Optimizing Estimated Loss Reduction for Active Sampling in Rank Learning

Pinar Donmez PINARD@CS.CMU.EDU
Jaime G. Carbonell JGC@CS.CMU.EDU
Language Technologies Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

Learning to rank is becoming an increasingly popular research area in machine learning. The ranking problem aims to induce an ordering or preference relations among a set of instances in the input space. However, collecting labeled data is growing into a burden in many rank applications since labeling requires eliciting the relative ordering over the set of alternatives. In this paper, we propose a novel active learning framework for SVM-based and boosting-based rank learning. Our approach suggests sampling based on maximizing the estimated loss differential over unlabeled data. Experimental results on two benchmark corpora show that the proposed model substantially reduces the labeling effort, and achieves superior performance rapidly with as much as 30% relative improvement over the margin-based sampling baseline.

1. Introduction

Learning to rank has recently drawn broad attention among machine learning researchers (Joachims, 2002; Freund et al., 2003; Cao et al., 2006). The objective of rank learning is to induce a mapping (ranking function) from a predefined set of instances to a set of partial (or total) orders. For instance, in recommendation systems each customer is represented with a set of features ranging from the income level to age and her preference order over a set of products (e.g. movies in Netflix). The ranking task is to learn a mapping from the feature space to a set of permutations of the products. The applications include document retrieval, collaborative filtering, product rating, and so on. In this paper, we are interested in IR applications, and focus on document retrieval. A number of queries are provided such that each query is associated with an ordering of documents indicating the relevance of each document to the given query. Like many other ranking applications, this requires a human expert to carefully examine the documents in order to assign relevance-based permutations. It is often unrealistic to spend extensive human effort and money for labeling in ranking. Thus, it is crucial to design methods that will considerably reduce the labeling effort without significantly sacrificing ranking accuracy.

The active learning paradigm addresses this type of problem. The central idea is to start with only a small amount of labeled examples and sequentially select new examples to be labeled by an oracle. The selected examples are then added to the training set. It is clear that labeling data in ranking requires a complete (or partial) ordering of data whereas in classification labeling considers only absolute class assignments. The target domain of a set of permutations is more complex than that of absolute classes. Hence, it is even more crucial to select the most informative examples to be labeled in order to learn a ranking model using fewer labeled examples.

In this paper, we propose a novel active sampling framework for SVM rank learning (Joachims, 2002), or RankSVM in short, and RankBoost (Freund et al., 2003). The proposed method considers the capacity of an unlabeled example to update the current model if rank-labeled and added to the training set. We show that this capacity can be defined as a function that estimates the error of a ranker introduced by the addition of a new example. The capacity function takes different forms in RankSVM and RankBoost due to different formulations of the ranking function. For example in the case of RankSVM, the ranking function is defined via a normal vector which is a weighted sum of the support vectors whereas the ranking func-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

tion is a weighted sum of weak learners in RankBoost. However, in both cases, the proposed strategy selects the samples which are estimated to produce a faster convergence from the current predicted ranking to the true ranking. Our empirical evaluations on two benchmark corpora from topic distillation tasks in TREC competitions show a significant advantage favoring our method against the margin-based sampling heuristic of (Brinker, 2004; Yu, 2005) and a random sampling baseline.

The rest of the paper is organized as follows: Section 2 provides a brief literature review to the related work. Section 3 motivates the choice of the proposed active learning framework and introduces two novel methods for active learning in the RankSVM and RankBoost settings. In Section 4, we report the experimental results and demonstrate the effectiveness of our methods on benchmark datasets. Finally, we offer our conclusions and next steps in Section 5.

2. Related Work

A number of strategies have been proposed for active learning in the classification framework. Some of those center around the version space (Mitchell, 1982) reduction principle (Tong & Koller, 2000): selecting unlabeled instances that limit the volume of the version space the most, or equivalently selecting the ones with the smallest margin. Some of the others adopt the idea of reducing the generalization error (Roy & McCallum, 2001; Xu et al., 2003; Nguyen & Smeulders, 2004; Donmez et al., 2007): the selection of the unlabeled data that has the highest affect on the test error, i.e. points in the maximally uncertain and highly dense regions of the underlying data distribution (Xu et al., 2003; Nguyen & Smeulders, 2004; Donmez et al., 2007).

Unfortunately, it is not straightforward to extend these theoretical principles to ranking problems. The generalization power of ranking functions is measured by different evaluation metrics than the ones used for classification. Moreover, the classical performance metrics for ranking, such as MAP (Mean Average Precision), precision at the k^{th} rank cut-off, NDCG (Normalized Discounted Cumulative Gain), etc., are harder to directly optimize than the classical loss functions for classification, i.e. log loss, 0/1 loss, squared loss, etc.

