Rule-Based Active Sampling for Learning to Rank

Rodrigo Silva, Marcos A. Gonçalves, and Adriano Veloso

Department of Computer Science, Federal University of Minas Gerais
{rmsilva,mgoncalv,adrianov}@dcc.ufmg.br

Abstract. Learning to rank (L2R) algorithms rely on a labeled training set to generate a ranking model that can be later used to rank new query results. Producing these labeled training sets is usually very costly as it requires human annotators to assess the relevance or order the elements in the training set. Recently, active learning alternatives have been proposed to reduce the labeling effort by selectively sampling an unlabeled set. In this paper we propose a novel rule-based active sampling method for Learning to Rank. Our method actively samples an unlabeled set, selecting new documents to be labeled based on how many relevance inference rules they generate given the previously selected and labeled examples. The smaller the number of generated rules, the more dissimilar and more "informative" is a document with regard to the current state of the labeled set. Differently from previous solutions, our algorithm does not rely on an initial training seed and can be directly applied to an unlabeled dataset. Also in contrast to previous work, we have a clear stop criterion and do not need to empirically discover the best configuration by running a number of iterations on the validation or test sets. These characteristics make our algorithm highly practical. We demonstrate the effectiveness of our active sampling method on several benchmarking datasets, showing that a significant reduction in training size is possible. Our method selects as little as 1.1% and at most 2.2% of the original training sets, while providing competitive results when compared to state-of-the-art supervised L2R algorithms that use the complete training sets.

1 Introduction

Many applications need ranking functions to order results before presenting them to their users, mainly when there is a large potential number of candidate results. Search engines and product recommendation systems, for instance, need to rank results based on their estimated relevance with respect to a query or based on a user profile and/or personal preferences. In the last few years, there has been considerable interest in the research community for machine learning techniques to learn to rank lists of documents or other data effectively [20]. Learning to Rank algorithms, which deliver superior performance when compared to more traditional approaches such as BM25 [13], use labeled training sets to build ranking models that are used to rank results at query time. The effectiveness of the learned functions is usually directly correlated with the amount of supervised training data available [9]. Yet it is very costly and laborious to produce training sets containing labeled instances, since they must be assessed by a human annotator. Some of the benchmarking datasets we use in our experimental evaluation, for

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 240–255, 2011.
© Springer-Verlag Berlin Heidelberg 2011

example, have training sets comprised of almost 90,000 labeled instances. In most real-life settings it is unfeasible to create such large sets from scratch.

Active learning techniques have been given much attention lately to help deal with the labeling effort problem [2, 3, 9, 21]. These techniques are used to actively sample an unlabeled dataset to select instances that maximize the effectiveness of learned rank functions. By carefully selecting and labeling instances it may be possible to use a (very) small training set and yet achieve highly effective learned functions, thus minimizing the labeling effort. The rationale behind active learning is that the instances selected are representative of the document corpus and somehow more "informative" to the learning algorithm. By using only a highly informative set, there may be a reduction of noise and uncertainty in the learning process, possibly yielding even more effective functions than those obtained using a large training set. In most active learning methods, samples are selected incrementally from an unlabeled set and are used to update an already existing learning function. At each step the most "informative" instances are selected to be labeled and added to the training set. Several strategies and heuristics have been proposed in the literature on how to determine the most informative samples. In some methods, the most ambiguous - or those instances for which the learner is most *uncertain* about - are selected [7]. In other settings, a query-by-committee (QBC) strategy is employed where competing learners vote on the label of the candidate samples and those about which they most disagree are selected [17]. Some algorithms select the samples that would cause the greatest change to the currently learned function [16]. Other methods select instances that would lead to the minimal expected future error or, similarly, optimize some other performance measure such as precision or recall [15].

There has been work proposing active learning methods for classification tasks [4,10, 11]. While classification functions output an specific class for each data item, ranking functions must produce partial orders of items either through some scoring function, pairwise ordering or listwise ordering of the items. Most active sampling methods for classification try to directly minimize the classification error, but this approach is not straightforward to extend to the ranking problem since position-based measures such as MAP and NDCG are usually non-continuous and non-differentiable [8]. Additionally, in most supervised learning settings, samples can be treated as independent of each other which is not the case for L2R where each sample represents a document *relative to a query*. Thus, in L2R, instances are conditionally independent given a query [9].

In this paper we propose a new active sampling technique based on association rules. Learning to rank using demand-driven association rules has been shown to provide competitive ranking quality [19]. The method proposed here uses association rules to actively select documents[1] from an unlabeled set based on how many inference rules are generated for each document. At each step, a new document is chosen for labeling as the most "dissimilar" with respect to the already selected samples (but with no need to create a model), with the goal of increasing diversity and representativeness. This is performed by choosing from the unlabeled data the document u_i that generates the smallest number of rules when considering a "projection" of the current selected training set with respect to the features of u_i. This projection aims at removing examples from the current training set that are not useful in producing rules for u_i.

[1] We use the terms documents, instances, samples, and examples interchangeably in this paper.

The more dissimilar the candidate document, the fewer the rules generated for it, meaning that few already labeled instances in the projected training set share common feature-values with the candidate. This process is repeated until the algorithm converges, which happens when an already selected document is selected again, i.e., there is no more information useful for generating rules from any other document in the unlabeled set.

Despite being very effective in several cases, as demonstrated in our experiments, our method may sometimes converge too fast, resulting in a very small labeled training set. We propose a strategy to delay this convergence and increase the size and diversity of the labeled set which consists in vertically partitioning the features in the unlabeled set, generating in practice several reduced (in terms of features, not instances) unlabeled sets, over which our active sampling method can be applied. Once the final reduced training set is created by the union of the documents selected in each partition, we apply the supervised on-demand association rule algorithm to rank the test set. One of the key advantages of our method is that it can be directly applied on an unlabeled set containing the extracted features for the documents in the corpus, without the need of an initial labeled set. Furthermore, once the unlabeled set is generated (i.e. feature extraction is performed on the corpus) and processed (i.e. discretized and partitioned), the method can be directly applied without extra parametrization since it naturally converges while selecting the training samples. This is different from most previous work where there is no clear stop criterion and the number of iterations has to be empirically determined.

