原著論文 Original Paper

Robust Online Learning to Rank via Selective Pairwise Approach Based on Evaluation Measures

Yoshihiko Suhara NTT Service Evolution Laboratories, NTT Corporation
suhara.yoshihiko@lab.ntt.co.jp

Jun Suzuki NTT Communication Science Laboratories, NTT Corporation
suzuki.jun@lab.ntt.co.jp

Ryoji Kataoka NTT Service Evolution Laboratories, NTT Corporation
kataoka.ryoji@lab.ntt.co.jp

keywords: information retrieval, learning to rank, online learning

Summary

Learning to rank is a supervised learning problem whose goal is to construct a ranking model. In recent years, online learning to rank algorithms have begun to attract attention because large-scale datasets have become available. We propose a selective pairwise approach to online learning to rank algorithms that offer both fast learning and high performance. The basic strategy of our method is to select the most effective document pair to minimize the objective function using an entered query present in the training data, and then updates the current weight vector by using only the selected document pair instead of using all document pairs in the query. The main characteristics of our method are that it utilizes adaptive margin rescaling based on the approximated NDCG to reflect the IR evaluation measure, the max-loss update procedure, and ramp loss to reduce the over-fitting problem. Finally, we implement our proposal, PARank-NDCG, in the framework of the Passive-Aggressive algorithm. We conduct experiments on the MSLR-WEB datasets, which contain 10,000 and 30,000 queries. Our experiments show that PARank-NDCG outperforms conventional algorithms including online learning to rank algorithms such as Stochastic Pairwise Descent, Committee Perceptron and batch algorithm such as RankingSVM on NDCG values. In addition, our method only takes 7 seconds to learn a model on the MSLR-WEB10K dataset. PARank-NDCG offers approximately 63 times faster training than RankingSVM on average.

1. Introduction

Ranking is a central part of information retrieval. Many ranking models have been developed, including BM25 [Robertson 94], PageRank [Brin 98] and so on. Modern web search engines commonly analyze and combine a large number of ranking features extracted from the submitted query and stored documents in order to appropriately rank the data against the given query [Büttcher 09]. The ranking function is a core component that outputs the final score used to determine the order of search results.

A recent research direction to construct appropriate ranking functions is to use the machine learning framework to construct appropriate ranking functions [Liu 11] since manual parameter tuning is usually very difficult, especially when there are many features. In general, training data of learning to rank problems consists of queries, their corresponding retrieved documents, and relevance judgments provided by human assessors. There are three major approaches for learning ranking functions, namely pointwise, pairwise and listwise approaches [Liu 11]. They differ in terms of the type of objective function used. The pointwise approach function so as to minimize a loss function that is defined on individual document relevance judgments. The pairwise approach trains a ranking function so as to minimize a loss function based on pairwise preferences. The ranking problem is transformed into the binary classification problem. The problem with this approach is that the objective function is formalized as minimizing errors in pairwise preference classification rather than the ranking of documents. The listwise approach trains a ranking function so as to minimize a loss function based directly or indirectly on IR measures. Since some listwise methods adopt evaluation measures such as NDCG or MAP as the objective function [Taylor 08, Valizadegan 09, Xu 07, Yue 07], the listwise methods are considered

to outperform the pairwise approach in general. Instead, listwise methods incur high computational cost since they calculate the loss function based on a document list in a query.

We focus on online learning to rank algorithms in this paper. There are two reasons for our focus on online learning: (1) enhanced learnability even when challenged with large-scale datasets. (2) fast and incremental learning. With regard to the first reason, large amounts of human relevance judgments can recently be easily obtained from crowdsourcing services (e.g., Amazon Mechanical Turks[*1]). In the machine learning framework, training with larger datasets is expected to achieve higher accuracy. With regard to the second reason, recency ranking, which immediately reflects users interests via real-time feedback, has attracted attention in recent years [Dong 10, Inagaki 10, Moon 10]. Given this background, fast and incremental learning algorithms are required to adjust existing models to suit the users' preferences.

Several online learning to rank methods have been described in previous studies including perceptron-based algorithms [Crammer 01, Elsas 08] and a stochastic algorithm [Sculley 09], which randomly samples document pairs to update weights. The advantage of the conventional online learning to rank algorithms is their lightweight parameter update procedure because the loss function is based on document pairs.

We aim to modify the conventional online pairwise approach to construct a *selective pairwise approach* that replicates the high performance of the listwise approach while matching the low computational cost of the pairwise approach. The novel online learning to rank algorithm proposed in this paper, PARank-NDCG, uses the NDCG measure, which is commonly used in information retrieval. To assign priority scores to the pairs, PARank-NDCG sets the margin size according to the NDCG measure. PARank-NDCG selects the pair that has the maximum loss value, and then updates the weight vector using the Passive-Aggressive algorithm (PA) [Crammer 06]. The reasons for choosing PA as the base algorithm are (1) we can model the priority as the margin size, and (2) it updates the weight in closed form so that only one update procedure is required to rank the selected pair correctly.

We also propose to introduce the ramp loss function to avoid noise effects in the training data. The pairwise approach is known to be sensitive to noise in the training data since a mis-labeled relevance judgment on a document would lead to a large number of mis-labeled document pairs [Chapelle 11]. As robust training of a ranking function is one of the chief goals in learning to rank [Chapelle 11], we resolve this problem to improve the performance of PARank-NDCG.

We conduct experiments on the large-scale MSLR-WEB dataset to evaluate our method and other existing online learning to rank algorithms including Stochastic Pairwise Descent [Sculley 09] and Committee Perceptron [Elsas 08] .

The major contributions of this paper are:

- We propose the selective pairwise approach that selects the max-loss pair in order of priority based on IR evaluation measures.
- We develop a novel online learning to rank algorithm that prepares different margin sizes for the pairs of different relevance levels to achieve high retrieval accuracy while holding the cost to linearithmic-time.
- We adopt the ramp loss function in the proposed method to avoid the negative effects of noise in the training data. We also verify that ramp loss works effectively in learning to rank tasks.

The rest of the paper is organized as follows: In Chapter 2, we describe learning to rank and IR evaluation measures. In Chapter 3, we present the proposed method and its algorithm. We report the experiments conducted and their results in Chapter 4, and discuss the results in Chapter 5. We review related works in Chapter 6 and conclude the paper in Chapter 7.

2. learning to rank

In this chapter, we summarize the framework of learning to rank and we introduce two conventional methods based on the pairwise approach. Next, we briefly introduce the IR evaluation measure used in this paper.

Learning to rank algorithms use training data in a supervised learning framework to generate a ranking function. Training data D consists of query set Q, search result documents for each query, and relevance judgments for each query-document pair. The documents returned for query $q \in Q$ are represented by $\mathbf{d}^{(q)} = (d_1^{(q)}, d_2^{(q)}, \ldots, d_{n_q}^{(q)})^T$, where n_q is the number of search result documents for query q. Each query-document pair is represented by feature vector $\phi(d_i^{(q)}) = \mathbf{x}_i^{(q)} \in \mathbb{R}^m$, where m is the number of ranking features. Note that each document may have different relevance levels due to the input query because relevance judgments are annotated in terms of the search request.