Recently, there have been attempts to address the challenges in active sampling for rank learning. Brinker (2004) uses a notion of the margin as an approximation to reducing the volume of the version space. The margin in the ranking scenario is defined as the minimum difference of scores between two instances assuming the ranking solution is a real-valued scoring function. Yu (2005) adopted the same notion of margin for SVM rank learning and proposed a batch mode for instance selection that minimizes the sum of the rank score differences of all data pairs within a set of samples. Yu (2005) proposed an efficient implementation which considers only the rank-adjacent pairs and showed that this strategy is optimal in terms of selecting the most ambiguous set of samples with respect to the ranking function. The major drawback of this margin-based sampling method of (Brinker, 2004; Yu, 2005) is that a scoring function for ranking may assign very similar scores to instances with the same rank label since the ranking function does not distinguish between the relative order of two relevant or two non-relevant examples. However, such instances do not carry any additional information for the rank learner to distinguish between the relevant and the non-relevant data.

Another recent development in active rank learning is the divergence-based sampling method of (Amini et al., 2006). The proposed method selects the samples at which two different ranking functions maximally disagree. One of the two functions is the current ranking function trained on the labeled data, and the other is a randomized function obtained by cross validation. The divergence-based strategy is effective only when provided with a sufficiently large initial labeled set, which is impractical for many real-world ranking applications, such as document retrieval.

3. Active Sampling in Rank Learning

3.1. Motivation

This section presents a novel method for active learning using RankSVM and RankBoost. Roy and McCallum (2001) argue that an optimal active learner is the one that asks for the labels of the examples that, once incorporated into training, would result in the lowest expected error on the test set. The expected error on the test set can be estimated using the posterior distribution $\hat{P}_D(y \mid x)$ of class labels estimated from the training set using some loss function L

$$E_{\hat{P}_D} = \int_x L(P(y \mid x), \hat{P}_D(y \mid x))P(x) \tag{1}$$

Their aim is then to select the point x^* such that when added to the training set with a chosen label y^*, the classifier trained on the new set $\{D + (x^*, y^*)\}$ would have less error than any other candidate x.

$$\forall (x, y) E_{\hat{P}_{D+(x^*,y^*)}} \leq E_{\hat{P}_{D+(x,y)}} \tag{2}$$

Since the true label y^* is unknown, the expectation calculation is carried out by calculating the estimated error for each possible label $y \in Y$, and then taking the average weighted by the current learner's posterior $\hat{P}_D(y \mid x)$. The naive implementation of this method would be quite inefficient and almost intractable on large datasets. Roy and McCallum (2001) address this problem using fast updates for a Naive Bayes classifier. Although efficient re-training procedures are available for some learners such as SVMs (Cauwenberghs & Poggio, 2000), it would still be infeasible for ranking tasks, especially considering the interactive nature of ranking systems. In this paper, we propose a method to estimate how likely the addition of a new example will result in the lowest expected error on the test set *without any re-training on the enlarged training set.* Our method is based on the likelihood of an example to change the current hypothesis significantly. There are a number of reasons why we believe this is a reasonable indicator for estimating that error:

- Adding a new data point to the labeled set can only change the error on the test set if it changes the current learner.
- The more significant that change, the greater chance to learn the true hypothesis faster.
- We note that a big change in the current hypothesis might not always lead to better generalization. However, as more data is sampled and the hypothesis gets closer to the truth, it is less likely that a single outlier could hurt the performance noticeably.

In the following sections, we briefly review the RankSVM and the RankBoost algorithms and propose a novel active learning method for each.

3.2. Preliminaries

Assume the data is represented as a set of feature vectors $\vec{x} \in \mathbb{R}^d$ and corresponding labels (ranks) $y \in Y = \{r_1, r_2, ..., r_n\}$ where n denotes the number of ranks. We assume binary relevance in this paper, though our framework can be generalized to multi-level ranking scenarios as long as the rank learner works on pairwise preference relationships, which is the case for the majority of rank learning algorithms. Features are numerical values for attributes in the data. Assume further that there exists a preference relationship between data vectors such that $y_i \succ y_j$ denotes $\vec{x}_i$ is ranked higher than $\vec{x}_j$. A perfect ranking function $f \in F$ preserves the order relationships between instances:

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

Suppose we are given a set of instances $D = \{(\vec{x}_i, y_i) : (\vec{x}_i, y_i) \in X \times Y\}_{i=1}^m$. The objective for rank learning is to learn a mapping $f : X \times Y \mapsto \mathbb{R}$ that minimizes a given loss function on the training data.

