International Journal of Innovative Computing, Information and Control
Volume **11**, Number **2**, April **2015**

 ISSN 1349-4198
pp. 759–772

TOPRANK: A NOVEL TOP-K RANKING METHOD BASED ON RANK AGGREGATION

Fan Cheng[1,2], Dengdi Sun[2] and Ying Xie[2]

[1]Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education
[2]School of Computer
Anhui University
No. 3, Feixi Road, Hefei 230039, P. R. China
chengfan@mail.ustc.edu.cn

Received June 2014; revised October 2014

Abstract. *This paper studies how to learn an accurate ranking model for Top-k ranking. Most previous work on learning to rank manages to optimize ranking on the whole permutation of objects, but in practical applications such as web search, correct ranking at the Top-k positions is much more important. To tackle the problem, in this paper we propose a novel Top-k ranking algorithm with rank aggregation. The algorithm is a multistage approach. In the first stage, we present a Listwise algorithm named TopSlack as the base ranker. It uses the structural SVM as optimization tool, and defines the objective function based on slack scaling which is believed can create more precise model than the margin one. The output of the TopSlack is re-ranked by multiple different rankers in the next stage, and lastly, the results of those rankers are combined together with Supervised Kemeny Aggregation technique to produce the final list. Experimental results on benchmark collections prove that the algorithm we proposed has significant advantage in comparison with several baseline methods, and is suitable for the Top-k ranking.*
Keywords: Learning to rank, Top-k ranking, Rank aggregation, Structural SVM, Cutting plane algorithm

1. **Introduction.** Learning to rank is the central problem in many applications including information retrieval, and recommender systems. It aims at learning a ranking model that given a query and a set of relevant documents, finds the appropriate ranking of documents according to their relevancy. There have been numerous learning to rank methods developed in the literature, which can be divided into three main categories: the Pointwise [1,2], the Pairwise [3-5], and the Listwise methods [6-11]. Empirical results on benchmark datasets have demonstrated that the Listwise ranking methods have very competitive ranking performances [6-11].

However, even with the Listwise methods, there is still a gap between the learning to rank and real ranking applications, where the correct ranking of the entire permutation is not needed [12]. For example, in web search, users usually care much more about the top ranking results and thus only correct ranking at the top positions is important. While the ranking methods above all spend some time in adjusting the exact preference orders among the rest results, to which the users pay no attention.

To fill the gap above, in this paper, we propose a novel ranking algorithm named TopRank which can produce an accurate Top-k ranking model. The main contributions of this paper are as follows.

(1) We propose a novel Top-k ranking algorithm with rank aggregating. Specifically, our approach is based on a mixed ranking model. Firstly, we construct a base ranking model to capture the total order of the candidates. Secondly, the top candidates of its

output are re-ranked by r other rankers. Lastly, the r rankers' results are aggregated and create a model to capture the relative of the Top-k items. The idea behind our strategy is that if the first ranker is reasonably accurate, the following re-ranking phase can focus on improving the precision at high ranks.

(2) To get an accurate base ranker, we design a new ranking algorithm named TopSlack. It belongs to Listwise method and uses structural SVM as optimization tool, which is commonly used in the ranking algorithms [7-11]. However, different from those algorithms which all adopt the margin scaling as the framework, TopSlack employs the slack scaling as its framework, which is believed can create more accurate model. For the challenge of finding the most violated constraint in slack scaling, we adopt a variational approximation to the slack loss so that the most violated labeling can be efficiently found using the same loss as in margin one.

(3) We demonstrate how multiple rankers can be effectively combined using rank aggregation techniques stemming from the Social Choice Theory literature. Specifically, we present to use the Supervised Kemeny Ranking technology for ranking aggregation to create the final rank list.

(4) We evaluate the proposed method on LETOR3.0 benchmark collections [13]. Experimental results manifest that TopRank we proposed can significantly outperform many other state-of-the-art ranking algorithms when measured by Top-k.

The remainder of the paper is organized as follows. In Section 2 the related work is presented. Section 3 discusses the design of TopRank in detail and the empirical results on the benchmark datasets are reported in Section 4. Section 5 concludes the paper and discusses the future work.

2. Related Work.

2.1. Learning to rank.

The process of learning to rank can be described as follows. Assume that we have a collection of m queries for training, denoted by $\{q_1, q_2, \ldots, q_m\} \in Q$. For each query q_i, we have a collection of n_i documents $\{d_i^1, d_i^2, \ldots, d_i^{n_i}\} \in D_i$, whose relevance to q_i is given by a vector $y_i = (y_i^1, y_i^2, \ldots, y_i^{n_i})$, and the relevance label $y_i^j \in \{r_1, r_2, \ldots, r_l\}$, with $r_l \succ r_{l-1} \succ \ldots r_1$. The goal is to learn from these examples a ranking function which, given a new query q, can rank the documents associated with the query such that more relevant documents are ranked higher than less relevant ones.

More formally, we shall assume a query-document feature mapping $\Phi(q_i, d_i^j)$ that maps each query-document pair to a d-dimensional feature vector. The learner then receives labeled training examples of the form $S = \{s_1, s_2, \ldots, s_m\}$, where $s_i = \{(\Phi(q_i, d_i^1), y_i^1), \ldots, (\Phi(q_i, d_i^{n_i}), y_i^{n_i})\}$ is the training sample associated with the ith query. The goal is to learn a ranking function $h : R^d \to R$ that ranks accurately documents associated with future queries. h is taken to rank a document d^j associated with a query q higher than a document d^k if $h(\Phi(q, d^j)) > h(\Phi(q, d^k))$, and lower than d^k otherwise. In this paper, we are interested in linear ranking function given by:

$$h(s_i) = h(q_i, d_i, y_i) = w^T \cdot \Psi(\Phi(q_i, d_i), y_i) \equiv w^T \cdot \Psi(x_i, y_i) \tag{1}$$

where $\Phi(q_i, d_i) = \sum_{j=1}^{n_i} \Phi(q_i, d_i^j)$, and Ψ represents a mapping function from input list to output list. Through designing appropriate supervised learning algorithms, people can get a ranking model.

