Active Learning for Web Search Ranking via Noise Injection

WENBIN CAI, MUHAN ZHANG, and YA ZHANG, Shanghai Jiao Tong University

Learning to rank has become increasingly important for many information retrieval applications. To reduce the labeling cost at training data preparation, many active sampling algorithms have been proposed. In this article, we propose a novel active learning-for-ranking strategy called ranking-based sensitivity sampling (RSS), which is tailored for Gradient Boosting Decision Tree (GBDT), a machine-learned ranking method widely used in practice by major commercial search engines for ranking. We leverage the property of GBDT that samples close to the decision boundary tend to be sensitive to perturbations and design the active learning strategy accordingly. We further theoretically analyze the proposed strategy by exploring the connection between the sensitivity used for sample selection and model regularization to provide a potentially theoretical guarantee w.r.t. the generalization capability. Considering that the performance metrics of ranking overweight the top-ranked items, item rank is incorporated into the selection function. In addition, we generalize the proposed technique to several other base learners to show its potential applicability in a wide variety of applications. Substantial experimental results on both the benchmark dataset and a real-world dataset have demonstrated that our proposed active learning strategy is highly effective in selecting the most informative examples.

Categories and Subject Descriptors: H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Active learning, noise injection, ranking, sensitivity sampling

ACM Reference Format:
Wenbin Cai, Muhan Zhang, and Ya Zhang. 2015. Active learning for web search ranking via noise injection. ACM Trans. Web 9, 1, Article 3 (January 2015), 31 pages.
DOI: http://dx.doi.org/10.1145/2697391

1. INTRODUCTION

With the rapid growth of Internet and web service technology, ranking has become essential to many information retrieval (IR) applications such as web search and recommendation. Learning to rank is to automatically generate ranking functions through supervised learning. Like many other supervised learning tasks, training a high-quality ranking function usually requires a large number of labeled examples. In a typical setting, training data are randomly selected with certain presumed distributions and annotated by trained editors. This data collection process is called passive learning. However, in many real-world learning-to-rank applications, it is very expensive to collect a sufficiently large labeled set to ensure model quality. In fact, not all of the data selected with passive learning contribute positively to the performance of the ranking model. To reduce the unnecessary annotation cost, active learning aims

A preliminary version of this article appeared as a short paper in Proceedings of the 21th ACM International Conference on Information and Knowledge Management (CIKM'12) [Cai and Zhang 2012].
Authors' addresses: W. Cai, M. Zhang, and Y. Zhang (corresponding author), Shanghai Key Laboratory of Multimedia Processing and Transmissions, Shanghai Jiao Tong University, Shanghai 200240, China; emails: {cai-wenbin, ufo000, ya_zhang}@sjtu.edu.cn.

DOI: http://dx.doi.org/10.1145/2697391

to selectively label the most informative examples. A typical active learning process can be summarized as follows: (1) generate a base model from a small initial training set, (2) select examples with a sampling function from a large set of unlabeled data and label them, and (3) add the newly labeled examples to the training set and retrain the model. This sampling process is repeated until a certain performance expectation is met or the labeling budget is used up. The key idea behind active learning is that if a learner is allowed to choose the data from which it learns, it can achieve better performance with fewer labeled instances.

So far, active learning has been extensively studied for classification problems. Uncertainty sampling, a widely adopted active learning strategy, aims to choose the examples that are closest to the decision boundary. Take binary classification for instance: uncertainty sampling is to choose the examples whose posterior probabilities are nearest 0.5 [Lewis and Gale 1994]. For nonprobabilistic learning classifiers such as support vector machines (SVMs), this strategy selects the data points that are closest to the separating hyperplane [Tong and Koller 2001]. Another classical active learning framework is query by committee (QBC), which generates a committee of models and selects the examples with the greatest disagreement among the members [Freund et al. 1997].

Compared to active learning for classification, active learning for ranking faces some unique challenges. First, there is no notion of margin in ranking, and therefore many of the margin-based active learning strategies are not readily applicable. Further, different from the classification-oriented performance metric where data examples are treated and evaluated independently of each other, the performance metrics for ranking are based on the ranked list; that is, the data examples are not independently and individually evaluated. Thus, even some straightforward active learning strategy, such as QBC, has not been justified for ranking problems. Finally, the dominant performance metrics for ranking such as Discounted Cumulative Gain (DCG) [Jarvelin and Kekalainen 2000] put more weights on the top-retrieved items in a ranked list. As a consequence, we need to take particular consideration of this ranking-specific characteristic into designing a data sampling function.

Focusing on ranking tasks, one can categorize existing learning-to-rank approaches into three major groups: pointwise approaches [Cossock and Zhang 2006], pairwise approaches [Cao et al. 2006], and listwise approaches [Xia et al. 2008]. In recent years, many active learning-for-ranking algorithms have been specifically proposed. Most of them perform the sample selection either at the query level [Yilmaz and Robertson 2009; Cai et al. 2011; Bilgic and Bennett 2012; Qian et al. 2013] or at the document level [Silva et al. 2011; Donmez and Carbonell 2008; Yu 2005; Aslam et al. 2009; Ailon 2011]. The query-level active learning selects all documents associated with an informative query, and the document-level active learning selects documents individually. Recently, with particular consideration of the query–document structure in ranking data, there is a small amount of work integrating the query-level and the document-level sampling [Long et al. 2010; Cai and Zhang 2012], which first selects the most informative queries at the query level and then selects the most informative documents related to the selected queries.

Gradient Boosting Decision Tree (GBDT), which has been extensively employed as the state-of-the-art model in many ranking tasks in recent research [Chapelle et al. 2010, 2011; Shen et al. 2013], represents a pointwise approach that is widely used in practice by commercial search engines such as Yahoo and Yandex. Therefore, an active learning-for-ranking strategy targeting GBDT as the base ranker is of great need in the ranking community.

Following the uncertainty-based sampling principle, we aim to identify the close-to-boundary examples, which is a nontrivial task for decision-tree-based models (GBDT-based pointwise ranking is regression in fact). First, unlike SVMs, there is no clear

definition of the distance between data points and decision boundaries in GBDT, and therefore the distance-based uncertainty sampling approach is unapplicable. Furthermore, the examples located in the same decision region will definitely receive the same predicted scores, and thus we cannot distinguish the close-to-boundary examples from others with predicted scores. To the best of our knowledge, there is very limited work on active learning for ranking targeting GBDT as the base ranker.