3.3. SVM Rank Learning

Assume $f \in F$ is a linear function, i.e. $f(\vec{x}) = \langle \vec{w}, \vec{x} \rangle$, that satisfies

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i \rangle > \langle \vec{w}, \vec{x}_j \rangle$$

The SVM model targeting this problem can be formulated as a Quadratic Optimization problem:

$$\min_{\vec{w}} \frac{1}{2} \|\vec{w}\|^2 + C \sum \xi_{ij} \tag{3}$$
$$\text{subject to } \langle \vec{w}, \vec{x}_i \rangle \geq \langle \vec{w}, \vec{x}_j \rangle + 1 - \xi_{ij} \text{ , } \xi_{ij} \geq 0 \; \forall i, j$$

The above optimization can be equivalently written by re-arranging the constraints and substituting the trade-off parameter C for $\lambda = \frac{1}{2C}$ as follows:

$$\min_{\vec{w}} \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, \vec{x}_k^1 - \vec{x}_k^2 \right\rangle\right]_+ + \lambda \|\vec{w}\|^2 \tag{4}$$

where $[\ldots]_+$ indicates the standard hinge loss. $\vec{x}^1 - \vec{x}^2$ is a pairwise difference vector whose label z is positive, i.e., $z = +1$ if $\vec{x}^1 \succ \vec{x}^2$ and $z = -1$ otherwise. K is the total number of such pairs in the training set. Finally, a ranked list is obtained by sorting the instances according to the output of the ranking function in descending order.

3.4. Active Sampling for RankSVM

Let us consider a candidate example $\vec{x} \in U$, where U is the set of unlabeled examples. Assume $\vec{x}$ is incorporated into the labeled set with a rank label $y \in Y$. We denote the total loss on the instance pairs that include $\vec{x}$ by a function of $\vec{x}$ and $\vec{w}$, i.e. $D(\vec{x}, \vec{w}) = \sum_{j=1}^{J_y} \left[1 - z_j \left\langle \vec{w}, \vec{x}_j - \vec{x} \right\rangle\right]_+$ where J_y is the number of examples in the training set with a different label than the label y of $\vec{x}$. For instance, J_y is the number of negative(non-relevant) examples in the training set if y is assumed to be positive(relevant), and vice versa. The objective function to be minimized by RankSVM then becomes:

$$\min_{\vec{w}} \left\{ \lambda \|\vec{w}\|^2 + \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, \vec{x}_k^1 - \vec{x}_k^2 \right\rangle\right]_+ + D(\vec{x}, \vec{w}) \right\} \tag{5}$$

Assume $\vec{w}^*$ is the solution to the optimization in Equation 4, and it is unique. Burges and Crisp (2000) show the necessary and sufficient conditions for the uniqueness of the SVM solution. There are only rare cases where uniqueness does not hold, thus it is a rather safe assumption to make. Since we do not actually re-run the optimization problem on the enlarged data, we restrict ourselves to the current solution(hypothesis) $\vec{w}^*$. Instead of re-optimizing, we estimate the effect of adding each candidate instance on the training loss using the current solution to tell how much incorporating x into the labeled set is likely to change the current hypothesis. First, let us consider two cases.

1. Assume $\vec{w}^* = \operatorname{argmin}_{\vec{w}} D(\vec{w}, \vec{x})$
   Then, $\vec{w}^*$ is also the solution to the optimization problem in Equation 5, combining the assumption with $\vec{w}^*$ being the solution to Equation 4. That means, adding $\vec{x}$ to the training set would not change the current hypothesis. From an active learning point of view, this example is useless since the learning algorithm is indifferent to its inclusion.

2. Assume $\vec{w}^* \neq \operatorname{argmin}_{\vec{w}} D(\vec{w}, \vec{x})$
   This is the situation where the current solution could be different if that example $\vec{x}$ were incorporated into training. The magnitude of the difference depends on the magnitude of the deviation of $D(\vec{w}^*, \vec{x})$ from its optimal value, $min_{\vec{w}} D(\vec{w}, \vec{x})$.