*1 https://www.mturk.com/mturk/

2·1 Pairwise Approach

In the pairwise approach, the algorithms use query-document pairs consisting of two documents that have different relevance levels. Assume that there are two documents ($\mathbf{x}_a^{(q)}$ and $\mathbf{x}_b^{(q)}$) with different relevance levels for the same query q. We can tackle the ranking problem as a binary classification problem by considering $\mathbf{x} = \mathbf{x}_a^{(q)} - \mathbf{x}_b^{(q)}$, $y = \mathrm{sign}(y_a^{(q)} - y_b^{(q)})$. In this paper, we omit (q) if it is obvious.

RankingSVM (RSVM) [Joachims 02] is a well-known pairwise learning to rank algorithm, and uses the pairwise framework to generate a ranking model. RSVM solves the following optimization problem:

$$\begin{aligned}&\text{min. } \|\mathbf{w}\|_2^2 + C\sum_{q\in Q}\sum_{a,b}\xi_{a,b}^{(q)}\\ &\text{s.t. } \mathrm{sign}(y_a^{(q)} - y_b^{(q)})\mathbf{w}^T(\mathbf{x}_a^{(q)} - \mathbf{x}_b^{(q)}) \geq 1 - \xi_{a,b}^{(q)} \forall q, a, b.\end{aligned}$$

This problem can be solved in the same way as standard SVM.

Online learning to rank algorithms based on the pairwise approach have been proposed. We explain Stochastic Pairwise Descent (SPD) [Sculley 09] framework with the Passive-Aggressive algorithm (PA) [Crammer 06] as its learner for updating.

SPD samples a query-document pair from all possible query-document pairs in each iteration and executes the update procedure. Perceptron, PA, or online SVM can be selected as the learner in the update procedure, Since our proposed method is an extension of the PA algorithm, we describe the standard PA algorithm for pairwise learning to rank.

PA is an online linear classification algorithm that performs minimal weight updating if the loss value of the current training instance is greater than zero. Let the feature expression of given document pair $\mathbf{x}_t$ be $\mathbf{x}_a - \mathbf{x}_b$ and the difference of the relevance levels be $y_t = \mathrm{sign}(y_a - y_b)$. The PA algorithm formalizes the weight update of $\mathbf{w}_t$ in the t-th iteration as the following optimization problem:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^m}{\mathrm{argmin}} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \text{ s.t. } \ell_t(\mathbf{w}_t; \mathbf{x}_t, y_t) = 0. \tag{1}$$

Here, ℓ_t is the hinge loss, which is calculated by

$$\ell_t(\mathbf{w}_t; \mathbf{x}_t, y_t) = \begin{cases} 0 & 1 \leq y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) & \text{otherwise.} \end{cases} \tag{2}$$

When $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ is larger than the margin size of 1, ℓ_t becomes zero, and weight update is not executed. Otherwise, it updates the weight vector to make loss ℓ_t equal to zero while it minimizes the L2 norm of the difference between the current weight and the updated weight. Using the Lagrange multipliers method, we can derive the following closed form to solve $\mathbf{w}_{t+1}$ from Eq. 1.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \text{ where } \tau_t = \frac{\ell_t}{\parallel \mathbf{x}_t \parallel^2}.$$

Here, $\mathbf{w}_{t+1}$ is set to correctly classify input $\mathbf{x}_t$. In this way, the PA algorithm updates a weight vector for error correction in one trial.

By using slack variable ξ to take account of constraint errors, we can derive a closed form of the weight update equation in the same way [Crammer 06]. PA-I, which corresponds to one-norm soft-margin SVM, updates the weight by using aggressiveness parameter C:

$$\tau_t = \min\left\{C, \frac{\ell_t}{\parallel \mathbf{x}_t \parallel^2}\right\}.$$

2·2 IR Evaluation Measure

In typical search engines, errors in higher positions in the search result are considered more critical than those in lower positions. Thus, document with higher relevance levels should be ranked higher than documents with lower relevance levels. IR evaluation measures such as NDCG reflect this demand. In this paper, we adopt Normalized Discounted Cumulative Gain (NDCG) [Järvelin 02], which is widely used in IR evaluations. Although Mean Average Precision (MAP) [Büttcher 09] is also commonly used in information retrieval, MAP does not evaluate multi-level relevance levels so it is not used in this paper. Many learning to rank algorithms also aim to improve the NDCG measures [Burges 06, Taylor 08, Xu 07]. NDCG is an evaluation measure that especially focuses on higher positions in the ranking list and higher relevance levels. The value in each position is given an exponential increase in the score against the relevance level whereas the value is decreased logarithmically against the rank. NDCG@k, which is the evaluation measure of the top k in the search results, is calculated as follows [Järvelin 02]:

$$\mathrm{DCG}_q@k \equiv \sum_{i=1}^{k} \frac{2^{y_i^{(q)}} - 1}{\log(1+i)}$$

$$\mathrm{NDCG}_q@k \equiv \frac{\mathrm{DCG}_q@k}{\mathrm{maxDCG}_q@k}$$

where $y_i^{(q)}$ is the relevance judgments of the i-th document in query q, and $\mathrm{maxDCG}_q@k$ is the DCG@k value of the ideal ranking for query q. After normalization, $\mathrm{NDCG@}k \in (0, 1]$. Since most search engines display ten results per page, the range NDCG@1-10 is commonly used to evaluate learning to rank algorithms.

3. PARank-NDCG

In this paper, we propose a novel online learning to rank algorithm that prepares a different margin size for each pair to assign priority, and then selects the max-loss pair for weight updating. We refer to our algorithm as PARank-NDCG (Passive-Aggressive Rank based on NDCG) since it focuses on optimizing approximated NDCG values. PARank-NDCG aims to learn the model that can properly rank documents by using different margin sizes for each query and each document pair. The motivation of having different margin sizes is based on the problem that existing pairwise methods give equivalent loss to all pairs regardless of their relevance levels. For example, they assume that pair $(y_a, y_b) = (5, 1)$ and pair $(y_a, y_b) = (2, 1)$ should be assigned the same loss. This is inappropriate in terms of the multi-level relevance IR evaluation measures such as NDCG.

We first explain how to reflect the NDCG measure in the margin size, and then introduce the max-loss update; we detail an efficient method of finding the max-loss pair. After we describe the ramp loss function, we show details of the PARank-NDCG algorithm.