In this article, we propose a novel active learning strategy tailored for ranking with GBDT. Unlike the max-margin-based models, decision-tree-based methods, where there exist clearly defined decision regions, tend to be sensitive to noise perturbations in the data. We attempt to leverage this property of GBDT in designing the active learning strategy. Moreover, considering that most of the ranking metrics are based on ranked lists and more weights are put on top-ranked items such as DCG, item rank is incorporated into the selection function to overweight the top-ranked items.

Leveraging the property that decision-tree-based methods are sensitive to noise perturbations, we utilize noise injection to perturb data and generate a set of noisy copies for each example. Clearly, if the corresponding noisy copies of an example have various predicted scores (i.e., crossing the decision boundary after noise perturbation), this data point is expected to be close to the decision boundary of the current model (i.e., sensitive to noise perturbation). We name this method of identifying the close-to-boundary data examples as sensitivity sampling (SS). Moreover, we theoretically analyze this technique by exploring the connection between the sensitivity defined for sample selection and the regularization to provide a potentially theoretical foundation w.r.t. the model's generalization performance.

For the ranking tasks, because we are mainly interested in the resulting ranking list focusing on the top-retrieved items rather than the actual predicted ranking scores, we tailor the proposed sensitivity sampling to derive a novel active learning strategy for ranking called ranking-based sensitivity sampling (RSS). First, the score distribution generated with noise injection is transformed to the rank distribution. Then, using a DCG-like gain function to measure each possible ranked list, the examples having the highest expected variation in the gain value are selected, that is, those most sensitive to noise perturbation in terms of ranking. Considering the query–document structure in ranking data, we investigate the RSS strategy at both the query level and the document level. Extensive experimental results on both the benchmark LETOR 4.0 dataset and a real web search ranking dataset from a commercial search engine have demonstrated that our proposed active learning strategy can achieve better performance than the state-of-the-art methods.

The proposed noise-injection-based active learning strategy is generic. In this study, we further generalize it to RankSVM and classification tree to show its applicability in a wide variety of applications. Empirical studies are performed on several benchmark datasets, and the results show the effectiveness of our proposed algorithms.

The main contributions of this article are summarized as follows:

—We propose a novel noise-injection-based method, SS, which is targeted on tree-based models to identify the close-to-boundary data examples.
—We theoretically analyze the proposed sensitivity sampling approach by deriving the connection between the sensitivity defined for sample selection and the model regularization to provide a potentially theoretical support of its generalization ability.
—We further tailor the proposed SS principle for learning to rank and derive a new active learning-for-ranking approach called RSS, which is highly effective in choosing the most informative examples in ranking.
—Moreover, we generalize the noise-injection-based active learning strategy to several other base learners to show its applicability in a wide range of applications.

The remainder of this article is organized as follows. Section 2 provides a brief review of the related work. Section 3 briefly introduces the GBDT model. Section 4 presents the SS to identify the close-to-boundary examples through noise injection. The proposed active learning-for-ranking strategy, ranking-based sensitivity sampling, is presented in Section 5. Section 6 presents the experiments and interprets the results. We generalize the noise-injection-based active learning strategy to several other popular learners in Section 7. Finally, Section 8 concludes the article and proposes future directions.

2. RELATED WORK

The objective of active learning is to achieve high model performance using as few labeled examples as possible, thereby minimizing the cost of data labeling. Active learning is well motivated in many supervised learning tasks where unlabeled data may be abundant but labeled data examples are expensive to obtain. In this section, we first briefly review related work on active learning and then discuss existing active learning-for-ranking algorithms.

2.1. Active Learning

So far, various active learning strategies have been proposed. A comprehensive active learning survey is given in Settles [2012].

One common strategy is called uncertainty sampling [Lewis and Gale 1994; Tong and Koller 2001], which aims to choose the examples whose labels the current model is least certain about. This strategy is usually straightforward for probabilistic models using entropy to measure the uncertainty. For the nonprobabilistic models such as SVMs, this strategy selects the examples which are close to the separating boundary [Tong and Koller 2001].

QBC is another typical active learning framework, which generates a committee of model members and select unlabeled data instances about which the models disagree the most [Freund et al. 1997]. A popular function to quantify the disagreement is vote entropy. To efficiently generate the committee, popular ensemble learning methods, such as Bagging and Boosting, have been employed [Abe and Mamitsuka 1998].

Another decision-theoretic active learning strategy is to minimize the generalization error of the model. Roy and McCallum [2001] proposed an optimal active sampling method to choose the example that leads to the lowest generalization error on the future test set once labeled and added to the training set. The weakness is that the computational cost of this method is extremely high. Instead of choosing the example yielding the smallest generalization error, Nguyen and Smeulders [2004] suggested to query the instance that has the largest contribution to the current error. Chon et al. [1996] proposed a statistically optimal active learning approach, which aims to choose the examples minimizing the output variance to reduce the generalization error.

2.2. Active Learning for Ranking

Compared to the traditional supervised learning problems, a unique query–document structure exists in the ranking data and leads to a unique data dependence relationship; that is, each query is independent of each other, and the query–document pairs are conditionally dependent given a query. Most of the existing active learning-for-ranking algorithms can be categorized into two types: the query-level active learning and the document-level active learning.

For query-level active learning, Yilmaz and Robertson [2009] empirically showed that having more queries but shallow documents performed better than having fewer queries but deep documents. Yang et al. [2009] presented a greedy sampling method for query selection by maximizing a linear combination of query difficulty, query density, and query diversity. Cai et al. [2011] proposed a query selection strategy by combining

domain adaptation and QBC-based active learning. Bilgic and Bennett [2012] introduced a query sampling algorithm that generalizes the usability of Expected Loss Optimization (ELO) [Long et al. 2010] to any base ranker. In recent development, Qian et al. [2013] introduced a pairwise query selection method under a layered hashing framework.

For document-level active learning, Aslam et al. [2009] empirically compared several document selection strategies for ranking, for example, depth-k pooling, uniform random sampling, and so forth. Yu [2005] proposed a novel document-level active sampling algorithm, which treats the document pairs with similar predicted relevance scores as the most informative examples. The document sampling is applied to RankSVM [Herbrich et al. 2000], which represents a classical pairwise learning-to-rank approach. Donmez and Carbonell [2008] proposed a theoretical document selection algorithm to query the documents that are expected to maximally change the current ranking model. The base ranking functions are RankSVM and RankBoost [Freund et al. 2003]. Based on statistical learning theory, Ailon [2011] analyzed the query complexity for pairwise ranking. Silva et al. [2011] proposed a novel document sampling algorithm based on association rules, which does not rely on any initial training seed.