We compare our method against a number of baselines that use the complete labeled training set, including some published by the LETOR dataset producers with many supervised L2R algorithms, and show that it is competitive when compared to several of them while selecting and using as little as 1.12% and at most 2.18% of the original training sets (average of 1.63% for all datasets considered). Best results reported in the literature with other active sampling strategies for L2R reported selecting at least 11% of the training set to produce similar competitive results. Moreover, in most cases our method improved the results of the original on-demand associative method (by as much as 13%), or produced similar results when compared to using the whole training set, confirming the hypothesis of noise reduction.

2 Related Work

Some researchers have recently proposed active learning schemes for L2R based on the optimization of approximations of position-based measures. The authors of [9], for example, propose a general active learning framework based on expected loss optimization (ELO). The framework uses function ensembles to select training examples that minimize a chosen loss function. The authors approach the unique challenges of active learning in L2R by separating their selection algorithm into query level and document level parts (what they call *Two-stage Active Learning*). To approximate their chosen metric, namely, DCG (*Discounted Cumulative Gain*), they use ensembles of learners to produce relevance scores and estimate predictive distributions for the documents in the active learning set. To produce the ensemble, they use a bootstrap technique that relies on an initial labeled set. Thus, their technique requires an initial labeled set to build the ensemble of learners, which needs to be large enough for the learners in the ensemble

to be minimally effective. In their case, the initial labeling sets contain thousands of instances.

In [21], an SVM-specific active learning method is proposed that interactively selects the most ambiguous set of samples with respect to the learned ranking function and a initial set of "real-world queries". At each round, those instances that minimize the support vector margin for the function learned so far are selected and the user is required to input a partial order for these instances. Compared to "traditional" active learning, this means a much more laborious labeling process for the user, who has to partially order results for every query in the initial set. This procedure is repeated an empirically established number of times with the function learned from all the user ordered instances, always selecting the most ambiguous instances for the current learned function. Although the algorithm could be modified to be used for document retrieval, in the paper it is tested using a data retrieval application which enables fuzzy search on relational databases.

Another SVM-specific strategy is presented in [2]. The authors rely on the relationship between the area under de ROC curve (AUC) and the hinge rank loss proposed by [18] to develop a loss minimization framework for active learning in ranking. Instead of testing each and every unlabeled sample to determine the one that has the smallest expected future error, the authors suggest selecting the examples that have the largest contribution to the estimated current error. These are potentially the ones that will bring more benefit when labeled for the functions that will be trained in the next rounds of the method. The proposed selection criterion is based on the hinge rank loss calculated on a per query basis and depends on the determination of a rank threshold that estimates the rank position that separates the lowest ranked relevant element from the highest ranked non-relevant example. The algorithm starts with a small labeled per query set and proceeds selecting unlabeled samples that have the highest uncertainty (as defined by the rank threshold). These samples are then labeled and added to the per query labeled sets and the process is repeated as many times as desired. The method is experimentally tested using the TD2003 and TD2004 datasets from LETOR and compared against four sampling baselines. The experimental setup starts with 11 labeled samples per query and selects 5 new samples per query per round. On the 20^{th} iteration, the selected set corresponds to approximately 11% of the original training sets in both collections.

A slightly different approach is proposed in [3]. Their method relies on the estimated risk of the ranking function on the labeled set after adding a new instance with all possible labels. The authors present results using this sampling technique with RankSVM and RankBoost. Their method, which also relies on an initial labeled training set and incrementally adds new samples, achieves competitive results with around 15% of the original training sets.

In contrast, our method does not use any initial labeled set and selects all samples from the unlabeled set. Furthermore, there is no need to empirically determine (using the validation or test sets) how many iterations are required to obtain competitive results, since the selection process naturally converges. Finally, our method achieves very good results with as little as 1.12% of the original training sets.

3 Selective Sampling Using Association Rules

In our context, the task of learning to rank is defined as follows. We have as input the *training data* (referred to as $\mathcal{D}$), which consists of a set of records of the form $< q, d, r >$, where q is a query, d is a document (represented as a list of m feature-values or $\{f_1, f_2, \ldots, f_m\}$), and r is the *relevance* of d to q. Features include BM25, Page Rank, and many other document and query-document properties. The relevance of a document draws its values from a discrete set of possibilities $\{r_0, r_1, \ldots, r_k\}$ (e.g. 0: not relevant, 1: somewhat relevant, 2: very relevant). The training data is used to build functions relating features of the documents to their corresponding relevance. The *test set* (referred to as $\mathcal{T}$) consists of records $< q, d, ? >$ for which only the query q and the document d are known, while the relevance of d to q is unknown. Ranking functions obtained from $\mathcal{D}$ are used to estimate the relevance of such documents to the corresponding queries.

3.1 Learning to Rank Using Association Rules

Ranking functions exploit the relationship between document features and relevance levels. This relationship may be represented by association rules. We denote as $\mathcal{R}$ a rule-set composed of rules of the form $\{f_j \wedge \ldots \wedge f_l \xrightarrow{\theta} r_i\}$. These rules can contain any mixture of the available features in the antecedent and a relevance level in the consequent. The strength of the association between antecedent and consequent is measured by a statistic, θ, which is known as *confidence* [1] and is simply the conditional probability of the consequent given the antecedent. In this section we discuss the use of association rules for the sake of learning to rank, and we present the algorithm LRAR (Learning to Rank using Association Rules) which extracts rules from $\mathcal{D}$ on a demand-driven basis and then combine these rules in order to estimate the relevance of each document in $\mathcal{T}$.

