Active Sampling for Rank Learning via Optimizing the Area Under the ROC Curve

Pinar Donmez and Jaime G. Carbonell

Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA, USA
{pinard,jgc}@cs.cmu.edu

Abstract. Learning ranking functions is crucial for solving many problems, ranging from document retrieval to building recommendation systems based on an individual user's preferences or on collaborative filtering. Learning-to-rank is particularly necessary for adaptive or personalizable tasks, including email prioritization, individualized recommendation systems, personalized news clipping services and so on. Whereas the learning-to-rank challenge has been addressed in the literature, little work has been done in an active-learning framework, where requisite user feedback is minimized by selecting only the most informative instances to train the rank learner. This paper addresses active rank-learning head on, proposing a new sampling strategy based on minimizing hinge rank loss, and demonstrating the effectiveness of the active sampling method for rankSVM on two standard rank-learning datasets. The proposed method shows convincing results in optimizing three performance metrics, as well as improvement against four baselines including entropy-based, divergence-based, uncertainty-based and random sampling methods.

Key words: Active learning, document retrieval, rank learning, AUC, hinge loss, performance optimization

1 Introduction

Dynamic ranking-based challenges abound in many applications. For instance, search engines must rank results for each query; review and recommendation sites rank competing products; Netflix ranks movie preferences based on prior user selections and feedback; Amazon ranks books based on collaborative filtering; service recommendation sites rank providers based on match to user's request, price, quality or reliability as judged by others, and geographical distance to the user. Rank-based learning methods are relatively recent. Rank learning thus far has mostly been applied to improving document retrieval, where a global ordering of documents is constructed based on the relevance scores of each document to each given query.

In supervised learning, it is usually necessary for users to examine and label large amounts of training data, but this process often proves impractical for most

real-life applications. Active learning is a paradigm that helps reduce the labeling effort, sometimes by orders of magnitude, via incrementally sampling from an unlabeled pool of instances and requesting the labels (or rank decisions) of only those instances that maximize the information value to the learning function [12, 25]. The learning function is updated each time new labeled instances are added to the training set via retraining. Relevance feedback in information retrieval can be regarded as a rudimentary case of active learning where the top k hits are returned for labeling. However, active learning can improve relevance feedback by selectively sampling the most informative instances for learning. Moreover, active learning can help the creation of retrieval test collections by reducing the number of relevance judgments required from human experts.

Active learning has been studied in the context of classification by several researchers [9, 19, 20, 24, 27, 29]. Most active sampling methods for classification try to minimize the classification error; hence do not take into account the rank order which is crucial for ranking tasks (e.g. an error at the top of the rank order is more consequential than one further down). Moreover, ranking problems are often applied to very skewed data distributions with relevant data being a small minority of the total data as is typical in IR problems. In this paper, we try to address these issues by proposing an active sampling approach for rank learning in the context of document retrieval.

A small number of different algorithms have been proposed for supervised rank learning in the literature [4, 11, 16]. Moreover, standard Support Vector Machine (SVM) training surprisingly leads to very good performance on various ranking problems with respect to the corresponding ordering metrics [2, 17, 23]. SVM solution relies on constructing a separating hyperplane in the input space to maximize the margin between two data sets with opposing class labels. The SVM objective is to minimize the training error measured with hinge loss plus a complexity term subject to some constraints. Recently, [26] has shown that minimizing hinge loss is an accurate approximation for maximizing the area under the ROC curve (AUC). Hence, in retrospect, SVMs should be good rankers, since they implicitly optimize a ranking quality measure, namely the AUC. This claim is also supported by several studies showing that SVM rank learning (RankSVM) [11, 16] provides state-of-the-art performance in learning ranking functions. Hence, we adopt RankSVM as the supervised rank learner in this paper.

The main focus of this paper is a new active sampling method for SVM rank learning, which relies on the relationship between the AUC and the hinge rank loss. We use the relationship proposed by [26] as a theoretically motivated justification for our loss minimization framework. Moreover, our method goes beyond [26]'s work by presenting a robust loss estimation that is crucial for highly skewed datasets typical in ranking tasks. Experimental results indicate a significant improvement over a random baseline and other active sampling methods on real-life corpora.