Taking particular consideration of the query-document structure in the ranking data, Long et al. [2010] proposed a two-stage framework that integrates the query-level active learning and the document-level active learning. Under the Bayesian framework, the ELO principle is introduced for active learning. Recently, Cai and Zhang [2012] proposed a novel active learning-for-ranking strategy, which is to choose the examples with the largest variance in terms of ranking through noise perturbation.

While most of the existing active learning-for-ranking algorithms are applied to pairwise learning-to-rank approaches (e.g., RankSVM), there is still very limited work targeting pointwise approaches such as GBDT. Due to the significant role of GBDT in ranking tasks, a targeted active learning strategy for GBDT is of great need. In this study, we extend our previous work [Cai and Zhang 2012] by two major additional contributions: (1) we theoretically analyze the proposed sensitivity sampling approach by deriving the connection between the sensitivity defined for sample selection and the model regularization to provide a potentially theoretical support of its generalization ability, and (2) we generalize the noise-injection-based active learning strategy to several other base learners to show its applicability in a wide spectrum of applications.

3. A BRIEF INTRODUCTION OF GRADIENT BOOSTING DECISION TREE

In this section, we briefly introduce the base ranker, GBDT, employed in this study. More details about GBDT can be found in Friedman [2001].

Decision tree model. Given the training set $\{(x_i, y_i), x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}_{i=1}^{n}$, the decision-tree-based model partitions the space of all joint predictor variable values into disjoint regions R_j, j=1, 2, ..., J as represented by the terminal nodes of the tree. A constant γ_j is assigned to each such region and the predictive rule is

$$x \in R_j \Rightarrow f(x) = \gamma_j. \tag{1}$$

The γ_j is the average of y_i in the terminal region R_j:

$$\gamma_j = \text{Ave}_{x_i \in R_j} y_i = \frac{1}{N_j} \sum_{x_i \in R_j} y_i, \tag{2}$$

where N_j is the number of data points within the region R_j. Thus, a J-terminal node tree can be formally expressed as

$$\text{tree}(x) = \sum_{j=1}^{J} \gamma_j \mathbf{1}(x \in R_j), \tag{3}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function.

Gradient Boosting Decision Tree. Under the functional gradient boosting framework, the crucial idea of GBDT is to repeatedly fit a J-terminal node regression tree to the residual: $r_{it} = y_i - f_{t-1}(x_i)$ (at the tth iteration). As in Friedman [2001], $f_0(x)$ is initialized as $f_0(x) = \sum y_i/n$, which can be regarded as a single terminal node tree. The update rule is

$$f_t(x) \leftarrow f_{t-1}(x) + \lambda_t \sum_{j=1}^{J_t} \gamma_{jt} \mathbf{1}(x \in R_{jt}), \tag{4}$$

where λ_t denotes the shrinkage factor, and γ_{jt} is the average of the residual in each terminal region R_{jt}:

$$\gamma_{jt} = \frac{1}{N_{jt}} \sum_{x_i \in R_{jt}} (y_i - f_{t-1}(x_i)). \tag{5}$$

Thus, GBDT can be formally expressed as an additive model with the special case that each base function is a regression tree:

$$f(x) = f_0(x) + \sum_{t=1}^{T} \lambda_t \sum_{j=1}^{J_t} \gamma_{jt} \mathbf{1}(x \in R_{jt}), \tag{6}$$

where T denotes the number of individual trees in GBDT. In recent development, GBDT has been extensively employed as the state-of-the-art learning algorithm in many ranking tasks [Chapelle et al. 2010, 2011; Shen et al. 2013]. In the following, we present our active learning strategy that is particularly tailored for GBDT.

4. SENSITIVITY SAMPLING VIA NOISE INJECTION

The classical uncertainty sampling strategy treats the examples that are close to the decision boundary as the most informative ones. This strategy is usually straightforward to implement for probabilistic models. Taking binary classification as an example, uncertainty sampling aims to select the examples whose posterior probabilities are closest to 0.5 [Lewis and Gale 1994]. For the nonprobabilistic models such as SVMs, this strategy selects the instances that are close to the separating hyperplane [Tong and Koller 2001]. However, in many state-of-the-art learning algorithms having clearly defined decision regions (e.g., the decision-tree-based models such as GBDT), it is nontrivial to identify the close-to-boundary data instances.

As mentioned in the previous section, decision trees are sensitive to noise perturbations in the data. For the decision-tree-based models, if a data example is close to the decision boundary, a small perturbation in feature space may make it cross the decision boundary and result in a change in the predicted score. On the contrary, if a data example is far away from the decision boundary, the predicted score under reasonable perturbation will remain consistent. Based on this intuition, we propose a novel method called sensitivity sampling, which leverages noise injection to identify the close-to-boundary instances.

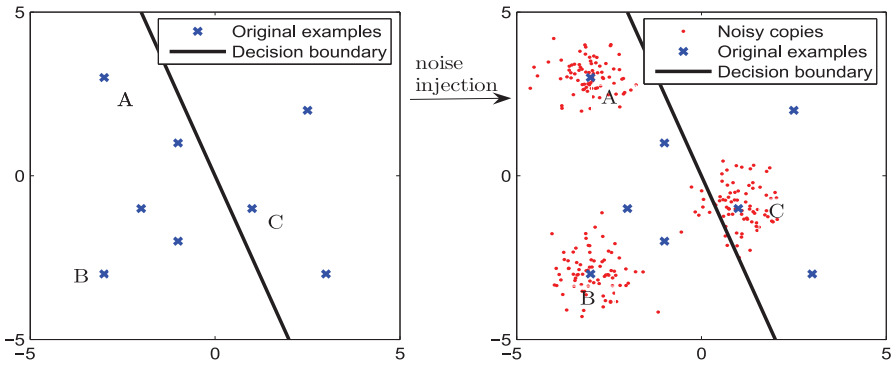


Fig. 1. An illustration of noise injection. Examples A and B are far away from the decision boundary, so that the noisy copies stay in the same region with the original examples. In contrast, example C is very close to the decision boundary, and hence its noisy copies tend to cross the boundary.

In the rest of this section, we first introduce the noise injection process. Then, we derive a novel approach called sensitivity sampling to identify the close-to-boundary examples. Finally, we theoretically analyze the proposed sensitivity sampling approach by deriving the connection between the sensitivity used for sample selection and model regularization to provide a potential guarantee of its generalization performance.