Demand-Driven Rule Extraction. The search space for rules is huge, and thus, computational cost restrictions must be imposed during rule extraction. Typically, a minimum support threshold (σ_{min}) is employed in order to select frequent rules (i.e., rules occurring at least σ_{min} times in $\mathcal{D}$) from which the ranking function is produced. This strategy, although simple, has some problems. If σ_{min} is set too low, a large number of rules will be extracted from $\mathcal{D}$, and often most of these rules are useless for estimating the relevance of documents in $\mathcal{T}$ (a rule $\{\mathcal{X} \rightarrow r_i\}$ is only useful to estimate the relevance of document $d \in \mathcal{T}$ if the set of features $\mathcal{X} \subseteq d$, otherwise the rule is meaningless to d). On the other hand, if σ_{min} is set too high, some important rules will not be included in $\mathcal{R}$, causing problems if some documents in $\mathcal{T}$ contain rare features (i.e., features occurring less than σ_{min} times in $\mathcal{D}$). Usually, there is no optimal value for σ_{min}, that is, there is no single value that ensures that only useful rules are included in $\mathcal{R}$, while at the same time important rules are not missed. The method to be proposed next deals with this problem by extracting rules on a demand-driven basis.

Demand-driven rule extraction is delayed until a set of documents is retrieved for a given query in $\mathcal{T}$. Then, each individual document d in $\mathcal{T}$ is used as a filter to remove irrelevant features and examples from $\mathcal{D}$. This process produces a projected training

data, $\mathcal{D}_d$, which is obtained after removing all feature-values not present in d. Then, a specific rule-set, $\mathcal{R}_d$ extracted from $\mathcal{D}_d$, is produced for each document d in $\mathcal{T}$.

Lemma 1: All rules extracted from $\mathcal{D}_d$ (i.e., $\mathcal{R}_d$) are useful to estimate r.

Proof: Since all examples in $\mathcal{D}_d$ contain only feature-values that are present in d, the existence of a rule $\{\mathcal{X} \rightarrow r_i\} \in \mathcal{R}_d$, such that $\mathcal{X} \nsubseteq d$, is impossible. ■

Theorem 1: The number of rules extracted from $\mathcal{D}$ depends solely on the number of features in $\mathcal{D}_d$, no matter the value of σ_{min}.

Proof: If an arbitrary document $d \in \mathcal{T}$ contains at most l features then any rule matching d can have at most l feature-values in its antecedent. That is, for any rule $\{\mathcal{X} \rightarrow r_i\}$, such that $\mathcal{X} \subseteq d$, $|\mathcal{X}| \leq l$. Consequently, for $\sigma_{min} \approx 0$, the number of possible rules matching d is $k \times (l + \binom{l}{2} + \ldots + \binom{l}{l}) = O(2^l)$, where k is the number of distinct relevances. Thus, the number of rules extracted for all documents in $\mathcal{T}$ is $O(|\mathcal{T}| \times 2^l)$. ■

Relevance Estimation. In order to estimate the relevance of a document d, it is necessary to combine all rules in $\mathcal{R}_d$. Our strategy is to interpret $\mathcal{R}_d$ as a poll, in which each rule $\{\mathcal{X} \xrightarrow{\theta} r_i\} \in \mathcal{R}_d$ is a vote given by a set of features $\mathcal{X}$ for relevance level r_i. Votes have different weights, depending on the strength of the association they represent (i.e., θ). The weighted votes for relevance level r_i are summed and then averaged (by the total number of rules in $\mathcal{R}_d$ that predict relevance level r_i), forming the score associated with relevance r_i for document d, as shown in Equation 1 (where $\theta(\mathcal{X} \rightarrow r_i)$ is the value θ assumes for rule $\{\mathcal{X} \rightarrow r_i\}$):

$$s(d, r_i) = \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{\mid \mathcal{R}_d \mid}, \text{ where } \mathcal{X} \subseteq d \tag{1}$$

Therefore, for a document d, the score associated with relevance r_i is given by the average θ values of the rules in $\mathcal{R}_d$ predicting r_i. The likelihood of d having a relevance level r_i is obtained by normalizing the scores, as expressed by $\hat{p}(r_i|d)$, shown in Equation 2:

$$\hat{p}(r_i|d) = \frac{s(d, r_i)}{\sum_{j=0}^{k} s(d, r_j)} \tag{2}$$

Finally, the relevance of document d is estimated by a linear combination of the likelihoods associated with each relevance level, as expressed by the ranking function $rank(d)$, which is shown in Equation 3:

$$rank(d) = \sum_{i=0}^{k} \left(r_i \times \hat{p}(r_i|d)\right) \tag{3}$$

The value of $rank(d)$ is an estimate of the true relevance of document d (i.e., r) using $\hat{p}(r_i|d)$. This estimate ranges from r_0 to r_k, where r_0 is the lowest relevance and r_k is the highest one. Relevance estimates are used to produce ranked lists of documents. All steps of LRAR are depicted in Algorithm 1.

Algorithm 1. Learning to Rank using Association Rules

Require: The training data $\mathcal{D}$, test set $\mathcal{T}$, and σ_{min} (≈ 0)
Ensure: $rank(d)$

1: **for all pair** $(d, q) \in \mathcal{T}$ **do**
2: $\mathcal{D}_d \Leftarrow \mathcal{D}$ projected according to d
3: $\mathcal{R}_d \Leftarrow$ rules extracted from $\mathcal{D}_d \mid \sigma \geq \sigma_{min}$
4: **for all** $i \mid 0 \leq i \leq k$ **do**
5: $s(d, r_i) \Leftarrow \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{|\mathcal{R}_d|}$
6: **end for**
7: $rank(d) \Leftarrow 0$
8: **for all** $i \mid 0 \leq i \leq k$ **do**
9: $rank(d) \Leftarrow rank(d) + r_i \times \hat{p}(r_i|d)$
10: **end for**
11: **end for**

3.2 Rule-Based Active Sampling

In this section we present a novel algorithm referred to as SSAR (Selective Sampling using Association Rules), which relies on an effective selective sampling strategy in order to deal with the high cost of labeling large amounts of examples. The key idea of SSAR is that it may provide results as effective (or even better) as LRAR by carefully choosing a much smaller set of training examples from which it learns the ranking functions. As we will see in detail below, SSAR takes each unlabeled document in turn, obtaining its projection from the current reduced training set and generating the rules that would be used to rank the document. After it has the rules for all unlabeled documents, it chooses the one that generated the fewest rules, obtaining its label and inserting it into the current selected and labeled training set. The goal is to label as few instances as possible, while providing equal or even improved ranking performance.

