QoRank: A Query-Dependent Ranking Model Using LSE-Based Weighted Multiple Hyperplanes Aggregation for Information Retrieval

Heli Sun,[1,*] Jianbin Huang,[2,†] Boqin Feng[1]
[1]*Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, People's Republic of China*
[2]*School of Software, Xidian University, Xi'an, People's Republic of China*

Ranking is a core problem for information retrieval since the performance of the search system is directly impacted by the accuracy of ranking results. Ranking model construction has been the focus of both the fields of information retrieval and machine learning, and learning to rank in particular has attracted much interest. Many ranking models have been proposed, for example, RankSVM is a state-of-the-art method for learning to rank and has been empirically demonstrated to be effective. However, most of the proposed methods do not consider about the significant differences between queries, only resort to a single function in ranking. In this paper, we present a novel ranking model named QoRank, which performs the learning task dependent on queries. We also propose a LSE (least-squares estimation) -based weighted method to aggregate the ranking lists produced by base decision functions as the final ranking. Comparison of QoRank with other ranking techniques is conducted, and several evaluation criteria are employed to evaluate its performance. Experimental results on the LETOR OHSUMED data set show that QoRank strikes a good balance of accuracy and complexity, and outperforms the baseline methods. © 2010 Wiley Periodicals, Inc.

1. INTRODUCTION

Over the past decade, the Web has grown exponentially in size. The sheer number of both good and bad pages on the Web has led to an increasing reliance on information retrieval systems for the discovery of useful information. Users rely on information retrieval systems not only to return pages related to their query but also to rank results to suggest the best pages first. Because of its importance, ranking

*Author to whom all correspondence should be addressed: e-mail: helisun.sunny@gmail.com.
†e-mail: jbhuang@xidian.edu.cn.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 26, 73–97 (2011)
© 2010 Wiley Periodicals, Inc.
View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.20455

has become the central problem of many information retrieval applications, such as document retrieval,[1] collaborative filtering,[2] key term extraction,[3] expert finding,[4] important email routing,[5] sentiment analysis,[6] product rating,[7] and anti Web spam.[8]

Ranking has been the focus of much attention for many years and several ranking functions emerged including inner product,[9] vector space model,[10] cosine,[11] probability[12] and BM25.[13] Some ranking methods are based on link analysis, such as PageRank[14] and HITS.[15] All these methods use a small number of features (e.g., term frequency, inversed document frequency, and document length) to tune ranking parameters empirically. Although a growing number of features such as structural features, title text, anchor text, and query independent features (e.g., PageRank and URL length) have been proved useful in document retrieval; empirical tuning of ranking functions sometimes leads to overfitting and has become increasingly difficult.

The approach of employing machine learning techniques to address the ranking problem naturally emerges as an effective solution. Recently, a method called learning to rank has gained increasing attention in both the fields of machine learning and information retrieval. The objective of learning to rank is to automatically learn a ranking model from training data, and the model can sort objects (e.g., documents) according to their degrees of relevance, preference, or importance as defined in a specific application. A typical setting in learning to rank is that feature vectors, and ranks are given as training data. When applied to document retrieval, learning to rank becomes a task as follows: In training, a ranking model is constructed with data consisting of queries, their corresponding retrieved documents, and relevance levels given by domain experts. In ranking, given a new query, the corresponding retrieved documents are sorted by the trained ranking model.

Several methods for learning to rank have been developed. Typical methods include RankSVM,[16] RankBoost,[17] RankNet,[18] and some improved methods such as MHR,[19] AdaRank,[20] and ListNet.[21] RankSVM and MHR are based on support vector machine (SVM), RankBoost and AdaRank are based on boosting, and RankNet and ListNet are based on neural nets. Recently, learning to rank functions have been a major issue in the machine learning community[22–25]and have produced many applications in information retrieval.[26–28]

However, it should be noted that in most of the previous work, a single ranking function is used to handle all instances, which belong to many different queries and ranks. This may not be appropriate, particularly for Web search:

- Queries in Web search have different semantics and intents. For example, queries can be navigational, informational, or transactional.[29]
- Queries in Web search differ in forms they appear. Queries can be short or long. Queries can be personal names, product names, or terminology. Queries can be phrases, combinations of phrases, or natural language sentences.
- The number of relevant documents can vary from query to query in Web search. Queries may have many relevant documents or only a few.
- The degree of relevance of a document to a query may fall into multiple levels. For example, we may get five rating levels from irrelevant to definitely relevant by users.

In this paper, we address learning to rank for document retrieval and propose an alternative approach to RankSVM. First, we discuss about RankSVM, which has been widely used in learning to rank because of its simplicity and effectiveness. However, RankSVM uses a single hyperplane to rank instances belonging to different ranks and queries would make compromises among the cases and result in lower ranking accuracy. In addition, the model training for RankSVM is expensive.

To address these limitations, we propose a new method QoRank, which performs the learning task in a query-dependent manner. It trains multiple query-dependent hyperplanes as the base decision functions and aggregates the rank lists from multiple hyperplanes as the final ranking. On the one hand, different from RankSVM and MHR, QoRank constructs hyperplanes depending on query and thus is able to provide a better separation of the instances of different queries. On the other hand, QoRank separates the training instance into small pieces and it gains the lowest training complexity. We next propose a LSE (least-squares estimation)-based weighted method for ranking aggregation, and the method assigns different weights to hyperplanes with weights reflecting the ranking accuracy.

The public data set LETOR OHSUMED is used to evaluate the effectiveness of our method. Experimental results show that the proposed query-dependent ranking model is very effective for information retrieval. Compared with traditional ranking model BM25 and other learning to rank methods, QoRank achieves better performance.

This paper is organized as follows: We start with a brief review on related research in Section 2. Then, we analyze the limitation of state-of-the-art learning to rank methods and describe our ranking model in Section 3. Section 4 introduces our ranking aggregation method for multiple hyperplanes. Section 5 conducts theoretical analysis. Section 6 describes the data set used in the experiment. Experimental evaluations are reported in Section 7, and the conclusion and future work are presented in the last section.

2. RELATED WORK

2.1. Learning to Rank

The key problem for information retrieval is ranking—specifically, how to create the ranking model that can sort documents based on their relevance to the given query. It is a common practice in information retrieval to tune the parameters of a ranking model using some labeled data and one performance measure.[4] For example, BM25[13] and LMIR.[30,31] There are some methods on how to compose a ranking function. For example, Zobel and Moffat[32] explored all possible combinations of feature weighting components to find such functions, which take into account a small number of features, including term frequency, inverse document frequency, and document normalizations. As more useful features and more labeled data become available, the ranking models become more sophisticated and how to tune or train ranking models becomes a challenging issue. The approach named learning to rank that uses all sorts of useful features and automatic parameter tuning naturally emerges as an effective solution and has become a topic of interest.

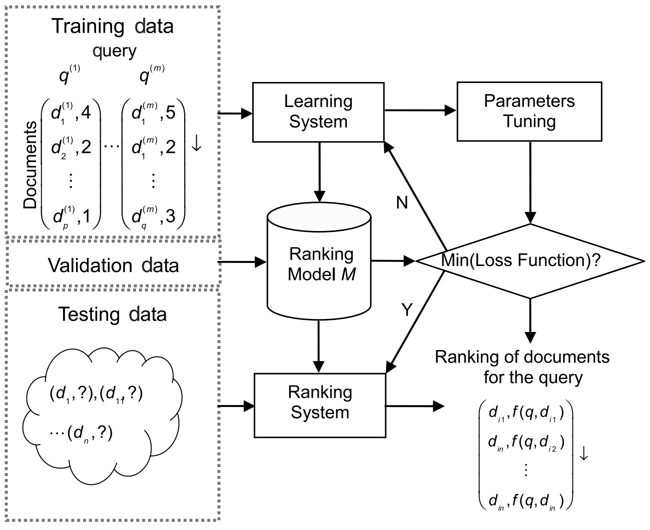


Figure 1. The general framework of learning to rank for document retrieval.

The general framework of learning to rank is described in Figure 1. A number of queries and their corresponding retrieved data are given. The ranks of the data are also provided. Partition the data set into several subsets, some subsets as a training set, some as a validation set, and others as a testing set. The training set will be used to learn the ranking model *M*, the validation set to tune the parameters, and the test set to report the ranking performance of *M*. In the learning phase, the objective is to construct a ranking model *M*, which yields the best results in ranking of the training data. After that, the ranking model *M* tunes the parameters on the basis of the performance measure of the results of the validation set. In the ranking phase, the model *M* returns a ranking list of retrieved results corresponding to the given query in descending order of the relevance scores.

