c}$ as $\alpha_t$
  (d) Update pair weight by Eq. (4)
Output: the final ranking function $H(x) = \sum_{t=1}^{k} \alpha_t h_t(x)$
```

Figure 2. The Algorithm of FRank

4. EXPERIMENTS

In this section we first introduce the evaluation metrics: precision at position n (P@n) [1], mean average precision (MAP) [1], and normalized discount cumulative gain (NDCG) [16][17]. Then, we briefly describe three comparison methods and one non-learning based method. We also describe the experiments for verifying the effectiveness of our proposed ranking algorithm on two datasets: TREC and Web search dataset.

4.1 Evaluation Measures

4.1.1 Precision at Position n (P@n)

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

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

For instance, if there are 5 returned documents for a query {relevant, irrelevant, irrelevant, relevant, irrelevant}, then P@n of the query is {1, 1/2, 1/3, 2/4, 3/5}. For a set of queries, we obtain P@n by means of averaging P@n values of all queries. In general, the present Internet 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 to evaluate the performance of each ranking algorithm.

4.1.2 Mean Average Precision (MAP)

For comparison, we also use MAP as evaluation metric for evaluating ranking methods. MAP is a widely used evaluation metric in conventional IR in which there are two categories for a document: relevant and irrelevant. MAP calculates the mean of average precisions over a set of queries. Given a query q_i, average precision is defined as the average of precision after each relevant document is retrieved. Given a query q_i, its average precision (AP) is calculated as:

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

where N is the number of docs retrieved, *P(j)* is the precision value at position *j* as described in previous subsection, and *pos(j)* is a binary function to indicates whether the documents in position *j* is relevant. Then, we obtain MAP by average the AP values of all the queries.

4.1.3 Normalized Discount Cumulative Gain

Considering that the web search dataset has multiple rating grades, we use the normalized discount cumulative gain (NDCG) [16][17] 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* is $n_i \sum_{j=1}^{i} \frac{(2^{r(j)} - 1)}{\log(1+j)}$

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 experiments.

4.2 Comparison Methods

In this study we choose three machine learning algorithms for comparison: RankBoost, RankNet, and RankSVM. For RankBoost, we use binary weak learner whose output is 0 and 1. For RankNet, we used linear neural net and two-layer net, and those are denoted as RankNet_Linear and RankNet_TwoLayer. For RankSVM, we directly used the binary code of SVMlight. For our proposed algorithm, FRank, we also use binary weak learner as weak learner for comparison.

To compare with conventional IR approach, we employ a widely used non-learning based ranking algorithm, BM25, as the baseline. BM25 computes the relevance score of a document as follow:

$$relevance = \sum_{T \in Q} \omega \frac{(k_1 + 1) tf (k_3 + 1) qtf}{(K + tf)(k_3 + qtf)}$$

where Q is a query consisting of terms T; *tf* is the occurrence frequency of the term T within the web page, *qtf* is the frequency of the term T within the topic from which Q was derived, and ω is the Robertson/Sparck Jones weight [25] of T in Q. K is calculated by

$$K = k_1 \left((1-b) + b \times dl / avdl \right)$$

where *dl* and *avdl* denote the page length and the average page length.

4.3 Experiment on TREC Dataset

In this section, we first describe TREC Web Track data collection, and then report experimental results on the collection. Due to the great impact of Text REtrieval Conference (TREC) [29] in information retrieval community, we evaluate the performance of FRank on TREC Web Track data collection.

This corpus 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 [9] as our query sets (with 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.

For learning algorithms, we extracted 14 features for each document. The details of these features are listed in Table 1.

Table 1. Extracted features for TREC data

Feature Name	Number of Features
BM25 [24]	1
MSRA1000 [26]	1
PageRank [22]	1
HostRank [30]	1
Relevance Propagation [23]	10

Since the size of TREC dataset seems inadequate for learning algorithm, we conduct 4-fold cross validation for each ranking algorithm. We also tune the parameters for BM25 with one of the trials and apply the parameters to other trials directly. The experimental results are plotted in Figure 3 in which the value is averaged score over the four trials.

From Figure 3, the proposed algorithm, FRank, outperforms all other algorithms in MAP, while the other learning algorithms get similar results. This may imply that these three state-of-the-art algorithms (RankSVM, RankBoost, and RankNet) have similar learning ability for information retrieval.