2.2. **Top-k ranking.**

2.2.1. *Motivation.* From the description above, we can see that the learning to rank methods manage to optimize ranking on the whole permutation of objects. While in the practical ranking applications, people often pay more attention to the top-ranked objects. Therefore, the correct ranking on the top positions is critically important. We take the web search as an example, modern web search engines only return top 1,000 results and 10 results in each page. According to a user study, 62% of search engine users only click on the results within the first page, and 90% of users click on the results within the first three pages [14], which means that if two ranked lists have the same ranking results for the top positions, they may provide the same experience to the users. Moreover, a good ranking on the top results is much more important than a good ranking on the others. Therefore, in the practical ranking applications, a good ranking method should pay more attention to the Top-k results, but less attention to the exact preference orders among the rest results. This, however, cannot be reflected in the traditional learning to rank, so we refer to it as the Top-k ranking problem.

Before we make further discussion, it should be noted that the Top-k ranking has wide application prospects. It can be applied not only to web search, but also to many other information retrieval areas such as collaborative filtering [15], document retrieval [16], and QA system [17]. Recently, it has been extended to the Bioinformatics task whose goal is to rank new chemical compounds such that active ones appear at the top of the list [18,19].

2.2.2. *Top-k ranking methods.* In contrast to the fact that there have been many attempts on the issue for learning to rank [1-11], as far as we know, there was only a little work focusing on the Top-k ranking. For example, Xia et al. first analyze the difference between Listwise ranking and Top-k ranking, then give the statistical consistence of Top-k ranking, and based on it, a Top-k ranking algorithm is proposed [12]. This approach is effective, but it is only suitable for the Listwise method, and whether it can be extended to the Pointwise or Pairwise method is still unknown. Niu et al. propose a novel Top-k ranking system, which involves Top-k labeling strategy and evaluation measure [20]. Although the experimental results demonstrate the effectiveness of algorithm, there are still some limits of this method. Firstly, the new "Top-k evaluation measures" in the paper are proposed subjectively, and the correctness of these measures has not been proved. Secondly, even with the new concept of "Top-k ground-truth", how to design a new ranking model remains a valuable problem to investigate. Different from the approach in the paper [20], we do not utilize the new concept of "Top-k ground-truth", "Top-k evaluation measures", etc. We adopt the same Top-k evaluation measures as the ones used in paper [12], such as Precision@k and NDCG@k, which are believed the most widely used measures in the Top-k ranking area. However, the algorithm we proposed has the significant difference with the one in paper [12]. The key of ours is based on the technique of the rank aggregation which ensures our approach can not only be applied to the Listwise method, but also can be extended to the Pairwise or Pointwise method. It should be noted that most of the existed ranking algorithms including the Top-k rankers in paper [12,20] all try to use one single ranker that performs best on all datasets, which is very hard to achieve in practice. The algorithm we proposed is a practical solution to this problem. The idea behind our algorithm is that it may be better to combine some different rankers to produce a more robust and accurate ranking. The whole procedure of our approach can be described as Figure 1, and in the next section we will depict it in detail.

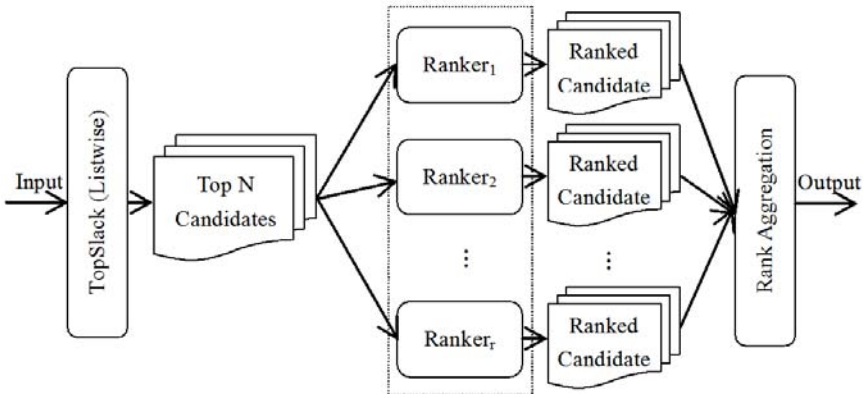


FIGURE 1. The learning scheme of TopRank

3. **TopRank-Direct Optimization of Top-k with Rank Aggregation.** As shown in Figure 1, the algorithm we proposed uses a cascading approach, where the ranking produced by one ranker is used as input to the next stage. The initial set of Top N candidates is firstly created by TopSlack, which is the base ranker of our algorithm.

3.1. **TopSlack – The base banker of the TopRank.** The framework of the TopSlack is based on structural SVM proposed by Tsochantaridis et al. [21]. In that paper, Tsochantaridis et al. proposed two formulations for maximum margin training of structured spaces: margin scaling and slack scaling. The margin scaling formulation has been widely used in recently "directly optimizing" ranking algorithms [7-11], since it can exploit the decomposability of the error function to find the most violated constraint using the maximum a posteriori (MAP) inference algorithm used for prediction. On the contrary, the slack scaling formulation is rarely used in ranking algorithms, for its non-decomposability nature. However, as pointed by Sarawagi and Gupta, slack scaling is believed to be more accurate and better behaved than margin one, because the margin scaling gives too much importance to instances which are already well separated from the margin [22]. Therefore, in this paper we present a novel "directly optimizing" ranking algorithm-TopSlack which is based on slack scaling formulation. It can be described as the following.