4.1. Noise Injection

Let x be a data example in the unlabeled dataset (denoted by pool set). Noise injection distorts x by adding some random noise ϵ to the features of x and generates m noisy copies around x. We formulate noise injection as follows:

$$x^k = x + \epsilon^k, \qquad k = 1, 2, \ldots, m, \tag{7}$$

where ϵ^k is the noise injected. In this study, we assume the noise ϵ to be a zero-centered Gaussian vector with independent coordinates:

$$\mathbb{E}[\epsilon] = \mathbf{0}, \ \mathbb{E}[\epsilon\epsilon^{\mathrm{T}}] = \sigma^2\mathbf{I}, \tag{8}$$

where $\mathbf{I}$ is the identity matrix. Our choice of Gaussian noise is motivated by previous work [Bishop 1995; Matsuoka 1992] on the theoretical demonstration of the connection between noise injection and the learning model's generalization performance.

Figure 1 illustrates the outcome of noise injection in a two-dimensional feature space. Points A and B are far away from the decision boundary, so that their corresponding noisy copies stay in the same region as the original examples. On the contrary, data example C is very close to the decision boundary, and thus its noisy copies tend to cross the boundary.

4.2. Sensitivity Sampling with Noise Injection

It is easy to see that a data point that crosses from one decision region to another will receive a different predicted value, and therefore examples near the decision boundary will be most prone to such phenomena after noise distortion.

The variation of the model's output with respect to the example x due to the perturbation with noise ϵ can be formulated as

$$\mathrm{Var}_\epsilon(x) = f(x + \epsilon) - f(x), \tag{9}$$

where $f(.)$ represents the current model. The sensitivity of the model with respect to a single example x is then defined as the expected variation after noise injection:

$$\mathrm{S}(x) = \mathbb{E}_\epsilon[||\mathrm{Var}_\epsilon(x)||^2] = \frac{1}{m}\sum_{k=1}^{m} ||f(x+\epsilon^k) - f(x)||^2, \tag{10}$$

where m is the number of noisy copies generated with Gaussian noise, and $\mathbb{E}_\epsilon[.]$ denotes the expectation over m noisy copies.

Clearly, if a data example x is close to the decision boundary, noise perturbation will result in a large value of sensitivity. Otherwise, the value of sensitivity will be small or even 0. Hence, the sensitivity defined is a reasonable measurement to estimate the distance between the data example x and the decision boundary. As a consequence, the SS principle, which aims at selecting the close-to-boundary examples, can be represented as

$$x^*_{\mathrm{SS}} = \arg\max_{x\in\mathcal{U}} \mathrm{S}(x), \tag{11}$$

where $\mathcal{U}$ is the pool set, and x^*_{SS} denotes the unlabeled example in the pool set, which is expected to be closest to the current decision boundary.

4.3. Theoretical Analysis

In machine-learning problems, the ultimate goal is to learn a model $f(.)$ with good generalization performance. The generalization error can be formulated as

$$\begin{aligned}\mathrm{Err} &= \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}[f(x), y(x)]dP(x,y)\\ &= \int_{\mathcal{X}}\int_{\mathcal{Y}} \mathcal{L}[f(x), y(x)]p(y|x)p(x)dydx,\end{aligned} \tag{12}$$

where $y(x)$ and $f(x)$ are the true label and the predicted label of the example x, respectively. $\mathcal{L}[f(x), y(x)]$ is a given loss function.

Here, we focus on the squared-error loss since the pointwise ranking is a regression problem. Thus, the generalization squared error can be written as

$$\mathrm{Err} = \int_{\mathcal{X}} \mathbb{E}_{\mathcal{Y}}[(f(x) - y(x))^2|x]p(x)dx, \tag{13}$$

where $\mathbb{E}_{\mathcal{Y}}[.]$ denotes the expectation over $p(y|x)$. Furthermore, the expectation inside the integral can be decomposed as follows [Hastie et al. 2001]:

$$\begin{aligned}\mathbb{E}_{\mathcal{Y}}[(f(x) - y(x))^2|x] &= \sigma_\epsilon^2 + (\mathbb{E}_{\mathcal{Y}}[f(x)] - y(x))^2 + (\mathbb{E}_{\mathcal{Y}}[f(x) - \mathbb{E}_{\mathcal{Y}}[f(x)]])^2\\ &= \text{Unavoidable Error} + \text{Bias}^2 + \text{Variance}.\end{aligned} \tag{14}$$

As shown, the last two terms compose the expected squared error of the learned regression function.[1] The bias term represents the structural error due to the model class itself, and the variance term denotes the approximation error on the finite training data due to the high complexity of the model. For a high-variance model with fixed complexity (e.g., a decision-tree-based model with given size), a practical solution to reduce the variance is to add more examples to the training set, therefore improving the model's generalization performance. For active learning, we are not able to change

[1]Although there is as yet no agreed-upon formalism for the definitions of bias and variance for other tasks such as classification, the bias-variance decomposition still holds (e.g., [Valentini and Dietterich 2004]).

the model complexity during the sampling process. Hence, a straightforward generalization of this practical solution is to actively select the examples lying in complex regions and add them to the training set, thereby reducing the prediction variance due to the complex parts of the model learned. In the following, we attempt to derive the connection between the sensitivity defined for sample selection and model complexity. In particular, we aim to show that the sensitivity is highly correlated with model regularization, which is usually introduced during parameter estimation in the model fitting step to control the model complexity.

We generalize the sensitivity for a single example x to the entire pool set to define the sensitivity of the model because the pool set is usually large enough to capture the data distributions in active learning cases:

$$\mathrm{S}(f) = \mathbb{E}_x[\mathrm{S}(x)] = \mathbb{E}_x\mathbb{E}_\epsilon[||\mathrm{Var}_\epsilon(x)||^2]. \tag{15}$$

Using the first-order Taylor expansion, the variation of the model's output with noise injection can be expressed as

$$\begin{aligned} \mathrm{Var}_\epsilon(x) &= \nabla f(x)^{\mathrm{T}}\epsilon + o(\epsilon^2) \\ &\approx \nabla f(x)^{\mathrm{T}}\epsilon, \end{aligned} \tag{16}$$

where $\nabla f(x)$ denotes the derivative of $f(.)$ with respect to x. According to Equation (16) , the sensitivity of the learning model can be represented as

$$\begin{aligned} \mathrm{S}(f) &= \mathbb{E}_x\mathbb{E}_\epsilon[\mathrm{Var}_\epsilon(x)^{\mathrm{T}}\mathrm{Var}_\epsilon(x)] \\ &\approx \mathbb{E}_x\mathbb{E}_\epsilon[(\nabla f(x)^{\mathrm{T}}\epsilon)^{\mathrm{T}}\nabla f(x)^{\mathrm{T}}\epsilon] \\ &= \mathbb{E}_x\mathbb{E}_\epsilon[\mathrm{tr}(\nabla f(x)^{\mathrm{T}}\epsilon\epsilon^{\mathrm{T}}\nabla f(x))] \\ &= \sigma^2\mathbb{E}_x[\mathrm{tr}(\nabla f(x)^{\mathrm{T}}\nabla f(x))] \\ &= \sigma^2\mathbb{E}_x[||\nabla f(x)||^2], \end{aligned} \tag{17}$$

where tr(.) stands for the trace calculation. In the following, we consider two cases to explore the relationship between the sensitivity defined previously and regularization: the linear model and the nonlinear model.