Sampling Function. Consider a large set of unlabeled documents $\mathcal{U}$=$\{u_1, u_2, \ldots, u_n\}$. The problem we investigate in this section is how to select a small subset of documents in $\mathcal{U}$, such that the selected documents carry almost the same information of all documents in $\mathcal{U}$. These highly informative documents will compose the training data $\mathcal{D}$, and, ideally, $|\mathcal{D}| \ll |\mathcal{U}|$. Particularly, SSAR exploits the redundancy in feature-space that exists between different documents in $\mathcal{U}$. That is, many documents in $\mathcal{U}$ may share some of their feature-values, and SSAR uses this fact to perform an effective selective sampling strategy.

Intuitively, if a document $u_i \in \mathcal{U}$ is inserted into $\mathcal{D}$, then the number of useful rules for documents in $\mathcal{U}$ that share feature-values with u_i will possibly increase. In contrast, the number of useful rules for those documents in $\mathcal{U}$ that do not share any feature-value with u_i will clearly remain unchanged. Therefore, the number of rules extracted for each document in $\mathcal{U}$ can be used as an approximation of the amount of redundant information between documents already in $\mathcal{D}$ and documents in $\mathcal{U}$. The sampling function employed by SSAR exploits this key idea, by selecting documents that contribute primarily with non-redundant information, and these informative documents are those

likely to demand the fewer number of rules from $\mathcal{D}$. More specifically, the sampling function $\gamma(\mathcal{U})$ returns a document in $\mathcal{U}$ according to Equation 4:

$$\gamma(\mathcal{U}) = \{u_i \text{ such that } \forall u_j : |\mathcal{R}_{u_i}| \leq |\mathcal{R}_{u_j}|\} \quad (4)$$

The document returned by the sampling function is inserted into $\mathcal{D}$, but it also remains in $\mathcal{U}$. In the next round of SSAR, the sampling function is executed again, but the number of rules extracted from $\mathcal{D}$ for each document in $\mathcal{U}$ is likely to change due to the document recently inserted into $\mathcal{D}$. The intuition behind choosing the document which demands the fewest rules is that such document should share less feature-values with documents that were already inserted into $\mathcal{D}$. That is, if only few rules are extracted for a document u_i, then this is evidence that $\mathcal{D}$ does not contain documents that are similar to u_i, and, thus, the information provided by document u_i is not redundant and u_i is a highly informative document. This simple heuristic works in a fine-grained level of feature-values trying to maximize the diversity in the training set. The extracted rules capture the co-occurrence of feature-values, helping in our goal of increasing diversity, since in this case, the document which demands the fewest rules is exactly the one which shares the least possible number of feature-values with documents already in the training data. In the case of a tie, the algorithm selects the document based on the size of the projection.

Notice that initially $\mathcal{D}$ is empty, and thus SSAR cannot extract any rules from $\mathcal{D}$. The first document to be labeled and inserted into $\mathcal{D}$ is selected from the set of available documents $\mathcal{U}$. In order to maximize the initial coverage of $\mathcal{D}$, the selected document is the one that maximizes the size of the projected data in $\mathcal{U}$, that is, it is the document d for which $\mathcal{U}_d$ is the largest. This is the document that shares more feature-values with the other documents of the collection and can be considered as the best representative of it. After the first document is selected and labeled, the algorithm proceeds using the fewest rules heuristic, as described above.

Natural Stop Condition. After selecting the first document and at each posterior round, SSAR executes the sampling function and a new example is inserted into $\mathcal{D}$. At iteration i, the selected document is denoted as $\gamma_i(\mathcal{U})$, and it is likely to be as dissimilar as possible from the documents already in $\mathcal{D}$=$\{\gamma_{i-1}(\mathcal{U}), \gamma_{i-2}(\mathcal{U}), \ldots, \gamma_1(\mathcal{U})\}$. The algorithm keeps inserting documents into the training data, until the stop criterion is achieved.

Lemma 2: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then $\gamma_i(\mathcal{U})$=$\gamma_j(\mathcal{U})$ $\forall j > i$.

Proof: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then the inclusion of $\gamma_i(\mathcal{U})$ does not change $\mathcal{D}$. As a result, any further execution of the sampling function must return the same document returned by $\gamma_i(\mathcal{U})$, and $\mathcal{D}$ will never change. ■

The algorithm stops when all available documents in $\mathcal{U}$ are less informative than any document already inserted into $\mathcal{D}$. This occurs exactly when SSAR selects a document which is already in $\mathcal{D}$. According to Lemma 2, when this condition is reached, SSAR will keep selecting the same document over and over again. At this point, the training data $\mathcal{D}$ contains the most informative documents, and LRAR can be applied to estimate the relevance of documents in $\mathcal{T}$. All steps of SSAR are shown in Algorithm 2.