Learning to rank is aimed at automatically creating the ranking model using some training data and machine learning techniques. Some early work tackled this problem as binary classification,[33] in which the assumption is made that a document is either relevant or irrelevant to the query, and the goal of learning is to classify relevant documents from irrelevant documents. However, in real-world applications, the degree of relevance of a document to a query can be discretized to multiple levels. And so, ranking is a problem different from classification. Many other methods based on machine learning techniques have been proposed and applied in information retrieval. According to Cao et al.,[1,21] the current methods fall into three categories: (i) pointwise, (ii) pairwise and (iii) listwise approaches. In the pointwise approach,[33,34] each training example is composed of a set of document features and its corresponding rank relative to a query. The learning process tries to map features into ranks. In the pairwise approach,[1,16–19,23,35] each training example is composed of pairs of instances and the preference relation among them. In this case, the goal is to classify each pair into correctly or incorrectly ranked categories. Finally, in the listwise approach,[20,21,36] a list of documents are used as training instances. A ranking function is learned, and then used to sort documents.

Several pairwise ranking methods have been proposed. For example, RankSVM[16] uses support vector machines to learn a ranking function from preference data. RankNet[18] applies neural network and gradient descent to obtain a ranking function. RankBoost[17] applies the idea of boosting to construct an efficient ranking function from a set of weak ranking functions. The studies reported in Ref. 1 proposed a framework called GBRank using gradient descent in function spaces,[37,38] which is able to deal with complicated features in the context of Web search. Our work in this paper can be viewed as an alternative approach to RankSVM developed for ranking in information retrieval.

2.2. Ranking Aggregation

Rank aggregation is a process of combining individual preferences to obtain a reasonable ultimate ranking. There exist various methods for ranking aggregation. In most cases, the strategies rely on the following information: (i) the ordinal rank assigned to an item in the rank list and (ii) the score assigned to an item in the rank list. In score-based methods, items are ranked in order of the assigned scores in the rank lists, or some transformation of those scores,[39–46] whereas in rank-based merging methods, items are ranked in order of the assigned ranks in the rank lists, or some transformation of those ranks.[39,43,47,56] Another orthogonal distinction of ranking aggregation methods is whether the methods rely on training data (e.g., the Bayes-fuse method,[39] the linear combination method,[45] and the preference rank combination method[41]) or not. Our method LSE-based weighted aggregation proposed in this paper can be viewed as a rank-based and weak training method.

3. QUERY-DEPENDENT RANKING WITH OPTIMIZED MULTIPLE HYPERPLANES

3.1. Analysis of RankSVM

Assuming that there exists an input space $X \in R^n$, where n denotes the number of features. An output space of ranks is represented by label set $Y = \{r_1, r_2, \ldots, r_m\}$, where m denotes the number of ranks. Furthermore, assume that there exists a total order between the ranks $r_1 \succ r_2 \succ \cdots \succ r_m$, where $\succ$ denotes the preference relationship described in Definition 1. In training phase, a set of labeled instances $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is given, where $x_i \in X$ is the feature vector of instance i, and $y_i \in Y$ is the rank label of instance i. If $y_i > y_j$, we say x_i is ranked ahead of x_j, denoted as $x_i \succ x_j$. Assume that F is the set of ranking functions, such that each of them can determine the preference relations between instances

$$x_i \succ x_j \Leftrightarrow f(x_i) > f(x_j) \tag{1}$$

DEFINITION 1. *Rank r_a is preferable to rank r_b if r_a is ranked higher than r_b, and the order relation between r_a and r_b is denoted as $r_a \succ r_b$.*

SVM is a general model of machine learning, which has been applied to ranking model construction and some promising results have been obtained. RankSVM[16] formalizes the above-mentioned learning problem as that of learning for classification on pairs of instances. Note that the relation $x_i \succ x_j$ between instance pairs x_i and x_j can be expressed by a new instance $x_i - x_j$. Next, take any instance pair and their relation to create a new instance and a new label.

$$\left(x_i - x_j, z = \begin{cases} +1 \; y_i > y_j \\ -1 \; y_j > y_i \end{cases}\right) \tag{2}$$

If x_i is ranked ahead of x_j, we assign a label $+1$, otherwise -1 to the new instance. In this way, RankSVM produces a new training data set. Constructing the SVM model is equivalent to solving the following problem:

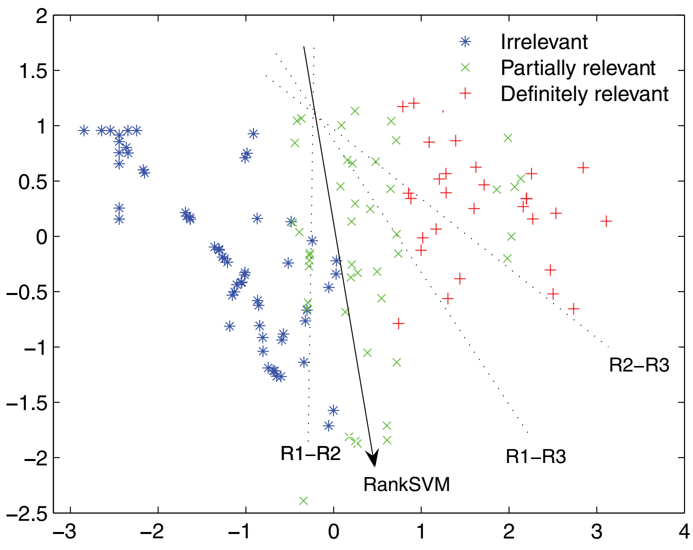
$$\begin{array}{l} \min_{\omega \xi_{i,j}} \frac{1}{2} \|\omega\|^2 + C \sum \xi_{ij} \\ s.t. \langle \omega, x_i - x_j \rangle > 1 - \xi_{ij}, \forall x_i \succ x_j, \xi_{ij} \geq 0 \end{array} \tag{3}$$

where $\|\omega\|^2$ denotes ℓ_2 norm measuring the margin of the hyperplane and ξ_{ij} denotes a slack variable. Suppose that the solution to (3) is ω^*, and then the ranking function is given by

$$f(x) = \langle \omega^*, x \rangle \tag{4}$$

RankSVM is unique in that it constructs one single hyperplane classifier on all instance pairs and uses it for ranking. The advantages of RankSVM are simplicity and effectiveness. However, RankSVM employs a single hyperplane for ranking and thus it is difficult to separate instances from many different ranks. In addition, RankSVM does not consider the diversity of queries. In reality, as shown in Figure 2, a single hyperplane is difficult to handle the ranking of instances from many different ranks and queries. Another problem is that the model training is generally costly. The order of the instance pairs it uses to train is quadratic in the training instance size.

To illustrate the limitation of RankSVM, two examples showing the distribution of instances for Query 5 and Query 65 of LETOR OHSUMED data set are presented in Figure 2. Principle component analysis (PCA) is performed on the data of LETOR OHSUMED data set, and the first and second principle components are displayed in the figure. In Figure 2, blue denotes "irrelevant" (R1), green denotes "partially relevant" (R2), and red denotes "definitely relevant" (R3). In Figure 2a, we can observe that RankSVM exploits a single hyperplane to rank all kinds of the documents and treats the instance pairs from all rank pairs equally. There are more instances in the ranks of R1 and R2, the ranking model of RankSVM tends to be close to that of R1–R2, i.e., the direction that separates R1 and R2. As a result, RankSVM could not well separate the instances in the ranks of R1 and R2, R2 and



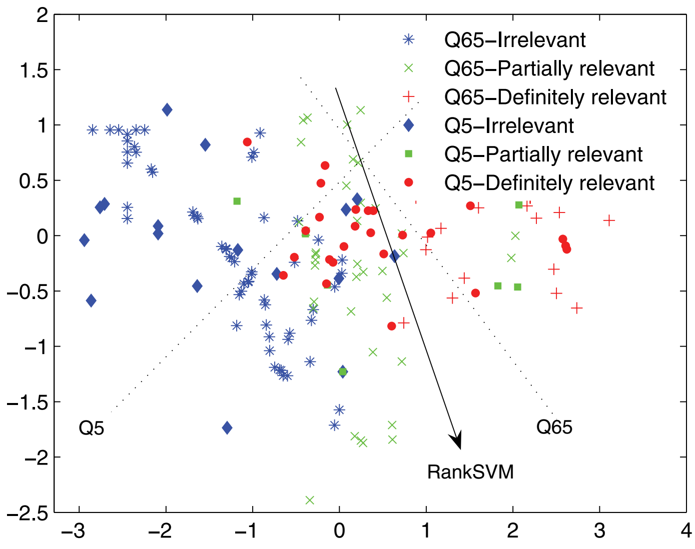


Figure 2. Instances distribution of Query 5 and Query 65 in LETOR OHSUMED data set.

R3. In Figure 2b, we can find that the total ranking model tends to be close to that of Query 65, which has more instances.