OP1. (Optimization Problem 1)

$$\min_{w,\xi} \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$$

$$\begin{aligned} &\text{s.t. } \bar{y_1} \in Y : w^T\left[\Psi(x_1, y_1) - \Psi\left(\bar{x_1}, \bar{y_1}\right)\right] \geq 1 - \frac{\xi_1}{\Delta\left(y_1, \bar{y_1}\right)} \\ &\qquad\qquad \cdots\cdots\cdots\cdots \\ &\text{s.t. } \bar{y_m} \in Y : w^T\left[\Psi(x_m, y_m) - \Psi\left(\bar{x_m}, \bar{y_m}\right)\right] \geq 1 - \frac{\xi_m}{\Delta\left(y_m, \bar{y_m}\right)} \end{aligned} \tag{2}$$

where w is the linear parameter to be learned, $C > 0$ is a constant that controls the trade-off between training error minimization and margin maximization, ξ_i is the slack variable for query q_i, $\bar{y_i}$ stands for any possible permutation of q_i, $\Delta : y_i \times \bar{y_i} \to R$ is a function used to measure the difference between the real output y_i and the possible

predicted output $\bar{y_i}$. This function must satisfy the following conditions:

$$① \text{ for } \forall(y = y'),\ \ \Delta(y, y') = 0; \quad ② \text{ for } \forall(y \neq y'),\ \ \Delta(y, y') > 0 \tag{3}$$

As explained in the Introduction, users mainly care about top results, so in TopSlack, we define the Ψ, Δ functions by paying more attention to the top relevant documents, so those two functions are defined as:

$$\Psi(x_i, y_i) = \sum_{j=1}^{n_i} y_i^j x_i^j l_i^j; \quad \Delta\left(y_i, \bar{y_i}\right) = 1 - \sum_{j=1}^{n_i} a_i^j b_i^j \tag{4}$$

Both l_i^j and a_i^j are decreasing functions which guarantee the higher ranked document d_i^j contributes more to the result. b_i^j denotes the gain of jth document in predicted list $\bar{y_i}$.

We can solve the OP1 (Optimization Problem 1) by substituting (4) with (2). However, unfortunately there is still a question: the number of possible output permutation $\bar{y_i}$ is exponential in the size, so the constraints of OP1: $\left(\bar{y_1}, \ldots, \bar{y_m}\right) \in Y^m$ could be extremely large. Fortunately, we may employ the cutting plane algorithm to solve OP1. The key of the cutting plane algorithm is that it iteratively introduces constraints until we can solve the original problem within a desired tolerance ε. The algorithm starts with no constraints, and iteratively finds for each example (x_i, y_i) the output $\bar{y_i}$ associated with the most violated constraint. If the corresponding constraint is violated by more than ε, then we introduce this $\bar{y_i}$ into the working set W_i of active constraints for example i, and re-solve formulation (2) using the updated $W_1, \ldots, W_m$. The whole cutting plane algorithm is described as the following.

Algorithm 1 Cutting Plane Algorithm for solving OP1 within tolerance ε

Step 1 Input: $\{(x_1, y_1), \ldots, (x_m, y_m)\}, C, \varepsilon > 0$

Step 2 For $i = 1, \ldots, m$ $\{W_i \leftarrow \phi, \xi_i \leftarrow 0\}$;

Step 3 repeat

Step 4 **For** $i = 1, \ldots, m$ do

Step 5

$$y_i^{\max} = \arg\max_{\bar{y_i}} \left[1 - w^T \left(\Psi(x_i, y_i) - \Psi\left(x_i, \bar{y_i}\right)\right) - \frac{\xi_i}{\Delta(y_i, \bar{y_i})}\right] \tag{5}$$

Step 6 **if** $\Delta\left(y_i, \bar{y_i}\right)\left[1 - w^T\left(\Psi(x_i, y_i) - \Psi\left(x_i, \bar{y_i}\right)\right)\right] > \xi_i + \varepsilon$ **then**

Step 7 $W_i \leftarrow W_i \cup \{y_i^{\max}\}$

Step 8

$$\begin{aligned}(w, \xi) \leftarrow \arg\min_{w, \xi \geq 0} & \tfrac{1}{2}||w||^2 + \tfrac{C}{m}\sum_{i=1}^{m}\xi_i \\ \text{s.t. } \forall\, \bar{y_1} \in W_1 : & w\Delta\left(y_1, \bar{y_1}\right)\left[\Psi(x_1, y_1) - \Psi\left(x_1, \bar{y_1}\right)\right] \geq \Delta\left(y_1, \bar{y_1}\right) - \xi_1 \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ \text{s.t. } \forall\, \bar{y_1} \in W_m : & w\Delta\left(y_m, \bar{y_m}\right)\left[\Psi(x_m, y_m) - \Psi\left(x_m, \bar{y_m}\right)\right] \geq \Delta\left(y_m, \bar{y_m}\right) - \xi_m\end{aligned}$$

Step 9 **end if**

Step 10 end for

Step 11 until no W_i has changed during iteration

Step 12 return (w, ξ).

It is proved that Algorithm 1's outer loop is guaranteed to halt within $O(\frac{1}{\varepsilon^2})$ iterations for any desired tolerance ε [21]. However, within the inner loop of cutting plane we have to

compute the formulation (5), which is known as the problem of "finding the most violated constraint". In margin scaling, $y_i^{\max}$ can be easily found by exploiting the decomposability of the error function with using the maximum a posteriori (MAP) inference algorithm [7,8]. While it is quite different in slack scaling, even with the decomposable loss and scoring functions, it is still difficult in finding $y_i^{\max}$ efficiently. To solve this problem, we adopt a variational approximation to the slack inference problem proposed by Liu et al. [23].

First of all, we rewrite the formulation (5) as:

$$\begin{aligned} y_i^{\max} &= \arg\max_{\bar{y_i} \in \widetilde{Y}} \left[1 - w^T \left(\Psi(x_i, y_i) - \Psi\left(x_i, \bar{y_i}\right) \right) - \frac{\xi_i}{\Delta\left(y_i, \bar{y_i}\right)} \right] \\ &= \arg\max_{\bar{y_i} \in \widetilde{Y}} \left[s_i\left(\bar{y_i}\right) - \frac{\xi_i}{\Delta\left(y_i, \bar{y_i}\right)} \right] \end{aligned} \tag{6}$$

where $s_i\left(\bar{y_i}\right) = w^T \Psi\left(x_i, \bar{y_i}\right)$, $\widetilde{Y} = \left\{ y : y \neq y_i, s_i(y) - \frac{\xi_i}{\Delta(y_i, y)} > s_i(y_i) - 1 \right\}$ is the set of all violated labeling. We can approximate the exact slack of formulation (6) with an upper bound as follows:

$$\max_{\bar{y_i} \in \widetilde{Y}} \left[s_i\left(\bar{y_i}\right) - \frac{\xi_i}{\Delta\left(y_i, \bar{y_i}\right)} \right] = \max_{\bar{y_i} \in \widetilde{Y}} \min_{\lambda_i \geq 0} F'\left(\bar{y_i}, \lambda_i\right) \tag{7}$$

$$\leq \min_{\lambda_i \geq 0} \max_{\bar{y_i} \in \widetilde{Y}} F'\left(\bar{y_i}, \lambda_i\right) \tag{8}$$

$$\leq \min_{\lambda_i \geq 0} F(\lambda_i) \tag{9}$$

where $F'\left(\bar{y_i}, \lambda_i\right) \equiv s_i\left(\bar{y_i}\right) + \lambda_i \Delta\left(y_i, \bar{y_i}\right) - 2\sqrt{\xi_i \lambda_i}$, and $F(\lambda_i) \equiv \max_{\bar{y_i} \neq y_i} F'\left(\bar{y_i}, \lambda_i\right)$. For the fixed λ_i, we can compute $F(\lambda_i)$ using the loss augmented MAP algorithm employed in margin scaling [7,8] to first find $y_{\lambda_i} = \arg\max_{\bar{y_i} \neq y_i} \left[s_i\left(\bar{y_i}\right) + \lambda_i \Delta\left(y_i, \bar{y_i}\right) \right]$ and then set $F(\lambda_i) = F'(y_{\lambda_i}, \lambda_i)$. We can compute $\min_{\lambda_i \geq 0} F(\lambda_i)$ with the efficient line search algorithm. In this paper, we adopt the Golden Search. During the search phase, for each λ_i that we encounter, we evaluate $F(\lambda_i)$ and thus get one labeling. Of all these labelings, we return the one with the highest $s_i\left(\bar{y_i}\right) - \frac{\xi_i}{\Delta\left(y_i, \bar{y_i}\right)}$ as $y_i^{\max}$. It has been proved that it is sufficient to perform the line search within the range $[\lambda_{low}, \lambda_{upper}]$. We omit the detail proof due to the lack of space. The interested reader can refer to Section 4 in Liu's paper.

By the approximation of slack loss above, we can compute Formula (5) efficiently.

3.2. **The rank aggregation.** As shown in Figure 1, the Top N candidates in output list of the TopSlack are re-ranked by r rankers, and the results of these rankers are aggregated to create the final rank list. By combining these different rankers, we expect to further improve the precision at high ranks. The core of above phase is how to aggregate the ranking properly. In this paper, we use the rank aggregation techniques stemming from the Social Choice Theory literature, which have been successfully applied in many different areas [24,25].

We begin by formally defining the rank aggregation task. Given a set of entities T, let V be a subset of T, and assume that there is a total ordering in V. We are given r

individual rankers $\Gamma_1, \Gamma_2, \ldots, \Gamma_r$ who specify their order preferences of the t candidates, where t is the size of V, i.e., $\Gamma_i = [d_1, d_2, \ldots, d_t]$, $i = 1, 2, \ldots, r$, if $d_1 > d_2 > \ldots > d_t$, $d_j \in V$, $j = 1, \ldots, t$. If d_i is preferred to d_j, we denote that by $d_i > d_j$. Rank aggregation function takes input orderings from r rankers and gives Γ, which is an aggregated ranking ordering. If V equals T, then Γ is called a full list (total ordering), and otherwise, it is called a partial list (partial ordering).

In this paper, we pay more attention to the top items, so we use the partial orderings corresponding to that only the Top candidates are re-ranked by different rankers. For the rank aggregation technique, we focus on the classical Kemeny Aggregation, which is an aggregation that has the minimum number of Pairwise disagreements with all rankers [26]. While in traditional Kemeny method, computing a Kemeny aggregation is NP-Hard for $r \geq 4$, so in practice, we use an approach that produces a 2-approximation of Kemeny optimal aggregation. Specifically, we employ the Supervised Kemeny Ranking which extends Approximate Kemeny aggregation to incorporate weights associated with each input ranking. The pseudo-code for Supervised Kemeny Ranking is presented in Algorithm 2.

Algorithm 2 Supervised Kemeny Ranking (SKR)

Step 1 Input: $\Gamma_i = [\Gamma_{i1}, \Gamma_{i2}, \ldots, \Gamma_{it}]$, $\forall i = 1, 2, \ldots, r$,
ordered arrangement of t candidates for r rankers,
$p = [p_1, p_2, \ldots, p_r]$: where p_i is the weight of ranker i,
$\mu = [\mu_1, \mu_2, \ldots, \mu_t]$: initial ordered arrangement of t candidates,
k: the number candidates to consider in each ranker's preference list ($k \leq t$)

Step 2 Initialize majority table $M_{i,j} \leftarrow 0$, $\forall i, j = 1, \ldots, t$
Step 3 For each ranker $a = 1$ to r
Step 4 For each candidate $i = 1$ to $k - 1$
Step 5 For each candidate $j = i + 1$ to k
Step 6 $M_{\tau_{ai}, \tau_{aj}} \leftarrow M_{\tau_{ai}, \tau_{aj}} + p_a$
Step 7 **end for**
Step 8 **end for**
Step 9 end for
Step 10 Quick sort μ, using M_{μ_i, μ_j}. If $M_{\mu_i, \mu_j} - M_{\mu_j, \mu_i} > 0$ then μ_i is greater than μ_j. If $M_{\mu_i, \mu_j} - M_{\mu_j, \mu_i} = 0$ then μ_i is equal to μ_j. If $M_{\mu_i, \mu_j} - M_{\mu_j, \mu_i} < 0$ then μ_i is less than μ_j.
Step 11 Return Γ: rank aggregated arrangement of candidates in decreasing order of relevance.

In Algorithm 2, the weights p_i correspond to the relative utility of ranker i. It is clear that the Supervised Kemeny Ranking turns back to Unsupervised Kemeny Ranking if we set $\forall i$, $p_i = \frac{1}{r}$. As we know, in general, different ranker has different performance on different set, so in the following experiments, we do not set a fixed value to each p_i. On the contrary, we define the weight p_i proportional to its performance on the validation set, because we want better ranker to have a bigger weight in the aggregation.