We now study the second case in more detail. Let $\hat{\vec{w}}$ be the weight vector that minimizes $D(\vec{w}, \vec{x})$, i.e. $\hat{\vec{w}} = \operatorname{argmin}_{\vec{w}} D(\vec{w}, \vec{x})$. Then, as the difference $\|\vec{w}^* - \hat{\vec{w}}\|$ increases it becomes less likely that $\vec{w}^*$ is optimal for Equation 5. In other words, the current solution $\vec{w}^*$ is in most need of updating in order to compensate for the loss on the new pairs. Let us write $\hat{\vec{w}}$ in terms of $\vec{w}^*$ as follows:

$$\hat{\vec{w}} = \vec{w}^* - \Delta w$$

Minimizing $D(\vec{w}, \vec{x})$ requires working with the hinge loss, the direct optimization of which is difficult due to the discontinuity of the derivative. However, it can still be solved using a gradient-descent-type algorithm[1]. Recall the objective function to be minimized:

$$\min_{\vec{w}} D(\vec{w}, \vec{x}) = \min_{\vec{w}} \sum_{j=1}^{J_y} \left[1 - z_j \langle \vec{w}, \vec{x}_j - \vec{x} \rangle\right]_+ \qquad (6)$$

The derivative of the above equation with respect to $\vec{w}$ at a single point $\vec{x}_j$, $\Delta\vec{w}_j$, is:

$$\Delta\vec{w}_j = \begin{cases} 0 & \text{if } z_j \langle \vec{w}, \vec{x}_j - \vec{x} \rangle \geq 1 \\ -z_j(\vec{x}_j - \vec{x}) & \text{if } z_j \langle \vec{w}, \vec{x}_j - \vec{x} \rangle < 1 \end{cases} \qquad (7)$$

Algorithm 1 RankBoost

```
  Input: initial data distribution $D_1$ over $X \times X$
  for $t = 1$ to $T$ do
     Train a weak learner on $D_t$
     Obtain the weak ranking $h_t : X \mapsto \mathbb{R}$
     Choose a weight $\alpha_t \in \mathbb{R}$ for $h_t$
     $D_{t+1}(\vec{x}^1, \vec{x}^2) = \frac{D_t(\vec{x}^1, \vec{x}^2) exp(-\alpha_t (h_t(\vec{x}^1) - h_t(\vec{x}^2)))}{Z_t}$
  end for
```

We substitute $\vec{w}$ in Equation 7 for the current weight vector $\vec{w}^*$ to estimate how the solution of Equation 6 deviates from it, i.e. $\|\vec{w}^* - \hat{\vec{w}}\| = \|\Delta\vec{w}\|$. We can now write the magnitude of the total derivative as a function of $\vec{x}$ and the rank label y as follows:

$$\begin{aligned} g(\vec{x}, y) &= \|\Delta\vec{w}\| = \sum_j \|\Delta\vec{w}_j\| \qquad (8) \\ &= \sum_{j=1}^{J_y} \begin{cases} 0 & \text{if } z_j \langle \vec{w}^*, \vec{x}_j - \vec{x} \rangle \geq 1 \\ \| - z_j(\vec{x}_j - \vec{x})\| & \text{if } z_j \langle \vec{w}^*, \vec{x}_j - \vec{x} \rangle < 1 \end{cases} \end{aligned}$$

$g(\vec{x}, y)$ estimates how likely the current hypothesis is to be updated to minimize the loss introduced as a result of the addition of the example $\vec{x}$ with the rank label y. Thus, we use this function to estimate the ability of each unlabeled candidate example to change the current learner if incorporated into training. Since the true labels of the candidate examples are unknown, we use the current learner to estimate the true label probabilities. Then, we can take the expectation of $g(\vec{x}, y)$ by taking the weighted sum over the current posterior $\hat{P}(y \mid \vec{x})$ for all $y \in Y$. Among all the unlabeled examples, we choose the one with the highest value for that expectation:

$$\begin{aligned} \vec{x}^* &= \operatorname*{argmax}_{\vec{x} \in U} \sum_{y \in Y} \hat{P}(y \mid \vec{x}) g(\vec{x}, y) \\ &= \operatorname*{argmax}_{\vec{x} \in U} \Big\{ \hat{P}(y = 1 \mid \vec{x}) g(\vec{x}, y = 1) + \qquad (9) \\ &\qquad \hat{P}(y = -1 \mid \vec{x}) g(\vec{x}, y = -1) \Big\} \end{aligned}$$

3.5. RankBoost Learning

RankBoost is a boosting algorithm designed for ranking problems. Like all algorithms in boosting family, RankBoost learns a weak learner on each round, and maintains a distribution D_t over the ranked pairs, $X \times X$, to emphasize the pairs whose relative order is the hardest to learn. An outline of the algorithm is given as Algorithm 1. Z_t is a normalization constant, and the final ranking is a weighted sum of the weak rankings $H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$. For more details and theoretical discussion see (Freund et al., 2003).