3.2. Query-Dependent Ranking with Optimized Multiple Hyperplanes

Let $Q = \{q_i | i = 1, \ldots, N_q\}$ denote a collection of N_q queries in the training data, and each query q_i is associated with a list of N_i instances $D^i = \{x_1^i, \ldots, x_{N_i}^i\}$, in which each instance x_j^i is manually judged with relevance $y_j^i \in Y$, where $Y = \{r_m | m = 1, \ldots, N_r\}$ and N_r denotes the number of ranks. For each query q_i, we train a hyperplane with the corresponding documents set $D^i = \{x_1^i, \ldots, x_{N_i}^i\}$, then the learning task is formalized as a quadratic programming problem shown below:

$$\begin{array}{l} \min_{\omega_i \xi_{i,j,k}} \frac{1}{2}\|\omega\|^2 + C\sum_{j,k} \xi_{i,j,k} \\ \text{s.t.} \langle \omega_i, x_j^i - x_k^i \rangle \geq 1 - \xi_{i,j,k}, \xi_{i,j,k} \geq 0 \end{array} \tag{5}$$

where i denotes ith query and ω denotes the parameter vector of base decision function, x_j^i and x_k^i denote jth and kth instance of D^i, $\xi_{i,j,k}$ is a slack variable.

Different from classification, the training instances of ranking are generally belong to certain rank, such as "definitely relevant," "partially relevant," and "irrelevant." There exist some relations between the ranks. Using the relations to optimize the multiple hyperplanes, both the number of hyperplanes and the number of training instances will be greatly reduced. First, we describe the adjacency relation of ranks in Definition 2.

Definition 2. *Rank r_m is adjacent to rank r_n iff r_m is preferable to r_n, and there is no rank between r_m and r_n, the relation between r_m and r_n is denoted as $r_m > r_n$.*

Let $\omega_{i,m,n}$ denote the parameter vector of base decision function for the rank pair r_m and r_n. When $r_m > r_n$, we can build up the base decision function as follows:

$$\begin{array}{l} \min_{\omega_{i,m,n} \xi_{i,j,k,m,n}} \frac{1}{2}\|\omega\|^2 + C\sum_{j,k} \xi_{i,j,k,m,n} \\ \text{s.t.} \langle \omega_{i,m,n}, x_j^m - x_k^n \rangle \geq 1 - \xi_{i,j,k,m,n}, \xi_{i,j,k,m,n} \geq 0 \end{array} \tag{6}$$

where x_j^m indicates an instance of rank r_m and x_k^n indicates an instance of rank r_n. We build a hyperplane for each rank pair r_m and r_n when $r_m > r_n$. If there are N_r ranks, then there will be only $N_r - 1$ base decision functions for the $N_r - 1$ rank pairs:

$$f_{i,m,n}(x) = \langle \omega_{i,m,n}, x \rangle \tag{7}$$

For example, when $N_r = 3$, the number of base decision functions is $N_r - 1 = 2$ and can be represented as $\omega_{1,2}$, $\omega_{2,3}$, respectively, corresponding to two rank pairs (1, 2), (2, 3). Figure 3 shows an example of the QoRank method, in a two-dimensional s pace described by the two principal coordinates. Considering the order relations, QoRank trains one hyperplane for every adjacent rank pair: One is between "definitely relevant" and "partially relevant;" the other is between"partially relevant" and "irrelevant."

Constructing one hyperplane for each query reduces the number of training instances for single hyperplane. Accordingly, the total number of training instance

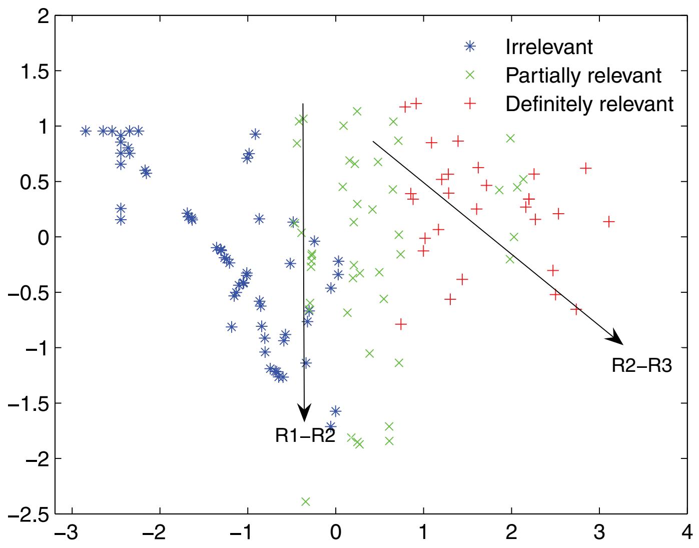


Figure 3. QoRank for query 65 of LETOR OHSUMED data set.

for all of the hyperplanes decreases dramatically, reducing the training time. The theory analysis is given in Section 5. Furthermore, building hyperplanes for part of the rank pairs conquers the problem of hyperplanes direction deviation for RankSVM caused by uneven distribution of training instance. The algorithm procedure for the method is described in Algorithm 1.

```
Input: training dataset $T = \{(x_i, y_i) | i = 1, 2, \dots, N_t\}$ where $x_i \in D$ and $y_i \in Y$, $D = \{x_1, \dots, x_{N_t}\}$,
    $Y = \{r_m | m = 1, \dots N_r\}$, query set $Q = \{q_i | i = 1, \dots, N_q\}$
Output: a set of decision functions
1  Divide the training dataset $T$ into some subsets $T_i (i = 1, \dots N_q)$ according to the query ;
2  for $i = 1, 2, \dots N_q$ do
3      for $j = 1, 2, \dots, N_r$ do
4          for $k = 1, 2, \dots, N_r$ do
5              if rank $r_j >$ rank $r_k$ then
6                  Choose the instances in $T_i$ belonging to rank $r_j$ and rank $r_k$ as subset $T_{ijk}$;
7              end
8          end
9      end
10     Calculate the vector differences in each subset ;
11     Select the working sets using the SMO method which meets $m(\alpha^k) - M(\alpha^k) \le \varepsilon$ ;
12     Carry out iterations until $\left| f(\alpha^k) - f(\vec{\alpha}) \right| \le \varepsilon$ ;
13     Calculate the parameters and select the support vectors;
14     Output the decision functions ;
15 end
```

Algorithm 1. QoRank.

4. LSE-BASED WEIGHTED RANKING AGGREGATION

After obtaining the base decision functions, the rank label for any new instance x is determined by LSE-based weighted aggregation, which assigns different weights to the base decision functions, and the weights are in proportion to their ranking accuracy. We propose to learn the weights as follows:

Let $X = \{x_i | i = 1, 2, \ldots, N_t\}$ be the set of training instances, where N_t is the number of instances in X. Let $\mathbf{r} = \{r_1, r_2, \ldots, r_{N_t}\}'$ be the rank label vector of X, i.e., the ground truth. Let $f_k (k = 1, 2, \ldots, K)$ be decision function set, where K is the number of decision functions.

Let the rank value vector given by the decision functions $f_k (k = 1, 2, \ldots, K)$ for X be $\mathbf{y} = \{y_1, y_2, \ldots, y_{N_t}\}'$, $\beta = \{\beta_1, \beta_2, \ldots, \beta_K\}'$ be the weight vector of the decision functions, and $\varsigma = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_K\}'$ be the vector of additive errors. We can obtain as follows:

$$\mathbf{y} = X\beta + \varsigma \tag{8}$$

Then we use least squares to fit a regression line to the data $\{x_i, r_i\}_{i=1}^{N_t}$, i.e., to find the regression coefficient estimates $\widehat{\beta}$ to minimize the criterion,

$$Q(\beta) = (\mathbf{y} - X\beta)'(\mathbf{r} - X\beta) \tag{9}$$

Taking derivatives with respect to β, and setting these to 0, we can obtain the normal equations as follows:

$$(X'X)\beta = X'\mathbf{y} \tag{10}$$

Apply the inverse of $X'X$ to both sides of Equation 10, then we get

$$\widehat{\beta} = (X'X)^{-1}X'\mathbf{y} \tag{11}$$

In testing, the rank label for any new instance x_i is determined by the following process: Let D denote the set of instances to be ranked and n the number of instances in D. $f_k(x_i)$ denotes the rank value of instance x_i given by decision function k. We can obtain the final rank value of instance x_i as

$$r\_value(x_i) = \frac{1}{K}\sum_{k=1}^{K} \beta_k f_k(x_i) \tag{12}$$

After the above-mentioned process, we sort the instances according to r_value. The larger the r_value is, the higher x_i in the final ranking list. An intuitive understanding of this process is, first, calculate the weight for each hyperplane according to its ranking accuracy. Then the base decision function gives a rank value to each instance. Finally, calculate the weighted average of the rank values received from

```
Input: training dataset $X = \{x_i | i = 1,2,\dots,N_t\}$, label vector $\mathbf{r} = \{r_1, r_2 \dots, r_{N_t}\}'$ of instances in $X$, test dataset $D$,
       the base decision functions $f_k (k = 1,2,\dots K)$
Output: the final rank list
1  for $i = 1,2,\dots N_t$ do
2      $y_i = 0$ ;
3      for $k = 1,2,\dots K$ do
4          $y_i = y_i + f_k(x_i)$ ;
5      end
6  end
7  $\widehat{\beta} = (X'X)^{-1}X'\mathbf{y}$;
8  for $i = 1,2,\dots,|D|$ do
9      $r\_value(x_i) = 0$ ;
10     for k = 1,2,...,K do
11         $r\_value(x_i) = r\_value(x_i) + \widehat{\beta} f_k(x_i)$;
12     end
13 end
14 $r\_value(x_i) = \frac{1}{K} r\_value(x_i)$;
15 Rank all of the instances according to their $r\_value(x_i)$;
```

Algorithm 2. LSE-based weighted aggregation.

all of the base decision functions for each instance and produce the final list. The LSE-base weighted aggregation is described in Algorithm 2.

