Adapting Ranking SVM to Document Retrieval

Yunbo CAO[1], Jun XU[2], Tie-Yan LIU[1], Hang LI[1], Yalou HUANG[2], Hsiao-Wuen HON[1]

[1]Microsoft Research Asia,
No.49 Zhichun Road, Haidian District
Beijing, China, 100080
{yunbo.cao, tyliu, hangli, hon}@microsoft.com

[2]College of Software, Nankai University,
No.94 Weijin Road, Nankai District
Tianjin, China, 300071
nkxj@hotmail.com, yellow@nankai.edu.cn

ABSTRACT

The paper is concerned with applying learning to rank to document retrieval. Ranking SVM is a typical method of learning to rank. We point out that there are two factors one must consider when applying Ranking SVM, in general a "learning to rank" method, to document retrieval. First, correctly ranking documents on the top of the result list is crucial for an Information Retrieval system. One must conduct training in a way that such ranked results are accurate. Second, the number of relevant documents can vary from query to query. One must avoid training a model biased toward queries with a large number of relevant documents. Previously, when existing methods that include Ranking SVM were applied to document retrieval, none of the two factors was taken into consideration. We show it is possible to make modifications in conventional Ranking SVM, so it can be better used for document retrieval. Specifically, we modify the "Hinge Loss" function in Ranking SVM to deal with the problems described above. We employ two methods to conduct optimization on the loss function: gradient descent and quadratic programming. Experimental results show that our method, referred to as Ranking SVM for IR, can outperform the conventional Ranking SVM and other existing methods for document retrieval on two datasets.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

Information retrieval, loss function, Ranking SVM

1. INTRODUCTION

Ranking functions in document retrieval traditionally use a small number of features (e.g., term frequency, inversed document frequency, and document length), which makes it possible to empirically tune ranking parameters [20]. Recently, however, a growing number of features such as structural features, title text, and anchor text, and query-independent features (e.g., PageRank and URL length) have proved useful in document retrieval while empirical tuning of ranking functions has become increasingly difficult.

Fortunately, in recent years more and more human-judged document retrieval results have become available. This makes it possible to employ supervised learning methodologies in the tuning of ranking functions. Many such efforts have been made using these approaches.

In one of such effort, document retrieval is formalized as classification of documents into two categories: relevant and irrelevant. Nallapati [12], for example, formalizes document retrieval as binary classification and solves the classification problem using Support Vector Machines (SVM) and Maximum Entropy (ME).

In another approach, document retrieval is formalized as a "learning to rank" problem in which documents are mapped into several ordered categories (ranks). OHSUMED [9] is a data collection that contains multiple data categories or ranks: definitely relevant, partially relevant, and irrelevant. Herbrich et al. [8], for instance, propose a method of learning to rank on the basis of SVM and apply their method to document retrieval. We refer to their method as Ranking SVM (or conventional Ranking SVM) in this paper. Specifically, Ranking SVM formalizes learning to rank as a problem of classifying *instance pairs* into two categories (correctly ranked and incorrectly ranked). Other methods within this approach have also been proposed [1, 19, 24].

We explore the problem of applying learning to rank to document retrieval and propose a new learning method on the basis of Ranking SVM. We refer to the method as Ranking SVM for IR.

We note two important factors to take into general consideration when applying Ranking SVM in a learning method for ranking documents being retrieved. Unfortunately, they are ignored in the existing methods, such as Ranking SVM.

(1) To have high accuracy on top-ranked documents is crucial for an IR system. Analysis on click-through data from search engines shows that users usually click on top-ranked documents among returned search results [16, 17, 18]. The Normalized Discounted Cumulated Gain (NDCG) measure [10] used in evaluation of document retrieval also reflects this preference. Therefore, it is necessary to perform training so that the top-ranked results (equivalently the ranked results with regard to the highest ranks) are generally accurate. However, in existing learning methods such as Ranking SVM, the losses (penalties) of incorrect ranking between higher ranks and lower ranks and incorrect ranking among lower ranks are defined the same.

(2) In reality some queries may have many relevant documents, while others may have a few. If we treat training data from different queries as equal, then the model trained is biased toward queries with many relevant documents. Therefore, it is necessary to put higher weight on data from queries with fewer relevant documents. However, in existing learning methods such as Ranking SVM, the losses (penalties) from incorrect rankings from different queries are defined the same.

To deal with the problems above, we propose using a new loss function for Ranking SVM used in document retrieval. The loss function is a modification of the Hinge Loss function in Ranking SVM. Thus our method can be viewed as an adaptation of Ranking SVM to document retrieval. Specifically, we set varying losses for misclassification of instance pairs between different rank pairs and for incongruent losses in the misclassification of instance pairs from multiple queries. To reduce errors on top rankings, the loss function heavily penalizes errors with regard to the highest ranked documents. To increase influences of queries with fewer relevant documents, the loss function heavily penalizes errors from queries. We propose two learning methods to optimize the cost function: gradient descent and quadratic programming.

Experimental results indicate that Ranking SVM for IR can outperform existing methods, including Ranking SVM on two datasets. One dataset is OHSUMED [9], and the other is from a commercial web search engine.

2. RELATED WORK

2.1 Document Retrieval

One key question in document retrieval is how to rank documents based on their degrees of relevance to a query. Much effort has been placed on the development of ranking functions.

Traditionally, document retrieval methods only use a small number of features (e.g., term frequency, inversed document frequency, and document length). Thus, it is possible to empirically tune the parameters of ranking functions [20]. In that sense, the methods are unsupervised. Okapi BM25 [14] and language models for information retrieval (LMIR) [13, 21] are such methods.