3·1 Margin Settings Based on NDCG

As described above, the general PA updates a weight with the loss prepared assuming that all document pairs have the same margin size. We propose to use different margin sizes based on approximated NDCG for each relevance level pair. Adapting margin sizes to the different samples is often used in cost-sensitive learning [Shen 05]. Using different margin sizes for each pair in terms of evaluation measures improves the performance of the proposed method. We prepare adaptive margin size $E_q(y_a, y_b)$ for query q and the combination of relevance levels y_a and y_b instead of the constant margin in Eq. 2. Thus we use the following loss function:

$$\ell_t(\mathbf{w}_t; \mathbf{x}_t, y_t) = \begin{cases} 0 & E_q \leq y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \\ E_q - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) & \text{otherwise.} \end{cases} \tag{3}$$

We provide a simple overview of setting a different margin size for each pair with different relevance levels in Figure 1. In this example, four-level relevance judgments are assigned to the documents constituting the search results of a query. There are margins of different sizes between the different relevance levels, and the aim is to set the appropriate margin size in terms of NDCG values.

We explain here how to calculate $E_q(y_a, y_b)$. Margin size $E_q(y_a, y_b)$ of the pair of relevance levels y_a and y_b (where $y_a > y_b$) for query q is calculated based on the drop in NDCG value when we swap the document of relevance level y_a and the document of relevance level y_b for query q.

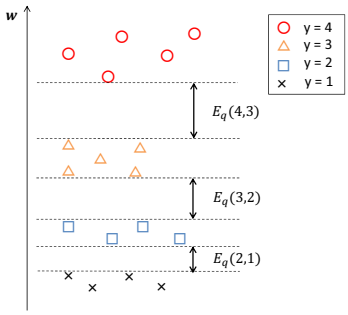


Fig. 1 Margins between the different relevance levels.

We consider the example in which there are (3, 3, 2, 3) documents for relevance levels (4, 3, 2, 1) in query q. In this example, there exist $3 \times 3 = 9$ patterns for swaping a document of relevance level four with a document of relevance level three.

Our method considers only one combination, the one that gives the maximum decrease in NDCG value compared to ideal ranking. That is, we utilize the NDCG value drops by swapping the highest positioned document with relevance level four and the lowest positioned document with relevance level three. We refer to this value as $\Delta\mathrm{NDCG}_q(4, 3)$. In this case, the NDCG value after swapping the documents becomes 0.88, thus, $\Delta\mathrm{NDCG}_q(4, 3) = 1 - 0.88 = 0.12$. In this way, ΔNDCG is calculated by

$$\Delta\mathrm{NDCG}_q(y_a, y_b) = 1 - \mathrm{NDCG}_q(y_a, y_b).$$

Here, scaling parameter E_{scale} is set to make the minimum margin 1 and we use this parameter in scaling:

$$E_q(y_a, y_b) = E_{\mathrm{scale}} \Delta\mathrm{NDCG}_q(y_a, y_b).$$

Since the distribution of relevance levels in any training data set varies with the query [Minka 08], we set the margin sizes to suit each query. Thus, we prepare margin sizes $E_q(y_a, y_b)$ for each pair in query q. We refer to this margin setting method as ΔNDCG.

3·2 Max-loss update and Fast selection method

The selective pairwise method has a mechanism that preferentially selects a document pair that could most affect the NDCG measure. We introduce the *max-loss update*, which is also used in PA [Crammer 06], to build the selective pairwise method. Different from SPD's use of random sampling of the document pair, max-loss update selects the document pair that has the maximum loss value

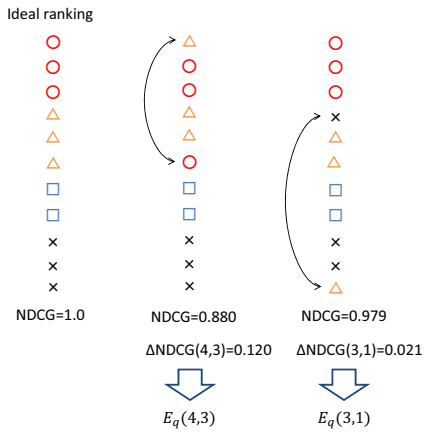


Fig. 2 Margin calculation method.

ℓ_t in the query, and then updates the weight in the manner of PA. This is intuitive; the correction of the pair that has the maximum loss value yields the effective minimization of the overall loss value. Since we use Eq. 3 to calculate the loss value, the mis-ranked document pair that decreases NDCG value is chosen as the max-loss pair. PA-based update then makes the weight value fit the selected pair.

The number of all document pairs for the query is of the order $O(n_q^2)$, where n_q is the number of documents returned for query q. In the worst case, the naive approach incurs $O(n_q^2)$ computation cost to calculate priorities for all pairs to find the max-loss pair. We will show that the margin setting method can decrease the computation cost to linearithmic-time against document size when finding the exact max-loss pair.

Since the margin is given for two different relevance levels, not for each pair, the loss value incurred by a margin size is identical for all pairs of two different relevance levels. Here, we let the higher relevance level be r_h and the lower relevance Level be r_l ($r_l < r_h$). The loss differs in $\mathbf{w}_t \cdot \mathbf{x}_t$, which can be decomposed to $\mathbf{w}_t \cdot \mathbf{x}_a - \mathbf{w}_t \cdot \mathbf{x}_b$. Thus, we can easily find the local max-loss pair with the combination of two relevance levels by selecting the minimum value of $\mathbf{w}_t \cdot \mathbf{x}_a$ in the higher relevance level (i.e., $y_a = r_h$) and the maximum value of $\mathbf{w}_t \cdot \mathbf{x}_b$ in the lower relevance level (i.e., $y_b = r_l$.) The global max-loss pair for the query certainly exists in the local max-loss pairs of all combinations of relevance levels. The number of local max-loss pairs is equivalent to the total combination number of relevance levels in the query.

We analyze the computation cost of this approach. Let m be the number of ranking features, and R the number of relevance levels in the training data. First, we calculate the score of each document given the current weight vector. The time needed is proportional to the document number, i.e., $O(n_q \cdot m)$. Here we can ignore all documents other than the two documents that have the maximum or the minimum score, in the relevance level. The calculation cost to find these two documents is $O(n_q)$. Note that we need not consider the maximum-scored document in the highest relevance level and the minimum-scored document in the lowest relevance level. Thus, only $\frac{1}{2}R(R-1)$ pairs need be considered. Finally, the overall computation cost is $O(n_q \cdot m + n_q + \frac{1}{2}R(R-1))$.

Since the number of relevance levels in the dataset is usually around three to five, at most ten, we can regard $\frac{1}{2}R(R-1)$ as a constant value compared to $n_q \cdot m$. Thus, the max-loss pair selection procedure of PARank-NDCG can be considered as a linear-time algorithm to document size. This analysis indicates that PARank-NDCG can scale to large-scale datasets.