[1] For a detailed discussion on solving SVM rank learning using gradient descent, see (Cao et al., 2006).

3.6. Active Sampling for RankBoost

This section introduces a similar method for active sampling for the RankBoost algorithm (Freund et al., 2003). Consider a candidate point $\vec{x} \in U$ and assume it is merged into the training set with rank label $y \in Y$. Unlike RankSVM, RankBoost algorithm does not directly operate with an optimization function. But the ranking loss with respect to the distribution at time t can be written as:

$$\sum_{\vec{x}^1, \vec{x}^2} D_t(\vec{x}^1, \vec{x}^2) I(H(\vec{x}^2) \geq H(\vec{x}^1)) \tag{10}$$

where I is defined to be 1 if the predicate holds and 0 otherwise. Hence, this is a sum over misranked pairs, assuming $\vec{x}^1 \succ \vec{x}^2$. The distribution at time $T+1$ can be written as:

$$D_{T+1}(\vec{x}^1, \vec{x}^2) = D_1(\vec{x}^1, \vec{x}^2) \frac{exp(H(\vec{x}^2) - H(\vec{x}^1))}{\prod_t Z_t} \tag{11}$$

The initial distribution term D_1 can be dropped without loss of generality, assuming it is uniform (which is reasonable given the fact that we do not have prior information about the data). Similarly to RankSVM, we would like to estimate how much the current ranking function would change if the point $\vec{x}$ were in the training set. We estimate this deviation by the difference in the ranking loss after enlarging the current labeled set with each example $\vec{x} \in U$. The ranking loss on the enlarged set with respect to the distribution D_{T+1} is:

$$\sum_{\vec{x}^1, \vec{x}^2} \frac{exp(H(\vec{x}^2) - H(\vec{x}^1))}{\prod_t Z_t} I(H(\vec{x}^2) \geq H(\vec{x}^1)) + \sum_{\vec{x}^j, \vec{x}} \frac{exp(H(\vec{x}^j) - H(\vec{x}))}{\prod_t Z_t} I(H(\vec{x}^j) \geq H(\vec{x})) \tag{12}$$

Note that the rank label y of $\vec{x}$ is assumed to be positive (relevant) with $\vec{x} \succ \vec{x}_j$ in this case. We have a similar calculation for the case where y is assumed to be negative (non-relevant). We adopt the distribution D_{T+1} because 1) it can easily be written in terms of the final ranking function, 2) it contains information about which pairs remain the hardest to determine after the iterative weight updates. Then, the difference in the ranking loss between the current and the augmented set simply becomes:

$$\Delta L(\vec{x}, y = 1) = \sum_{\vec{x}^j, \vec{x}} \frac{exp(H(\vec{x}^j) - H(\vec{x}))}{\prod_t Z_t} I(H(\vec{x}^j) \geq H(\vec{x})) \tag{13}$$

This difference indicates how much the current ranking function needs to be modified to compensate for the loss incurred by including this example. Note that $I(x \geq 0) \leq e^x$ for $\forall x \in \mathbb{R}$ (Freund et al., 2003). Therefore, the upper bound on ΔL can be written as:

$$\Delta L(\vec{x}, y = 1) \leq \sum_{\vec{x}^j, \vec{x}} \frac{exp(2(H(\vec{x}^j) - H(\vec{x})))}{\prod_t Z_t} \tag{14}$$

$\Delta L(\vec{x}, y = -1)$ can be similarly bounded, e.g. $\Delta L(\vec{x}, y = -1) \leq \sum_{\vec{x}, \vec{x}^m} \frac{exp(2(H(\vec{x}) - H(\vec{x}^m)))}{\prod_t Z_t}$. Now, the loss difference can be estimated by taking the expectation over the possible rank labels of $\vec{x}$ with respect to the current ranker's posterior, $\hat{P}(y \mid \vec{x})$:

$$E_{\hat{P}}(\Delta L(\vec{x})) = \hat{P}(y = 1 \mid \vec{x}) \Delta L(\vec{x}, y = 1) + \hat{P}(y = -1 \mid \vec{x}) \Delta L(\vec{x}, y = -1) \tag{15}$$