The remainder of this paper is organized as follows: In Section 2, we recapitulate the relationship between the AUC and the hinge rank loss proposed by [26]

and present our SVM active sampling method for ranking. Experimental evaluations are discussed in Section 3. Finally, we offer our conclusions and potential future directions in Section 4.

2 Loss Minimization

2.1 Hinge Rank Loss and AUC

The hinge loss of a real-valued classifier is defined as $L^H = \sum_{i=1}^{N}[1-y_i(c_i-\theta)]_+$. $c_i \in \mathbb{R}$ is the classifier output, $y_i \in \{-1,+1\}$ are the binary class labels, θ is the real-valued decision threshold, and N is the total number of training instances. $[.]_+$ denotes the positive part, i.e. $[a]_+ = a$ if $a > 0$, and 0 otherwise. Let the classifier outputs c_i be sorted in ascending order, i.e. the smallest output value is assigned the lowest rank. Then, the rank version of the standard hinge loss proposed by [26] becomes:

$$L^{HR} = \sum_{i=1}^{N}[\frac{1}{2} - y_i(r_i - \bar{\theta})]_+ \tag{1}$$

r_i is the rank of the data point $\boldsymbol{x}_i$, $\bar{\theta}$ is the rank threshold defined as $\bar{\theta} = max\{r_i : c_i \leq \theta\} + \frac{1}{2}$ which is half way between two neighboring rank positions where one belongs to the positive(negative) class, and the other belongs to the other class. Note that L^{HR} increases linearly in r_i tracking the standard hinge loss in c_i.

The AUC measure is equivalent to the probability that a randomly chosen member of class $+1$ will have a smaller estimated probability of belonging to class -1 than a randomly chosen member of class -1 [10]. Moreover, AUC is equivalent to the Wilcoxon-Mann-Whitney test statistic [18, 28]; thus it can be written in terms of pairwise comparison of ranks:

$$A = \frac{1}{n^+n^-}\sum_{j=1}^{n^+}\sum_{i=1}^{n^-} I(r_j^+ > r_i^-) \tag{2}$$

where I is the indicator function where $I(a) = 1$ if a is true, and 0 otherwise. n^+ and n^- denote the number of positive(relevant) and negative(nonrelevant) examples, respectively. Steck shows that AUC can be written in terms of the hinge rank loss defined in Equation 1 as follows [26]:

$$A \geq 1 - \frac{L^{HR} - C}{n^+n^-} \tag{3}$$

where C is a constant, independent of the rank order (see [26] for further details). The hinge rank loss is the decisive term in the lower bound on the AUC. Hence, minimizing the hinge rank loss guarantees maximizing the AUC. Similarly, the bipartite ranking error R adopted by [22] is directly coupled with the AUC; i.e. $R = 1 - AUC$. Hence, effectively reducing the bipartite loss guarantees an increase in the AUC. This is supported empirically in Section 3.2 where both our method and the bipartite ranking loss based method of [22] improve the AUC.

2.2 SVM Active Learning for Ranking

Relying on the relationship between the hinge rank loss and the AUC, we propose selecting examples that will minimize the expected hinge rank loss in order to maximize rank-learning as measured by the AUC. Expected loss minimization has been studied before for active learning, but in classification [9, 19, 20], rather than in ranking. Unfortunately, active sampling designed for classification error cannot directly apply to the ranking scenario. Ranking loss is based on the relative position of the entities instead of the absolute class label. The rank position of an error matters significantly since the top of the ordered list is more important than the bottom. Moreover, it is crucial to take into account the data skew typical in ranking datasets when designing sampling algorithms for ranking. In this section, we describe a loss minimization algorithm for active learning in ranking to address these issues.