Currently, additional features have proved useful for document retrieval, including structural features (e.g., title, anchor text, and URL) and query-independent features (e.g., PageRank and URL length). This increase in features makes empirical tuning of parameters difficult. The paradigm of employing supervised learning in construction of document retrieval models has drawn recent attention.

For instance, document retrieval is formalized as classification. Documents are judged within two categories: relevant and irrelevant. Nallapati [12] formalizes document retrieval as binary classification and solves it using SVM and Maximum Entropy. Gao et al. [5] propose employing discriminative training in creating a ranking model.

For another example, document retrieval is regarded as learning to rank. Herbrich et al. [8] propose addressing the issue by means of SVM and applying it to information retrieval. We refer to the method as Ranking SVM in this paper. Joachims [11] also applies the method to document retrieval. He utilizes click-through data to deduce pair-wise training data for learning Ranking SVM models. Burges et al. [1] propose employing relative entropy as a loss function and gradient descent as an algorithm to train a neural network model for document retrieval.

2.2 Learning to Rank

In learning to rank a number of categories are given and a total order is assumed to exist over the categories. Labeled instances are provided. Each instance is represented by a feature vector, and each label denotes a rank. Existing methods fall into two categories. They are referred to in this paper as "point-wise training" and "pair-wise training".

In point-wise training, each instance (and its rank) is used as an independent training example. The goal of learning is to correctly map instances into intervals.

For instance, Crammer et al. [4] propose a Perceptron-based learning algorithm called PRank. PRank trains a Perceptron model retaining not only a weight vector $\vec{w}$ but also a threshold vector $\vec{c}$. The objective of learning in PRank is to find a Perceptron model that successfully projects all the instances into k subintervals defined by $\vec{c}$. See also [2, 3, 6, 15].

In pair-wise training each instance pair is used as a training example and the goal of training is to correctly find the differences between ranks of instance pairs.

For instance, the Herbrich et al. Ranking SVM [8] is such a method. That model formalizes learning to rank as learning for classification on pairs of instances and tackles the classification issue by using SVM. See also [1, 15, 19].

For other work on learning to rank, refer to[22,23,24,25,26,27].

3. RANKING SVM

Assume that there exists an input space $X \in R^n$, where n denotes number of features. There exists an output space of ranks (categories) represented by labels $Y = \{r_1, r_2, \cdots, r_q\}$ where q denotes number of ranks. Further assume that there exists a total order between the ranks $r_q \succ r_{q-1} \succ \cdots \succ r_1$, where $\succ$ denotes a preference relationship. A set of ranking functions $f \in F$ exists and each of them can determine the preference relations between instances:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow f(\vec{x}_i) > f(\vec{x}_j), \tag{1}$$

Suppose that we are given a set of ranked instances $S = \{(\vec{x}_i, y_i)\}_{i=1}^{t}$ from the space $X \times Y$. The task here is to select the best function f^* from F that minimizes a given loss function with respect to the given ranked instances.

Herbrich et al. [8] propose formalizing the above learning problem as that of learning for classification *on pairs of instances.*

First, we assume that *f* is a linear function.

$$f_{\vec{w}}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle, \tag{2}$$

where $\vec{w}$ denotes a vector of weights and $\langle \cdot, \cdot \rangle$ stands for an inner product. Plugging (2) into (1) we obtain

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i - \vec{x}_j \rangle > 0 \tag{3}$$

Note that the relation $\vec{x}_i \succ \vec{x}_j$ between instance pairs $\vec{x}_i$ and $\vec{x}_j$ is expressed by a new vector $\vec{x}_i - \vec{x}_j$. Next, we take any instance pair and their relation to create a new vector and a new label. Let $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ denote the first and second instances, and let $y^{(1)}$ and $y^{(2)}$ denote their ranks, then we have

$$\left(\vec{x}^{(1)}-\vec{x}^{(2)}, z=\begin{cases}+1 & y^{(1)}\succ y^{(2)}\\ -1 & y^{(2)}\succ y^{(1)}\end{cases}\right) \qquad (4)$$

From the given training data set *S,* we create a new training data set *S′* containing ℓ labeled vectors.

$$S' = \{\vec{x}_i^{(1)} - \vec{x}_i^{(2)}, z_i\}_{i=1}^{l} \qquad (5)$$

Next, we take *S’* as classification data and construct a SVM model that can assign either positive label $z = +1$ or negative label $z = -1$ to any vector $\vec{x}^{(1)} - \vec{x}^{(2)}$.

Constructing the SVM model is equivalent to solving the following Quadratic Optimization problem:

$$\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$$
$$\text{subject to } \xi_i \geq 0, \quad z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle \geq 1-\xi_i \quad i=1,\cdots,\ell \qquad (6)$$

Note that the optimization is (6) is equivalent to that in (7), when $\lambda = \frac{1}{2C}$ [7].

$$\min_{\vec{w}} \sum_{i=1}^{\ell}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle\right]_+ + \lambda\|\vec{w}\|^2, \qquad (7)$$

where subscript “+” indicates the positive part. The first term is the so-called empirical Hinge Loss and the second term is regularizer. In Figure 2, the solid line denotes the Hinge Loss function of Ranking SVM.