Note the similarity with Equation 9 in the SVM case. Finally, we select the instance $\vec{x}$ that has the highest expected loss differential, e.g. $\vec{x}^* = \mathrm{argmax}_{\vec{x}}\, E_{\hat{P}}(\Delta L(\vec{x}))$. For notational clarity, we take the maximum over the upper bound in Equation 14 as follows:

$$\vec{x}^* = \underset{\vec{x} \in U}{\mathrm{argmax}} \Big\{ \hat{P}(y = 1 \mid \vec{x}) (\sum_{\vec{x}^j, \vec{x}} exp(2(H(\vec{x}^j) - H(\vec{x})))) + \hat{P}(y = -1 \mid \vec{x}) (\sum_{\vec{x}, \vec{x}^m} exp(2(H(\vec{x}) - H(\vec{x}^m)))) \Big\} \tag{16}$$

For simplicity, we leave out the normalization constant $\prod_t Z_t$ since we are interested in the relative expectation rather than the absolute expectation.

3.7. Final Selection

The sample selection in both RankSVM and RankBoost requires estimating a posterior label distribution. We adopt a sigmoid function to estimate that posterior in the SVM case, as suggested by (Platt, 1999):

$$\hat{P}(y \mid \vec{x}) = \frac{1}{1 + exp(-y * f(\vec{x}) + C)}$$

where $f(\vec{x})$ is the real-valued score of the ranking algorithm, and C is a constant for calibrating the estimate. C is tuned on a separate corpus not used for evaluation in this paper. The final ranking in RankBoost is a sum of weak learners with the corresponding weights. When the weights are too small (or too large), the posterior gets close to the extreme (either 0 or 1) regardless of the example. Hence, we normalize the RankBoost output dividing by the maximum possible

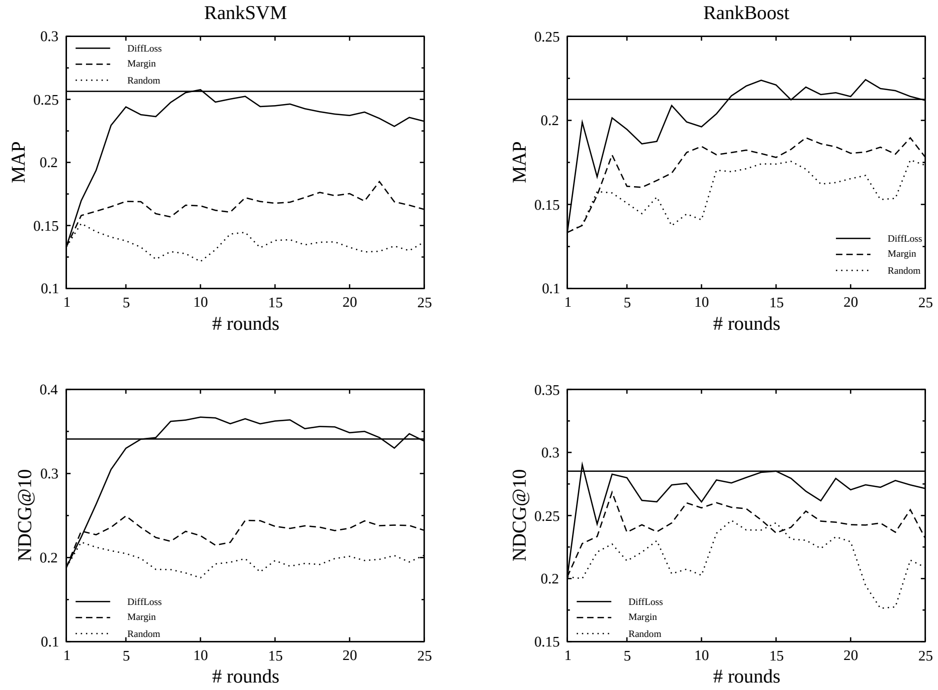

Figure 1. Comparison of different active learners on TREC 2003. The horizontal line indicates the performance when the entire training data is used. Only ∼ 15% of the training data is actively labeled in total by each method.

rank score without changing the rank order:

$$\hat{P}(y \mid \vec{x}) = \frac{1}{1 + exp(-y * \frac{H(\vec{x})}{\sum_{t=1}^{T} \alpha_t} + C)}$$

Note $max_{\vec{x}} H(\vec{x}) = max_{\vec{x}} \sum_{t=1}^{T} \alpha_t h_t(\vec{x}) = \sum_{t=1}^{T} \alpha_t$ since the weak learner $h_t(\vec{x})$ in RankBoost is a {0,1}-valued function defined on the ordering information provided by the corresponding feature (Freund et al., 2003).