The expected loss minimization criterion requires each unlabeled example to be tested separately in order to calculate the expected future error if it were chosen for a rank-label. Clearly, this is not efficient for large datasets. Nguyen and Smeulders [20] proposed selecting the examples that have the largest contribution to the current estimated error instead of choosing the sample that produces the smallest future error; $s = \text{argmax}_{i \in I_U} E_{y|\boldsymbol{x}}[(y_i - \hat{y}_i)^2 \mid \boldsymbol{x}_i]$ where I_U is the set of indices of the unlabeled data. We adopt a similar approach but our selection criterion is based on the hinge rank loss rather than the typical loss functions used for classification such as squared loss. The optimization problem for SVM Rank Learning [4, 17] can be written as a loss minimization problem as follows:

$$L^H = \sum_{i=1}^{l} [1 - z_i \langle \boldsymbol{w}, \boldsymbol{x}_i^{(1)} - \boldsymbol{x}_i^{(2)} \rangle]_+ \tag{4}$$

plus a complexity penalty [1]. $\boldsymbol{x}_i^{(1)}$ and $\boldsymbol{x}_i^{(2)}$ correspond to two different examples (i.e. one relevant, one nonrelevant) for a given query (we omit the query subindex for notational simplicity). L^H is specific to a given query. $z_i = +1$ if $\boldsymbol{x}_i^{(1)} \succ \boldsymbol{x}_i^{(2)}$, and $z_i = -1$ otherwise. By algebraic reformulations:

$$L^H = \sum_{i=1}^{l} [1 - z_i \langle \boldsymbol{w}, \boldsymbol{x}_i^{(1)} \rangle + z_i \langle \boldsymbol{w}, \boldsymbol{x}_i^{(2)} \rangle]_+$$

$$L^H = \sum_{i=1}^{l} [(1 - y_i^{(1)} \langle \boldsymbol{w}, \boldsymbol{x}_i^{(1)} \rangle) + (1 - y_i^{(2)} \langle \boldsymbol{w}, \boldsymbol{x}_i^{(2)} \rangle) - 1]_+$$

[1] The decision threshold θ is typically chosen as 0 without loss of generality.

where $y_i^1 = z_i$ and $y_i^2 = -z_i$. The rank version of the above loss function then becomes[2]:

$$L_{HR} = \sum_{i=1}^{l} [(\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\theta})) + (\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\theta})) - 1]_+$$

$$L_{HR} \leq \sum_{i=1}^{l} [\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\theta})]_+ + [\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\theta})]_+ \quad (5)$$

where the rank threshold $\bar{\theta}$ is specific to a given query q. Since the RankSVM implementation takes as input vectors corresponding to individual data points (documents), we use, for convenience, the right hand side of the above inequality as the loss function instead of Equation 4 that uses pairwise difference vectors. In our active learning scenario, this corresponds to selecting the example pair that has the largest expected hinge rank loss $E_{y|\boldsymbol{x}}[[\frac{1}{2} - y_i^{(1)}(r_i^{(1)} - \bar{\theta})]_+ + [\frac{1}{2} - y_i^{(2)}(r_i^{(2)} - \bar{\theta})]_+ \mid (\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)})]$, where the expectation is taken over the posterior distribution of y given $\boldsymbol{x}$. However, picking an optimal pair requires $O(n^2)$ comparisons in a set of size n; hence it is impractical for large-scale ranking applications. Therefore, we proceed with selecting individual example(s) per query with the largest expected loss. A selected example may not be optimal compared to the pair selected according to Equation 5; however, it is a reasonable choice for performance-time tradeoff. In fact, our empirical results show that this strategy is quite effective for learning a good ranker with few labeled instances.

$$E\big[[\frac{1}{2} - y_k(r_k - \bar{\theta})]_+ \mid \boldsymbol{x}_k\big] =$$

$$\hat{P}(y_k = 1 \mid \boldsymbol{x}_k)[\frac{1}{2} - (r_k - \bar{\theta})]_+ + \hat{P}(y_k = -1 \mid \boldsymbol{x}_k)[\frac{1}{2} + (r_k - \bar{\theta})]_+ \quad (6)$$

Sampling according to Equation 6 favors points with the highest uncertainty. RankSVM optimizes pairwise preferences, and it may not learn a reasonable decision threshold. Thus, the estimated decision boundary may not be in correspondence with the true rank threshold. This bias may not affect the ranking performance as long as the correct order is obtained. However, it presents a larger problem in the active-learning-to-rank context. The idea of uncertainty has a different interpretation in ranking since the most uncertain points in ranking problems can be considered as the points whose rankings are closest to the rank threshold. This corresponds to multiple thresholds in a multi-level rating scenario with uncertain points being specific to each threshold. Therefore, the rank threshold should define the decision boundary. In order to simulate this effect, we propose a normalized rank distance measure and incorporate it into

[2] The transformation from Equation 4 to 5 is possible when the data has binary relevance judgments, which is the case for the majority of the benchmark test collections including ones used in this paper. An extension of this work to multi-level ratings is projected future work.