Case 1 (linear case). A linear learning model is the one that involves a linear combination of the input features

$$f(x) = \theta^{\mathrm{T}}x + b. \tag{18}$$

The bias term b is usually omitted for simplicity. In fact, it is easy to employ the bias by padding an extra dimension of all 1s. The derivative of the linear function is

$$\nabla f(x) = \nabla(\theta^{\mathrm{T}}x) = \theta. \tag{19}$$

Combining Equation (17) and Equation (19), we have

$$\mathrm{S}(f) \approx \sigma^2||\theta||^2, \tag{20}$$

which is equivalent to the classical regularization term added to the loss function in the model fitting process, with the coefficient of the regularizer determined by the noise variance σ^2.

Case 2 (nonlinear case). Here, we take the GBDT model, which is employed as the base ranking function in this study, as an example. According to Equation (6), the

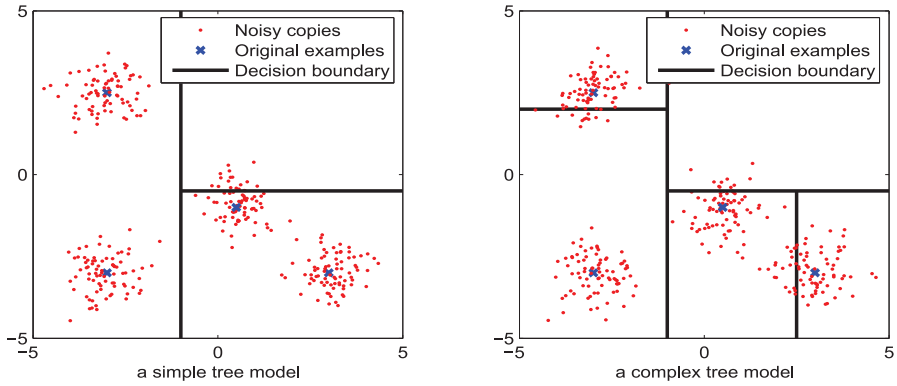


Fig. 2. An example to illustrate the relationship between the sensitivity and the size of a decision tree model. The left one is a small tree model, and the right one is a relatively complex tree model with more decision boundaries. With more close-to-boundary data examples, the sensitivity of the large tree model is higher than the small decision tree.

derivative is calculated as

$$\nabla f(x) = \sum_{t=1}^{T} \lambda_t \nabla \sum_{j=1}^{J_t} \gamma_{jt} \mathbf{1}(x \in R_{jt})$$
$$= \sum_{t=1}^{T} \lambda_t \sum_{j=1}^{J_t} \frac{1}{N_{jt}} \nabla \sum_{x \in R_{jt}} r_{it} \mathbf{1}(x \in R_{jt}). \tag{21}$$

Because the function $\sum_{x \in R_{jt}} r_{it} \mathbf{1}(x \in R_{jt})$ is not continuous, we cannot calculate the derivative $\nabla f(x)$ directly. However, we see that the derivative is directly correlated with the term $1/N_{jt}$. In the decision-tree-based model, the number of examples N_{jt} located in a single region R_{jt} is inversely related to the size of the tree. Consequently, the derivative $\nabla f(x)$ can be rewritten as a function taking the tree size as a variable. After the expectation calculation via Equation (17), it is straightforward to derive that the sensitivity is indeed a function of the size of the boosted trees. In other words, the sensitivity defined is highly correlated with the classical regularization term, the size of the tree, for the tree-based model. Figure 2 presents an illustrative example to show the link between the sensitivity and the size of a decision tree model. The left one is a small tree model (denoted as f_1), and the right one (denoted as f_2) is a relatively large tree model with more decision boundaries. Compared to the small tree f_1, there are more data instances that tend to cross decision boundaries after noise perturbation, and therefore the sensitivity of f_2 is higher than f_1.

We have shown that the sensitivity defined closely connects to the classical regularization term. Our approach aims to pick data points located in the complex regions (e.g., the complex regions of the GBDT with many decision boundaries) to greatly reduce the prediction variance due to the complex parts of the model learned, thereby improving the model's generalization performance.

5. SENSITIVITY SAMPLING FOR RANKING

In ranking applications, because we are mainly interested in the ranked list focusing on the top-ranked items rather than the absolute value of the predicted ranking score, we tailor the proposed sensitivity sampling principle for ranking tasks. In this section, we first detail the process to transform the score distribution, which is generated with noise

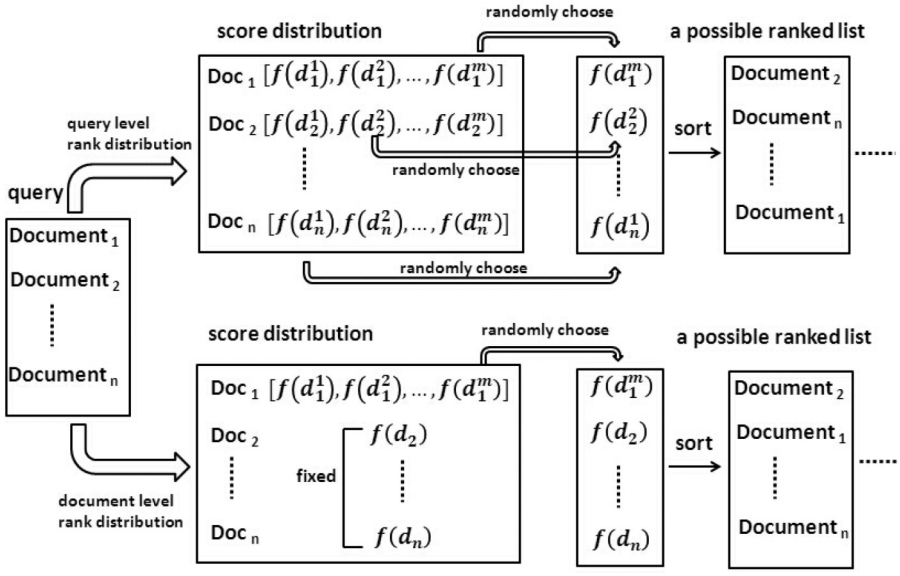


Fig. 3. The process to generate rank distribution.

injection, to the rank distribution. We then derive a novel active learning-for-ranking strategy called ranking-based sensitivity sampling. Considering the query–document structure in the ranking data, the RSS approach is derived at both the query level and the document level.