Algorithm 2. Selective Sampling using Association Rules

Require: Unlabeled data $\mathcal{U}$, and σ_{min} (≈ 0)
Ensure: The training data $\mathcal{D}$

continue
  for all document $u_i \in \mathcal{U}$ **do**
    $\mathcal{D}_{u_i} \Leftarrow \mathcal{D}$ projected according to u_i
    $\mathcal{R}_{u_i} \Leftarrow$ rules extracted from $\mathcal{D}_{u_i} \mid \sigma \geq \sigma_{min}$
  end for
  if $\mathcal{D} = \emptyset$ **then**
    $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{U}_{u_i}| \geq |\mathcal{U}_{u_j}|\}$
  else
    $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{R}_{u_i}| \leq |\mathcal{R}_{u_j}|\}$
  end if
  if $\gamma_i(\mathcal{U}) \in \mathcal{D}$ **then break**
  else append $\gamma_i(\mathcal{U})$ **to** $\mathcal{D}$

4 Experimental Evaluation

4.1 LETOR Datasets

To evaluate the effectiveness of our method, we did extensive experimentation using the Learning to Rank (LETOR) benchmark datasets version 3.0. LETOR 3.0 is composed of 6 separate web datasets plus the OHSUMED corpus. The web datasets contain labeled instances selected from web pages obtained from a 2002 crawl of the .gov TLD. These collections are separated in three search tasks: topic distillation (TD), homepage finding (HP) and named page finding (NP) and contain 2 sets each (namely, TREC2003 and TREC2004). The collections contain instances represented by 64 features for the top 1.000 documents returned for a specific set of queries using the BM25 model [12]. These datasets use a binary relevance judgment indicating whether a document is or is not relevant to a given query. We evaluate our method on all the largest, more diverse, LETOR 3.0 web datasets. In all datasets we have used the query-based normalized versions as suggested by the producers of the LETOR benchmarking datasets. We also use 5-fold cross validation for all results reported as well as the evaluation script provided in the LETOR package to generate the final precision, MAP and NDCG metrics.

4.2 Results

As described in Section 3, SSAR, our rule-based active sampling algorithm processes a given list of unlabeled instances selecting the one that produces the fewest rules. In this setup, the training set for the selection process is initially empty and grows one instance at a time as they are selected from the unlabeled set and labeled. The algorithm eventually converges when it selects an instance it has already selected before. Once that happens, the selected items can be used as a reduced training set to rank the test set using LRAR (i.e. the supervised rule-based rank-learner). All results presented here use, therefore, 2 distinct sets. The original training set is used as the unlabeled set from which instances are selected by SSAR. The test set is then ranked by the LRAR using

Table 1. SSAR MAP Results and Statistics

	SSAR	LRAR	LBM25	Random	G%	Sel	Utot	Sel%	Rsel%	R%	R25%
TD2003	0.2032	0.2459	0.1402	0.1181±0.0234	44.94	157	29,435	0.53	6.80	0.82	13.22
TD2004	0.1792	0.2463	0.1452	0.1267±0.0163	23.47	141	44, 488	0.32	7.82	1.50	11.32
NP2003	0.7202	0.6373	0.5346	0.4981±0.0688	34.72	207	89, 194	0.23	3.04	0.10	25.32
NP2004	0.4993	0.5155	0.2672	0.3695±0.0575	35.15	181	44, 300	0.41	3.18	0.10	5.76
HP2003	0.6487	0.7083	0.5230	0.5486±0.0493	18.25	218	88, 564	0.25	3.62	0.12	26.04
HP2004	0.6332	0.5443	0.3712	0.3117±0.0496	70.55	222	44, 645	0.50	1.68	0.11	4.24

Table 2. SSAR with Partitions (SSARP) MAP Results and Statistics

	SSARP	LRAR	LBM25	Random	G%	Sel	Utot	Sel%	Rsel%	R%	R25%
TD2003	0.2689	0.2459	0.2104	0.1573±0.0198	27.84	642	29,435	2.18	4.40	0.82	7.70
TD2004	0.2006	0.2463	0.1818	0.1707±0.0112	10.37	633	44,488	1.42	5.72	1.50	6.00
NP2003	0.6960	0.6373	0.6321	0.5573±0.0423	10.09	995	89,194	1.12	2.42	0.10	7.28
NP2004	0.5499	0.5155	0.4064	0.4038±0.0580	35.29	860	44,300	1.94	1.80	0.10	2.16
HP2003	0.7411	0.7083	0.6487	0.5825±0.0460	14.25	1091	88, 564	1.23	2.00	0.12	6.90
HP2004	0.6168	0.5443	0.3685	0.3718±0.0479	65.89	855	44,645	1.91	1.68	0.11	1.74

the selected and labeled instances as training. Observe that our method does not use an initial labeled set to learn an initial model. The labeling effort is restricted to the examples selected by the algorithm directly from the unlabeled set[2].

Experimental Setup and First Results. The association rule mining algorithm uses nominal values to generate the inference rules used in ranking the results or in selecting samples. Therefore it is necessary to discretize the original LETOR data. Since all our data is unlabeled, we need to use an unsupervised discretization algorithm. Simple algorithms such as Uniform Range Discretization (URD) or Uniform Frequency Discretization (UFD) could be used, but being oversimplistic, they may cause some loss of information. Instead, we use the Tree-based Unsupervised Bin Estimator (TUBE) proposed in [14]. TUBE is a greedy non-parametric density estimation algorithm that uses the log-likelihood measure and a top-down tree building strategy to define varying-length bins for each attribute in the dataset. The algorithm can automatically determine the number of bins using cross-validation and the log-likelihood. Since this approach can lead to too many bins in some cases, we chose to use 10 varying-length bins for all attributes in all datasets (which is the default for the URD algorithm implemented in Weka). The results shown in tables 1 and 2 were obtained using TUBE to discretize each feature from the training set into 10 bins. After the bins were determined using the training sets in each fold, the test sets were discretized using the same bins.