4. Evaluation

4.1. Data and Settings

We used two datasets in the experiments: TREC 2003 and 2004 topic distillation tasks in LETOR (Liu et al., 2007). The topic distillation task in TREC is very similar to web search where a page is considered relevant to a query if it is an entry page of some web site relevant to the query. The relevance judgments on the web pages with respect to the queries are binary. There are 44 features, e.g. content and hyperlink features, each of which is extracted from each document-query pair and normalized into $[0, 1]$. There are 50 and 75 queries with 1% and 0.6% relevant documents in TREC03 and TREC04, respectively. The total number of documents per query is ∼ 1000 for both datasets. We used the standard train/test splits over 5 folds in LETOR. For each fold, we randomly picked 16 documents including exactly one relevant document per query for initial labeling. The remaining training data is considered as the unlabeled set. We compared our method with the margin-based sampling of (Brinker, 2004; Yu, 2005) and random sampling baselines. Each method selects 5 documents per query for labeling at each round, e.g. our method selects the top 5 documents according to the criteria in Equation 9 and 16. Then, the ranking function is re-trained, and evaluated on the test set. This process is repeated for 25 iterations which corresponds to labeling only ∼ 15% of the entire training data. The reported results are averaged over 5 folds.

We adopted two standard, widely used performance metrics for evaluation, namely the Mean Average Precision (MAP) and the Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002). For a single query, average precision is defined as the average of the precision as computed at each rank for all relevant documents: $AP = \frac{\sum_{n=1}^{N}(P(r)*rel(r))}{\text{\# relevant documents for this query}}$ where r is the rank, N is the number of documents retrieved, $rel()$ is a binary function on the relevance of a given rank, and $P(r) = \frac{\text{\# relevant docs in top r results}}{r}$ is the precision at

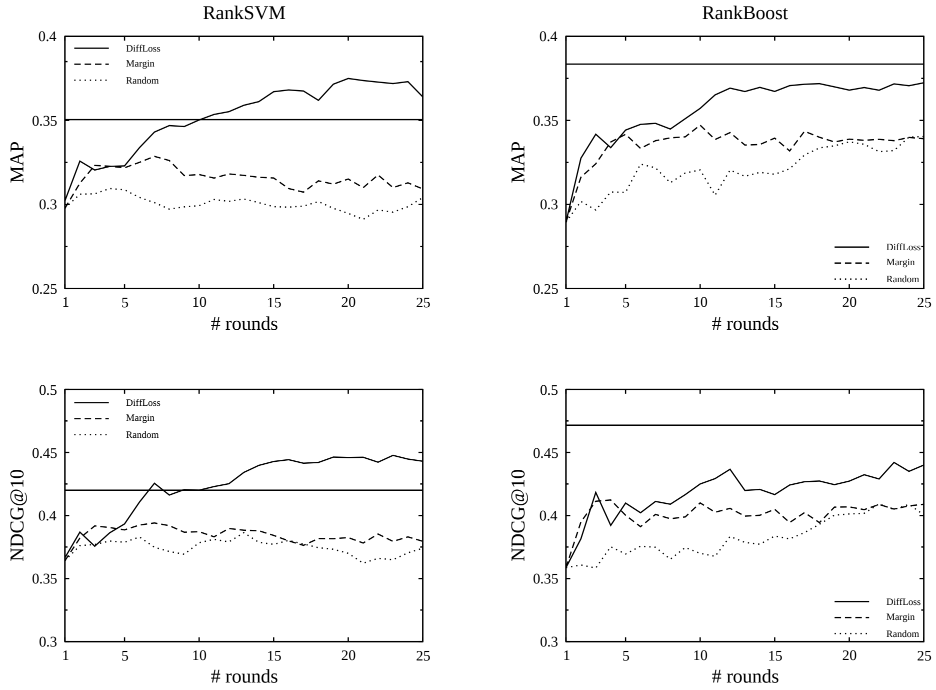

Figure 2. Comparison of different active learners on TREC 2004. The horizontal line indicates the performance when the entire training data is used. Only $\sim 15\%$ of the training data is actively labeled in total by each method.

the rank cut-off r. MAP is obtained by averaging the AP values for all queries. NDCG is cumulative and discounted since the overall utility of a list is measured by the sum of the gain of each relevant document, but the gain is discounted as a function of rank position. The NDCG value of a rank list at position n is given as follows: $NDCG@n = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{log(1+j)}$ where $r(j)$ is the rank of the j^{th} document in the list, and Z_n is the normalization constant so that a perfect ranking yields an NDCG score of 1.