Note that all learning-based methods outperform BM25. The MAP value of BM25 is about 0.13. We note that learning algorithm is cable to obtain at least 40% improvement over non-learning BM25. That would suggest that learning to rank is a promising direction for IR.

We also can note that FRank outperforms other ranking algorithms on precision, especially precision at position 1. The result indicates that the proposed method is also suitable for conventional IR in which relevant document should be ranked as high as possible. Moreover, FRank obtains better performance of NDCG value at top position such as NDCG@1 and NDCG@3.

Table 2. Details of Data Set

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

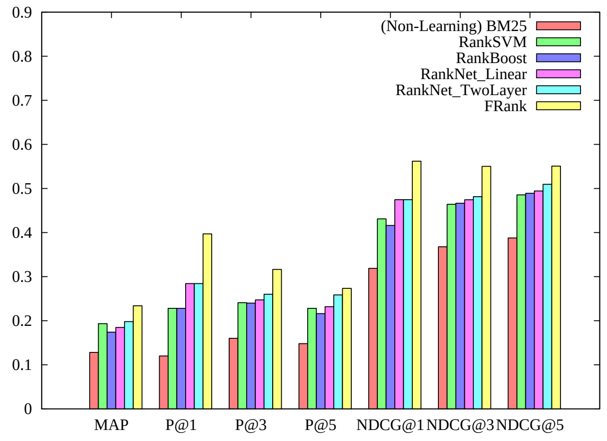


Figure 3. Results of FRank on TREC Dataset

4.4 Experiment on Web Search Data

In this subsection we describe the dataset collected from a commercial search engine. We also describe the experiments for verifying the effectiveness of our proposed ranking algorithm. We first examine the training performance of FRank on the huge Web search dataset. Then, we compare the performance of FRank with that of RankNet and RankBoost; moreover, we also test the performance of standard IR model, BM25 [5][24][25], for a comparison of learning-based and non-learning ranking methods. Finally, we perform experiments to investigate how the number of training queries affects the performance of the ranking algorithms.

4.4.1 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 (irrelevant) 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. Since several features are a larger number, the whole features are preprocessed with global normalization. Considering that these features reflect the business secrete of that search engine company, we would not describe them in detail. However, this does not influence our experimental study since all the learning methods under investigation actually operate on the same feature set.

For ease of experimental study, we divide the aforementioned data into three sets: 12,000 queries for training, 3,800 queries for validation, and 3,800 queries for testing. We did not conduct 4-fold cross validation in web search dataset since the size of dataset is too large. Since some unlabeled documents are highly relevant, performance is affected if we use the whole unlabeled documents for training. Therefore, 20 unlabeled documents are randomly selected as the poorly relevant documents for training. This results in our training of 385,293 documents; it totally contains 2,635,976 pairs. Table 2 summarizes the details of the training, validation, and testing datasets.

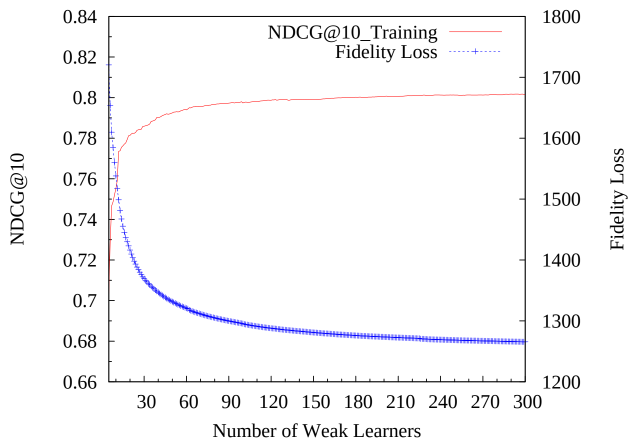


Figure 4. Results of FRank on Training Data

4.4.2 Training Process of FRank

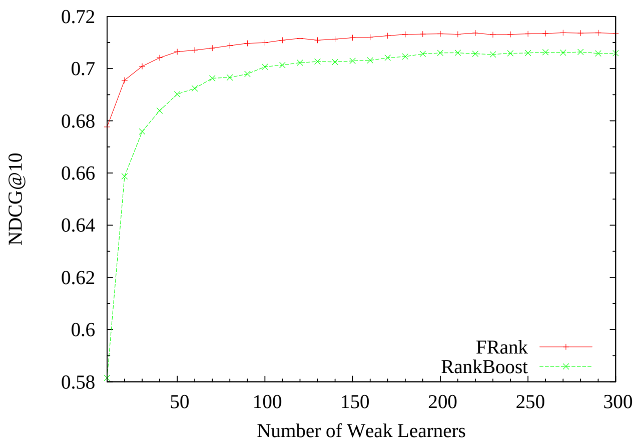
We check whether the fidelity loss function is minimized using our proposed generalized additive model, and whether the fidelity loss function is consistent with the evaluation of IR, i.e. NDCG. The corresponding result is shown in Figure 4, in which the number of weak learners starts from 5. The figure shows the fidelity loss decreases with the increasing number of weak learners; on the other hand, the value of NDCG@10 increases when the fidelity loss decreases. This indicates FRank really can reduce fidelity loss and boost NDCG value.