Equation 6 to obtain the following:

$$E\big[[\frac{1}{2} - y_k(r_k - \bar{\theta})]_+ \mid \boldsymbol{x}_k\big] = \Big\{\hat{P}(y_k = 1 \mid \boldsymbol{x}_k)\frac{[\frac{1}{2} - (r_k - \bar{\theta})]_+}{|r_{min} - \bar{\theta}|}(1-\lambda) + \hat{P}(y_k = -1 \mid \boldsymbol{x}_k)\frac{[\frac{1}{2} + (r_k - \bar{\theta})]_+}{|r_{max} - \bar{\theta}|}\lambda\Big\} \quad (7)$$

where $r_{min} = 1$ and $r_{max} = |I_{U_q}|$ since the most relevant examples have the highest rank and vice versa. $|I_{U_q}|$ denotes the size of the unlabeled set for the query q. The normalization in Equation 7 regularizes the effect of the points that are ranked further below in the rank order, and those ranked at the top. Generally, the number of points that are ranked above the threshold would be small since there are only a handful of positive(relevant) examples compared to the large amount of negative(non-relevant) examples in tasks such as document retrieval. Without normalization, the points with rank $r_k > \bar{\theta}$ have little chance of being selected since the rank distance $r_k - \bar{\theta}$ is small. Dividing both distances by their maximum renormalizes them into the same scale, favoring a more balanced estimation. $0 < \lambda < 1$ is a trade-off parameter that controls the weight of the examples on either side of the rank threshold. Setting $\lambda > 0.5$ gives more weight to the examples that are mistakenly ranked above the threshold but are in fact negative(nonrelevant). We tuned the λ parameter on a small dataset not reported in this paper and that resulted in fixing λ at 0.6. Better tuning on a validation set could further improve our results. The outline of our selection algorithm is given in Figure 1.

Input: Labeled data L, Unlabeled data U, # rounds T
Output: A ranking function $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$
for t=1:T
1. Learn a ranking function f on L
2. Rank the examples $\boldsymbol{x}_k \in U$ in ascending order acc. to $f(\boldsymbol{x}_k)$
3. Estimate their posterior, i.e. $\hat{P}(y_k \mid \boldsymbol{x}_k)$
4. Select the top l examples, $U^{(l)}$, when sorted in descending order w.r.t.:
   $\hat{P}(y_k = 1 \mid \boldsymbol{x}_k)\frac{[\frac{1}{2} - (r_k - \bar{\theta})]_+}{|r_{min} - \bar{\theta}|}(1-\lambda) + \hat{P}(y_k = -1 \mid \boldsymbol{x}_k)\frac{[\frac{1}{2} + (r_k - \bar{\theta})]_+}{|r_{max} - \bar{\theta}|}\lambda$
6. Remove $U^{(l)}$ from U and update $L = L \cup U^{(l)}$
end

Fig. 1. Our active sampling algorithm

The class probability $\hat{P}(y_k \mid \boldsymbol{x}_k)$ in step 3 of the algorithm in Figure 1 can be estimated by fitting a sigmoid to the ranking function output:

$$\hat{P}(y_k \mid \boldsymbol{x}_k) = \frac{1}{1 + exp(-y_k * f(\boldsymbol{x}_k))} \quad (8)$$

In this paper, we propose a simple method to construct a calibrated estimate for the posterior class distribution. First, we propose a way to estimate the rank

threshold $\bar{\theta}$ and then we use it to calibrate the posterior. We assume that the true ranking function maximizes the score difference between the lowest ranked relevant and the highest ranked non-relevant examples. We sort the data in ascending order of rank scores and compute the absolute difference of the scores of two neighboring examples. The threshold is then chosen as summarized in Figure 2.