5. THEORETICAL ANALYSIS

We can also construct the ranking hyperplanes by other means, such as (1) *One against all*: Construct a hyperplane to separate one specific rank with the remaining ranks. (2) *One against one*: Construct a hyperplane to separate pairs of ranks, such as MHR.[19] (3) *Ensemble SVM*: Construct hyperplanes via different training data sets that selected by a certain scheme, e.g., bagging and boosting. We next analyze the principle and training complexity of RankSVM, One-against-all, One-against-one, Ensemble SVM-Bagging, Ensemble SVM-Boosting, and QoRank method, respectively.

Suppose that $Y = \{r_m | m = 1, \dots, N_r\}$ is the set of rank labels, where N_r denotes the number of ranks. $Q = \{q_n | n = 1, \dots, N_q\}$ is the set of queries, where N_q is the number of queries. Furthermore, suppose that R_i denotes the set of instances whose rank label is r_i. R_i^j denotes the set of instances of query j and whose rank label is r_i, and the number of items in R_i^j is denoted as $|R_i^j|$. In training, parameter C is set to 1, and training complexity of SVM can be obtained from,[2]

$$\varsigma = O\left(N_S^3 + N_S M + N_S d_M M\right) \tag{13}$$

where N_S is the number of support vectors, M the number of training instances, and d_M the dimension of the input data.

In learning to rank, the training instances of SVM are document pairs, and the number of support vectors almost equals the number of training instances, that is,

Table I. The principle and complexity of hyperplane construction methods.

	Principle	Complexity
RankSVM	Construct single hyperplane to deal with the instance of all ranks	$O([\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(\lvert R_i\rvert \times \lvert R_j\rvert)]^3 + a[\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(\lvert R_i\rvert \times \lvert R_j\rvert)]^2))$
One-against-all	Construct one hyperplane for each rank to separate it from others	$O(\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}[(\lvert R_i\rvert \times \lvert R_j\rvert)^3 + a(\lvert R_i\rvert \times \lvert R_j\rvert)^2])$
One-against-one	Construct one hyperplane for each rank pair	$O(\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}[(\lvert R_i\rvert \times \lvert R_j\rvert)^3 + a(\lvert R_i\rvert \times \lvert R_j\rvert)^2])$
Ensemble SVM-Bagging	Construct K hyperplanes independently with different training data sets selected via a bootstrap method	$O(K\{[\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(\lvert R_i\rvert \times \lvert R_j\rvert)]^3 + a[\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(\lvert R_i\rvert \times \lvert R_j\rvert)]^2\})$
Ensemble SVM-Boosting	Construct K hyperplanes with the training sample subsets that selected via boosting method	$O(K\{[\sum_{i'=1}^{N_r-1}\sum_{j'=i'+1}^{N_r}(\lvert R_{i'}\rvert \times \lvert R_{j'}\rvert)]^3 + a[\sum_{i'=1}^{N_r-1}\sum_{j'=i'+1}^{N_r}(\lvert R_{i'}\rvert \times \lvert R_{j'}\rvert)]^2\})$
QoRank	Construct one hyperplane for each adjacent rank pairs according to query	$O(\sum_{\substack{i=1 \\ j \rhd i}}^{N_y-1}\sum_{q=1}^{N_q}((\lvert R_i^q\rvert \times \lvert R_j^q\rvert)^3 + a(\lvert R_i^q\rvert \times \lvert R_j^q\rvert)^2))$

$N_S \approx M$. The training complexity of ranking SVM can be written as follows, where a is a constant:

$$\varsigma = O(M^3 + aM^2) \tag{14}$$

Table I summarizes the principle and training complexity for all of the ranking hyperplane construction methods. Because of space limitation, only calculation of complexity for RankSVM, MHR, and QoRank are described below, and the comparison of training complexity is presented in the Appendix.

The number of instances used in RankSVM training is

$$M_{\text{RankSVM}} = \sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(\lvert R_i\rvert \times \lvert R_j\rvert)$$

The training complexity of RankSVM is $\varsigma_{\text{RankSVM}} = O(M^3_{\text{RankSVM}} + aM^2_{\text{RankSVM}})$.

The number of instances used in hyperplane $\omega_{i,j}$ training of MHR is $M^{i,j}_{\text{MHR}} = \lvert R_i\rvert \times \lvert R_j\rvert$. From Equation 15, the training complexity of hyperplane $\omega_{i,j}$ is $\varsigma_{i,j} = O((M^{i,j}_{\text{MHR}})^3 + a(M^{i,j}_{\text{MHR}})^2)$ and that of MHR is

$$\varsigma_{\text{MHR}} = O\left(\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}[(M^{i,j}_{\text{MHR}})^3 + a(M^{i,j}_{\text{MHR}})^2]\right)$$

The number of instances used in hyperplane $\omega'_{i,j}$ training of the QoRank method is $M^{i,j}_{\text{QoRank}} = |R_i| \times |R_j|$, where $j \triangleright i$. From Equation 15, the training complexity of hyperplane $\omega'_{i,j}$ is $\varsigma_{i,j} = O((M^{i,j}_{\text{QoRank}})^3 + a(M^{i,j}_{\text{QoRank}})^2)$, where $j \triangleright i$; then the training complexity of QoRank is

$$\varsigma_{\text{QoRank}} = O\left(\sum_{\substack{i=1 \\ j \triangleright i}}^{N_r-1} \sum_{q=1}^{N_q} [(|R_i^q| \times |R_j^q|)^3 + a(|R_i^q| \times |R_j^q|)^2]\right)$$

6. DATA SET

In our experiment, we use one data set of the LETOR[50] package called the OHSUMED collection,[51] which was created for information retrieval research. It is a subset of MEDLINE, a database on medical publications. The LETOR OHSUMED collection consists of 348,566 records from 270 medical journals during the period of 1987–1991. The fields of a record include title, abstract, MeSH indexing terms, author, source, and publication type. There are 106 queries. For each query, there are a number of associated documents. Each query is about a medical search need and thus is also associated with patient information and topic information. The documents' degrees of relevance with respect to the queries are judged, by humans, on three levels: definitely relevant, possibly relevant and irrelevant. There are a total of 16,140 query-document pairs with relevance judgments.

In the LETOR OHSUMED collection, each document is defined as below:

- *I* Sequential identifier
- *U* MEDLINE identifier
- *M* Human-assigned MeSH terms
- *T* Title
- *P* Publication type
- *W* Abstract
- *A* Author
- *S* Source

For each query in the LETOR OHSUMED collection, the patient and topic information are defined in the following way:

- *I* Sequential identifier
- *B* Patient information
- *W* Information request

Each query-document pair in LETOR OHSUMED data set consists of a vector of features which includes both "low-level" and "high-level" features used in document retrieval.[4,52,53] The features are summarized in Tables II and III. Low-level features include term frequency, inverse document frequency, document length, and

Table II. Low-level features for LETOR OHSUMED data set.

Feature	Description
L1	$\sum_{q_i \in q \cap d} c(q_i, d)$
L2	$\sum_{q_i \in q \cap d} \log(c(q_i, d) + 1)$
L3	$\sum_{q_i \in q \cap d} \frac{c(q_i, d)}{\lvert d \rvert}$
L4	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{\lvert d \rvert} + 1\right)$
L5	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$
L6	$\sum_{q_i \in q \cap d} c(q_i, C) \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$
L7	$\sum_{q_i \in q \cap d} \log\left(\log\left(\frac{\lvert C \rvert}{df(q_i)}\right)\right)$
L8	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{c(q_i, C)} + 1\right)$
L9	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{\lvert d \rvert} \frac{\lvert C \rvert}{c(q_i, C)} + 1\right)$
L10	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{\lvert d \rvert} \log\left(\frac{\lvert C \rvert}{df(q_i)}\right) + 1\right)$

Table III. High-level features for LETOR OHSUMED data set.

Feature	Description
H1	$BM25$
H2	$\log(BM25)$
H3	LMIR with DIR smoothing
H4	LMIR with JM smoothing
H5	LMIR with ABS smoothing

their combinations. High-level features include the outputs of BM25 and LMIR algorithms. In total, there were 25 features: 10 from title, 10 from abstract, and 5 from "title and abstract." We can see that some classic information retrieval models such as BM25 and LMIR are within the feature set of learning to rank method. By including the output of the model as a feature, learning to rank method can easily incorporate any new retrieval model, which is highly desired for real search engines.