4.4.3 Performance Comparisons

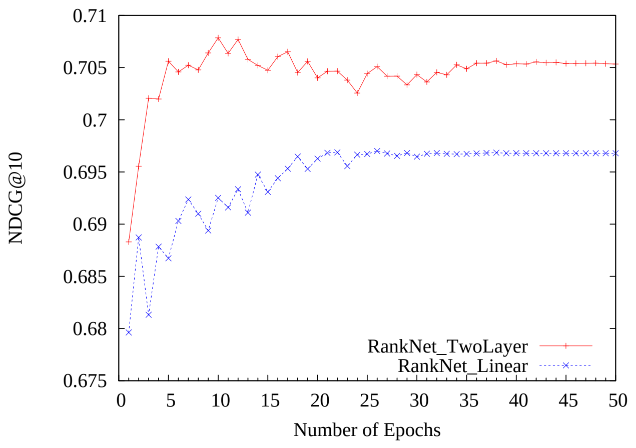
We compare FRank with RankBoost, RankSVM, and RankNet verifying that the fidelity loss function is superior to the loss functions of these reference algorithms. For RankBoost [11], we implemented binary weak learner which is denoted by RankBoost. According to [7], we also implemented RankNet_Linear and RankNet_TwoLayer. For RankSVM [18][19], we directly adopt the binary code of SVMlight. For FRank, we used the binary weak learner, which output is 0 or 1. Since the number of pairs in the training set (with 12,000 queries) is 2,635,976, RankSVM ran out of memory, even with linear kernel. Therefore, we only report the experimental results for FRank, RankBoost, and RankNet.

To evaluate the performance of these ranking methods, we first ran experiments on validation set to select the best parameter setting for each method. For example, for FRank and RankBoost, we determine how many weak learners are used in the final ranking function. In addition, for RankNet, we need to determine the number of training epochs. The validation data is taken to guarantee the most effective generalization performance of these ranking algorithms. We adopt NDCG@10 as the evaluation criterion to select the best parameter setting on the validation set.

Figure 5 plots the NDCG@10 curves of three ranking algorithms on validation data; the number of weak learners starts from 10 in this figure. Figure 5(a) shows that FRank performs better than RankBoost on the validation set. In addition, when the number of weak learners is smaller than 20, RankBoost obtains worse performance. However, when the number of weak learners is over 20, RankBoost eventually gets better performance on the validation dataset. That is because the weak learner in RankBoost only has 0/1 outputs, with few weak learners, the corresponding power of representation is quite limited.



(a)



(b)

Figure 5. Results on Validation Data: (a) FRank and RankBoost, (b) RankNet

When the number of weak learners continues to increase, the non-linear nature of RankBoost eventually makes the model more complex and representative. On the other hand, we note from this figure that all the curves have a flat long tail when the number of weak learners continues to increase. To make sure generalizations about our model are correct, we select some point in the middle of this flat tail as our best parameter settings. Accordingly, we finally select 224 weak learners for RankBoost, and 271 weak learners for FRank.

In Figure 5(b), we observe that RankNet_TwoLayer performs well. Its result is more accurate than RankNet_Linear. However, although RankNet_TwoLayer is accurate, it seems sensitive to the dataset. The performance of RankNet_TwoLayer drops when the number of epoch was more than 10. In contrast, the ranking algorithms based on generalized additive model such as RankBoost and FRank are robust against this problem. The robustness also corresponds to the essential property of an additive model. According to Figure 5(b), we selected 25 epochs for RankNet_Linear and 9 epochs for RankNet_TwoLayer.