5.1. Rank Distribution Generation

Similar to SoftRank [Taylor et al. 2008], we assume that each ranking score is nondeterministic and is sampled from a certain score distribution. Previous work [Taylor et al. 2008] has assumed that the distribution of scores follows a given equal variance Gaussian distribution. However, the assumption in most cases is unrealistic because it implies that the score distribution is independent of the ranking model. In this work, we propose to approximate the score distribution leveraging noise injection; that is, $[f(x^1), f(x^2), \ldots, f(x^m)]$ is treated as an approximation of the score distribution for each example, which is then transformed to the rank distribution.

5.1.1. The Process to Generate Rank Distribution. Considering the query–document structure in the ranking data, the rank distribution is derived at two levels: the query-level rank distribution $p(r|q)$ and the document-level rank distribution $p(r|d)$.

Suppose there are n documents related to a query. To generate the query-level rank distribution, we randomly sample a score from the score distribution of each document to generate a score vector $[f(d_1), f(d_2), \ldots, f(d_n)]$. We then sort the documents according to the score vector to generate a ranked list for the query. By performing the previous process M times, we get an approximation of the query-level rank distribution:

$$p(r|q) = \frac{\#r}{M}, \quad r \leftarrow \text{sort}_{f(d_i)\sim[f(d_i^1),\ldots,f(d_i^m)],i=1,2,\ldots,n}[f(d_1), \ldots, f(d_n)]. \tag{22}$$

Figure 3 shows the process to generate the query-level rank distribution (top panel).

Given a query–document pair, to generate its document-level rank distribution, the predicted ranking scores of other documents related to the given query are fixed to be

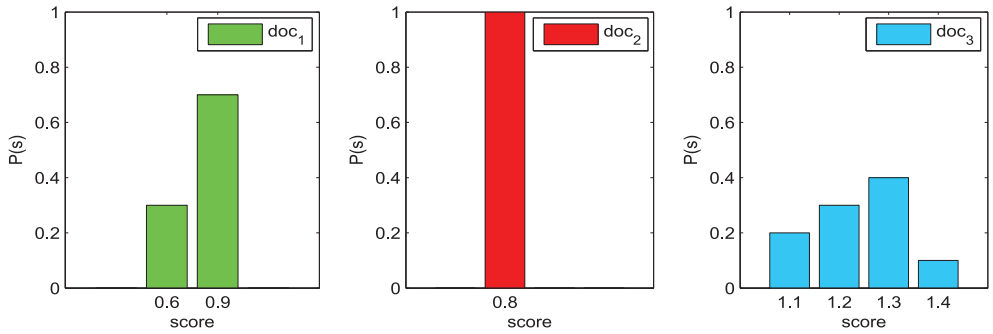


Fig. 4. The score distributions of the three example documents. doc_1 may be close to a decision boundary because it is assigned two predicted scores after a small perturbation. Similarly, doc_2 may be far away from any decision boundary, and doc_3 may be close to the intersection of multiple decision boundaries.

the predicted scores without noise perturbation. We then randomly sample the score distribution of the given document to generate a score vector $[f(d_1), f(d_2), \ldots, f(d_n)]$ and sort it to produce a ranked list for the document. Again, the sampling and sorting process is performed M times to generate an approximation of the document-level rank distribution:

$$p(r|d) = \frac{\#r}{M}, \quad r \leftarrow \text{sort}_{f(d)\sim[f(d^1),\ldots,f(d^m)],d\in q}[f(d_1), \ldots, f(d_n)]. \tag{23}$$

The bottom panel of Figure 3 presents the process of generating the rank distribution of Doc_1.

5.1.2. An Illustrative Example. Figure 4 shows the score distributions of three example documents. After noise injection, the data example doc_1 is assigned two predicted ranking scores, with a probability of 0.3 and 0.7, respectively. This implies that doc_1 may be close to a decision boundary and hence a small perturbation leads its copies to cross the boundary. Similarly, the noise perturbation of doc_3 results in four predicted ranking scores, suggesting that doc_3 may be close to the intersection of multiple decision boundaries. On the contrary, doc_2 may be far away from any decision boundary because its ranking scores are very consistent under perturbation.

Suppose a query q only matches the previous three documents d_1, d_2, and d_3. The ranked list for the query q is either $r_1 = [d_3, d_2, d_1]$ or $r_2 = [d_3, d_1, d_2]$. We repeat the sampling and sorting process to generate its query-level rank distribution. In this particular example, the probabilities of the two ranked lists could be approximated with 0.3 and 0.7, respectively, that is, $p(r_1|q) = 0.3$ and $p(r_2|q) = 0.7$.

Assume the predicted scores for d_1, d_2, and d_3 without perturbation are 0.6, 0.8, and 1.2, respectively. Through repeated sampling and sorting, the ranked list of d_1 is either $r_1 = [d_3, d_2, d_1]$ or $r_2 = [d_3, d_1, d_2]$ with a probability of 0.3 and 0.7, respectively, and thus we have $p(r_1|d_1) = 0.3$ and $p(r_2|d_1) = 0.7$. For d_2 and d_3, the ranked list will always be $r = [d_3, d_2, d_1]$, that is, $p(r|d_2) = 1$ and $p(r|d_3) = 1$.

5.2. Ranking-Based Sensitivity Sampling

As discussed before, if an example is close to the decision boundary, noise injection is more likely to vary the predicted ranking score. However, change in ranking score does not necessarily lead to variation in the final ranking. In the example shown in Figure 4, doc_3 maintains a stable ranking despite a relatively large variation in its predicted scores. Because we are mainly interested in the ranking of the examples rather than the actual ranking scores, we compute the sensitivity in terms of ranking rather than

ALGORITHM 1: RSS at Query Level

Input: the small labeled training set $\mathcal{D}$, the unlabeled pool data set $\mathcal{U}$, the ranking function $f(.)$ trained with $\mathcal{D}$.
Output: The query q^* having the highest sensitivity.
Inject noise ϵ drawn from an $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ Normal to each unlabeled example via Equation (7);
for *each* q *in* $\mathcal{U}$ **do**
  Generate query-level rank distributions via Equation (22);
  for *each* $r \in \mathcal{R}$ **do**
    Calculate the gain value via Equation (24);
    Calculate the variation in terms of ranking via Equation (25);
  end
  Calculate the sensitivity via Equation (26);
end

the predicted ranking scores. The details of the RSS algorithms at both the query level and the document level are provided in the following section.