Input: a ranking function f, unlabeled data U

Output: the estimated rank threshold $\hat{\bar{\theta}}$

1. Sort the examples $\boldsymbol{x} \in U$ acc. to $f(\boldsymbol{x})$ to obtain a rank order, i.e. $\boldsymbol{x}_1 \prec \boldsymbol{x}_2 \prec ... \prec \boldsymbol{x}_{r_{max}}$
2. Compute $|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i+1})|\ \forall i = 1, 2, ..., r_{max} - 1$
3. The threshold then becomes: $\hat{\bar{\theta}} = \text{argmax}_{i=1,...,r_{max}-1} |f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i+1})|$

Fig. 2. Posterior calibration

Now we can calibrate the estimate in Equation 8 by adding the output score of the instance whose rank is equal to the estimated threshold, i.e.

$$\hat{P}(y_k \mid \boldsymbol{x}_k) = \frac{1}{1 + exp(-y_k * f(\boldsymbol{x}_k) + f(\boldsymbol{x}_{\hat{\bar{\theta}}}))}$$

We substitute the above estimate into Equation 7 for active instance selection. Now, it should be clear that Equation 7 favors points with the highest uncertainty with respect to the current ranker. This is consistent with many other active sampling methods proposed for classification in which uncertainty-based selection criterion plays an effective role [9, 27, 29], although none of them has previously adopted a normalized uncertainty-based criterion for rank-learning.

3 Experimental Evaluation

3.1 Data and Problem Setup

In order to assess the effectiveness of our active-sampling method, we used the Learning to Rank (LETOR) Benchmark dataset [13]. We report results of our studies on the TREC 2003 and TREC 2004 topic distillation tasks [6, 7] in LETOR, namely TD2003 and TD2004. The relevance assessments are binary and created by human judges. There are 44 features for each document-query pair. In our evaluation, we used query-based normalization into the $[0, 1]$ interval for the features, as suggested by the producers of the LETOR [13] package. There are 50 and 75 queries, each with $\sim$ 1000 documents, in TD2003 and TD2004, respectively. The percentage of relevant documents is 1% in TD2003 and 0.6% in TD2004. The TD2003 and TD2004 datasets come with standard train and test splits divided into 5 folds. In each fold, we randomly picked 11 documents

(one relevant and 10 non-relevant) for each query from the given training data to construct the initial labeled set. The remaining training data is used as the unlabeled set. Each sampling method selects $l = 5$ unlabeled instances per query at each round. Then, the selected instances are labeled and added to the current training set. The performance of the ranker is re-evaluated on the testing data. This procedure is repeated for 20 iterations on every fold, and the averaged results are reported.

We tested the performance of our method (denoted by *LossMin*) against four baselines: the entropy-based sampling method of [22] (denoted by *Entropy*), the uncertainty sampling heuristic of [30] (denoted by *Uncertain*), the divergence-based sampling strategy of [1] (denoted by *Diverse*), and random sampling (denoted by *Random*). *Entropy* method [22] samples the most confusing instances for the current ranker which are identified via estimating the bipartite ranking error [8] that counts an error each time a relevant instance is ranked lower than an irrelevant one. The selection mechanism of [30] favors the most ambiguous set of samples (data pairs that are closest in the rank scores and thus most ambiguous) with respect to the current ranker. *Diverse* method selects samples exhibiting maximal divergence (disagreement) between the current hypothesis and a randomized one [1]. We report AUC, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) as the evaluation measures. The NDCG measure was evaluated at the 10^{th} rank cut-off. The performance at the beginning is the same for all methods since they start with the same initial random samples. The ranking implementation in SVMLight [15] was used with a linear kernel and default parameter settings.

3.2 Empirical Results

Figure 3 shows the performance comparison on the TD2004 and TD2003 datasets. Our method outperforms the others on both datasets. In fact, these results are significant ($p < 0.0001$ on TD2003 and $p < 0.001$ on TD2004 w.r.t. MAP and NDCG10) according to a two-sided paired t-test at 95% confidence conducted over the entire operating range. Furthermore, we can order the methods according to the significance of the results with respect to three evaluation metrics. We denote $p < 0.01$ significance level by $\gg$, $p < 0.05$ significance level by $>$, and statistical indifference by $\approx$. Table 1 shows the relative performance of each active ranker on both datasets with respect to all three evaluation criteria. Unfortunately, *Uncertain* and *Diverse* have low performance, similar to random sampling. *Uncertain* selects instances with the most similar scores, but ignores the fact that examples with the same rank label are likely be assigned similar scores. However, such examples do not provide any additional information to the rank learner, leading to a poor performance. A similar behavior is also observed by [1]. On the other hand, the low performance of *Diverse* is perhaps due to the heavy dependence of this algorithm on a sufficiently large initial training set. *Diverse* divides the initial training set into folds and a ranking function is trained on each fold. These functions are not reliable when the training set size is small; hence do not help improve the performance. Our method, on the other hand,