Table 3. Parameter Settings for Each Ranking Algorithm

Ranking Algorithm	Parameter	NDCG@10
FRank	271	0.713802
RankNet_TwoLayer	9	0.707845
RankBoost	224	0.706398
RankNet_Linear	25	0.697004

Table 3 summarizes the parameter setting and the corresponding NDCG@10 value on the validation set for each ranking algorithm. As shown in this table, our proposed method outperforms the other ranking algorithms; moreover, the models with probabilistic loss functions (i.e. FRank and RankNet) also perform better than those with conventional pair-based loss functions (i.e. RankBoost).

After the parameter tuning on the validation set, we use the best parameter setting to examine the performance of the ranking algorithms on the testing data. In the test trail we calculate the value of NDCG from 1 to 10 comprehensively to evaluate the performance for each ranking algorithm.

For comparison of learning-based and non-learning methods, we also conduct an experiment of applying standard IR model [5][24][25], i.e. BM25, which is the well-recognized IR model without learning technique on the testing dataset. Moreover, we also simply use a linear combination of BM25 and PageRank to rank the documents. The better performance is obtained when the parameter is 0.2 (i.e. 0.8 * BM25 + 0.2 * PageRank) after we try many different combination parameters. Table 4 summarizes the results of learning-based and non-learning methods. From these results, it is inadequate to simply use a standard IR model for the large-scale IR problem, especially for the Web searching. In other words, learning-based methods have their advantages in leveraging large number of features to boost search performance.

Figure 6 plots NDCG values from 1 to 10 of learning-based methods on the testing dataset. This figure shows that FRank outperforms the other ranking algorithms from NDCG@1 to NDCG@10. RankNet_TwoLayer also performs well on this large-scale dataset. RankBoost was in the third position. These results indicate that the loss function based on the probabilistic ranking framework is more accurate than that used in RankBoost. In addition, although FRank and RankNet both are based on the probabilistic ranking framework, FRank can obtain more accurate ranking function than RankNet; this is consistent with the discussion about the superiority of fidelity in Section 3.2.

To verify whether the above improvements are statistically significant, we further perform t-test for FRank and RankNet_TwoLayer with a confidence level of 98%. 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 information retrieval, FRank is significantly better than RankNet_TwoLayer, and thus significantly better than other ranking algorithms under investigation.

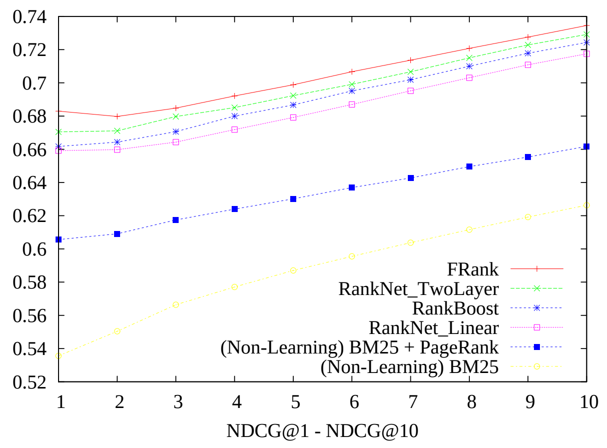


Figure 6. Results of Ranking Algorithms on Test Data

4.4.4 *Experiment on Different Size of Training Data*

Considering RankSVM ran out of memory on our large training set, we further conduct several experiments on smaller-scale datasets to provide more insight about RankSVM. It is also meaningful to investigate how the number of training queries will affect the performance of other ranking algorithms. For this purpose we separately trained these referenced ranking algorithms on 1,000, 2,000, 4,000, 8,000, and 12,000 queries. The detail information of the training data is listed in Table 5. Note that, according to our experiments, RankSVM can output reasonable models when training with 1,000, 2,000, and 4,000 queries; however, when the number of training queries is more than 8,000, the binary code of RankSVM runs out of memory. This is mainly because RankSVM has to construct as many constraints as the document pairs, and then the number of variables in the dual problem becomes voluminous when training on large-scale dataset. For those cases that RankSVM can operate upon, we use the linear kernel and tune the parameter C on the validation set; other experimental settings are similar to those in Section 4.4.3.