4. Experiments.

4.1. Experiments settings.

4.1.1. *Datasets and baselines.* In this section, we empirically evaluate the TopRank on LETOR3.0 which are considered as benchmark datasets in rank learning area [13]. Specifically, we select OHSUMED, TD2003, and TD2004 in the LETOR3.0 to perform the experiments. OHSUMED dataset consists of articles from medical journals. There are 106

queries in the collection. For each query, there are a number of associated documents. The relevance degrees of documents with respect to the queries are given by humans, on three levels: definitely, possibly, or not relevant. There are 16,140 query-document pairs with relevance labels. In LETOR, the data is represented as feature vectors and their corresponding relevance labels. Features in OHSUMED dataset consist of low-level features and high-level features. Low-level features include term frequency (tf), inverse document frequency (idf), document length (dl), and their combinations. High-level features include BM25 and LMIR scores. In total, there are 25 features. TD2003 and TD2004 datasets are from the topic distillation task of TREC 2003 and TREC 2004. TD2003 has 50 queries and TD2004 has 75 queries. For each query, there are about 1,000 associated documents. Each query document pair is given a binary judgment: relevant or irrelevant. The features of TD2003 and TD2004 datasets include low-level features such as term frequency (tf), inverse document frequency (idf), and document length (dl), as well as high-level features such as BM25, LMIR, PageRank, and HITS. In total, there are 44 features.

For the sets in LETOR, five partitions are provided to conduct the five-fold cross validation, each including training, test and validation sets. In our experiments, we conduct five-fold cross validation, following the guideline of LETOR.

The results of a number of state-of-the-art learning to rank algorithms are also provided in the LETOR page. Since these baselines include most of the well-known learning to rank algorithms from each category (Pointwise, Pairwise and Listwise), we select some of them as comparison algorithms to study the performance of TopRank. Here is the list of these baselines (the details can be found in the LETOR web page).

Regression: This is a simple linear regression which is a basic Pointwise algorithm [1].

Ranking SVM: Ranking SVM is a Pairwise approach using Support Vector Machine [5].

ListNet: ListNet is a Listwise algorithm. It uses cross-entropy loss as its loss function [6].

Besides these three baselines, we adopt two other algorithms for comparison. The first one is TopMargin, which is similar to TopSlack-the first stage ranker of TopRank. The main difference between TopMargin and TopSlack is that the former selects the margin scaling as the framework, while the latter employs the slack one. We use the TopMargin as the fourth comparison algorithm, in order to see whether the slack scaling formulation can really bring better behavior than the margin one. The other comparison algorithm is Top-10 ListMLE which is from paper [12]. As mentioned before, it is a novel effective Top-k ranker in the learning to rank and has very good performance on the LETOR datasets. We choose it as the fifth example because we are interested in how our ranking algorithm performs, when compared with another good Top-k ranker. Note that we do not use the ranker in the paper [20], because it defines different "top-k ground-truth" and "top-k evaluation measures" from ours.

4.1.2. *Evaluation measures.* Based on the three datasets, we compare our algorithm with those five ranking algorithms and the evaluation measures are carried out with two standard metrics in top ranking: Precision@k and NDCG@k.

(1) Precision@k

Precision@k [27] is a measure for evaluating top k positions of a ranked list using two grades (relevant and irrelevant) of relevance judgment:

$$pre@k = \frac{1}{k}\sum_{j=1}^{k} r_j \tag{10}$$

where k denotes the truncation position and

$$r_j = \begin{cases} 1 & \text{if document in } j\text{th is relevant} \\ 0 & \text{otherwise} \end{cases}$$

(2) NDCG@k

Different from Precision which only considers two grades, NDCG (Normalized Discounted Cumulative Gain) is an evaluation measure that can leverage the relevance judgment in terms of multiple ordered categories. It is defined as:

$$NDCG@k = N_k^{-1} \sum_{j=1}^{k} g(r_j) \cdot l(j) \tag{11}$$

where N_k denotes the maximum of $\sum_{j=1}^{k} g(r_j) \cdot l(j)$, r_j denotes the relevance level of the document ranked at jth position, $g(r_j)$ is a gain function, and $l(j)$ is a discount function. In the following experiments, we adopt $g(r_j) = 2^{r_j} - 1$ and $l(j) = \frac{1}{\log(1+j)}$, which are defined by Jarvelin and Kekalainen [28].

From the definitions above, we can see that it is obvious that these two measures are Top-k related and are suitable for evaluating the ranking performance in Top-k ranking problems.

Note that in following experiments, due to the space limitation, we only list the results of N (NDCG) at positions 1, 3, 10, and P (Precision) at positions 1, 3, and 10, which are the same as the ones in paper [12].

4.1.3. *The parameter settings of TopRank.* It should be noted that when using the TopRank approach, we need further define some parameters. Firstly, we shall define the parameters ε, C and formulation (4) in the base ranker. In our experiment, we fix $\varepsilon = 0.001$, and choose C on the validation sets in the set of $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$. For the formulation (4), we consider two kinds of function respectively. Namely, if the ranking measure is Precision, then the formulations are defined based on Precision@k, which denote:

$$l_j = a_j = \begin{cases} \frac{1}{n} & \text{if } j \le n \\ 0 & \text{else} \end{cases} \quad \text{and} \quad b_j = \begin{cases} 1 & \text{if } d^j \text{ is relevant} \\ 0 & \text{else} \end{cases} \tag{12}$$

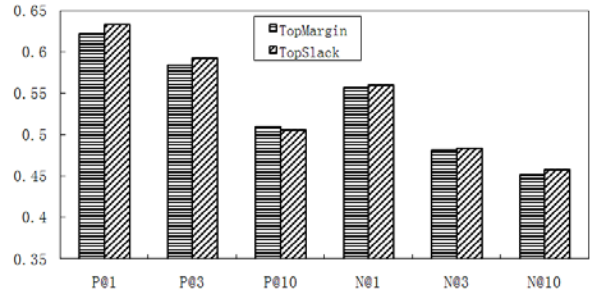
If the ranking measure is NDCG, we then define the formulations as:

$$l_j = a_j = \begin{cases} \frac{1}{\log(1+j)} & \text{if } j \le n \\ 0 & \text{else} \end{cases} \quad \text{and} \quad b_j = \frac{2^{y_i^j} - 1}{Nor(y_i)} \tag{13}$$