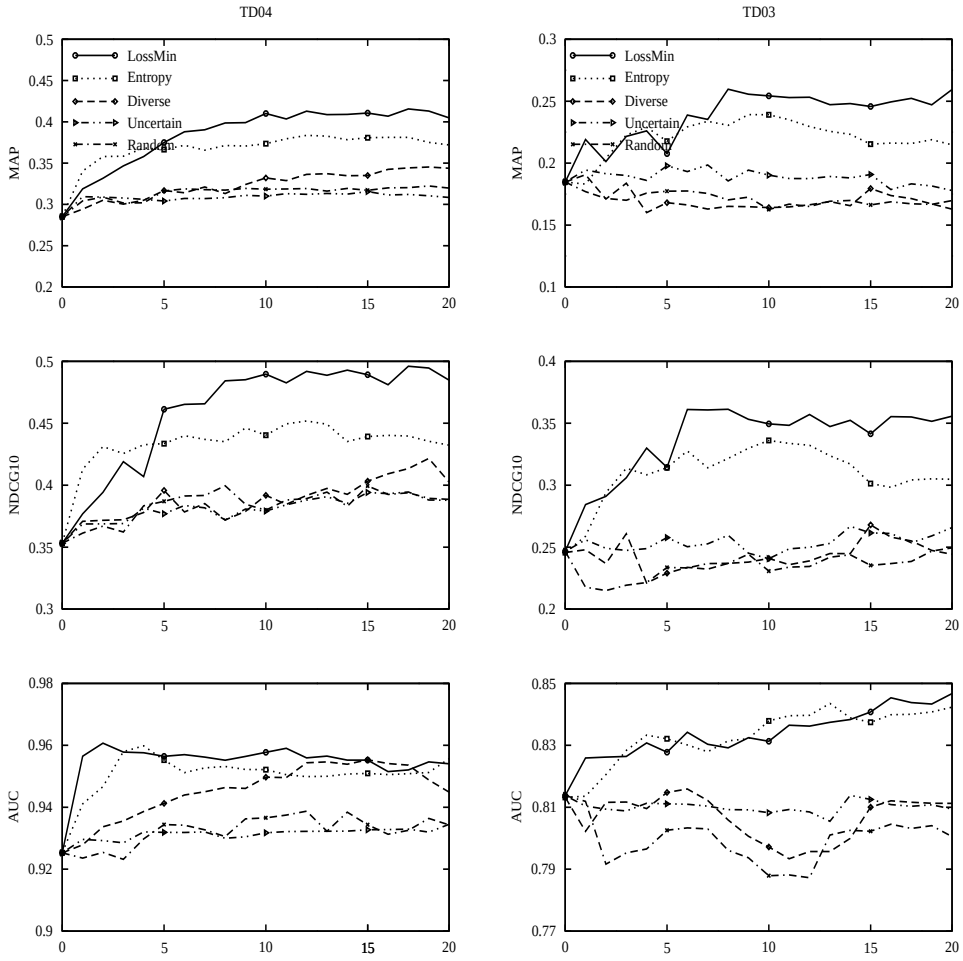

Fig. 3. Average results on TD2004 (left figure) and TD2003 (right figure) datasets. X-axis shows the # of iterations. 5 instances per query are selected per iteration.

effectively increases the performance even when starting with minimal labeled data, which makes it quite useful for many ranking applications that otherwise would require extensive labeling effort, such as document retrieval.

Our method also increases the AUC on both datasets. The increase is quite sharp on TD2004 with a slight decline at the end while it is continuous and gradual on TD2003 dataset. This supports the theoretical claim that our method tries to optimize the AUC metric by sampling the instances that have the largest effect on the expected hinge rank loss, and thereby on maximizing the expected AUC score. A similar improvement is also apparent for the MAP and NDCG@10 metrics, indicating that they might be well correlated with the AUC. On the other hand, *Entropy* method achieves comparable AUC score to ours even though

Table 1. Relative Performance Comparison. $\gg$ denotes $p < 0.01$, $\approx$ denotes indifference, and $>$ denotes $p < 0.05$.