Table 4. NDCG Values of Ranking Algorithms on Test Data

NDCG	FRank	RankNet TwoLayer	RankBoost	RankNet Linear	BM25 + PageRank	BM25
1	0.682	0.670	0.661	0.659	0.605	0.535
2	0.679	0.671	0.664	0.659	0.609	0.550
3	0.684	0.679	0.670	0.664	0.617	0.566
4	0.692	0.685	0.679	0.671	0.623	0.577
5	0.698	0.692	0.686	0.679	0.630	0.587
6	0.706	0.699	0.695	0.686	0.636	0.595
7	0.713	0.706	0.701	0.695	0.642	0.603
8	0.720	0.715	0.710	0.703	0.649	0.611
9	0.727	0.722	0.717	0.710	0.655	0.619
10	0.734	0.729	0.724	0.717	0.661	0.626

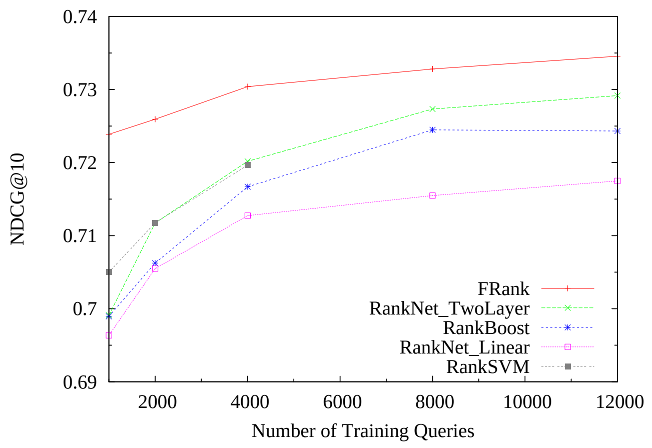


Figure 7. Testing Results on the Different Training Data

Figure 7 plots the results of each of the ranking algorithms trained on different numbers of queries. Table 6 also summaries the value of NDCG@10 on different training datasets. From this figure, we observe the following things:

(1) When the number of training queries is 1,000, the linear model performs better than the complex non-linear models. This is because the non-linear models are too complex, and a small number of training data cannot lead to reliable models. As a result, the complex models are statistically incomplete or tend to over-fit the data. In contrast, since FRank introduces query-level normalization in the loss function, there is some kind of query-level smoothing within it. Therefore, FRank performs relatively well on a small number of training sets. As shown in the figure, FRank far outperforms other algorithms when the number of training queries is 1,000.

(2) The performance of each ranking algorithm increases when the number of training queries increases. However, when the number of queries is more than 8,000, the performance only slightly improves, and sometimes even decreases (e.g. RankBoost). This is interesting because it is not always worth using more training data if considering the tradeoff between effectiveness and scale of training.

(3) RankSVM performs as well as RankNet_TwoLayer when the number of training queries is small. In this regard, we can predict that if the scalability issue of RankSVM can be fixed, it may be an effective candidate for learning to rank.

(4) The methods based on the probabilistic ranking framework perform well when the amount of training data is large. This is because more pair-wise information can make the corresponding calculations of the probabilities more accurate.

(5) Our proposed FRank method obtains more accurate ranking functions than other algorithms for all cases, and it was also more stable with respect to the number of training queries. This strongly suggests that FRank is more suitable for ranking purposes in information retrieval.

Table 5. Details of Different Training Data

Number of training queries	Number of docs	Number of pairs
1000	27,745	141,305
2000	55,344	269,700
4000	117,702	654,253
8000	250,625	1,648,314
12000	385,293	2,635,976

Table 6. NDCG@10 on the Different Size of Training Queries

# of Training Queries	FRank	RankNet TwoLayer	RankNet Linear	RankBoost	RankSVM
1,000	0.723	0.699	0.696	0.698	0.704
2,000	0.725	0.711	0.705	0.706	0.711
4,000	0.730	0.720	0.712	0.716	0.719
8,000	0.732	0.727	0.715	0.724	None
12,000	0.734	0.729	0.717	0.723	None

5. CONCLUSIONS AND FUTURE WORK

This paper presented an approach to learning the underlying ranking function with the goal of improving the accuracy of information retrieval. On the basis of the probabilistic ranking framework, we propose a novel loss function named Fidelity to measure the loss of ranking, and accordingly derive 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 large numbers of queries and large numbers of features.

Several issues remain for future work.

(1) For theoretical aspects, we hope to investigate how to prove the generalization bound based on the probabilistic ranking framework.