5.2.1. RSS at Query Level. The main idea of the query-level active learning is described as follows. Given the query-level rank distribution, if the ranked list of the query is stable after noise perturbation, it suggests that the current ranking model is insensitive about the query. Otherwise, the query is sensitive under the model, and we treat it as the informative one.

Inspired by the DCG function [Jarvelin and Kekalainen 2000], we define the gain function g of the ranked list as

$$g(r) = \sum_{i=1}^{n} \left(2^{s(r_i)} - 1\right) / log_2(1 + r_i), \tag{24}$$

where $s(r_i)$ is the predicted ranking score of the document without noise perturbation at rank r_i in the ranked list, and n is the number of documents related to the list. We then represent the variation of the ranked list associated with a query due to noise injection as the DCG-like gain variation:

$$\mathrm{Var}_\epsilon(q) = g(r|q + \epsilon) - g(r|q). \tag{25}$$

It is easy to see that variation in the DCG-like gain implicitly emphasizes sensitivity at the top of ranking. More specifically, if a query is ranked differently for its top-ranked documents, the variation in the gain value will be large. Otherwise, it will result in a relatively small variation.

We calculate the gain value for each possible ranked list of the query q and compute the sensitivity as the expected variation in the gain values. The query selection criteria can be expressed as

$$q^*_{\mathrm{RSS}} = \arg\max_{q \in \mathcal{U}} \sum_{r \in \mathcal{R}} p(r|q) ||g(r|q + \epsilon) - g(r|q)||^2, \tag{26}$$

where $\mathcal{R}$ denotes all possible ranked lists, and q^*_{RSS} is the selected query. The pseudo-code for query sampling is shown in Algorithm 1.

In active learning settings, a potentially important issue of the interactive learning process is the stopping criterion, that is, deciding when to stop learning, which is still an open question and less researched [Zhu et al. 2010a]. Very often, for many practical applications, the active learning process is stopped when a given performance threshold is reached or a certain labeling budget is exhausted. In this study, we simply stop the sampling process with a predefined number of iterations, which is a commonly adopted

ALGORITHM 2: RSS at Document Level

Input: the small labeled training set $\mathcal{D}$, the unlabeled pool dataset $\mathcal{U}$, the ranking function $f(.)$ trained with $\mathcal{D}$.
Output: The document d^* having the highest sensitivity.
Inject noise ϵ drawn from an $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ Normal to each unlabeled example via Equation (7);
for *each d in* $\mathcal{U}$ **do**
  Generate document-level rank distributions via Equation (23);
  for *each* $r \in \mathcal{R}$ **do**
    Calculate the gain value via Equation (24);
    Calculate the variation in terms of ranking via Equation (27);
  end
  Calculate the sensitivity via Equation (28);
end

convention. In principle, defining an appropriate stopping criterion for active learning is to make a tradeoff between the data annotation cost and the model quality, and we consider it as our future work.

5.2.2. RSS at Document Level. The main motivation behind the document-level active learning is described as follows. The query-level sampling chooses all documents associated with the selected query. However, a sensitive query may still contain documents about which the ranking model is insensitive, for example, doc_2 and doc_3 shown in Figure 4. Given the document-level rank distribution, we aim to select only documents about which the ranking model is sensitive as the informative examples, for example, doc_1 in Figure 4.

Similar to the query selection, the variation of the ranked list related to a document due to noise perturbation is represented as the gain variation:

$$\mathrm{Var}_\epsilon(d) = g(r|d+\epsilon) - g(r|d). \tag{27}$$

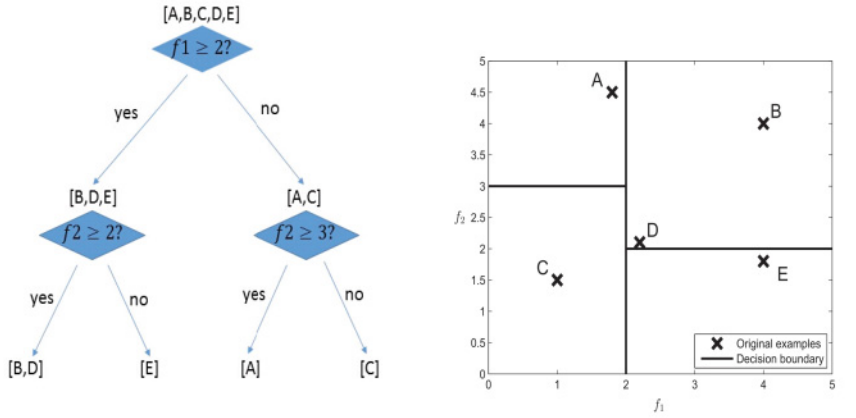
We then use the expected variation in the gain values over each possible ranked list to measure the sensitivity of the document, and the document selection criteria can be formulated as

$$d^*_{\mathrm{RSS}} = \arg\max_{d\in\mathcal{U}} \sum_{r\in\mathcal{R}} p(r|d)||g(r|d+\epsilon) - g(r|d)||^2, \tag{28}$$

where d^*_{RSS} is the selected document. The pseudo-code for document sampling is given in Algorithm 2.

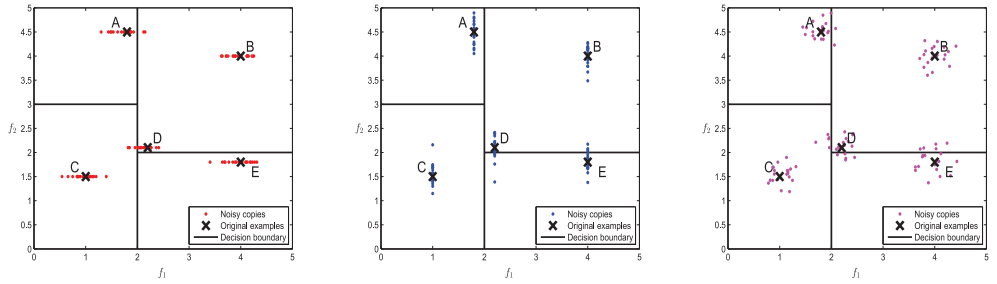
5.2.3. RSS at Two Stage. Both the query-level active learning and the document-level active learning have their own disadvantages. The query-level sampling selects all documents associated with a query. It may include some noninformative documents since there are usually a large number of documents related to a selected query, especially in the web search ranking applications. Because the quality of a ranking model is determined mainly by the top-ranked documents, most of them are noninformative. The document-level sampling ignores the query–document structure and selects documents independently. This sampling strategy simply ignores the query–document structure and the data dependency relationship, and hence the result may not be optimal.