Metric	Dataset	Relative Results
MAP	TD04	LossMin $\gg$ Entropy $\gg$ Diverse $\gg$ Uncertain
	TD03	LossMin $\gg$ Entropy $\gg$ Uncertain $\gg$ Diverse
NDCG10	TD04	LossMin $\gg$ Entropy $\gg$ Diverse $\approx$ Uncertain
	TD03	LossMin $\approx$ Entropy $\gg$ Uncertain $>$ Diverse
AUC	TD04	LossMin $>$ Entropy $>$ Diverse $\gg$ Uncertain
	TD03	LossMin $\approx$ Entropy $\gg$ Uncertain $>$ Diverse

Table 2. Performance and Selection Time Comparison. Iter: the # of iterations. LossMin: the proposed method, Ent: entropy-based method, Diverse: Divergence-based sampling, Un: maximum-uncertainty sampling. Time: training time + ranking time + instance selection time.

Time (cpu-sec)				MAP				Iter
LossMin	Ent	Diverse	Un	LossMin	Ent	Diverse	Un	
1.347	1.347	1.347	1.347	0.184	0.184	0.184	0.184	0
1.977	45.234	36.072	1.947	0.219	0.183	0.190	0.194	1
2.003	48.576	39.127	1.953	0.201	0.204	0.171	0.191	2
2.038	52.865	41.973	1.968	0.221	0.222	0.183	0.189	3
2.110	55.038	44.665	2.001	0.226	0.228	0.160	0.186	4

our method has significantly better MAP and NDCG scores. This is not a very surprising result since the *Entropy* method is suited for the bipartite ranking loss R, which is inversely correlated with the AUC, i.e. $R = 1 - AUC$. But, the main advantage of our method is its use of the normalized rank distance resulting in a more balanced selection for the highly skewed datasets. This sampling favors the mistakenly ranked instances at the top of the ordered list, hence boosts the metrics sensitive to the high ranks, such as MAP and NDCG, without damaging results on the full range as evidenced by our AUC results.

Table 2 summarizes the average results on TD2003 dataset. Due to space constraints, we only show the MAP score and the average selection time for each method for the first 5 iterations. The selection time is calculated as the time each algorithm spends to train the rank learner on the current labeled set, and then to assign scores to the unlabeled examples and finally to select new instances to be labeled. Our method achieves greater learning efficiency with modest computation time in comparison with the other baselines. Particularly, our method has very similar performance to that of the entropy-based method for the early iterations despite the greater complexity of the latter. Nevertheless, our method reaches a significantly better final performance.

4 Conclusions and Future Directions

This paper presented an active learning approach to ranking problem in the context of document retrieval, which is in principle extensible to any other partially

(or totally) ordered ranking task. The novelty of our approach lies in relying on expected loss minimization for rank learning via the use of a normalized ranking loss estimation. Experimental results on real-life corpora show that our sampling algorithm works well in practice on two different datasets, successfully learning a ranking function with many fewer labeling requests than the other baselines, except for comparable performance with the entropy-based method on AUC. On other metrics, our method dominates all the baselines.

This paper takes a step towards showing that active learning is an effective and promising area of research in rank learning. It also opens further questions regarding potential future directions. For instance, we observe that our method while optimizing AUC also improves MAP and NDCG@10 significantly, but its closest competitor (entropy-based method) does not. Further investigation should shed more light on why this is the case. Moreover, if either MAP and NDCG were the real objective function (vs. the precision-recall balanced AUC metric), we would adjust our sampling strategy, or develop a new one that more directly optimizes these metrics.

Ranking problems with a complete order (e.g. a declining-preference ranking for a recommendation system) pose challenges for active learning beyond those based on binary relevance judgments. In such a scenario, the system can ask the oracle (e.g. human user) only about the partial (relative) order of a set of points. In subsequent work, we plan to investigate these scenarios in order to develop effective sampling methods for a wider range of ranking challenges.