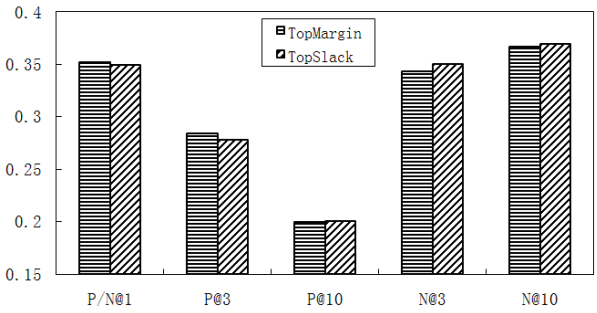
where $Nor(y_i)$ is a normalized factor.

Next, we shall define the parameters in Algorithm 2. We fix $r = 3$. Specifically, we use the Regression, Ranking SVM, ListNet with rank aggregation. For the weight parameter p_i, as mentioned before, if the ranking measure is Precision, we define the p_i proportional to the P@10 computed on the validation sets. If ranking measure is NDCG, the p_i is defined proportional to the N@10. For the parameters t, k, we set $t = \min_{1 \le i \le m} (100, \min(n_i))$, and k is set to the 50% of t.

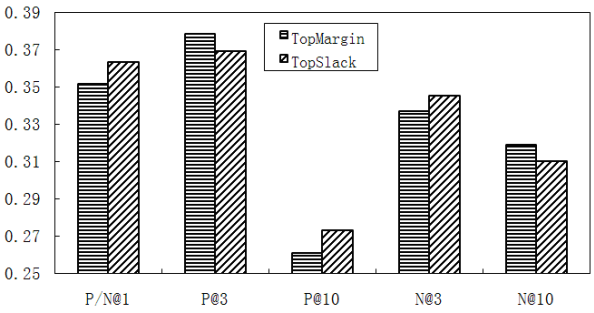
4.2. **Experimental results.** In this section, we empirically evaluate our proposed ranking algorithm from the following three aspects.



(a) On OHSUMED dataset



(b) On TD2003 dataset



(c) On TD2004 dataset

FIGURE 2. The performance comparison between TopMargin and TopSlack

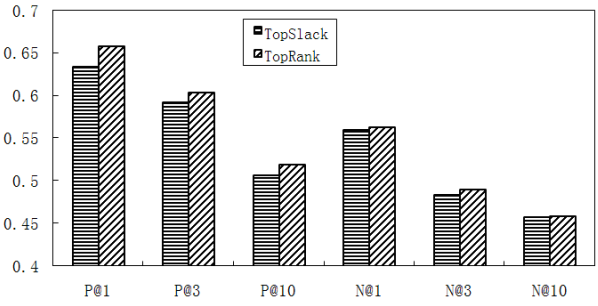
4.2.1. *Comparison between TopMargin and TopSlack.* We design this comparison in order to see whether the algorithm based on slack scaling can be better-behaved than the margin one. The experimental results on OHSUMED, TD2003, and TD2004 datasets are illustrated in Figure 2[1].

From the figures, we can see that the TopSlack performs better than TopMargin with most of the statistical data. We would like to point some statistics in detail. On OHSUMED set, for all the six statistical points, TopSlack outperforms TopMargin with five points, and only worse than TopMargin at P@10. On TD2004 set, TopSlack is better than TopMargin with four out of six points, and is inferior to TopMargin at P@3 and N@10. While on TD2003 set, the two algorithms win three out of six respectively. TopSlack performs better at P@10, N@3, N@10. TopMargin is better at P/N@1 and P@3. The comparison results above prove that in general, the slack scaling can be more accurate and better behaved than margin one. The reason lies in the following fact. Margin scaling gives too much importance to instances which are already well separated from the margin. This hurts because the loss ξ_i is determined by a single most violated labeling. In contrast, the slack scaling loss will ignore instances that are separated by the margin

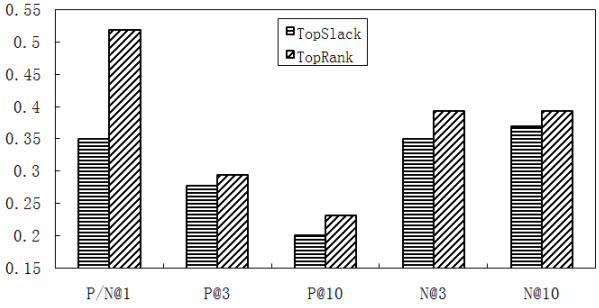
[1]On datasets with only two ratings such as TD2003 and TD2004, N@1 equals P@1.

of 1, and the loss ξ_i is determined by labeling that matter because of their being close to the margin.

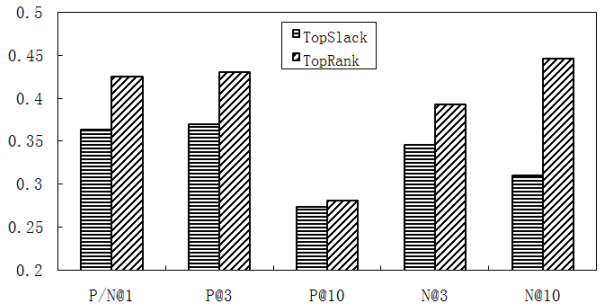
4.2.2. *Comparison between TopSlack and TopRank.* We design this comparison because we are concerned about whether the rank aggregation technology can really improve the performances of the Top-k results. The experimental results on the three datasets are illustrated in Figure 3.