4.2. Results

Figure 1 and 2 plot the performance of the proposed method (denoted by *DiffLoss*), and as comparative baselines, the margin-based sampling and random sampling strategies on TREC 2003 and 2004 datasets. DiffLoss has a clear advantage over margin-based and random sampling in all cases with respect to different evaluation metrics. The differences over the entire operating range are also statistically significant ($p < 0.0001$) according to a two-sided paired t-test at 95% confidence level. DiffLoss especially achieves 30% relative improvement over the margin-based sampling for RankSVM on TREC 2003 dataset.

The horizontal line in each figure indicates the performance if all the training data was used, which we call the "optimal" performance. The performance of DiffLoss for RankBoost is comparable to the "optimal" on TREC 2003 and 2004 datasets. In case of RankSVM, DiffLoss is close to the "optimal" on TREC 2003, and outperforms it on TREC 2004 dataset. More precisely, DiffLoss using RankSVM reaches the optimal performance (even surpassing it on TREC 2004) after 10 rounds of labeling on average (labeling 5 documents per query at each round). DiffLoss using RankBoost, on the other hand, reaches 95% and 90% of the optimal performance on MAP and NDCG@10, respectively on TREC 2004 dataset after 10 rounds. This suggests that carefully chosen samples might lead to a higher level of accuracy than blindly using large amounts of training data. This is an important development over traditional supervised rank learning since it not only reduces the expensive labeling effort, but also may lead to greater generalization power. As follow-up work, we intend to explore methods that will automatically tell the sampling algorithm when to stop so that maximum gain with minimum cost is obtained, as well as exploring the underlying criteria for measuring the quality of actively selected examples.

We conducted another set of experiments to test the hypothesis that selecting a diverse set of samples might lead to better results. We adopted the maximal marginal relevance principle of (Carbonell & Goldstein, 1998), originally proposed for text summarization. The idea is to select samples for labeling such that they have both the maximum potential to change the current ranking function and are maximally dissimilar to each other. See (Carbonell & Goldstein, 1998) for more details. However, incorporating this diversity principle into our selection criteria only slightly improved our results at the very beginning of the learning curve, but the improvement vanished afterwards. Thus, we do not report these results here in this paper.

5. Conclusion

We proposed two novel active sampling methods based on SVM rank learning and RankBoost. Our framework relies on the estimated risk of the ranking function on the labeled set after adding a new instance with all possible labels. The samples with the largest expected risk(loss) differential are selected to maximize the degree of learning at the fastest rate. Empirical results on two standard test collections indicate that our method significantly reduces the required number of labeled examples to learn an accurate ranking function. Possible extensions of this work include a study of the risk minimization in terms of direct optimization of ranking performance metrics, such as MAP, NDCG, precision@k, etc. and self-regulating algorithms that can decide when to terminate.

Acknowledgments

This material is based in part upon work supported by the Defense Advanced Projects Research Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

Amini, M., Usunier, N., Laviolette, F., Lacasse, A., & Gallinari, P. (2006). A selective sampling strategy for label ranking. *ECML '06* (pp. 18–29).

Brinker, K. (2004). Active learning of label ranking functions. *ICML '04* (pp. 17–24).

Burges, C., & Crisp, D. (2000). Uniqueness of the svm solution. *NIPS '00* (pp. 223–229).

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). 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).

Carbonell, J., & Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. *SIGIR '98* (pp. 335–336).

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *NIPS '00* (pp. 409–415).

Donmez, P., Carbonell, J., & Bennett, P. (2007). Dual strategy active learning. *Proceedings of the European Conference on Machine Learning* (pp. 116–127).

Freund, Y., Iyer, R., Schapire, R., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transaction on Information Systems*, *20(4)*, 422–446.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02).*

Liu, T., Xu, J., Qin, T., Xiong, W., Wang, T., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *SIGIR '07 Workshop: Learning to Rank for IR.*

Mitchell, T. (1982). Generalization as search. *Journal of Artificial Intelligence*, *18*, 203–226.

Nguyen, H., & Smeulders, A. (2004). Active learning with pre-clustering. *ICML '04* (pp. 623–630).

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 61–74.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *ICML '01* (pp. 441–448).

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *Proceedings of International Conference on Machine Learning* (pp. 999–1006).

Xu, Z., Yu, K., Tresp, V., Xu, X., & Wang, J. (2003). Representative sampling for text classification using support vector machines. *ECIR '03.*

Yu, H. (2005). Svm selective sampling for ranking with application to data retrieval. *SIGKDD '05* (pp. 354–363).