Table 1 presents the results for this initial setup. The first column, "SSAR", shows the MAP obtained using only the samples selected from the unlabeled set. The number of selected samples appears in the "Sel" column. The "UTot" column shows the number

[2] In our experiments, the presence of the user who would provide the labels is simulated; we use instead the original labels available in the collection after a document is selected.

of instances in the unlabeled set (from which the samples were selected) and the "Sel%" column indicates the percentage of the unlabeled set that the selected examples represent. The "RSel %" column shows the proportion of relevant samples in the selected set while the "R%" field shows the proportion of relevant documents in the full training set. "R25%" shows the proportion of relevant samples selected by the BM25 baseline described below. As a reference result, the "LRAR" column contains the MAP obtained by the LRAR algorithm using the complete training set and supervised discretization. We show this result for comparison, since it uses a very effective supervised discretization method [5] and provides a target to measure our hypothesis of noise reduction. The baselines appear on the 3^{rd} and 4^{th} columns: "LBM25" shows the resulting MAP from running LRAR with the same amount of samples selected by SSAR as training but selected using the value of the BM25 attribute in descending order (i.e. instances with the highest BM25 were used); "Random" shows the MAP obtained by randomly selecting the same amount of samples selected by SSAR and using these as the training set for supervised LRAR ranking. Finally, the "G%" column indicates the gain obtained by SSAR over the best value from the LRAR BM25 and Random baselines. The Random baseline was produced by randomly sampling the same amount of instances selected by SSAR at least 20 times for each fold and averaging the resulting MAPs. We present the mean obtained from all runs and all folds as well as the confidence interval for a confidence level of 95%. The LRAR with BM25 baseline tries to select more interesting instances by using the BM25 attribute value as a measure of instance quality.

From Table 1 we can see that the method converges very fast, selecting from 0.23 to 0.53% of the instances in the unlabeled set. Compared to the baselines, it performs very well as can be seen on the "G%" column. SSAR even beats the LRAR reference result in NP2003 and HP2004. Notice also that SSAR tends to select a much higher proportion of relevant instances than present in the original sets (RSel% vs. R%). These datasets have an extremely reduced amount of relevant instances and SSAR seems to single them out based on the "fewer rules" heuristic. On the other hand, the BM25-based selection obtains an even higher proportion of relevant documents, showing the power of this classic IR measure. But from the LBM25 results we can see that it is not only a matter of using many relevant instances, but also about the "quality" of the instances used. Our rule-based selection method does choose many relevant instances, but it does so based on the rules created from all attributes. These initial results are promising, but the algorithm is converging too fast in some collections for the selected samples to have sufficient representativeness and diversity. Next we propose a simple and yet effective strategy to delay the convergence.

Increasing Sample Diversity. The approach described above has two potential issues: first, it may take some time to run on datasets with too many features, since the active sampling algorithm's complexity depends on the number of features and the size of the unlabeled set. Second, as we can see in Table 1, the number of instances selected using this method is very small, ranging from 0.23 to 0.53% of the unlabeled set size. There's no doubt that the selected samples are very informative, since the results obtained by SSAR are usually reasonably better than the baselines. Nonetheless, the resulting training dataset may still be too small to provide, in some collections, the minimum diversity of examples for the supervised algorithm to reach a performance comparable to that

obtained using the complete training set (i.e. column "LRAR" in Table 1). By delaying convergence and increasing the number of sampled instances we can improve the diversity of the selected set.

We propose partitioning the unlabeled set into vertical sections containing subsets of the features[3]. Each of these partitions contain all unlabeled instances, but only a group of each instance's features. The active sampling algorithm can then be run on each partition, selecting instances based on distinct feature sets. This simple strategy increases the number of selected instances, since for each partition the algorithm will choose those that are most informative given the features in that partition. It also improves the diversity of the samples, as they are selected based on distinct characteristics. To apply this strategy, we need to determine how to separate the features into the partitions and how many partitions should be used. The features need to be divided in such a way as to allow the algorithm to select informative samples regardless of which feature set is used. However, features have diverse informational values, with some being more informative to the rank learning algorithm than others. Ideally, we want to distribute the most informative features through all partitions in order to maximize the quality and variety of the instances selected by SSAR from each partition. There are many ways to estimate which features provide more information. We chose to estimate the χ^2 value of each feature. Since we do not have relevance information, we cannot calculate the χ^2 in relation to the label. What we do instead is to calculate the χ^2 of each feature in relation to the others (i.e. how well one feature performs in predicting another feature's value). Thus we produce a $n \times n$ matrix containing in line i the $n-1$ features ranked in descending order of their χ^2 value in relation to the i_{th} feature. We then combine the n rankings by scoring each feature according to its positions in all rankings. If a feature appears at the first position in a ranking we add 1 to its score, otherwise, we add $1/log(10 \times j)$ where j is the feature's position in that ranking. Then we order the features in descending order of score, obtaining the final ranking.

With this ranked list of features, we can now vertically partition the unlabeled set by spreading the features into the partitions in the ranked order. Thus, given the ranked list of m features $\mathcal{F} = \{f_1, f_2, ..., f_m\}$, and a set of n partitions $\mathcal{P} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n\}$, we place feature f_1 (the one in the first position of the χ^2 ranking) in partition $\mathcal{U}_1$ then f_2 in $\mathcal{U}_2$, eventually reaching partition $\mathcal{U}_n$ and starting over by placing the next feature, f_i into $\mathcal{U}_1$ and f_{i+1} into $\mathcal{U}_2$ and so on. We now run SSAR using each $\mathcal{U}_k$ set as input, and obtaining n sets of selected documents $\mathcal{D}_k$. We then obtain the original set of features for all selected documents from $\mathcal{U}$ and run LRAR using this new set as training.

The next step is to determine how many partitions to use. If very few partitions are used, then the increase in document diversity and the number of selected instances may be small, yielding only marginal gains. If we use too many partitions, then the informational value of each document is diluted and SSARP (SSAR with Partitions) will select many instances that are not informative as a whole, but only when considered in the context of the reduced feature set (i.e. the partition from which it was selected). Thus it is important to determine how many partitions to use to obtain good diversity while selecting informative samples. We performed preliminary tests using 2 collections, and

[3] Another strategy would be to select documents *per query*, but this would entail a large number of actively selected samples, thus hurting our goal of significantly reducing the labelling effort.