(a) On OHSUMED dataset



(b) On TD2003 dataset

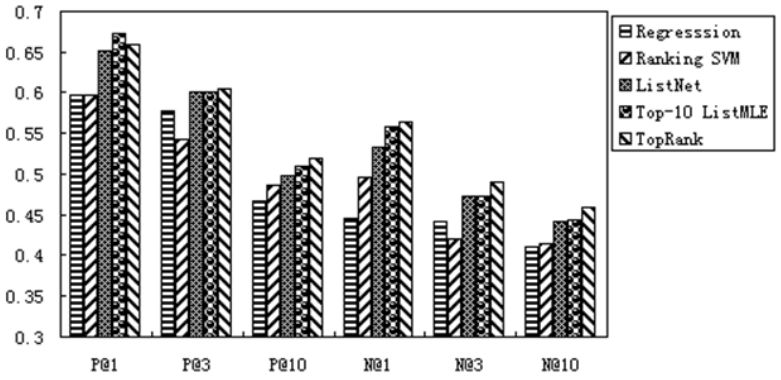


(c) On TD2004 dataset

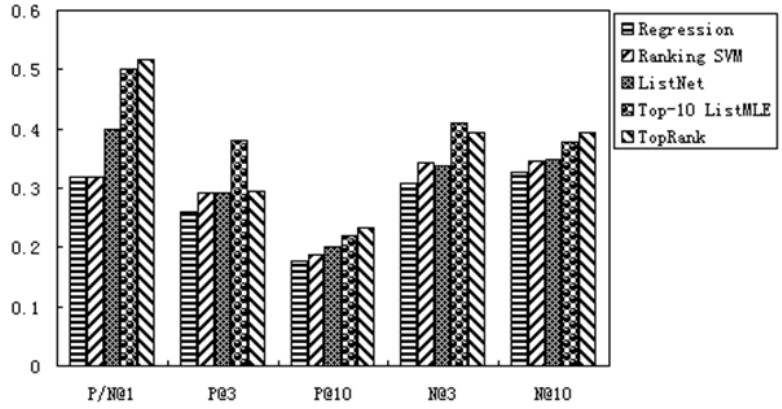
FIGURE 3. The performance comparison between TopSlack and TopRank

By comparing the TopRank with TopSlack, we can find that on all the three datasets, our aggregation algorithm TopRank consistently outperforms TopSlack in terms of all the measures. Due to the space limitation, we only take TD2004 as an example and report statistics measured by NDCG. For N@1, N@3, N@10, when compared to TopSlack, TopRank increases 17.2%, 5.6%, 43.6% in performance respectively. Those experimental results show that TopRank, as a rank aggregating algorithm, does effectively boost the ranking accuracies of TopSlack, and once again prove the strategy that we can improve the precision at the top of list by re-ranking the top results from an initial ranker.

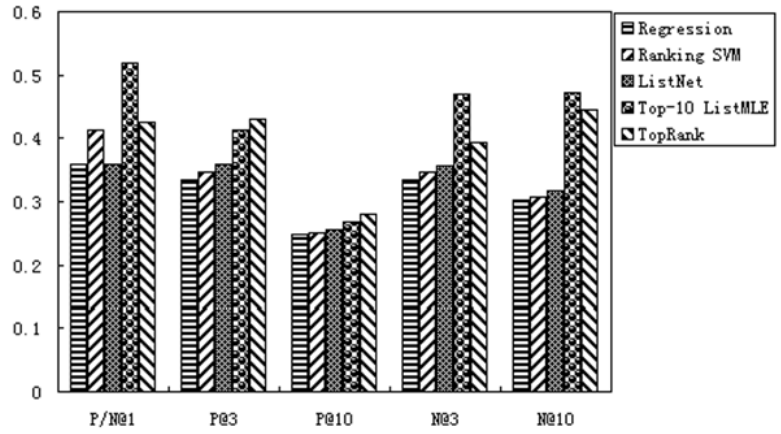
4.2.3. *Comparison with baselines.* In the last set of experiments, we are interested in how our ranking algorithm performs, when compared with other ranking algorithms. Specifically, we compare TopRank with the baselines mentioned in Section 4.1.1, and Figure 4 provides the results on OHSUMED, TD2003, and TD2004 datasets.



(a) On OHSUMED dataset



(b) On TD2003 dataset



(c) On TD2004 dataset

FIGURE 4. The performance comparison between TopRank and baselines

From the figures above, we can see that different algorithms have different performances on the different sets. For example, on OHSUMED dataset, the algorithms based on Listwise method (ListNet, Top-10 ListMLE, TopRank) are significantly better than the two non-Listwise algorithms (Regression, Ranking SVM) with all the statistical points. However, on the other two datasets: TD2003, TD2004, the results are a bit different.

Some Listwise algorithms are worse than Ranking SVM with some statistical points. For instance, on TD2004 set, Ranking SVM performs much better than ListNet and TopSlack in terms of P/N@1. One possible explanation is that OHSUMED is a multi-level dataset, and may be more suitable for the Listwise algorithms, while TD2003 and TD2004 are both binary sets, and the Ranking SVM which is also a binary algorithm may perform better on them.

Moreover, it can be seen from the figures that no matter on OHSUMED or on TD2003, TD2004, the two Top-k rankers (Top-10 ListMLE, TopRank) are significantly better than the other algorithms, in terms of all measures, which indicates the effectiveness of Top-k ranking strategy. Especially, we should point out that even compared with Top-10 ListMLE, the aggregation ranker we proposed is better with most of the statistics. We take OHSUMED dataset as an example, and for all the six statistical points, TopRank outperforms Top-10 ListMLE with five points, and is only worse than Top-10 ListMLE at P@1. In fact, among all the algorithms in comparison, TopRank appears to be the most stable method across all the datasets. Statistics show that TopRank performs best with 11 out of all the 18 points, and is the second best algorithm with the remainder. All those prove the excellent effectiveness of our ranking approach, and demonstrate that TopRank, as a mixed ranking model, can effectively cope with the Top-k ranking problem.

5. **Conclusion.** In this paper, we have proposed a novel Top-k ranking algorithm based on rank aggregation technique. Firstly, a structural SVM based method, termed TopSlack is proposed as a main ranker. Different from other "directly optimizing" ranking algorithms which used margin scaling as framework, TopSlack defines the objective function based on slack scaling, which is believed to be more accurate and better behaved. For the problem that slack scaling is difficult to optimize, we introduce the cutting plane algorithm for outer optimization, and adopt a variational approximation to the slack scaling for inner finding the most violated constraint. The top candidates in output list of the TopSlack are then re-ranked by multiple rankers, and we presented to use the Supervised Kemeny Ranking technology for ranking aggregation to create the final rank list. The experiments on the benchmark have shown that the algorithm we proposed is indeed better than the several baseline methods, and is suitable for the Top-k ranking.