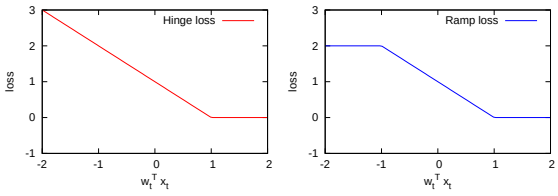
We show a real example using the MSLR-WEB10K dataset in the following experiments. In the dataset, one query has, on average, 120 documents (5,264 document pairs) and the dataset contains five-level relevance judgments. Our method requires only 10 calculations of the loss value to find the global maximum-loss pair while the naive approach requires 5,264 calculations at most. We note that the number of pairs is less than n_q^2 if there are more than two relevance levels and the naive method can make use of the inner-product cache to avoid redundant calculations in determining the loss value.

3·3 Robust Learning using Ramp Loss

The learning to rank task is known to suffer from noise in the training data since it degrades the generated ranking functions [Chapelle 11, Xu 10]. Since we employ the selective pairwise approach, our method can be expected to suffer from noisy data. That is, if the algorithm selects a misjudged or meaningless document pair when optimizing the evaluation measure, the weight update might degrade the final output.

To resolve this problem, we propose to introduce the ramp loss function [Collobert 06]. Ramp loss gives constant loss values to the samples whose loss values exceed a threshold in the hinge loss function. Although ramp loss was originally proposed to reduce the number of support vectors and thus shorten the training time of SVM, it is also known to offer robustness against noisy data [Collobert 06, Wang 10].

We show the differences between hinge loss and ramp loss in Figure 3. In this figure, x-axis plots the $\mathbf{w}_t \cdot \mathbf{x}_t$ value and y-axis plots the size of losses. In this example, the margin size for each loss function is set to 1. Both



(i) (ii)

Fig. 3 Loss functions: (i) hinge loss, (ii) ramp loss.

loss functions give no loss value to the samples outside the margins ($\mathbf{w}_t \cdot \mathbf{x}_t \geq 1$). With hinge loss, the loss value is proportional to the distance from the margin. By contrast, the ramp loss assigns an identical loss value to the data whose errors exceed a certain level ($\mathbf{w}_t \cdot \mathbf{x}_t \leq -1$). Thus, ramp loss has the effect of preventing excessively large loss values from being assigned to greatly misranked pairs.

With ramp loss, we still use $E_q(y_a, y_b)$ for margin size and use constant value (-1) as the start point of constancy in the usual manner of applying ramp loss. Thus, the loss value of weight vector $\mathbf{w}_t$ in the t-th iteration for the pair consisting of document a with relevance level y_a and document b with relevance level y_b is calculated as

$$\ell_t^{ramp}(\mathbf{w}_t; \mathbf{x}_a, \mathbf{x}_b, y_a, y_b, q) = \begin{cases} 0 & \text{if } E_q(y_a, y_b) \leq \mathbf{w}_t \cdot \mathbf{x}_t \\ E_q(y_a, y_b) - \mathbf{w}_t \cdot \mathbf{x}_t & \text{if } -1 < \mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b) \\ 1 + E_q(y_a, y_b) & \text{if } \mathbf{w}_t \cdot \mathbf{x}_t \leq -1. \end{cases} \tag{4}$$

According to [Wang 10], the update equation is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \mathbf{x}_t \quad \text{, where}$$
$$\tau_t = \begin{cases} \min\left\{C, \frac{\ell_t^{ramp}}{\|\mathbf{x}_t\|^2}\right\} & \text{if } -1 < \mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b) \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

We note that with the ramp loss function, our max-loss update procedure selects the document pair that has the highest loss value while it holds $-1 < \mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b)$ as the max-loss pair because the document pair that holds $\mathbf{w}_t \cdot \mathbf{x}_t \leq -1$, or $E_q(y_a, y_b) \leq \mathbf{w}_t \cdot \mathbf{x}_t$ does not change weight value as in Eq. 5. For example, consider that our fast selection method finds that the pair that has the maximum $\mathbf{w}_t \cdot \mathbf{x}_t$ value consists of two documents ($\mathbf{x}_a$ and $\mathbf{x}_b$). $\mathbf{x}_a$ has the minimum score $\mathbf{w}_t \cdot \mathbf{x}_a$ in the relevance level of y_a and $\mathbf{x}_b$ has the highest score $\mathbf{w}_t \cdot \mathbf{x}_b$ in relevance level y_b. If $\mathbf{w}_t \cdot (\mathbf{x}_a - \mathbf{x}_b)$ does not hold $-1 < \mathbf{w}_t \cdot (\mathbf{x}_a - \mathbf{x}_b) < E_q(y_a, y_b)$, the selection is not appropriate in terms of the update with ramp loss. Thus, the naive use of fast selection may not work.

To enable fast selection while using ramp loss, we sort the documents in the relevance level by the calculated score $\mathbf{w}_t \cdot \mathbf{x}$ for each relevance level. Thus, we can easily find the pair that has the maximum loss value in the pairs that hold $\mathbf{w}_t \cdot \mathbf{x}_t \leq -1$. Precisely, the sort cost is $\sum_i^R n_{q,i} \log n_{q,i}$ where $n_{q,i}$ is the document number in the i-th relevance level for query q (thus, $\sum_i n_{q,i} = n_q$). Since $\sum_i^R n_{q,i} \log n_{q,i} < n_q \log n_q$, we describe $O(n_q \log n_q)$. Thus, the overall computation cost is $O(n_q \cdot m + n_q \log n_q + \frac{1}{2}R(R-1))$ and we can regard $\frac{1}{2}R(R-1)$ as a constant value compared to $n_q \cdot (m + \log n_q)$. Then, PARank-NDCG with the ramp loss can be considered as a linearithmic-time algorithm to document size.

Since the documents in the same relevance level are sorted by the value of $\mathbf{w}_t \cdot \mathbf{x}_a$ $(a = 1, \ldots, n_q)$, the procedure repeatedly switches document pair and evaluates $\mathbf{w}_t \cdot \mathbf{x}_a - \mathbf{w}_t \cdot \mathbf{x}_b$ until it finds the document pair that satisfies $\mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b)$. Although it takes $O(n_q)$ calculation cost in the worst case, the procedure usually can find the document pair that satisfies $\mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b)$ early in the procedure. Thus this procedure is still much faster than the naive method, which calculates loss values for all document pairs to find the document pair that satisfies $\mathbf{w}_t \cdot \mathbf{x}_t < E_q(y_a, y_b)$. To investigate the training efficiency of our fast selection method, we compare the actual calculation time of these methods using the real dataset in Section 4·3.