7. EXPERIMENTAL EVALUATIONS

7.1. Evaluation Criteria

In information retrieval, ranking results are usually evaluated in terms of performance measures such as Mean Average Precision (MAP)[4] and Normalized Discounted Cumulative Gain (NDCG).[54] MAP is widely used in IR when there are two ranks: positive (relevant) and negative (irrelevant), and it measures the precision of ranking results.

$$P@n = \frac{\text{Number of relevant instances in top } n}{n} \tag{15}$$

Precision at n measures accuracy of top-n results for a query. Given a query q_i, average precision is defined as the average of precision after each relevant instance is retrieved, and its average precision $AvgP_i$ is defined as

$$AvgP_i = \sum_{n=1}^{N} \frac{P@n \times pos(n)}{\text{Number of relevant instances}} \tag{16}$$

where n is position, N is number of instances retrieved, and $pos(n)$ is a binary function indicating whether the instance at position n is positive. MAP is defined as the mean of average precisions over a set of queries.

NDCG is also a measure commonly used in IR, when there are more than two categories in relevance ranking. While evaluating a ranking list, NDCG follows two rules: (1) Highly relevant documents are more valuable than marginally relevant documents; (2) the lower the ranking position of a document, the less the value of the document for the user. Given a query q_i, the NDCG score at position n in the ranking of documents is defined as

$$NDCG@n = Z_n \sum_{j=1}^{n} (2^{R(j)} - 1)\big/\log(1 + j) \tag{17}$$

where $R(j)$ is the rating of the jth document and Z_n is a normalization constant. Z_n is chosen to guarantee that a perfect ranking's NDCG score at position n is 1.

7.2. Experimental Results

In the experiments, we use LibSVM[55] as SVM tools. The experiments are conducted on a PC with 3.0 GHz CPU and 1G memory. To conduct fivefold cross-validation, we partition the LETOR OHSUMED collection into five parts, denoted as D1, D2, D3, D4, and D5. For each fold, we use three parts for training, one part for validation, and the remaining part for testing. All results reported in this section are obtained by averaging five trials.

To show the advantage of learning to rank over traditional model, we present the experimental result of BM25 and some SVM-based learning to rank methods, such as RankSVM and MHR. Table IV shows the $P@n$ ($n = 1, 2, \ldots, 10$) value of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set.

Table IV. The $P@n$ value of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set.

	$P@1$	$P@2$	$P@3$	$P@4$	$P@5$	$P@6$	$P@7$	$P@8$	$P@9$	$P@10$
BM25	0.453	0.442	0.438	0.411	0.391	0.372	0.367	0.352	0.331	0.310
RankSVM	0.634	0.610	0.590	0.592	0.570	0.543	0.533	0.525	0.520	0.507
MHR	0.644	0.600	0.574	0.560	0.539	0.533	0.530	0.522	0.501	0.482
QoRank-WV	0.667	0.653	0.645	0.615	0.600	0.585	0.571	0.564	0.543	0.512

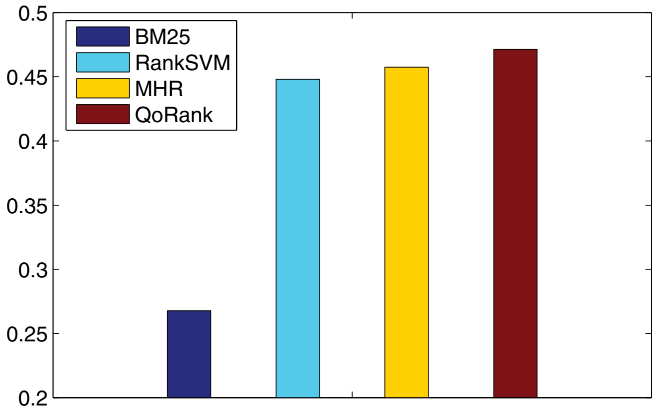


Figure 4. The MAP value of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set.

Figure 4 describes the comparison of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set in terms of MAP, which reflects the average ranking precision of all of the query results. In MAP calculation, we define the "definitely relevant" ratings as positive and the other two categories as negative. For MAP value, by combining BM25 with other features, the learning to rank algorithms significantly outperforms the single feature of BM25. It validates the capability of combining large number of features and automatic parameter tuning of leaning to the rank method. QoRank obtains the best performance compared with the other two SVM-based ranking methods, outperforms MHR with almost 2%, and outperforms RankSVM with more than 3%.

Figure 5 shows the NDCG value of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set. According to the rules of NDCG measure, highly

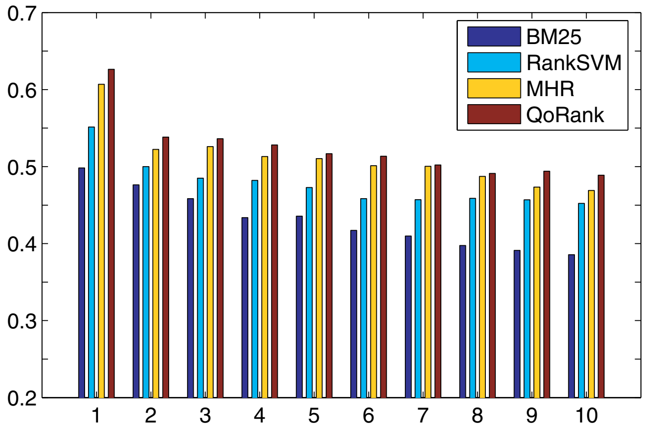


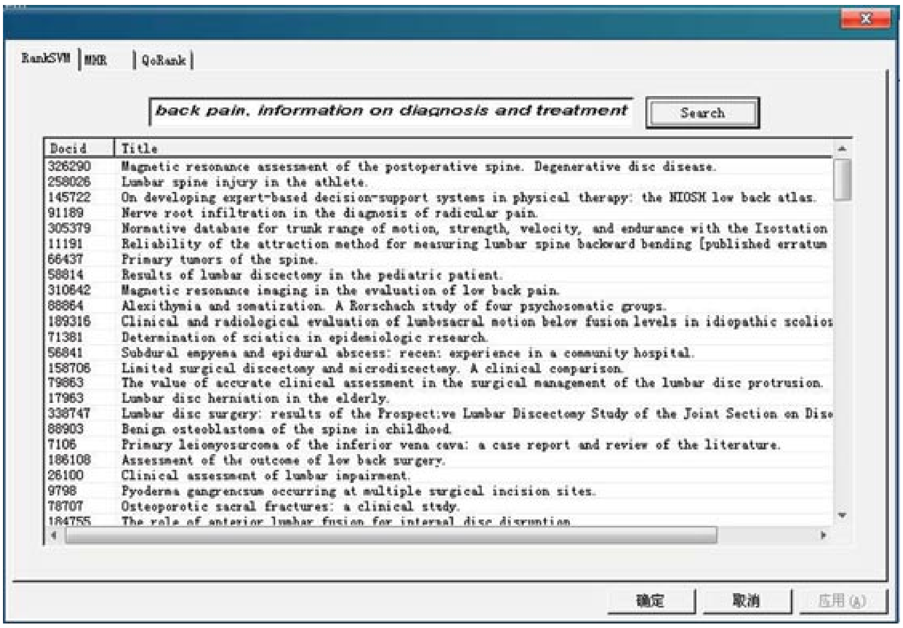
Figure 5. The NDCG value of BM25, RankSVM, MHR, and QoRank on the LETOR OHSUMED data set.

Table V. The relevance of top 12 results by RankSVM, MHR, and QoRank for the Query 43 of LETOR OHSUMED data set.

	1	2	3	4	5	6	7	8	9	10	11	12
RankSVM	N	P	P	P	N	P	P	N	P	N	D	N
MHR	D	N	P	N	D	D	D	N	P	P	P	N
QoRank	D	N	D	D	D	N	D	P	D	P	N	N

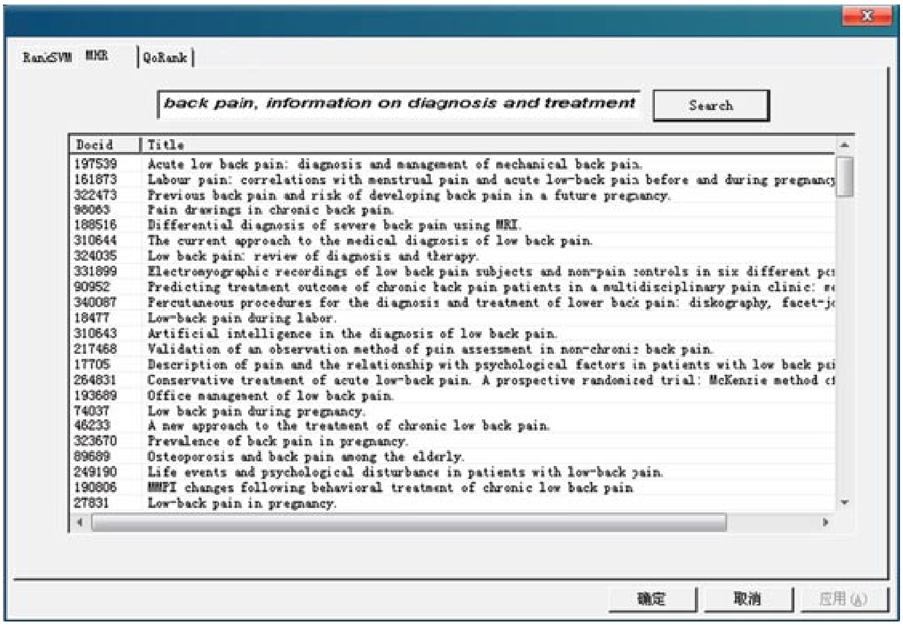
relevant documents are more valuable than marginally relevant document; the lower the ranking position of a document, the less the value of the document for the user, we present the NDCG value of top 10 positions. From Figure 5, we can see that the QoRank significantly outperforms other methods on NDCG@1. In most position, QoRank outperforms BM25 with more than 6%, outperforms RankSVM with 2%, and outperforms MHR with more than 1%. These results indicate that not only the relation but also the LSE-based weighted aggregation method help to improve the ranking performance.