Suppose that $\vec{w}^*$ is the weights in the SVM solution. Geometrically $\vec{w}^*$ forms a vector orthogonal to the hyperplane of Ranking SVM. We utilize $\vec{w}^*$ to form a ranking function $f_{\vec{w}^*}$ for ranking instances.

$$f_{\vec{w}^*}(\vec{x}) = \left\langle \vec{w}^*, \vec{x}\right\rangle \qquad (8)$$

When Ranking SVM is applied to document retrieval, an instance (feature vector) is created from one query-document pair [1]. Each feature is defined as a function of query and document. For instance, the feature of term frequency is determined by the number of times in which the query term occurs in the document. The instances from all queries are then combined in training. There is no difference in treatments toward the instances from different queries. Furthermore, there is no difference in treatments between instance pairs from different rank pairs.

4. PROBLEM ANALYSIS

There are two important factors to consider when applying learning to rank to IR. Let us take Ranking SVM as example and use data from the OHSUMED collection to look at the problems. OHSUMED [9] is data collection on document retrieval, as will be further explained in Section 6.

In learning of Ranking SVM for IR, each instance $\vec{x}$ is generated from one query-document pair and is labeled with one rank. There are three possible ranks: definitely relevant (r_3), partially relevant (r_2), and irrelevant (r_1). We plot the data to observe its tendencies. Specifically, we conduct principle component analysis (PCA) on the data and display its first and second principle components.

Figure 1 (a) shows the results for two randomly selected queries (query number 12 and 50). The horizontal axis and vertical axis represent the first and second principle components of the data. The circle, square, and triangle represent definitely relevant partially relevant and irrelevant instances. The solid and empty points represent the instances of query 12 and query 50.

The first factor is to intensify the training on the top rankings. It is a problem that Ranking SVM does not take into consideration. Figure 1 (b) shows the ranking models between pairs of ranks:

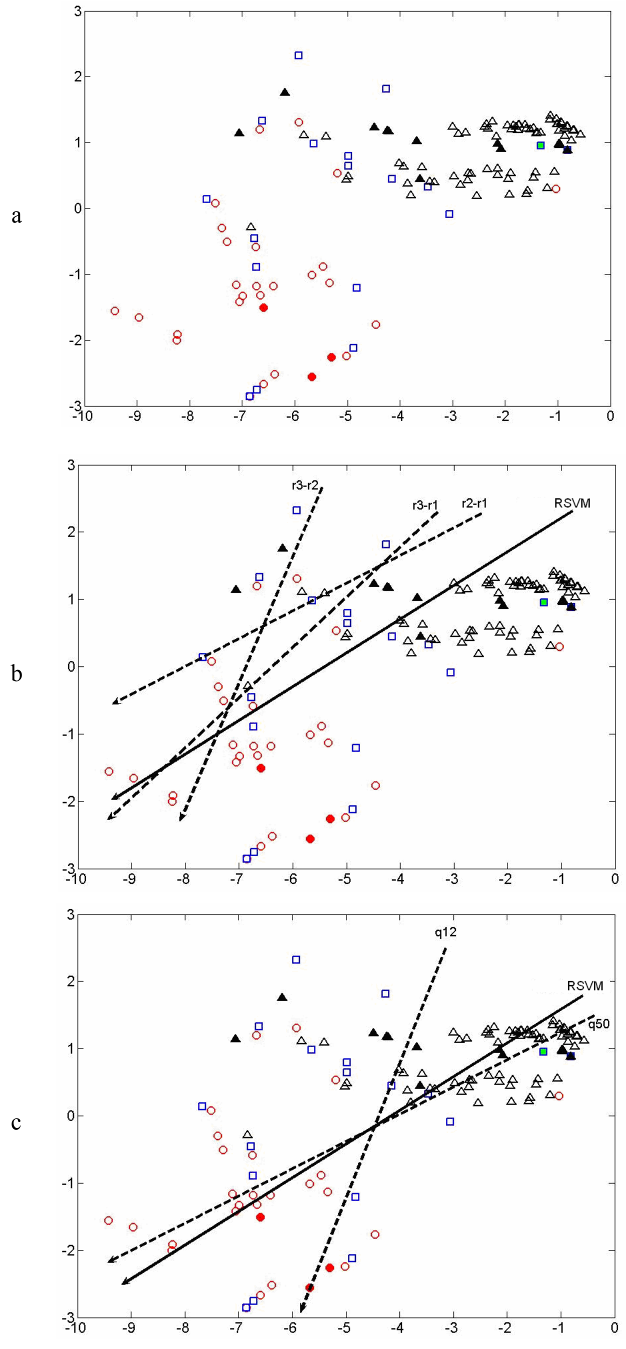

Figure 1: OSHUMED data (queries 12 and 50)

r_3-r_2, r_3-r_1, and r_2-r_1 (they are equivalent to the conventional SVM). It also gives the ranking model of Ranking SVM. Because Ranking SVM treats the instance pairs from all rank pairs equally and there are more instances in the ranks of r_2 and r_1, the ranking model of Ranking SVM tends to be close to that of r_2-r_1. This is not desirable, since it means that Ranking SVM emphasizes ranking on the middle and bottom. Actually, it should do the opposite. One solution to adjust the bias is to change the cost or penalty of misclassification of instance pairs between different rank pairs.

The second factor is to avoid training a model heavily influenced by queries with many relevant documents. It is also a problem which Ranking SVM does not consider. Figure 1 (c) shows the ranking models of Ranking SVM created individually by the data of the two queries individually. It also provides the ranking model created when the data of the two queries are combined together. We see that the total ranking model tends to be close to that of query 50, which has more relevant documents (and thus more instance pairs), although it should not have such a bias. One solution to the problem is to add a different weight to data from queries with varied relevant documents.