3·4 Algorithm

We tag PARank-NDCG as Algorithm 1. PARank-NDCG receives training data D, the number of repetitions T, and parameter C as input. First, it calculates E_q values for each query in the training data. Next, it updates the weight vector for each incoming query. In the update phase, it selects the document pair that provides the maximum loss value, and then updates the weight. Here,

$$\mathbf{Z} = \{(\mathbf{x}_a, \mathbf{x}_b, y_a, y_b) | \mathbf{x}_a, \mathbf{x}_b \in \mathbf{X}, y_a, y_b \in \mathbf{y}, y_a > y_b\}$$

, which implies a set of document pairs of $y_a > y_b$ for query q. The algorithm executes the above process for all queries and the process is repeated T times. Finally, the algorithm outputs the average of all weight vectors following the strategy of Averaged Perceptron [Collins 02]. Here, $|Q|$ indicates the number of queries in the training data.

Algorithm 1 PARank-NDCG algorithm.

Input: D, T, C

Output: $\mathbf{w}^*$

create E_q for all queries q in Q
$\mathbf{w}_0 \leftarrow \mathbf{0}$
for $i = 1$ to T **do**
  for all queries q in Q **do**
    $(\mathbf{x}_a, \mathbf{x}_b, y_a, y_b) =$
      $\mathrm{argmax}_{(\mathbf{x}_a, \mathbf{x}_b, y_a, y_b) \in \mathbf{Z}} \ell_t^{ramp}(\mathbf{w}_t; \mathbf{x}_a, \mathbf{x}_b, y_a, y_b, q)$
    $\mathbf{x}_t = \mathbf{x}_a - \mathbf{x}_b$
    $\tau_t = \min\left\{C, \frac{\ell_t^{ramp}}{\|\mathbf{x}_t\|^2}\right\}$
    $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \mathbf{x}_t$
  end for
end for
$\mathbf{w}^* = \frac{1}{T|Q|} \sum_{t=1}^{T|Q|} \mathbf{w}_t$
return $\mathbf{w}^*$

Table 2 Summary of datasets.

dataset	#query	#doc/#query	#feature
MSLR-WEB10K	10,000	120.0	136
MSLR-WEB30K	30,000	125.7	136

4. Evaluation

This section describes experiments conducted with PARank-NDCG, and compares its performance to conventional learning to rank algorithms. First, we compare our method to the conventional online learning to rank algorithms on large-scale datasets to verify its performance as an online algorithm. Second, we compared the learning curves and calculation time of these methods to evaluate the efficiency of PARank-NDCG.

4·1 Datasets

The experiments were conducted using the recently released Microsoft Learning to Rank (MSLR-WEB) benchmark dataset*2, which consists of MSLR-WEB10K and MSLR-WEB30K.

In this paper, we use the MSLR-WEB dataset to evaluate our method because (1) it is large enough to provide a valid assessment, (2) the dataset is suitable for the evaluation of actual web search engine settings.

We summarize the datasets in Table 2. #query is the total number of queries in the dataset, #doc/#query is the average number of query-document pairs per query. All query-document pairs in the MSLR-WEB dataset were assigned five-level relevance judgments by human assessors. The default setting of the these five-level relevance judgments in the attached evaluation tool for NDCG calculation was (15, 7, 3, 1, 0). A detailed explanation of the feature sets in the MSLR-WEB dataset is publicly available on the web page*3. Some of the document-query features provided in this collection are query term frequency in title or abstract, BM25 score, and language model scores with various smoothing parameters. We normalized the feature values to the $[0, 1]$ interval by linear scaling.

4·2 Experiments on MSLR-WEB

As the baseline online learning to rank algorithms, we selected Committee Perceptron (ComP) [Elsas 08] and Stochastic Pairwise Descent (SPD) [Sculley 09]. We implemented ComP by C++ and we used the `sofia-ml`*4 as the SPD implementation. For ComP, we used the weighted averaging strategy, which uses a validation set to calculate weights for hypotheses in terms of NDCG since it performs as well as BordaFuse method [Elsas 08] but unlike BordaFuse it does not need to store all the hypotheses when calculating the final ranking. We selected Pegasos-SVM (SPD-SVM) and PA-I (SPD-PA) as the learning methods for SPD and used `rank` as the sampling method since it outperformed `query-norm-rank` in preliminary experiments. As the baseline batch learning to rank algorithm, we selected RSVM. We used `svm_rank`*5 as the RSVM implementation.

MSLR-WEB10K and MSLR-WEB30K have been already split into five for cross-validation (three training sets, one validation set, and one test set). In our experiments, we used the split dataset to conduct 5-fold cross validations. The T value of PARank-NDCG was set to 10 for both MSLR-WEB10K and MSLR-WEB30K. Since the training dataset of each fold contains 6,000 queries on MSLR-WEB10K and 18,000 queries on MSLR-WEB30K, these settings yield 60,000 and 180,000 iterations for each dataset. The iteration number of ComP, SPD-SVM, and SPD-PA was set to 100,000 according to [Sculley 09] for the MSLR-WEB10K dataset, and 300,000 for the MSLR-WEB30K dataset. Trade-off parameter C for PARank-NDCG, SPD-SVM, SPD-PA and RSVM was chosen from $\{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0\}$ so as to achieve the highest NDCG@10 value in the validation set. Committee number for ComP was selected from $\{10, 30, 50, 100\}$ so as to maximize the NDCG@10 value in the validation set.

We evaluated each method by NDCG@1-10, and applied t-test with significant levels of 0.001, 0.01, and 0.05. We show the results in Table 1. Values in bold face are the

∗2 http://research.microsoft.com/en-us/projects/mslr/

∗3 http://research.microsoft.com/en-us/projects/mslr/feature.aspx
∗4 http://code.google.com/p/sofia-ml/
∗5 http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 1 Results from MSLR-WEB10K and MSLR-WEB30K datasets.

(a) MSLR-WEB10K

	NDCG									
	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
PARank-NDCG	**0.3182**	**0.3211**	**0.3276**	**0.3330**	**0.3380**	**0.3423**	**0.3471**	**0.3503**	**0.3541**	**0.3572**
SPD-SVM	0.2871	0.2947	0.3039	0.3104	0.3161	0.3216	0.3267	0.3310	0.3350	0.3384
SPD-PA	0.2830	0.2987	0.3061	0.3131	0.3193	0.3246	0.3297	0.3338	0.3379	0.3415
ComP	0.2409	0.2515	0.2620	0.2697	0.2760	0.2820	0.2876	0.2923	0.2969	0.3008
RSVM	0.2990	0.3118	0.3214	0.3281	0.3339	0.3390	0.3442	0.3483	0.3527	0.3564

(b) MSLR-WEB30K

	NDCG									
	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
PARank-NDCG	**0.3208**	**0.3247**	**0.3304**	**0.3357**	**0.3408**	**0.3448**	**0.3490**	**0.3529**	**0.3567**	**0.3598**
SPD-SVM	0.2845	0.2929	0.2997	0.3061	0.3123	0.3173	0.3218	0.3262	0.3304	0.3339
SPD-PA	0.2869	0.2997	0.3075	0.3147	0.3283	0.3344	0.3397	0.3445	0.3491	0.3533
ComP	0.2461	0.2585	0.2674	0.2752	0.2814	0.2869	0.2922	0.2971	0.3014	0.3056
RSVM	0.3008	0.3118	0.3212	0.3283	0.3344	0.3397	0.3445	0.3491	0.3533	0.3571

maximum achieved by the methods. From Table 1, we can confirm the following results:

- For the MSLR-WEB10K dataset, PARank-NDCG is a significant improvement over SPD-SVM, SPD-PA, ComP for NDCG@1-10 ($p < 0.001$), RSVM for NDCG@1-4 ($p < 0.01$), NDCG@5-7 ($p < 0.05$).
- For the MSLR-WEB30K dataset, PARank-NDCG is a significant improvement over SPD-SVM, SPD-PA, Comp and RSVM for NDCG@1-10 ($p < 0.001$).