We have developed a prototype research system with OHSUMED data set on the platform of Visual C++ 2008 and SQL Server 2005, including three ranking methods: RankSVM, MHR, and QoRank. Figures 6–8 show the ranking results of RankSVM, MHR, and QoRank for Query 43 "back pain, information on diagnosis and treatment" separately. Table V shows the relevance level of top 12 documents.



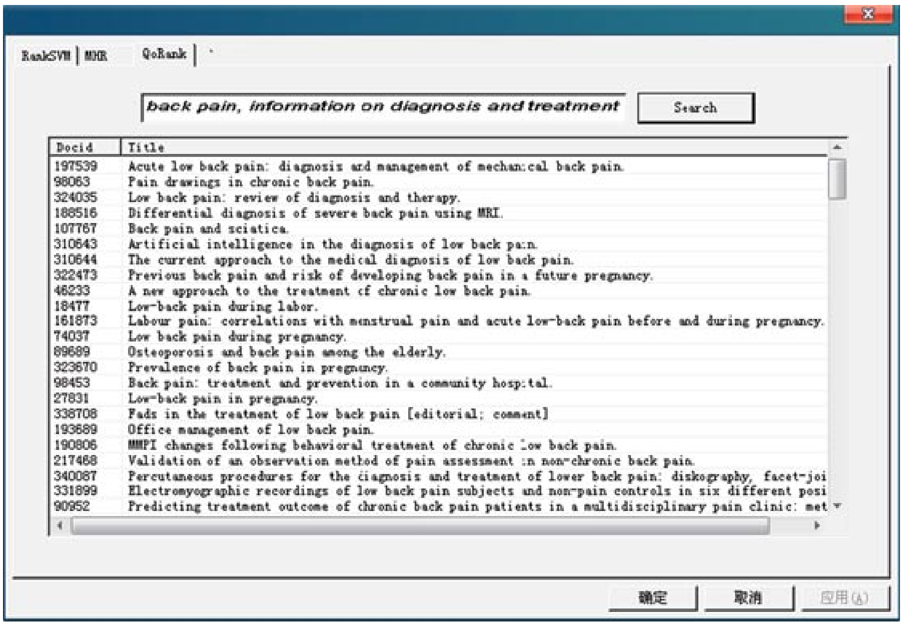
Docid	Title
326290	Magnetic resonance assessment of the postoperative spine. Degenerative disc disease.
258026	Lumbar spine injury in the athlete.
145722	On developing expert-based decision-support systems in physical therapy: the NIOSH low back atlas.
91189	Nerve root infiltration in the diagnosis of radicular pain.
305379	Normative database for trunk range of motion, strength, velocity, and endurance with the Isostation
11191	Reliability of the attraction method for measuring lumbar spine backward bending [published erratum
66437	Primary tumors of the spine.
58814	Results of lumbar discectomy in the pediatric patient.
310642	Magnetic resonance imaging in the evaluation of low back pain.
88864	Alexithymia and somatization. A Rorschach study of four psychosomatic groups.
189316	Clinical and radiological evaluation of lumbosacral motion below fusion levels in idiopathic scolios
71381	Determination of sciatica in epidemiologic research.
56841	Subdural empyema and epidural abscess: recen: experience in a community hospital.
158706	Limited surgical discectomy and microdiscectomy. A clinical comparison.
79863	The value of accurate clinical assessment in the surgical management of the lumbar disc protrusion.
17963	Lumbar disc herniation in the elderly.
338747	Lumbar disc surgery: results of the Prospective Lumbar Discectomy Study of the Joint Section on Dis
88903	Benign osteoblastoma of the spine in childhood.
7106	Primary leiomyosarcoma of the inferior vena cava: a case report and review of the literature.
186108	Assessment of the outcome of low back surgery.
26100	Clinical assessment of lumbar impairment.
9798	Pyoderma gangrenosum occurring at multiple surgical incision sites.
78707	Osteoporotic sacral fractures: a clinical study.
184755	The role of anterior lumbar fusion for internal disc disruption

Figure 6. Query results ranking by RankSVM for Query 43 of LETOR OHSUMED data set.



Docid	Title
197539	Acute low back pain: diagnosis and management of mechanical back pain.
161873	Labour pain: correlations with menstrual pain and acute low-back pain before and during pregnanc
322473	Previous back pain and risk of developing back pain in a future pregnancy.
98063	Pain drawings in chronic back pain.
188516	Differential diagnosis of severe back pain using MRI.
310644	The current approach to the medical diagnosis of low back pain.
324035	Low back pain: review of diagnosis and therapy.
331899	Electromyographic recordings of low back pain subjects and non-pain controls in six different po
90952	Predicting treatment outcome of chronic back pain patients in a multidisciplinary pain clinic: m
340087	Percutaneous procedures for the diagnosis and treatment of lower back pain: diskography, facet-jo
18477	Low-back pain during labor.
310643	Artificial intelligence in the diagnosis of low back pain.
217468	Validation of an observation method of pain assessment in non-chronic back pain.
17705	Description of pain and the relationship with psychological factors in patients with low back pa
264831	Conservative treatment of acute low-back pain. A prospective randomized trial: McKenzie method o
193689	Office management of low back pain.
74037	Low back pain during pregnancy.
46233	A new approach to the treatment of chronic low back pain.
323670	Prevalence of back pain in pregnancy.
89689	Osteoporosis and back pain among the elderly.
249190	Life events and psychological disturbance in patients with low-back pain.
190806	MMPI changes following behavioral treatment of chronic low back pain
27831	Low-back pain in pregnancy.

Figure 7. Query results ranking by MHR for Query 43 of LETOR OHSUMED data set.



Docid	Title
197539	Acute low back pain: diagnosis and management of mechanical back pain.
98063	Pain drawings in chronic back pain.
324035	Low back pain: review of diagnosis and therapy.
188516	Differential diagnosis of severe back pain using MRI.
107767	Back pain and sciatica.
310643	Artificial intelligence in the diagnosis of low back pain.
310644	The current approach to the medical diagnosis of low back pain.
322473	Previous back pain and risk of developing back pain in a future pregnancy.
46233	A new approach to the treatment of chronic low back pain.
18477	Low-back pain during labor.
161873	Labour pain: correlations with menstrual pain and acute low-back pain before and during pregnancy.
74037	Low back pain during pregnancy.
89689	Osteoporosis and back pain among the elderly.
323670	Prevalence of back pain in pregnancy.
98453	Back pain: treatment and prevention in a community hospital.
27831	Low-back pain in pregnancy.
338708	Fads in the treatment of low back pain [editorial; comment]
193689	Office management of low back pain.
190806	MMPI changes following behavioral treatment of chronic low back pain.
217468	Validation of an observation method of pain assessment in non-chronic back pain.
340087	Percutaneous procedures for the diagnosis and treatment of lower back pain: diskography, facet-joi
331899	Electromyographic recordings of low back pain subjects and non-pain controls in six different posi
90952	Predicting treatment outcome of chronic back pain patients in a multidisciplinary pain clinic: met

Figure 8. Query results ranking by QoRank for Query 43 of LETOR OHSUMED data set.

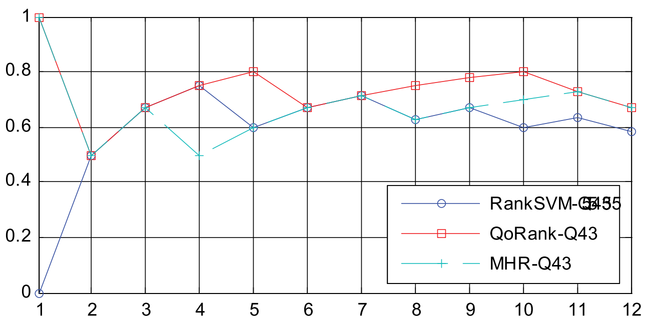


Figure 9. The $P@n$ Value of RankSVM, MHR, and QoRank methods for Query 43 of LETOR OHSUMED data set.