Here, we show the results of two queries for ease of explanation. In general, we observe that the same tendencies exist for all queries in OSUMED.

One may argue that the characteristics of data depend on the features used and with more features available the problems described above may disappear. It is hard to anticipate what will happen when more features are available. In our current data analysis, we use all the conventional features utilized in document retrieval and our experimental results indicate that, with the features, we can achieve the same performance as those of state-of-the-arts methods. Therefore, we can say that, at least with the standard feature set, we observe the phenomena explained above.

We also note that other learning to rank [1, 19, 24] methods also do not take into consideration these two factors.

5. Ranking SVM for IR

5.1 Loss Function

To deal with the two problems described in Section 4, we define a new loss function on the basis of Hinge Loss. In the loss function as shown in (9), we add rank parameters τ 's to adjust the bias between rank pairs and add query parameters μ 's to adjust the bias across queries. We re-formalize the learning problem of Ranking SVM for IR as that of minimizing the following loss function.

$$\min_{\vec{w}} L(\vec{w}) = \sum_{i=1}^{\ell} \tau_{k(i)}\mu_{q(i)} \left[1 - z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle\right]_+ + \lambda \|\vec{w}\|^2, \qquad (9)$$

where $k(i)$ denotes the type of ranks of instance pair i, $\tau_{k(i)}$ denotes the rank parameter for $k(i)$, $q(i)$ denotes the query of instance pair i, and $\mu_{q(i)}$ denotes the query parameter for $q(i)$. The penalty received by the i^{th} pair is determined by the product of $\tau_{k(i)}$ and $\mu_{q(i)}$: $\tau_{k(i)}\mu_{q(i)}$. The idea of using different losses for misclassification of input instances into different classes has been proposed for other problems [28]. To the best of our knowledge, this is the first attempt of it in document retrieval.

Figure 2 plots the shapes of different Hinge Loss functions with different penalty parameters $\tau \cdot \mu$'s. The x-axis represents $z \cdot f(\vec{x}^{(1)} - \vec{x}^{(2)})$ and the y-axis represents loss. When $z \cdot f(\vec{x}^{(1)} - \vec{x}^{(2)}) \ge 1.0$, all the losses are zero. When $z \cdot f(\vec{x}^{(1)} - \vec{x}^{(2)}) < 1.0$, the losses are linearly decreasing functions, with different slops of $\tau \cdot \mu$'s. If the product $\tau \cdot \mu$ equals 1.0, then the Hinge Loss function becomes the same as that in Ranking SVM.

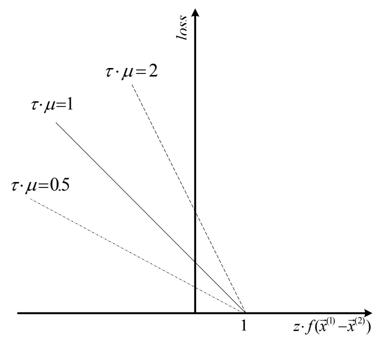

Figure 2: Hinge Loss functions with different penalty parameters

5.2 Determining Parameter Values

We then consider how to determine the values of the parameters τ 's and μ 's.

For τ 's, we propose a heuristic method for estimating the parameters on the basis of simulation. Suppose that NDCG is used in evaluation. (In principle, any other measure can be used). We take the averaged drops in terms of NDCG as the parameter values when randomly changing the positions of documents in rankings. We make use of a set of *labeled* data. To calculate the value of the parameter for a specific rank pair, we repeat the following processes. For each query, we first find its perfect ranking (NDCG@1 of the ranking is 1.0). We then randomly select one document from each of the ranks and reverse their positions. In this way we obtain a new ranking and we can calculate its NDCG@1. Usually there is a drop in NDCG@1. We calculate the average drops in NDCG over all the queries in the dataset. Finally, we take the average performance drop as the value of rank parameter for the rank pair.

As for μ 's, we define it as follows:

$$\mu_{q(i)} = \frac{\max_j \#\{\text{instance pairs associated with } q(j)\}}{\#\{\text{instance pairs associated with } q(i)\}} \qquad (10)$$

With such a parameter, we can more penalize the errors on the queries with fewer instance pairs. As a result, the training will be conducted equally over all queries.

5.3 Optimization Methods

In this section, we present two methods for optimizing the loss function (9). The two methods are gradient descent and quadratic programming.

5.3.1 Gradient Descent

The loss function in (9) can be rewritten as

$$L(\vec{w}) = \sum_{i=1}^{\ell} l_i(\vec{w}) + \lambda \|\vec{w}\|^2, \quad where\ l_i(\vec{w}) = \tau_{k(i)}\mu_{q(i)} \left[1 - z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle\right]_+ \qquad (11)$$

By differentiating (11) with respect to parameters $\vec{w}$, we obtain

$$\frac{\partial L}{\partial \vec{w}} = \sum_{i=1}^{\ell} \frac{\partial l_i(\vec{w})}{\partial \vec{w}} + 2\lambda \vec{w},$$
$$where\ \frac{\partial l_i(\vec{w})}{\partial \vec{w}} = \begin{cases} 0 & if\ z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle \ge 1 \\ -z_i \tau_{k(i)}\mu_{q(i)}(\vec{x}_i^{(1)} - \vec{x}_i^{(2)}) & if\ z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle < 1 \end{cases} \qquad (12)$$