Although the empirical studies on benchmark collections justified the effectiveness of the proposed algorithm, the statistical consistency of the ranking aggregation for Top-k ranking is still unknown. In future work, we will focus on the consistency analysis on the ranking aggregation method. We try to provide the sufficient and necessary condition for ranking aggregation consistency, and based on it, derive the consistent ranking algorithm for Top-k ranking.

Acknowledgement. This work is supported by the Humanities and Social Sciences Project of Chinese Ministry of Education (Grant No. 13YJC870003), and the Natural Science Foundation of China (Grant No. 60875027), and Special Foundation for Young Scientist of Anhui province. (Grant No. 2012SQRL016), Youth Foundation of Anhui University. (Grant No. KJQN1119), the Doctor Foundation of Anhui University, and the Foundation for the Key Teacher by Anhui University.

REFERENCES

[1] R. Nallapati, Discriminative models for information retrieval, *Proc. of the International ACM SIGIR Conference on Information Retrieval*, pp.64-71, 2004.

[2] P. Li, C. J. C. Burges, Q. Wu et al., McRank: Learning to rank using multiple classification and gradient boosting, *NIPS*, pp.845-852, 2007.

[3] C. Burges, T. Shaked and E. Renshaw, Learning to rank using gradient descent, *Proc. of ICML*, pp.89-96, 2005.
[4] Y. Freund, R. Iyer and R. E. Schapire, An efficient boosting algorithm for combining preferences, *Journal of Machine Learning Research*, vol.3, pp.933-969, 2003.
[5] T. Joachims, Optimizing search engines using clickthrough data, *Proc. of the ACM Conference on Knowledge Discovery and Data Mining*, pp.134-142, 2002.
[6] Z. Cao, T. Qin, T. Liu, M. Tsai and H. Li, Learning to rank: From pairwise approach to listwise approach, *Proc. of the International Conference on Machine Learning*, pp.129-136, 2007.
[7] Q. V. Le and A. J. Smola, Direct optimization of ranking measures, *Journal of Machine Learning Research*, vol.1, pp.1-18, 2010.
[8] O. Chapelle and M. Wu, Gradient descent optimization of smoothed information retrieval metrics, *Information Retrieval Journal*, vol.13, pp.216-235, 2010.
[9] J. Weston and J. Blitzer, Latent structured ranking, *arXiv preprint arXiv:1210.4914*, 2012.
[10] B. Geng, L. Yang, C. Xu et al., Ranking model adaptation for domain-specific search, *IEEE Trans. Knowledge and Data Engineering*, vol.24, no.3, pp.745-758, 2012.
[11] W. Yao, J. He, G. Huang et al., SoRank: Incorporating social information into learning to rank models for recommendation, *Proc. of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, pp.409-410, 2014.
[12] F. Xia, T.-Y. Liu and H. Li, Statistical consistency of top-k ranking, *NIPS*, pp.2098-2106, 2010.
[13] T.-Y. Liu, T. Qin, J. Xu, W. Xiong and H. Li, Letor: Benchmark dataset for research on learning to rank for information retrieval, *LR4IR*, Amsterdam, Netherlands, pp.137-145, 2007.
[14] G. Cong, C. S. Jensen and D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, *Proc. of the VLDB Endowment*, pp.337-348, 2009.
[15] C. J. C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak and Q. Wu, Learning to rank using an ensemble of lambda-gradient models, *Journal of Machine Learning Research*, vol.3, pp.223-235, 2011.
[16] O. Chapelle and Y. Chang, Yahoo! learning to rank challenge overview, *Journal of Machine Learning Research Proceedings Track*, vol.14, pp.81-87, 2011.
[17] A. Agarwal, H. Raghavan and K. Subbian, Learning to rank for robust question answering, *Proc. of CIKM*, pp.139-149, 2012.
[18] S. Agarwal, D. Dugar and S. Sengupta, Ranking chemical structures for drug discovery: A new machine learning approach, *Journal of Chemical Information and Modeling*, vol.5, pp.716-731, 2010.
[19] T. Takenouchi, O. Komori and S. Eguchi, An extension of the receiver operating characteristic curve and AUC-optimal classification, *Neural Computation*, no.8, pp.2789-2824, 2012.
[20] S. Niu, J. Guo, Y. Lan et al., Top-k learning to rank: Labeling, ranking and evaluation, *Proc. of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.751-760, 2012.
[21] I. Tsochantaridis, T. Joachims, T. Hofmann et al., Large margin methods for structured and interdependent output variables, *Journal of Machine Learning Research*, vol.6, no.2, pp.1453-1484, 2006.
[22] S. Sarawagi and R. Gupta, Accurate max-margin training for structured output spaces, *Proc. of the 25th International Conference on Machine Learning*, pp.888-895, 2008.
[23] S. Liu, B. Zhai, S. Chan et al., Approximated slack scaling for structural support vector machines in scene depth analysis, *Mathematical Problems in Engineering*, no.1, pp.1-13, 2013.
[24] K. Subbian and P. Melville, Supervised rank aggregation for predicting influence in networks, *arXiv preprint arXiv:1108.4801*, 2011.
[25] A. Agarwal, H. Raghavan, K. Subbian et al., Learning to rank for robust question answering, *Proc. of the 21st ACM International Conference on Information and Knowledge Management*, pp.833-842, 2012.
[26] C. Dwork, R. Kumar, R. Naor and D. Sivakumar, Rank aggregation methods for the web, *International World Wide Web Conference*, pp.613-622, 2001.
[27] C. Buckley and E. M. Voorhees, *Retrieval System Evaluation, Chapter TREC: Experiment and Evaluation in Information Retrieval*, MIT Press, 2005.
[28] K. Jarvelin and J. Kekalainen, Ir evaluation methods for retrieving highly relevant documents, *Proc. of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.41-48, 2000.