(2) Considering that many approaches can be applied to minimize the fidelity loss function, we would like to study whether it is better to combine the Fidelity loss with other machine learning methods, such as kernel methods.

(3) On scalability issues, we plan to implement a parallel version of FRank that can handle even larger training datasets.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] Baeza-Yates, R., Ribeiro-Neto, B. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] Bartell, B.T. *Opitmizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval*. Ph.D. Thesis, University of California, San Diego, 1994.

[3] Baum, E., and Wilczek, F. Supervised learning of probability distributions by neural networks. *Neural Information Processing Systems* (pp. 52-61), 1988.

[4] Birrell, N.D., Davies, P.C.W. *Quantum Fields in Curved Space*. Cambridge University Press, 1982.

[5] Bookstein, A. Outline of a general probabilistic retrieval model. *Journal of Documentation. Volume 39* (pp. 63-72), 1983.

[6] Borlund, P. The Concept of Relevance in IR. *Journal of the American Society for Information Science and Technology 54(10)* (pp. 913-925), 2003.

[7] Burges, C., Shaked, T., Renshaw, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to Rank using Gradient Descent. *Proceedings of 22nd International Conference on Machine Learning*, Bonn, 2005.

[8] Crammer, K., and Singer, Y. Pranking with ranking. *NIPS 14*, MIT Press, 2002.

[9] Craswell, N., Hawking, D., Wilkinson, R., and Wu, M. Overview of the TREC 2003 Web Track. In NIST Special Publication 500-255: *The Twelfth Text REtrieval Conference (TREC 2003)*, pages 78–92, 2003.

[10] Dekel, O., Manning, C., and Singer, Y. Loglinear models for label-ranking. *NIPS 16*, MIT Press, 2004.

[11] Freund, Y., Iyer, R., Schapire, R., and Singer, Y. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 2003.

[12] Friedman, J., Hastie, T., and Tibshirani, R. *Additive logistic regression: a statistical view of boosting*. Dept. of Statistics, Stanford University Technical Report, 1998.

[13] Fuhr, N. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems* (pp. 183-204), 1989.

[14] Hastie, T., Tibshirani, R., and Friedman, J. The Elements of Statistical Learning, *Springer*, New York, 2001.

[15] Herbrich, R., Graepel, T., and Obermayer, K. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classiers*, MIT Press (pp. 115-132), 2000.

[16] Jarvelin, K., and Kekalainen, J. IR evaluation methods for retrieving highly relevant documents. *Proceeding of 23rd ACM SIGIR*, 2000.

[17] Jarvelin, K., and Kekalainen, J. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 2002.

[18] Joachims, T., Schölkopf, B., Burges, C., and Smola, A. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999.

[19] Joachims, T. Optimizing Search Engines Using Clickthrough Data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2002.

[20] Nallapati, R. Discriminative models for information retrieval. *Proceedings of SIGIR 2004* (pp. 64–71), 2004.

[21] Nielsen, M.A., and Chuang, I.L. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[22] Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Stanford University, Stanford, CA, 1998.

[23] Qin, T., Liu, T.Y., Zhang, X.D., Chen, Z., and Ma, W.Y. A study of relevance propagation for web search. In SIGIR 2005: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. Salvador, Brazil, August 2005.

[24] Robertson, S.E. The probability ranking principle in IR. *Journal of Documentation. Volume 33 (pp. 294–304)*, 1977.

[25] Robertson, S.E., and Walker, S. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR* (pp. 345–354), 1994.

[26] Song, R., Wen, J. R., Shi, S. M., Xin, G. M., Liu, T. Y., Qin, T., Zheng, X., Zhang, J. Y., Xue, G. R., and Ma, W. Y. Microsoft Research Asia at Web Track and Terabyte Track of TREC 2004, in the *13th TREC*, 2004

[27] Sormunen, E. Liberal relevance criteria of TREC- Counting on negligible documents? In *Proceedings of the 25th Annual International ACM SIGIR*, 2002.

[28] Vapnik, V. *Statistical learning theory*. Addison Wiley, 1998.

[29] Voorhees, E.M. and Harman, D.K. TREC: Experiment and Evaluation in Information Retrieval. MIT Press, 2005.

[30] Xue, G.R., Yang, Q., Zeng, H.J., Yu, Y., and Chen, Z. Exploiting the hierarchical structure for link analysis. In *Proc. of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. Salvador, Brazil, August 2005.