We then define the iteration equations of gradient descent as

$$\begin{aligned} &\vec{w}^{(k+1)} = \vec{w}^{(k)} + \eta_k \Delta\vec{w}^{(k)} \\ &\Delta\vec{w}^{(k)} = -\nabla L(\vec{w}^{(k)}) = -\sum_{i=1}^{\ell} \frac{\partial l_i(\vec{w}^{(k)})}{\partial \vec{w}} - 2\lambda \vec{w}^{(k)} \\ &\eta_k : L(\vec{w}^{(k)} + \eta_k \Delta\vec{w}^{(k)}) = \min_{\eta \geq 0} L(\vec{w}^{(k)} + \eta\Delta\vec{w}^{(k)}) \end{aligned} \tag{13}$$

Upon each iteration we reduce the cost function along its descent direction, as in (12). To determine the step size of iteration, we conduct a line search along the descent direction, as described in (13). In practice, instead of calculating each η_k we make all η's fixed. Figure 3 provides the algorithm.

```
Input:
  Training sample $S = \{(\vec{x}, y)\}^m$, where $(\vec{x}, y) \in X \times \{r_q \succ r_{q-1} \succ \cdots \succ r_1\}$
  Learning rate $\eta > 0$, cost parameters $\{\tau_{k(i)}\}$ and $\{\mu_{q(i)}\}$, penalty weight $\lambda$

Make $S' = \{((\vec{x}_i^{(1)}, \vec{x}_i^{(2)}), z_i)\}_{i=1}^{\ell}$ from $S$;
$\vec{w} = \vec{0}$;
while (stop_condition isn't met){
  $\Delta\vec{w} = \vec{0}$;
  for($i = 0$; $i < \ell$; $i{+}{+}$){
    if($z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle < 1$)   $\Delta\vec{w} = \Delta\vec{w} + z_i \tau_{k(i)} \mu_{q(i)} (\vec{x}_i^{(1)} - \vec{x}_i^{(2)})$;
  }
  $\Delta\vec{w} = \Delta\vec{w} - 2\lambda w$;
  $\vec{w} = \vec{w} + \eta\Delta\vec{w}$;
}
return $\vec{w}$;
```

Figure 3: Gradient descent algorithm

5.3.2 *Quadratic Programming*

In our method, instead of directly solving (9), we solve the equivalent Quadratic Optimization problem as described below.

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{i=1}^{\ell} C_i \cdot \xi_i \\ &\text{subject to } \xi_i \geq 0, \quad z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle \geq 1 - \xi_i \quad i = 1, \cdots, \ell \end{aligned} \tag{14}$$

This is because the following theorem holds.

Theorem 1: Criteria (*9*) *and* (*14*) *are equivalent, with* $C_i = \frac{\tau_{k(i)}\mu_{q(i)}}{2\lambda}$.

We will give the proof in the full version of the paper. It turns out that the method creates a SVM model as solution.

For the optimization problem in (14), the Lagrange Function can be written as

$$L_P = \frac{1}{2}\|\vec{w}\|^2 + \sum_{i=1}^{\ell} C_i \cdot \xi_i - \sum_{i=1}^{\ell} \alpha_i \left[z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle - (1 - \xi_i) \right] - \sum_{i=1}^{\ell} \mu_i \xi_i \tag{15}$$

Then, the objective is to minimize (15) with respect to $\vec{w}$ and ξ_i. Setting the respective derivatives to zero, we get

$$\vec{w} = \sum_{i=1}^{\ell} \alpha_i z_i \langle \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle \tag{16}$$

$$\alpha_i = C_i - \mu_i \quad i = 1, \cdots, \ell \tag{17}$$

as well as the positive constraints $\alpha_i, \mu_i, \xi_i \;\; i = 1, \cdots, \ell$. Substituting (16) and (17) into (15), we obtain the Lagrange dual function.

$$L_D = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{i'=1}^{\ell} \alpha_i \alpha_{i'} z_i z_{i'} \langle \vec{x}_i^{(1)} - \vec{x}_i^{(2)}, \vec{x}_{i'}^{(1)} - \vec{x}_{i'}^{(2)} \rangle \tag{18}$$

It gives a lower bound on the objective function (10). We maximize L_D subject to the constraints

$$0 \leq \alpha_i \leq C_i \quad i = 1, \cdots, \ell \tag{19}$$

The objective function here is similar to that in Ranking SVM. The only difference lies on the use of the box constraints on parameter $\vec{\alpha}$ (19). For Ranking SVM, the upper bounds of all $\alpha_i, i = 1, \cdots, \ell$ are the same ($0 \leq \alpha_i \leq C, i = 1, \cdots, \ell$). Here, the upper bounds vary according to different types of pairs. A larger C_i corresponds to a more important rank pair or a pair from a query with fewer instances.

6. EXPERIMENTAL RESULTS

6.1 Evaluation Measures

As one measure for evaluating the results of ranking methods, we used Normalized Discounted Cumulative Gain (NDCG) [10]. NDCG is a measure commonly used in IR, when there are more than two categories in relevance ranking. Given a query q_i, the NDCG score at the position m in the ranking of documents provided by a retrieval method is defined as

$$N_i = n_i \sum_{j=1}^{m} (2^{r_{(j)}} - 1) / \log(1 + j) \tag{20}$$

where $r_{(j)}$ is the rating of the jth document and n_i is a normalization constant. n_i is chosen so that a perfect ranking's NDCG score is 1. For queries whose returned documents are less than m, the NDCG score is only calculated for the returned documents. In our experiments we measured NDCG at the positions of 1, 3, and 5.