Table 3. MAP for SVM using selected samples, BM25 and Random Baselines

	SSARP	**SVMS**	**SBM25**	**Random**	**G%**
TD2003	0.2689	0.2194	0.1568	0.1417±0.0285	39.95
TD2004	0.2006	0.1957	0.1335	0.1687±0.0145	16.01
NP2003	0.6960	0.6428	0.6587	0.5739±0.0237	-2.41
NP2004	0.5499	0.5929	0.5811	0.5787±0.0329	2.04
HP2003	0.7411	0.6747	0.7090	0.5798±0.0592	-4.84
HP2004	0.6168	0.6734	0.6731	0.5406±0.0357	0.05

Table 4. MAP for SSARP and LETOR Baselines

	SSARP	**RBoost**	**FRank**	**REG**
TD2003	***0.2689***	*0.2274*	*0.2031*	*0.2409*
TD2004	0.2006	0.2614	0.2388	0.2078
NP2003	**0.6960**	0.7074	*0.6640*	*0.5644*
NP2004	*0.5499*	0.5640	0.6008	*0.5142*
HP2003	***0.7411***	*0.7330*	*0.7095*	*0.4968*
HP2004	*0.6168*	0.6251	0.6817	*0.5256*
Avg.	*0.5122*	0.5197	0.5163	*0.4250*

decided to set the number of features per partition to be from around 8 to 12. Using 5 partitions for all datasets, we have around 12 features per set for LETOR 3.0[4].

Table 2 presents the results for SSARP. All results were produced by splitting the unlabeled sets in 5 partitions, selecting instances from the partitions, creating a new labeled training set comprised of all the selected samples with all features and running LRAR on the test set with 5-fold cross validation. Again we compare the resulting MAPs with two baselines: LRAR BM25 and Random. The baselines were ran again, since the number of documents selected has changed for all datasets. Notice that the results for "LRAR" are the same as in Table 1, since it uses the complete training set.

As we can see, SSARP selects from 4 to 5 times the amount of instances selected by SSAR although the percentages relative to the unlabeled set (column "Sel%") remain very low. These percentages now vary from 1.12 (NP2003) to 2.18% (TD2003). Not only the size of the selected set increased, but there is more diversity in the selected set as instances were chosen based on different feature groups. As a consequence, the MAP for SSARP is better for 4 datasets. The improvement over SSAR was 32% for TD2003, 14% for HP2003, 11% for TD2004 and 10% for NP2004. For NP2003 and HP2004, there was a small reduction of around 3%. The proportion of relevant instances selected is also lower for both SSARP and the BM25 baseline. Our method is not only still better than the baselines but also beats LRAR using the full training sets on all datasets except TD2004. This is an indication that there is some noise reduction in the active selection process. With only 2% of the original training set it is possible to obtain better results than using the full training set with a supervised method such as LRAR.

5 Discussion

5.1 Does the Proposed Sampling Technique Work with Other L2R Methods?

Once the instances are actively selected by SSARP, it is possible to use other supervised learning algorithms to rank the test sets. Of course, different algorithms will find the instances selected by SSARP more or less "informative". Active learning methods currently proposed in the literature are inherently algorithm-specific as illustrated by the

[4] For datasets with more features, more partitions should be used.

SVM-specific techniques discussed in section 2. For instance, the approach proposed in [21] selects, at each round, the samples that minimize the support vector margin. What if we run an SVM ranking algorithm using the selected instances as training[5]? Table 3 shows the MAP resulting from running SVMRank[6] [6] using the examples selected by SSARP as training sets. The column "SSARP" repeats the results from Table 2 for reference. "SVMS" shows the MAP obtained from running SVMRank using the samples selected by SSARP (see Table 2 for other information such as the number of instances selected, etc.). Again, we ran two baselines for comparison: "SBM25" contains the results from running SVMRank with the same amount of samples as selected by SSARP, but picked by their BM25 value. "Random" shows the average of 20 runs where the instances are randomly selected and includes the confidence interval for a 95% confidence level. "G%" indicates the gain of SVMS over the best baseline.

From the results we can observe that SVMS fares very well on the TD2003 and TD2004 datasets as compared to the baselines. For the other datasets, it basically ties with the SBM25 baseline. This may be due to the fact that the BM25 method selects a high proportion of relevant instances which are extremely rare in the original NP and HP datasets (column "R%" of Tables 1 and 2). This is one of the difficulties in ranking these datasets and both SSARP and the BM25 selection methods may help as they tend to select a larger proportion of relevant samples. Though these results can be improved, they illustrate that our method chooses informative instances that are useful even for completely different algorithms.

5.2 How Does the Proposed Method Fare against Supervised LETOR Baselines?

To put SSARP results in perspective, Table 4 shows the MAP results for SSARP and those for three supervised algorithms from the LETOR 3.0 published baselines (that use the full training sets). The producers of the LETOR datasets have published results containing precision, MAP and NDCG obtained using 12 L2R algorithms on the LETOR 3.0 datasets [12]. We chose to show the results for three algorithms: RankBoost ("RBoost" in Table 4), FRank and Regression ("REG"). The first two were selected mainly because, similarly to our method, they use non-linear ranking functions, so they all lie in the same class of algorithms, making this a more fair comparison. RankBoost achieves very good results, beating all other 11 algorithms in 2 of the 6 datasets (TD2004, NP2003). FRank has average results if compared to the other algorithms and Regression obtains a lower average then the other algorithms, although it still beats some of the them in some datasets. We use the results of these algorithms as high, medium and low watermarks to show that SSARP obtains competitive results selecting only a very small fraction of each dataset. In Table 4, SSARP results that appear in *italic* are equal or better than one of the 3 baselines. Numbers in **bold** indicate that the result is equal or better that 2 of the baselines. Finally, the results in ***bold and italic*** are equal or better then all three baselines. We indicate which results from the baselines were matched or surpassed by showing them in *italic*. From Table 4 we can see that SSARP beats the chosen supervised baselines in TD2003 and HP2003. It also obtains

[5] One reason to use RankSVM is that it is one of the best performing methods on LETOR 3.0.
[6] Best parameters were determined using cross-validation on the training sets.