The character "D" in red represents "definitely relevant," character "P" in blue represents "partially relevant," and character "N" in black represents "not relevant."

It is easy to find that both QoRank and MHR get eight "definitely relevant" or "partially relevant" documents in the top 12 items, and the corresponding number of RankSVM is 7. In the ranking results of QoRank, there are six "definitely relevant" documents and all ranked in the first ten positions. MHR gets four "definitely relevant" documents and RankSVM gets only one. According to the rule that the document in the higher ranking position is much easier to be checked by user, QoRank will be more competitive in real-world applications. The results indicate that constructing ranking models depending on the query is helpful to improve the ranking accuracy. Figure 9 shows the $P@n$ value of RankSVM, MHR, and QoRank for Query 43.

As aforementioned, QoRank utilizes the order relations between ranks to generate the query-dependent base decision functions. In this way, fewer instance pairs are used and the computation time in training is greatly reduced. We also compare the training time between QoRank, MHR, and RankSVM. Table VI shows the results. In each trial, the total training time of QoRank is only 1/30 of RankSVM and 1/10 of MHR, which corresponds with the analysis in Section 5. In summary, the experimental results in this section demonstrate the effectiveness of QoRank: It has good ranking performance with low training complexity.

Table VI. The training time for the LETOR OHSUMED data set.

		MHR				
Minutes	RankSVM	$\omega_{1,2}$	$\omega_{2,3}$	$\omega_{1,3}$	sum	QoRank
Trial 1	**1728.64**	306.16	51.64	228.43	**586.23**	**67.721**
Trial 2	**2013.57**	456.41	38.72	283.57	**778.7**	**79.182**
Trial 3	**1845.36**	329.31	35.44	260.85	**625.60**	**58.726**
Trial 4	**906.53**	157.74	17.23	139.27	**314.24**	**39.270**
Trial 5	**1113.72**	202.01	14.63	163.51	**380.15**	**43.375**
Average	**1521.56**	290.33	31.53	215.13	**536.98**	**57.6548**

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a two-stage learning to rank method QoRank for information retrieval. In the first stage, it performs the training depending on query and using the relation between ranks to optimize the training process. In the second stage, we propose a LSE-based weighted method to aggregate the rank lists of base decision functions for the final ranking. Experimental results on LETOR OHSUMED data set show the great importance of building ranking models depending on query. It not only makes the model construction more adaptable but also reduces the training time.

There are several avenues for future research. One possible direction is the base decision function construction. In this paper, we have only investigated one type of method. A comprehensive investigation of base decision function construction is a natural next step. The optimization of the loss function needs further investigation. Developing a more effective ranking aggregation method is also to be included.

Appendix

Proof of $\varsigma_{\text{QoRank}} < \varsigma_{\text{MHR}} < \varsigma_{\text{RankSVM}}$

$$\begin{aligned}\varsigma_{\text{RankSVM}} &= O\left(M_{\text{RankSVM}}^3 + aM_{\text{RankSVM}}^2\right)\\ &= O\left(\left[\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(|R_i| \times |R_j|)\right]^3 + a\left[\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}(|R_i| \times |R_j|)\right]^2\right)\end{aligned}$$

$$\begin{aligned}\varsigma_{\text{MHR}} &= \sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r} O\left(\left(M_{\text{MHR}}^{i,j}\right)^3 + a\left(M_{\text{MHR}}^{i,j}\right)^2\right)\\ &\approx O\left(\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}\left[\left(M_{\text{MHR}}^{i,j}\right)^3 + a\left(M_{\text{MHR}}^{i,j}\right)^2\right]\right)\\ &= O\left(\sum_{i=1}^{N_r-1}\sum_{j=i+1}^{N_r}\left[(|R_i| \times |R_j|)^3 + a(|R_i| \times |R_j|)^2\right]\right)\end{aligned}$$

$$\varsigma_{\text{QoRank}} = \sum_{\substack{i=1\\ j\rhd i}}^{N_r-1}\sum_{q=1}^{N_q} O\left(\left(M_{\text{QoRank}}^{i,j}\right)^3 + a\left(M_{\text{QoRank}}^{i,j}\right)^2\right)$$

$$\approx O\left(\sum_{\substack{i=1\\ j\rhd i}}^{N_r-1}\sum_{q=1}^{N_q}\left[\left(M_{\text{QoRank}}^{i,j}\right)^3 + a\left(M_{\text{QoRank}}^{i,j}\right)^2\right]\right)$$

$$= O\left(\sum_{\substack{i=1\\ j\rhd i}}^{N_r-1}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_j^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_j^q\right|\right)^2\right]\right)$$

First, we can prove that $\varsigma_{\text{QoRank}} < \varsigma_{\text{MHR}}$.

$$\because (a_1 \times a_2 \times \cdots \times a_i \times \cdots \times a_p)^m + (b_1 \times b_2 \times \cdots \times b_j \times \cdots \times b_q)^m$$

$$< \left[(a_1 + a_2 + \cdots + a_i + \cdots + a_p) \times (b_1 + b_2 + \cdots + b_j + \cdots + b_q)\right]^m, m > 1$$

$$\therefore \sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^2\right] < \left[\left(\sum_{q=1}^{N_q}\left|R_i^q\right|\right)\right.$$

$$\left.\times\left(\sum_{q=1}^{N_q}\left|R_{i+k}^q\right|\right)\right]^3 + a\left[\left(\sum_{q=1}^{N_q}\left|R_i^q\right|\right) \times \left(\sum_{q=1}^{N_q}\left|R_{i+k}^q\right|\right)\right]^2$$

$$\because \sum_{q=1}^{N_q}\left|R_i^q\right| = |R_i|$$

$$\therefore \sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^2\right] < \left(\left|R_i\right| \times \left|R_{i+k}\right|\right)^3$$

$$+ a(|R_i| \times |R_{i+k}|)^2$$

$$\therefore \sum_{i=1}^{N_r-1}\sum_{k=1}^{N_r-i}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+k}^q\right|\right)^2\right]$$

$$< \sum_{i=1}^{N_r}\sum_{k=1}^{N_r-i}\left[(|R_i| \times |R_{i+k}|)^3 + a(|R_i| \times |R_{i+k}|)^2\right]$$

$$\text{and} \sum_{i=1}^{N_r-1}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^2\right]$$

$$\ll \sum_{i=1}^{N_r-1}\sum_{k=1}^{N_r-i}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times |R_{i+k}^q|\right)^3 + a(|R_i^q| \times |R_{i+k}^q|)^2\right]$$

$$\therefore \sum_{i=1}^{N_r-1}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^2\right]$$

$$\ll \sum_{i=1}^{N_r}\sum_{k=1}^{N_r-i}\left[\left(\left|R_i\right| \times \left|R_{i+k}\right|\right)^3 + a\left(\left|R_i\right| \times \left|R_{i+k}\right|\right)^2\right]$$

$$\therefore O\left(\sum_{i=1}^{N_r-1}\sum_{q=1}^{N_q}\left[\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^3 + a\left(\left|R_i^q\right| \times \left|R_{i+1}^q\right|\right)^2\right]\right)$$

$$\ll O\left(\sum_{i=1}^{N_r}\sum_{k=1}^{N_r-i}[(|R_i| \times |R_{i+k}|)^3 + a(|R_i| \times |R_{i+k}|)^2]\right)$$

$\therefore \varsigma_{\text{QoRank}} < \varsigma_{\text{MHR}}$.

Second, $\varsigma_{\text{MHR}} < \varsigma_{\text{RankSVM}}$.

$$\because a_1^m + a_2^m + \cdots + a_i^m + \cdots a_n^m < (a_1 + a_2 + \cdots + a_i + \cdots a_n)^m, m > 1$$

$$\therefore \sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}[(|R_i| \times |R_k|)^3 + a(|R_i| \times |R_k|)^2] < \left[\sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}(|R_i| \times |R_k|)\right]^3$$

$$+ a[\sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}(|R_i| \times |R_k|)]^2$$

$$\therefore O\left(\sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}[(|R_i| \times |R_k|)^3 + a(|R_i| \times |R_k|)^2]\right)$$

$$< O\left(\left[\sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}(|R_i| \times |R_k|)\right]^3 + a\left[\sum_{i=1}^{N_r-1}\sum_{k=i+1}^{N_r}(|R_i| \times |R_k|)\right]^2\right)$$

$\therefore \varsigma_{\text{MHR}} < \varsigma_{\text{RankSVM}}$.

Then we can get $\varsigma_{\text{QoRank}} < \varsigma_{\text{MHR}} < \varsigma_{\text{RankSVM}}$. □

Acknowledgments

We would like to thank Prof. Jiawei Han for his comments on our paper. This work was supported by the Nation 863 High Technology Program of China (2008AA01Z131) and Natural Science Basic Research Plan in Shaanxi Province of China (No. SJ08-ZT14).