The results show that our method outperforms the conventional online learning to rank algorithms and pairwise batch-learning method.

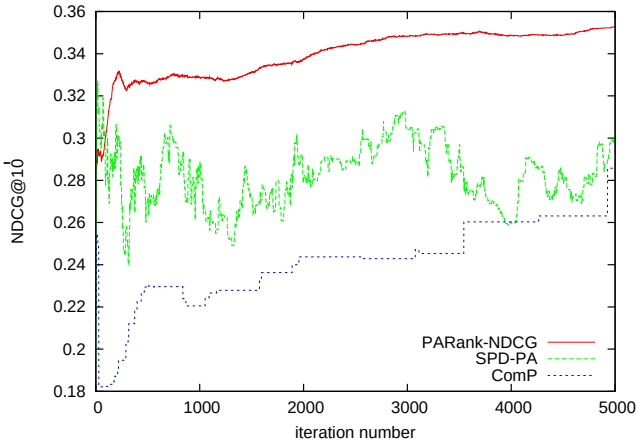


Fig. 4 Learning curve on Fold1 of the MSLR-WEB10K dataset.

4·3 Efficiency Analysis

We compared the learning curves of PARank-NDCG, SPD-PA and ComP to verify the learning efficiency of PARank-NDCG. We used Fold1 of the MSLR-WEB10K dataset. We created a model using the training data, and then evaluated the NDCG@10 value on the test dataset. Each method used the parameter that achieved the highest NDCG@10 value of three methods on the validation set, i.e., $C = 10.0$ for PARank-NDCG, $C = 0.01$ for SPD-PA and 10 committees for ComP.

We show the transition of NDCG@10 value on the test dataset for each iteration in Figure 4. We confirmed that PARank-NDCG offers a more rapid increase in NDCG@10 than SPD-PA or ComP. PARank-NDCG also offers stable performance after 300 iterations. Since PARank-NDCG uses the training data of an incoming query for each iteration, 300 iterations correspond to the 300 queries and they are only $1/20$ of the whole training dataset. This indicates that early termination of training in PARank-NDCG would not harm the final result.

Figure 4 also shows that PARank-NDCG with only 3,000 iterations achieves the same performance as SPD-PA with 100,000 iterations. This result indicates that PARank-NDCG has very fast learning and indeed the one-pass procedure is possible with PARank-NDCG. This means that PARank-NDCG needs to store only the partial data that corresponds to the current query, not the entire training data set.

We also evaluated the calculation cost of the proposed method. As we have already confirmed that PARank-NDCG outperforms SPD-PA and ComP even with one-pass update, we compare the training time of one-pass PARank-NDCG to those of the other algorithms, SPD-PA, ComP and RSVM. Calculation time is the average training time of Fold1-5 on the MSLR-WEB10K dataset. To verify the effectiveness of the fast selection method, we also calculated the training time of PARank-NDCG without the fast

Table 3 Average training time on the MSLR-WEB10K dataset (Fold1-5).

method	training time [sec.]
PARank-NDCG	6.94
PARank-NDCG_naive	44.73
SPD-SVM/SPD-PA	1.16
ComP	37.06
RSVM	442.49

selection method, that is, the method calculates the loss value for all document pairs for a query to find the max-loss pair. We refer to this variant as PARank-NDCG_naive. Note that PARank-NDCG_naive makes use of inner-product cache to avoid the redundant calculation of $\mathbf{w}_t \cdot \mathbf{x}_t$. We show the results in Table 3. Table 3 confirms that PARank-NDCG offers significantly faster training than ComP or RSVM. We also confirm that the fast selection method greatly reduces the training time, from 44.73 sec. to 6.94 sec.

Note that the training time of PARank-NDCG includes the margin setting time. Since it takes 5.7 seconds on average to calculate the margin sizes on the MSLR-WEB10K dataset, these results verify that the weight update procedure of PARank-NDCG is as fast as the random sampling strategy such as SPD-SVM and SPD-PA while PARank-NDCG outperforms them.

5. Discussions

The results gained from MSLR-WEB show that PARank-NDCG achieves higher retrieval accuracy in terms of NDCG than SPD-SVM, SPD-PA and RSVM. From the finding that NDCG@k with small k values achieves better results than with large k values, our method emphasizes the improvement in search results with higher positions. In this chapter, we investigate the effect of (1) margin setting based on evaluation measures, and (2) ramp loss.

First, we verified the effect of basing the margin setting method on ΔNDCG values. We compared the ΔNDCG margin (ΔNDCG) to constant margins (const.), which are used by conventional PA algorithms such as SPD-PA. To verify the effect clearly, we also tested two methods whose algorithm is based on hinge loss in addition to ramp loss. Figure 5 shows the loss functions of those settings.

We show the NDCG@1-10 values of the four methods in Figure 6. Experimental settings are the same as in Section 4·2. Figure 6 shows that regardless of which loss function is selected, the methods with ΔNDCG outperform constant margins for both MSLR-WEB10K and MSLR-WEB30K ($p < 0.001$). This result confirms that

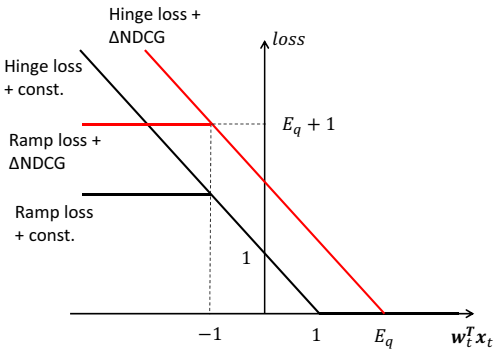


Fig. 5 Comparison of different loss functions.

our margin setting method can improve ranking effectiveness.

We also verified the effect of the ramp loss. The comparison of method (a) to (b) in Figure 6 showed that method (b), which uses ramp loss, outperforms (a) with respect to NDCG@2-10 for both MSLR-WEB10K and MSLR-WEB30K ($p < 0.001$). Also, the comparison of (c) to (d) in Figure 6 showed that ramp loss improves NDCG values even with constant margins ($p < 0.001$). This result shows that using ramp loss as the loss function of PARank-NDCG can improve search accuracy, as well as showing that ramp loss is effective in learning to rank tasks.