We also used Mean Average Precision (MAP) as evaluation measure for evaluating ranking methods. MAP is widely used in IR and is based on the assumption that there are two categories: positive (relevant) and negative (irrelevant) in ranking of instances (documents). 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 positive (relevant) instance is retrieved. Given a query q_i, its average precision (AvgP$_i$) is calculated as:

$$AvgP_i = \sum_{j=1}^{M} \frac{P(j) \times \text{pos}(j)}{\text{number of positive instances}} \tag{21}$$

where *j* is the rank, *M* is the number of instances retrieved, pos(*j*) is a binary function to indicates whether the instance in the rank *j* is positive (relevant), and *P*(*j*) is the precision at the given cut-off rank *j*:

$$P(j) = \frac{\text{number of positive instances in top } j \text{ positions}}{j} \tag{22}$$

6.2 Our Method and Baseline Methods

We denote our method as Ranking SVM for IR (RSVM-IR). We further denote RSVM-IR with the two optimization options, gradient descent, and quadratic programming, as RSVM-IR-GD and RSVM-IR-QP.

Obviously, we can have a method in which we only consider the first factor in Section 4. That is, we set different rank parameters for different rank pairs and set all the query parameters as 1.0. We denote it as RSVM-IR-Rank. Similarly, we can have a method of RSVM-IR-Query. In RSVM-IR-Rank and RSVM-IR-Query, we only employ quadratic programming as our optimization method.

Table 1: Features in ranking models. $C(w, d)$ represents count of word w in document d; C represents the entire collection; n denotes the number of terms in the query; $|.|$ denotes the size of function; and $idf(.)$ denotes the inverse document frequency.

FEATURE			
1	$\sum_{q_i \in q \cap d} \log(c(q_i, d)+1)$	2	$\sum_{q_i \in q \cap d} \log(\frac{\|C\|}{c(q_i, C)}+1)$
3	$\sum_{q_i \in q \cap d} \log(idf(q_i))$	4	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i,d)}{\|d\|}+1)$
5	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i,d)}{\|d\|} \cdot idf(q_i)+1)$	6	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i,d)}{\|d\|} \cdot \frac{\|C\|}{c(q_i,C)}+1)$
7	$\log(BM25\ score)$		

Table 2: p-values in Sign Test

p-values	BM25	LMIR	RSVM
RSVM-IR-QP	2.44E-13	2.28E-12	0.0391

Table 3: Top 5 documents with respect to query 9

	RSVM	RSVM-IR-QP
Top 5 ranked doc	*p d d p n*	*d p d n p*
NDCG@1	0.3333	1.0

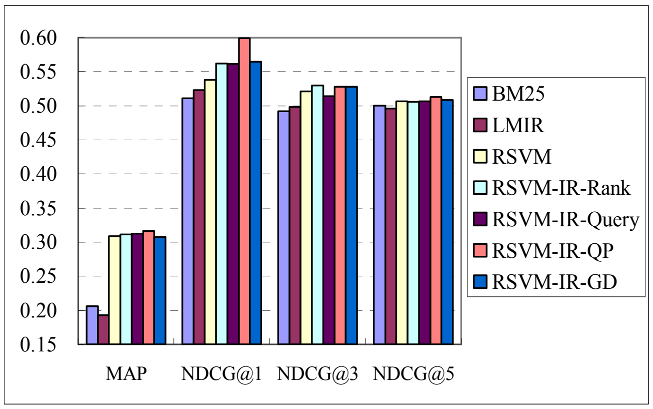

Figure 4: Ranking accuracies on OHSUMED data

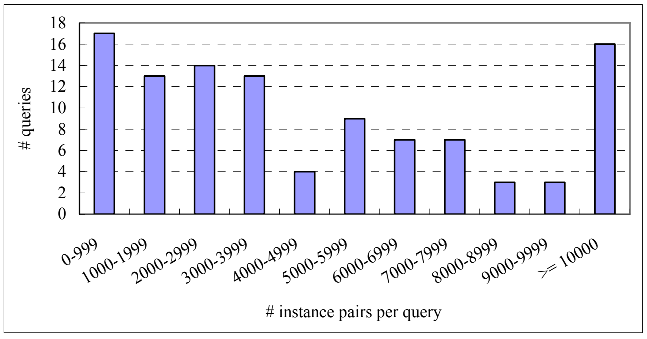

Figure 5: Distribution of queries over numbers of instance pairs

We use Ranking SVM (RSVM) as a baseline method in all the experiments. Actually, Ranking SVM is a special case of Ranking SVM for IR (RSVM-IR) when all the $\tau \cdot \mu$ values equals 1.0. We also compare our methods with BM25 [14] and language model for information retrieval (LMIR) [21]. For both BM25 and LMIR, we used the tool Lemur[1] (Language Models Toolkits for Information Retrieval).

6.3 Experiment with OHSUMED Data

In the experiment, we made use of the OHSUMED collection [9]. It is a collection of documents and queries on medicine, consisting of 348,566 references and 106 queries. There are a total of 16,140 query-document pairs upon which relevance judgments where made. The relevance judgments are either d (*definitely relevant*), p (*possibly relevant*), or n (*not relevant*). The data have been used in many experiments in IR, for example, the TREC-9 filtering track [14].

Each instance consists of a vector of features, determined by a query and a document. We adopted the standard features used in document retrieval [12]. Table 1 shows all the features. For example, *tf* (term frequency), *idf* (inverse document frequency), *dl* (document length), and their combinations are features. BM25 score is another feature, which is calculated using the ranking method of BM25 [14]. We took log on the feature values in order to reduce the effects of large numbers. This does not change the tendencies of the results, based on preliminary experiments. Stop words are removed and stemming is conducted in indexing and retrieval.