References

1. Amini, M., Usunier, N., Laviolette, F., Lacasse, A., Gallinari, P.: A selective sampling strategy for label ranking. In: ECML '06, pp. 18–29. (2006)
2. Brefeld, U., Scheffer, T.: AUC maximizing support vector learning. In: ICML Workshop on ROC Analysis in Machine Learning, (2005)
3. Brinker, K.: Active Learning of Label Ranking Functions. In: ICML '04, pp. 17–24. (2004)
4. Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., Hon, H.-W.: Adapting ranking svm to document retrieval. Proceedings of the international ACM SIGIR Conference on Research and Development in information retrieval (SIGIR'06), pp. 186–193. (2006)
5. Chu, W., Ghahramani, Z.: Extensions of Gaussian Processes for Ranking: Semi-supervised and Active Learning. In: Proceedings of the NIPS 2005 Workshop on Learning to Rank, pp. 29–34. (2005)
6. Craswell, N., Hawking, D., Wilkinson, R., Wu, M.: Overview of the trec 2003 web track. In: Text Retrieval Conference (TREC'03). (2003)
7. Craswell, N., Hawking, D.: Overview of the trec 2004 web track. In: Text Retrieval Conference (TREC'04). (2004)
8. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. In: Journal of Machine Learning Research, vol. 4, pp. 933–969. (2003)
9. Donmez, P., Carbonell, J. G., Bennett, P. N.: Dual strategy active learning. In: Proceedings of the European Conference on Machine Learning, pp. 116–127. (2007)

10. Hand, D. J., Till, R. J.: A simple generalization of the area under the ROC curve for multiple class classification problems. Machine Learning, pp. 171–186. (2001)
11. Gao, J., Qi, H., Xia, X., Nie, J.-Y.: Linear discriminant model for information retrieval. In: Proceedings of the international ACM SIGIR Conference on Research and Development in information retrieval (SIGIR'05), pp. 290–297. (2005)
12. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: SIGIR '94, pp. 3–12. (1994)
13. Liu, T. Y., Xu, J., Qin, T., Xiong, W., Wang, T., Li, H.: `http://research.microsoft.com/users/tyliu/LETOR/`
14. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. In: ACM Transaction on Information Systems, vol. 20(4), pp. 422–446. (2002)
15. Joachims, T.: `http://svmlight.joachims.org/`
16. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02), (2002)
17. Joachims, T.: A support vector method for multivariate performance measures. In: Proceedings of the International Conference on Machine Learning (ICML'05), pp. 377-384. (2005)
18. Mann, H. B., Whitney, D. R.: On a test whether one of two random variables is stochastically larger than the other. Annals of Mathematical Statistics, pp. 50–60. (1947)
19. McCallum, A., Nigam, K.: Employing EM and pool-based active learning for text classification. In: ICML '98, pp. 359–367. (1998)
20. Nguyen, H. T., Smeulders, A.: Active learning with pre-clustering. In: ICML '04, pp. 623–630. (2004)
21. Radlinski, F., Joachims, T.: Active Exploration for Learning Rankings from Clickthrough Data. In: KDD '07, pp. 570–579. (2007)
22. Rajaram, S., Dagli, C. K., Petrovic, N., Huang, T. S.: Diverse Active Ranking for Multimedia Search. In: Computer Vision and Pattern Recognition (CVPR '07). (2007)
23. Rakotomamonjy, A.: Optimizing the area under ROC curve with SVMs. In: ECAI Workshop on ROC Analysis in AI, (2004)
24. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: ICML '01, pp. 441–448. (2001)
25. Seung, H. S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pp. 287–294. (1992)
26. Steck, H.: Hinge rank loss and the area under the ROC curve. In: Proceedings of the European Conference on Machine Learning (ECML'07), pp. 347–358. (2007)
27. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In: Proceedings of International Conference on Machine Learning, pp. 999–1006. (2000)
28. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics 1, pp. 80–83. (1945)
29. Xu, Z., Yu, K., Tresp, V., Xu, X., Wang, J.: Representative sampling for text classification using support vector machines. In: Proceedings of the European Conference on Information Retrieval, (2003)
30. Yu, H.: SVM selective sampling for ranking with application to data retrieval. In: SIGKDD '05, pp. 354–363. (2005)