6. Related work

A large number of studies have been conducted on learning to rank and its application to information retrieval. Developing algorithms that directly or approximately optimize evaluation measures such as NDCG or MAP is now one of the central issues in learning to rank research. Burges et al. [Burges 06] proposed LambaRank, which models IR evaluation measures such as NDCG and MAP as gradient calculation to achieve direct optimization of the evaluation measures. Xu et al. [Xu 07] proposed Boosting-based method AdaRank, which prepares a weight distribution for each query in the training data. The algorithm uses the distribution to create a weak ranker for each iteration, and then combines them to generate a final ranking function. There are several works that utilize the structured output learning framework for learning to rank algorithms to optimize IR evaluation measures [Chapelle 07, Chakrabarti 08, Yue 07]. Joachims [Joachims 05] proposed large margin method to optimize multivariate performance measures. Yue et al. [Yue 07] extended the framework to develop SVM-MAP, which optimizes MAP value. Chapelle et al. [Chapelle 07] proposed a method that optimizes NDCG directly using a structured output learning framework. Those methods potentially incur $O(n!)$

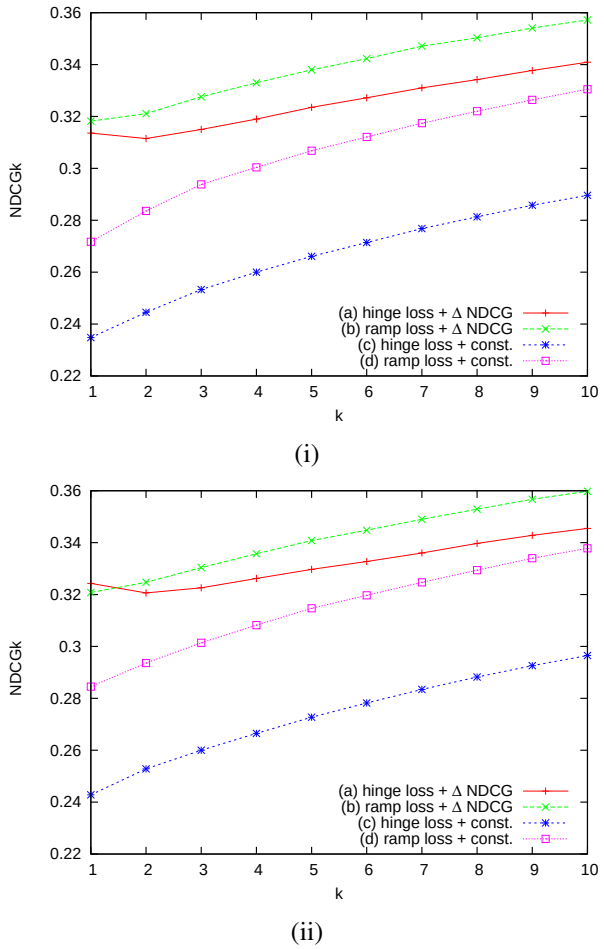


Fig. 6 Comparison of margin size setting on (i) MSLR-WEB10K dataset and (ii) MSLR-WEB30K dataset.

computational cost to obtain the permutation that achieves the maximum weighted value for each update iteration.

We conducted partial experiments using MQ2007 dataset [Liu 07] to compare AdaRank_NDCG [Xu 07], which directly optimizes NDCG value. The experimental settings are basically same as in Section 4·2. We confirmed that our method outperforms AdaRank_NDCG with regard to NDCG@1-10 values. Sepcifically, PARank-NDCG outperforms AdaRank-NDCG by 0.012 points on NDCG@1 and 0.005 points on NDCG@1-10 average.

Since IR evaluation measures such as NDCG or MAP are indifferentiable as regards the weight parameter, we cannot obtain gradient values to optimize these evaluation measures. Thus, some previous works prepare an approximate objective function to optimize IR evaluation measures indirectly. Taylor et al. [Taylor 08] proposed a smoothed function called SoftNDCG, which is differentiable as regards the weight parameter to enable the approximate optimization of NDCG. Valizadegan et al. [Valizadegan 09] proposed a Boosting-based method called NDCG_Boost to optimize the lower bound of NDCG.

Previous works on online learning to rank algorithms include: the perceptron-based algorithm [Crammer 01, Elsas 08] and the stochastic pairwise descent algorithm [Sculley 09]. Crammer and Singer [Crammer 01] proposed the PRank algorithm, which uses a perceptron-based algorithm for model training; it predicts rank level from ordered categories. Elsas et al. [Elsas 08] proposed the Committee Perceptron algorithm, which holds a predefined number of weight vectors (committee) in the training phase, and calculates the weighted-average of them using the validation set to incorporate the evaluation measures into a final ranking function. Sculley [Sculley 09] proposed the random sampling strategy called stochastic pairwise descent, which optimizes a predefined loss function such as the hinge loss.

Similar to our approach, Cao et al. [Cao 06] propose to modify the hinge loss of RSVM by considering two factors: relevance levels from which the pairs were obtained; the number of pairs in the query. Their method, called RSVM-IR, calculates the loss penalty value by using the averaged dropping value of NDCG@1, which is obtained by randomly selecting pairs from the selected relevance levels. Our method differs in two points: First, our method incorporates the decrease of NDCG values in margin size, not the degrees of slopes. Second, the algorithm takes the averaged decrease in NDCG values compared to ideal ranking when randomly swapping the positions of documents in the rankings for all queries. On the other hand, our method prepares margin sizes using the worst dropping NDCG value in the ideal ranking for each query.

7. Conclusion

In this paper, we proposed a selective pairwise approach that selects the most effective pair in order of priority based on IR evaluation measures. We developed PARank-NDCG, a novel online learning to rank algorithm that prepares different margin sizes for pairs at different relevance levels based on approximated NDCG values. PARank-NDCG adopts the max-loss update strategy to select the pair that has the highest priority in terms of the approximated NDCG. It also adopts ramp loss as its loss function to counter the problems caused by noise in the training data.

Experiments on two large-scale datasets, MSLR-WEB10K and MSLR-WEB30K, showed that PARank-NDCG outperforms key conventional online learning to rank algorithms including Stochastic Pairwise Descent, Committee Perceptron and the batch learning to rank algorithm RankingSVM. An efficiency analysis showed that the weight update procedure of PARank-NDCG with the fast selec-

tion method is as fast as the random sampling strategy while PARank-NDCG outperforms the random sampling methods such as SPD-PA, SPD-SVM for NDCG@1-10.

Another contribution of this paper is that it verifies the effectiveness of ramp loss in learning to rank problems. Best to our knowledge, this is the first work to make use of ramp loss in the learning to rank task.