References

1. Cao Y, Xu J, Liu TY, Li H, Huang Y, Hon HW. Adapting ranking SVM to document retrieval. In: Proc 29th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2006. pp 186–193.
2. Harrington EF. Online ranking/collaborative filtering using the perceptron algorithm. In: Machine Learning-International Workshop Then Conference-; 2003. Vol 20, pp 250–257.
3. Collins M. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In: Proc 40th Annual Meeting on Association for Computational Linguistics. July, 2002. pp 7–12.
4. Baeza-Yates R and Ribeiro-Neto B. Modern information retrieval. ACM Press, New York 1999.
5. Chirita PA, Diederich J, W Nejdl. MailRank: using ranking for spam detection. In: Proc 14th ACM Int Conf on Information and Knowledge Management. New York: ACM; 2005. pp 373–380.
6. B Pang. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL 2005. The Association for Computer Linguistics. 2005.
7. Dave K, Lawrence S, Pennock DM. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: Proc 12th Int Conf on World Wide Web. New York: ACM; 2003. pp 519–528.
8. Gyongyi Z, Garcia-Molina H, Pedersen J. Combating Web spam with trustrank. In: Proc 13th Int Conf on Very Large Data Bases. VLDB Endowment: Toronto, Canada; 2004. pp 576–587.
9. Witten IH, Moffat A, Bell TC. Managing gigabytes: Compressing and indexing documents and images. San Francisco, CA: Morgan Kaufmann; 1999.
10. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. Commun ACM 1975;18(11):613–620.
11. Harman D. Ranking algorithms. Information Retrieval: Data Structures & Algorithms. Pearson Education, India 1992. pp 363–392.
12. Robertson SE, Jones KS. Relevance weighting of search terms. J Am Soc for Inform Sci Technol 1976;27(3):129–146.
13. Robertson SE, Walker S, Jones S, Hancock-Beaulieu MM, Gatford M. Okapi at TREC-4. In: Proc 4th Text Retrieval Conf. Diane Pub Co., Pennsylvania 1996. pp 73–97.
14. Page L, Brin S, Motwani R, Winograd T. The pagerank citation ranking: bringing order to the Web. Stanford InfoLab, California 1998.
15. Kleinberg JM. Authoritative sources in a hyperlinked environment. J ACM 1999;46(5):604–632.
16. Herbrich R, Graepel T, Obermayer K. Large margin rank boundaries for ordinal regression. In: KDD'02: Proc 8th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM; 2002. pp 133–142.
17. Freund Y, Iyer R, Schapire RE, Singer Y. An efficient boosting algorithm for combining preferences. J Machine Learning Res 2003;4(11):933–969.
18. Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G. Learning to rank using gradient descent. In: Proc 22nd Int Conf on Machine Learning. New York: ACM; 2005. pp 89–96.
19. Qin T, Zhang XD, Wang DS, Liu TY, Lai W, Li H. Ranking with multiple hyperplanes. In: Proc 30th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2007. pp 279–286.

20. Xu J, Li H. Adarank: a boosting algorithm for information retrieval. In: Proc 30th Annual Int ACM SIGIR Conf Research and Development in Information Retrieval. New York: ACM; 2007. pp 391–398.
21. Cao Z, Qin T, Liu TY, Tsai MF, Li H. Learning to rank: from pairwise approach to listwise approach. In: Proc 24th Int Conf on Machine Learning. New York: ACM; 2007. pp 129–136.
22. Burges C, Ragno R, Le QV. Learning to rank with nonsmooth cost functions. MIT Prsss, Massachusetts; 2007. pp 193–200.
23. Tsai MF, Liu TY, Qin T, Chen HH, Ma WY. Frank: a ranking method with fidelity loss. In: Proc 30th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2007. pp 383–390.
24. Xia F, Liu TY, Wang J, Zhang W, Li H. Listwise approach to learning to rank: theory and algorithm. In: Proc 25th Int Conf on Machine Learning New York: ACM; 2008. pp 1192–1199.
25. Zheng Z, Zha H, Zhang T, Chapelle O, Chen K, Sun G. A general boosting method and its application to learning ranking functions for Web search. In: Proc 21st Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT Press; 2008. pp 1697–1704.
26. Geng X, Liu TY, Qin T, Arnold A, Li H, Shum HY. Query dependent ranking using k-nearest neighbor. In: Proc 31st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2008. pp 115–122.
27. Qin T, Liu TY, Zhang XD, Wang DS, Xiong WY, Li H. Learning to rank relational objects and its application to web search. In: Proc 17th Int Conf on World Wide Web. New York: ACM; 2008. pp 407–416.
28. Veloso AA, Almeida HM, Gonçalves MA, Meira Jr W. Learning to rank at query-time using association rules. In: Proc 31st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, New York: ACM; 2008. pp 267–274.
29. Rose DE, Levinson D. Understanding user goals in web search. In: Proc 13th Int Conf on World Wide Web. New York: ACM; 2004. pp 13–19.
30. Lafferty J, Zhai CX. Document language models, query models, and risk minimization for information retrieval. In: Proc 24th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2001. pp 111–119.
31. Ponte JM, Croft WB. A language modeling approach to information retrieval. In: Proc 21st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 1998. pp 275–281.
32. Zobel J, Moffat A. Exploring the similarity space. In: ACM SIGIR Forum. New York: ACM; 1998. Vol 32, p 34.
33. Nallapati R. Discriminative models for information retrieval. In: Proc 27th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2004. pp 64–71.
34. Crammer K, Singer Y. A new family of online algorithms for category ranking. In: Proc 25th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, New York: ACM; 2002. pp 151–158.
35. Joachims T. Optimizing search engines using clickthrough data. In: Proc 8th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM; 2002. pp 133–142.
36. Yue Y, Finley T, Radlinski F, Joachims T. A support vector method for optimizing average precision. In: Proc 30th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2007. p 278.
37. Friedman JH. Greedy function approximation: a gradient boosting machine. Annals Stat. Institute of Mathematical Statistics, Maryland 2001; 29:1189–1232.
38. Friedman JH. Stochastic gradient boosting. Comput Stat Data Anal 2002;38(4):367–378.
39. Aslam JA, Montague M. Models for metasearch. In: Proc 24th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2001. pp 276–284.

40. Callan JP, Lu Z, Croft WB. Searching distributed collections with inference networks. In: Proc 18th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, New York: ACM; 1995. pp 21–28.
41. Fagin R, Wimmers EL. Incorporating user preferences in multimedia queries. In: Proc 6th Int Conf on Database Theory. London: Springer-Verlag, 1997. pp 247–261.
42. Fox JA, Shaw E. Combination of multiple sources: The trec-2 interactive track matrix experiment. In: ACM SIGIR-94. 1994.
43. Lee JH. Analyses of multiple evidence combination. In: ACM SIGIR Forum. New York: ACM; 1997. Vol 31, pp 267–276.
44. Manmatha R, Rath T, Feng F. Modeling score distributions for combining the outputs of search engines. In: Proc 24th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2001. pp 267–275.
45. Vogt CC, Cottrell GW. Fusion via a linear combination of scores. Inform Retrieval 1999;1(3):151–173.
46. Voorhees E, Gupta NK, Johnson-Laird B, Princeton NJ. The collection fusion problem. In: Overview of the Third Text REtrieval Conference (TREC-3). Diane Pub Co; Pennsylvania; 1995. pp 95–104.
47. Dwork C, Kumar R, Naor M, Sivakumar D. Rank aggregation methods for the web. In: Proc 10th Int Conf on World Wide Web, New York: ACM; 2001. pp 613–622.
48. Yager RR, Rybalov A. On the fusion of documents from multiple collection information retrieval systems. J Am Soc Inform Sci Technol 1998;49(13):1177–1184.
49. Burges C. A tutorial on support vector machines for pattern recognition. Data Mining Knowl Discovery 1998;2(2):121–167.
50. Liu TY, Xu J, Qin T, Xiong W, Li H. Letor: benchmark dataset for research on learning to rank for information retrieval. In: Proc SIGIR 2007 Workshop on Learning to Rank for Information Retrieval. ACM Press, New York; 2007. pp 3–10.
51. Hersh W, Buckley C, Leone TJ, Hickam D. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In: Proc 17th Annual Int ACM SIGIR Conf Research and Development in Information Retrieval. New York: Springer-Verlag; 1994. pp 192–201.
52. Robertson S, Hull DA. The TREC-9 filtering track final report. NIST Special Publication SP, TREC, Maryland; 2001. pp 25–40.
53. Zhai CX and Lafferty J. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In: Proc 24th Annual Int ACM SIGIR Conf Research and Development in Information Retrieval New York: ACM; 2001. pp 334–342.
54. Jarvelin K, Kekalainen J. IR evaluation methods for retrieving highly relevant documents. In: Proc 23rd Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM; 2000. pp 41–48.
55. Chang CC, Lin CJ. LIBSVM: a library for support vector machines; 2001. Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm.
56. Yager RR, Rybalov A. On the fusion of documents from multiple collection information retrieval systems. J Am Soc Inform Sci Technol 1998;49(13):1177–1184.