In MAP calculation, we define the Category d as positive (relevant) and the other two categories as negative (irrelevant).

We conducted 4-fold cross-validation. We tuned the parameters for LMIR and BM25 with one of the trials and applied them to the other trials directly. The results reported in Figure 4 are the averaged of four trials. From Figure 4, we see our methods (both RSVM-IR-QP and RSVM-IR-GD) outperform Ranking SVM, BM25 and LMIR in terms of all measures. Furthermore, RSVM-IR-QP and RSVM-IR-GD are better than RSVM-IR-Rank and RSVM-IR-Query. The results indicate our method improves the baseline methods. We effectively deal with the two problems from which Ranking SVM suffers. We conducted a Sign Test on the improvements of RSVM-IR-QP over BM25, LMIR, and RSVM in terms of NDCG@1. The results shown in Table 2 indicate that the improvements are statistically significant (p-value < 0.05) in terms of NDCG@1. .

Statistics on the distribution of instances in the OHSUMED dataset include 2,252 query-documents labeled as d, 2,585 labeled as p, and 11,303 labeled as n. Note that rank n has the largest number of instances, followed by p and d. The findings confirm our observation in Figure 1(b). It explains why RSVM-IR-Rank performs better than Ranking SVM (RSVM). We also analyze the results and find that our method provides better rankings on the highest ranks than Ranking SVM. For example, for query 9 ("t-cell lymphoma associated with autoimmune symptoms"). The top five documents returned by ranking SVM and our method are listed in Table 3 (the scores of NDCG@1 are also given). We note that both the ranking methods incorrectly rank the two document pairs. Two (d, p) pairs reversed in the ranking by Ranking SVM and one (d, p) and one (p, n) pairs are reversed in

[1] http://www.lemurproject.org/

the ranking by our method. However, the errors in our method are less problematic, since they are not at the top. The top-ranked results by Ranking SVM contain more errors than those of our method, because our method is trained to better perform at the top.

We also provide statistics on the number of instance pairs per query, as shown in Figure 5. The queries are grouped into different categories based on the ranges of the number of instance pairs. For example, all the queries in the category 1000-1999 should have instance pairs in between 1,000 and 1,999. We can see that the numbers of instance pairs may vary from query to query. As discussed in Section 4, we should consider a way to adjust the discrepancy. The statistics also explain why RSVM-IR-Query performs better than Ranking SVM.

In the example data in Section 4 (queries 12 and 50), we also applied Ranking SVM and our methods to it. Figure 6 shows the result in terms of NDCG@N (N = 1, 5, 10, …, 30). We clearly observe that our method outperforms Ranking SVM, indicating that it effectively deals with the problems Ranking SVM suffers from.

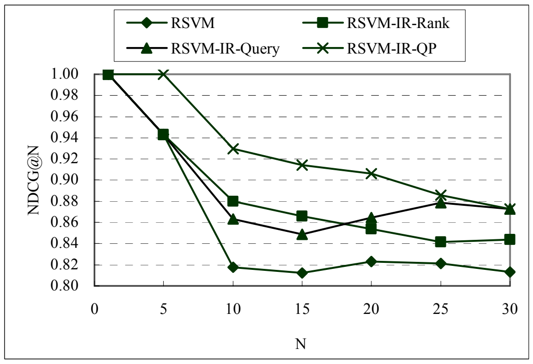

Figure 6: NDCG curves with respect to queries 12 and 50

6.4 Experiment with Web Search Data

In the experiment, we made use of a data collection from a commercial web search engine. The collection consists of 2,198 queries. Human judges have made relevance judgments on documents related to all queries. There are six ranks: “definitive”, “excellent”, “good”, “fair”, “bad”, and “detrimental”. In total, there are 74,276 query-document pairs (instances). Feature vectors are also generated from the query-document pairs. There are 426 features including those described in Section 6.3, as well as those made from hyperlink, anchor text, URL, and PageRank.

We randomly split the data into training set and test set. The training set had 1,109 queries and the test set had 1,089. We did not use LMIR as the baseline.

For MAP calculation we used the top ranks “definitive”, “excellent” and “good” as positive and the other ranks as negative.

Figure 7 shows the performances on the test set. We see that our methods (RSVM-IR-QP and RSVM-IR-GD) outperform all the baselines in all measures. This indicates again that our approach is effective for improving real IR problems. (We were not able to apply LMIR to the data because we obtained the data as feature vectors, not as documents. We were able to get results for BM25 because BM25 score exists as one of the features.)

In the collection, 8,990 query-documents are labeled as “definitive”, 4,403 as “excellent”, 3,735 labeled as “good”, 20463 labeled as “fair”, 36,375 as “bad”, and 310 labeled as “detrimental”. RSVM-IR-Rank makes larger improvements over Ranking SVM.

In contrast, we can also see that RSVM-IR-Query does not make much improvement. In Figure 8, the statistics on the dataset suggest the reason. Most of the queries fall into the range of 0-99 and thus the second factor in Section 4 is not severe for this dataset.