As future work, we plan to theoretically analyze the selective pairwise approach. We also aim to develop a listwise online learning to rank algorithm that directly optimizes IR evaluation measures while matching the training speed of existing online learning to rank algorithms.

♢ References ♢

[Brin 98] Brin, S. and Page, L.: The anatomy of a large-scale hypertextual Web search engine, in *Proceedings of the seventh international conference on World Wide Web 7*, pp. 107–117 (1998)

[Burges 06] Burges, C. J. C., Ragno, R., and Le, Q. V.: Learning to Rank with Nonsmooth Cost Functions, in *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pp. 193–200 (2006)

[Büttcher 09] Büttcher, S., Clarke, C. L. A., and Cormack, G. V.: *Information Retrieval: Implementing and Evaluating Search Engines*, MIT Press (2009)

[Cao 06] Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., and Hon, H.-W.: Adapting ranking SVM to document retrieval, in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 186–193 (2006)

[Chakrabarti 08] Chakrabarti, S., Khanna, R., Sawant, U., and Bhattacharyya, C.: Structured learning for non-smooth ranking losses, in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 88–96 (2008)

[Chapelle 07] Chapelle, O., Le, Q., and Smola, A.: Large margin optimization of ranking measures, in *Proceedings of the NIPS '07 Workshop on Machine Learning for Web Search* (2007)

[Chapelle 11] Chapelle, O., Chang, Y., and Liu, T.-Y.: Future directions in learning to rank, in *Proceedings of the ICML '10 Workshop on Yahoo! Learning to Rank Challenge*, Vol. 14 of *JMLR Workshop and Conference Proceedings*, pp. 91–100 (2011)

[Collins 02] Collins, M.: Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms, in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pp. 1–8, Stroudsburg, PA, USA (2002)

[Collobert 06] Collobert, R., Sinz, F., Weston, J., and Bottou, L.: Trading convexity for scalability, in *Proceedings of the 23rd international conference on Machine learning*, pp. 201–208 (2006)

[Crammer 01] Crammer, K. and Singer, Y.: Pranking with Ranking, in *Proceedings of the 14th annual conference on Neural Information Processing Systems*, pp. 641–647 (2001)

[Crammer 06] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y.: Online Passive-Aggressive Algorithm, *Mach. Learn.*, Vol. 7, pp. 551–585 (2006)

[Dong 10] Dong, A., Chang, Y., Zheng, Z., Mishne, G., Bai, J., Zhang, R., Buchner, K., Liao, C., and Diaz, F.: Towards recency ranking in web search, in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 11–20 (2010)

[Elsas 08] Elsas, J. L., Carvalho, V. R., and Carbonell, J. G.: Fast learning of document ranking functions with the committee perceptron, in *Proceedings of the international conference on Web search and web data mining*, pp. 55–64 (2008)

[Inagaki 10] Inagaki, Y., Sadagopan, N., Dupret, G., Dong, A., Liao, C., Chang, Y., and Zheng, Z.: Session Based Click Features for Recency Ranking, in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (2010)

[Järvelin 02] Järvelin, K. and Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques, *ACM Trans. Inf. Syst.*, Vol. 20, No. 4, pp. 422–446 (2002)

[Joachims 02] Joachims, T.: Optimizing search engines using clickthrough data, in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142 (2002)

[Joachims 05] Joachims, T.: A support vector method for multivariate performance measures, in *Proceedings of the 22nd international conference on Machine learning*, pp. 377–384 (2005)

[Liu 07] Liu, T.-Y., Qin, T., Xu, J., Xiong, W., and Li, H.: LETOR: Benchmark dataset for research on learning to rank for information retrieval,, in *Proceedings of SIGIR2007 Workshop on Learning to rank for information retrieval* (2007)

[Liu 11] Liu, T.-Y.: *Learning to Rank for Information Retrieval*, Springer (2011)

[Minka 08] Minka, T. and Robertson, S.: Selection bias in the LETOR datasets, in *Proceedings of SIGIR2008 Workshop on Learning to rank for information retrieval* (2008)

[Moon 10] Moon, T., Li, L., Chu, W., Liao, C., Zheng, Z., and Chang, Y.: Online learning for recency search ranking using real-time user feedback, in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1501–1504 (2010)

[Robertson 94] Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M.: Okapi at TREC-3, in *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pp. 109–126 (1994)

[Sculley 09] Sculley, D.: Large scale learning to rank, in *Proceedings of the NIPS '09 Workshop on Advances in Ranking* (2009)

[Shen 05] Shen, L. and Joshi, A. K.: Ranking and Reranking with Perceptron, *Mach. Learn.*, Vol. 60, pp. 73–96 (2005)

[Taylor 08] Taylor, M., Guiver, J., Robertson, S., and Minka, T.: SoftRank: optimizing non-smooth rank metrics, in *Proceedings of the international conference on Web search and web data mining*, pp. 77–86 (2008)

[Valizadegan 09] Valizadegan, H., Jin, R., Zhang, R., and Mao, J.: Learning to Rank by Optimizing NDCG Measure, in Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. eds., *Advances in Neural Information Processing Systems 22a*, pp. 1883–1891 (2009)

[Wang 10] Wang, Z. and Vucetic, S.: Online Passive-Aggressive Algorithms on a Budget, in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 908–915 (2010)

[Xu 07] Xu, J. and Li, H.: AdaRank: a boosting algorithm for information retrieval, in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 391–398 (2007)

[Xu 10] Xu, J., Chen, C., Xu, G., Li, H., and Abib, E. R. T.: Improving quality of training data for learning to rank using click-through data, in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 171–180 (2010)

[Yue 07] Yue, Y., Finley, T., Radlinski, F., and Joachims, T.: A support vector method for optimizing average precision, in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 271–278 (2007)

〔担当委員：Takamura Hiroya〕

Received May 18, 2012.

Author's Profile



Suhara, Yoshihiko (Member)

received a B.E., and M.E. from Keio University, Japan, in 2006, and 2008, respectively. In 2008, he joined Nippon Telegraph and Telephone Corporation. His research interests include machine learning and information retrieval.



Suzuki, Jun

received a B.E., and M.E. from Keio University, Japan, in 1999, and 2001, respectively and a Ph.D. in engineering from the Graduate School of Information Science, Nara Institute of Science and Technology, Japan, in 2005. In 2001, he joined NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation. His research interests include machine learning and natural language processing.



Kataoka, Ryoji

received the B.E. and M.E. degrees in electronic engineering from Chiba University, Japan, in 1985 and 1987, respectively. He then joined Nippon Telegraph and Telephone Corporation where he has been engaged in research on transaction processing, multimedia information systems, and information retrieval. He is currently a Senior Research Engineer, Supervisior in NTT Service Evolution Laboratories. He is a member of the Information Processing Society of Japan.