Table 5. NDCG for SSARP and LETOR Baselines

	SSARP			**RankBoost**			**FRank**			**Regression**		
	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10
TD2003	***0.3800***	***0.3255***	***0.3302***	*0.2800*	*0.3149*	*0.3122*	*0.3000*	*0.2468*	*0.2690*	*0.3200*	*0.2984*	*0.3263*
TD2004	*0.3600*	*0.3404*	*0.3061*	0.5067	0.3878	0.3504	0.4933	0.3629	0.3331	*0.3600*	*0.3257*	*0.3031*
NP2003	**0.5867**	***0.7850***	**0.7918**	0.6000	*0.7818*	0.8068	*0.5400*	*0.7595*	*0.7763*	*0.4467*	*0.6423*	*0.6659*
NP2004	*0.4133*	**0.6546**	*0.6805*	0.4267	*0.6512*	0.6914	0.4800	0.6870	0.7296	*0.3733*	*0.6135*	*0.6536*
HP2003	***0.7200***	**0.7809**	**0.7982**	*0.6667*	0.8034	0.8171	*0.6533*	*0.7780*	*0.7970*	*0.4200*	*0.5463*	*0.5943*
HP2004	**0.5200**	*0.6981*	*0.7111*	*0.5067*	0.7211	0.7428	0.6000	0.7486	0.7615	*0.3867*	*0.6130*	*0.6468*

very good results for NP2003, beating 2 baselines and almost reaching the performance of the 3^{rd} (RankBoost). Overall, SSARP's average performance is only 1.4% below the best algorithm (RankBoost) but using on average 98% less training. For further performance comparisons, Table 5 provides the NDCG@1, @5 and @10 using the same baseline algorithms and markup scheme.

5.3 How Does SSARP Compare to Other Active Learning Methods for L2R?

Some of the most recent active learning strategies for L2R [2,3] were run on the TD2003 and TD2004 datasets, thus allowing some comparison with our strategy. Both works use a similar overall sampling strategy: an initial per query labeled seed set is used and the algorithms iteratively select new samples to be labeled. The initial seed sets contain 11 ([2]) or 16 ([3]) samples per query with one of the samples being necessarily relevant (amounting to 1.1 and 1.6% of the training sets for both datasets, respectively). The algorithms can be run for as many rounds as necessary, always selecting 5 new samples *per query* at each round. By the 20^{th} round for the first article, 11% of the training sets are selected and labeled. For the second paper, by the 25^{th} round, 15% of the training sets are selected. In both cases, there is no way to know *a priori* how many iterations are necessary for convergence. In contrast, our method does not require initial labeled sets nor do we have to empirically determine how many rounds are needed to achieve good results. For TD2003, SSARP selects only 2.18% of the original training set and achieves a higher MAP then those reported in both papers. TD2004 was the hardest collection for our method and we did not beat the results reported in [2], but SSARP uses only 1.42% of the original training set to obtain a reasonable performance, while in that work around 11% of the training data had to be actively labeled.

6 Conclusions

We have proposed a rule-based active sampling method that is practical and effective, selecting very few documents to be labeled and yet offering competitive results when compared to established supervised algorithms using the complete training sets. Since the method does not depend on an initial labeled set, it can be easily used in real-life applications where labeling many documents can be prohibitively expensive or unpractical. For future work we intend to investigate the use of other discretization techniques.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD 1993, pp. 207–216 (1993)
2. Donmez, P., Carbonell, J.G.: Active sampling for rank learning via optimizing the area under the ROC curve. In: SIGIR 2009, pp. 78–89 (2009)
3. Donmez, P., Carbonell, J.G.: Optimizing estimated loss reduction for active sampling in rank learning. In: ICML 2008, pp. 248–255 (2008)
4. Donmez, P., Carbonell, J.G., Bennett, P.N.: Dual strategy active learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 116–127. Springer, Heidelberg (2007)
5. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI 1993, pp. 1022–1029 (1993)
6. Joachims, T.: Optimizing search engines using clickthrough data. In: SIGKDD 2002, pp. 133–142 (2002)
7. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: SIGIR 1994, pp. 3–12 (1994)
8. Liu, T.: Learning to rank for information retrieval. Found. Trends Inf. Retr. 3(3), 225–331 (2009)
9. Long, B., Chapelle, O., Zhang, Y., Chang, Y., Zheng, Z., Tseng, B.: Active learning for ranking through expected loss optimization. In: SIGIR 2010, pp. 267–274 (2010)
10. Mccallum, A.K.: Employing EM in pool-based active learning for text classification. In: ICML 1998, pp. 350–358 (1998)
11. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: ICML 2004, p. 79 (2004)
12. Qin, T., Liu, T., Xu, J., Li, H.: LETOR: a benchmark collection for research on learning to rank for information retrieval. Inf. Retr. 13, 346–374 (2010)
13. Robertson, S.E., Walker, S., Hancock-Beaulie, M.M.: Large test collection experiments on an operational, interactive system: Okapi at TREC. IP&M 31, 345–360 (1995)
14. Schmidberger, G., Frank, E.: Unsupervised discretization using tree-based density estimation. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 240–251. Springer, Heidelberg (2005)
15. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison
16. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1289–1296. MIT Press, Cambridge (2008)
17. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: COLT 1992, pp. 287–294 (1992)
18. Steck, H.: Hinge rank loss and the area under the ROC curve. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 347–358. Springer, Heidelberg (2007)
19. Veloso, A.A., Almeida, H.M., Gonçalves, M.A., Meira Jr., W.: Learning to rank at query-time using association rules. In: SIGIR 2008, pp. 267–274 (2008)
20. Wang, L., Lin, J., Metzler, D.: Learning to efficiently rank. In: SIGIR 2010, pp. 138–145 (2010)
21. Yu, H.: SVM selective sampling for ranking with application to data retrieval. In: SIGKDD 2005, pp. 354–363 (2005)