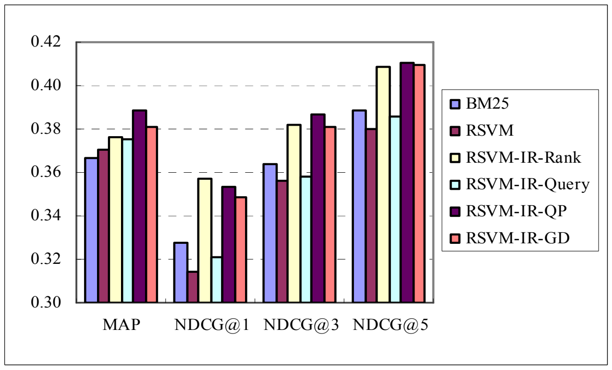

Figure 7: Ranking accuracies on web data

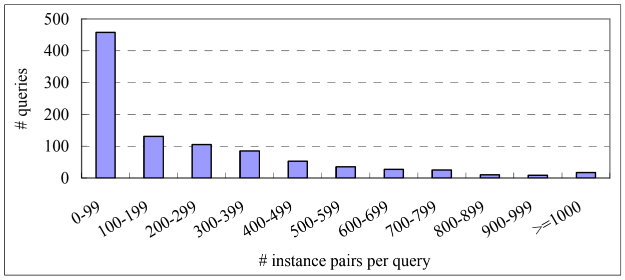

Figure 8: Distribution of queries over numbers of instance pairs

7. CONCLUSION AND FUTURE WORK

In the paper, we have proposed a new method for document retrieval on the basis of learning to rank.

There are two important factors that must be taken into consideration when applying learning to rank to document retrieval. One is to intensify training for top-ranked documents, and the other is to avoid training a model biased toward queries with many relevant documents. Existing learning to rank methods for document retrieval, including Ranking SVM, does not consider the two factors. We propose using a new loss function to deal with ranking problems. The new loss function naturally incorporates the two factors into the Hinge Loss function used in Ranking SVM, with two types of additional parameters. We employ gradient descent and quadratic programming to optimize the loss function. Experimental results indicate that our methods significantly outperform Ranking SVM and other existing methods in document retrieval.

We hope to explore future work in several areas, such as conducting theoretical analysis on the proposed method, applying the ideas to other loss functions, and developing loss functions more suitable for document retrieval.

REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank using

Gradient Descent. Proceedings of the 22nd International Conference on Machine Learning, 2005.

[2] W. Chu and Z. Ghahramani. Gaussian Process for Ordinal Regression. Technical Report. Univ. College London, 2004.

[3] W. Chu and S. Keerthi. New Approaches to Support Vector Ordinal Regression. Proceedings of International Conference on Machine Learning, 2005.

[4] K. Crammer and Y. Singer. Pranking with Ranking. Advances in Neural Information Processing Systems 14, pages 641-647, 2002.

[5] J. Gao, H. Qi, X. Xia, and J. Nie. Linear Discriminant Model for Information Retrieval. Proceedings of the 28th Annual ACM SIGIR Conference, 2005.

[6] E. F. Harrington. Online Ranking/Collaborative filtering using the Perceptron Algorithm. Proceedings of the 20th International Conference on Machine Learning, pages 250-257, 2003.

[7] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: data mining, inference and prediction. Springer-Verlag, 2001.

[8] R. Herbrich, T. Graepel, and K. Obermayer. Large Margin Rank Boundaries for Ordinal Regression. Advances in Large Margin Classifiers, pages 115-132, 2000.

[9] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. Proceedings of the 17th Annual ACM SIGIR Conference, pages 192-201, 1994.

[10] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pages 41-48, 2000.

[11] T. Joachims. Optimizing Search Engines Using Clickthrough Data. Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, 2002.

[12] R. Nallapati. Discriminative models for information retrieval. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pages 64-71, 2004.

[13] J. Ponte and W. B. Croft. A language model approach to information retrieval. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pages 275-281, 1998.

[14] S. Robertson and D. A. Hull. The TREC-9 Filtering Track Final Report. Proceedings of the 9th Text REtrieval Conference, pages 25-40, 2000.

[15] A. Shashua and A. Levin. Taxonomy of Large Margin Principle Algorithm for Ordinal Regression Problems. Advances in Neural Information Processing Systems 15. Cambridge, MA: MIT Press, 2000.

[16] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a Very Large AltaVista Query Log. Technical Report SRC 1998-014, Digital Systems Research Center, 1998.

[17] A. Spink, B. J. Jansen, D. Wolfram, and T. Saracevic. From e-sex to e-commerce: web search changes. IEEE Computer, 35(3), pages 107-109, 2002.

[18] A. Spink, D. Wolfram, B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. Journal of the American Society of Information Science and Technology, 52(3), pages 226-234, 2001.

[19] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 4, pages 933–969, 2003.

[20] G. Salton. The SMART Retrieval System: Experiments in automatic document processing. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[21] J. Lafferty and C. Zhai Document Language Models, Query Models, and Risk Minimization for Information Retrieval. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pages 111-119, 2001.

[22] D. Grangier and S. Bengio. Exploiting Hyperlinks to Learn a Retrieval Model. Proceedings of NIPS Workshop, 2005.

[23] S. Rajaram and S. Agarwal Generalization Bounds for k-Partite Ranking. Proceedings of NIPS Workshop, 2005.

[24] C. Burges. Ranking as Learning Structured Outputs. Proceedings of NIPS Workshop, 2005.

[25] N. Usunier, V. Truong, R. A. Massih, and P. Gallinari, Ranking with Unlabeled Data: A First Study. Proceedings of NIPS Workshop, 2005.

[26] W. Chu and Z. Ghahramani Extensions of Gaussian Processes for Ranking: Semi-Supervised and Active Learning. Proceedings of NIPS Workshop, 2005.

[27] S. Yu, K. Yu, and V. Tresp, Collaborative Ordinal Regression. Proceedings of NIPS Workshop, 2005.

[28] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. Proceedings of ICML, 1999.