To address the problem, Long et al. [2010] proposed a two-stage active learning framework, which first selects the most informative queries at the query level and then selects the most informative documents related to the selected queries. In this study, we follow this two-stage active learning strategy in designing our proposed RSS algorithm.



(a) a 2-layer decision tree (b) the partition of a 2-dimensional space

Fig. 5. A two-layer decision tree model, together with the corresponding partition of the two-dimensional space. The examples ABCDE are located in different decision regions.



(a) noise on the root feature (b) noise on the leaf feature (c) noise on root & leaf features

Fig. 6. The results after adding Gaussian noise on different features.

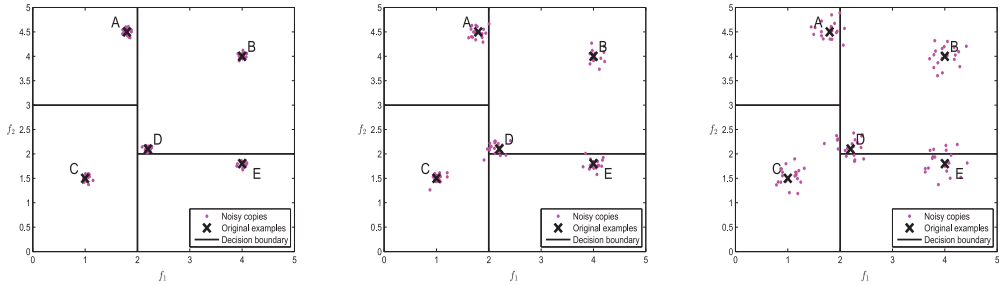
6. EXPERIMENTS

In this section, we first use a synthesized example to illustrate the idea and process of noise injection. Then, we present extensive empirical studies by applying the proposed ranking-based sensitivity sampling algorithm to ranking tasks.

6.1. Synthesized Example for Noise Injection

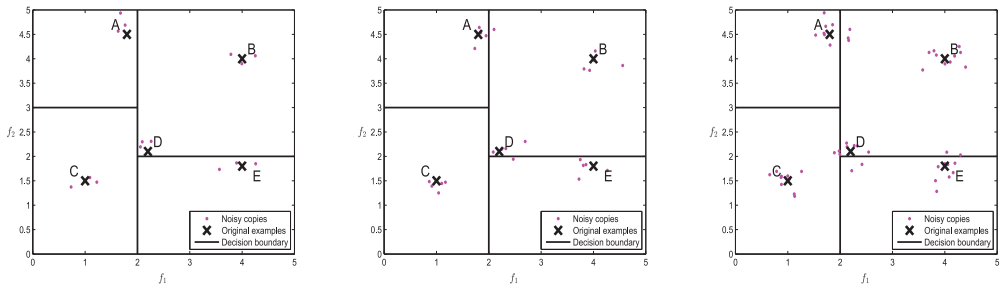
The first experiment is performed on a set of synthetic data in order to illustrate the general idea of noise injection for the decision-tree-based model and gain insight into the proposed active learning method. We build a two-layer decision tree model with two features and artificially generate five two-dimensional unlabeled data points (A = [1.8 4.5], B = [4 4], C = [1 1.5], D = [2.2 2.1], and E = [4 1.8]) located in different decision regions (as shown in Figure 5).

First, we conduct the experiments of adding noise to different features, that is, root feature versus leaf feature. Figure 6(a) presents the results after only adding noise on the root feature f_1, which are straightforward to understand. However, for the example E that is close to the decision boundary constructed with the leaf feature f_2, its predicted score definitely remains consistent due to the fact that it is far away from the root decision boundary. Thus, we are not able to identify the close-to-boundary example E in this case. Figure 6(b) shows the results of only adding noise to the leaf feature f_2, and similarly the close-to-boundary example A cannot be identified



(a) $\sigma = 0.05$ (b) $\sigma = 0.1$ (c) $\sigma = 0.2$

Fig. 7. The results after adding Gaussian noise with different standard deviations.



(a) $m = 3$ (b) $m = 5$ (c) $m = 10$

Fig. 8. The results after adding Gaussian noise with a different number of noisy copies.

as well. In contrast, Figure 6(c) shows the results of adding the noise drawn from a two-dimensional $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ Normal, so that the noisy copies are generated around the original examples. Therefore, each of the close-to-boundary examples can be easily identified.

In addition, we experiment with different parameters of the Gaussian noise to better understand the noise injection process. There are several important parameters in the Gaussian noise: the mean μ, the covariance matrix $\Sigma = \sigma^2\mathbf{I}$, and the number of noisy copies m. Clearly, the mean vector μ is set to be $\mathbf{0}$ because we aim at generating noisy copies around the original examples. Figures 7(a), 7(b), and 7(c) plot the results with different standard deviations σ ($\sigma = 0.05, \sigma = 0.1, \sigma = 0.2$), respectively. It is observed that the number of samples crossing the boundary after noise injection generally increases with the increasing value of the standard deviation σ. The result definitely matches the intuition. If an example is very close to the decision boundary, a small perturbation will make it cross the boundary. Otherwise, a larger perturbation is required.

Fixing the standard deviation as $\sigma = 0.2$, we further experiment with different values of m. Figure 8(a) presents the results after adding Gaussian noise with a small value of m ($m = 3$), and we see that there are no examples that cross the decision boundary after perturbation. This is because the noise may be occasionally added on the inverse direction to the boundary. When the value of m increases, the close-to-boundary examples are more likely to cross the boundary as shown in Figures 8(b) and 8(c).

Table I. The Statistics of the Two Learning-to-Rank Datasets

Dataset	AL Dataset	# Queries	# Documents	# Features
LETOR 4.0	base set	60	2,000	46
	pool set	1,940	66,383	
	test set	297	10,262	
WEB-SEARCH	base set	200	4,102	36
	pool set	3,000	60,609	
	test set	564	11,363	

6.2. Real Ranking Datasets

We use two learning-to-rank datasets to validate our proposed active learning algorithms. The first one is the LETOR 4.0 dataset,[2] a benchmark dataset on learning to rank. Each query–document pair is represented by 46 features, including both the low-level features, such as term frequency, inverse document frequency, and their combinations, and the high-level features, such as BM25 and PageRank. The query–document pairs are labeled with a three-level relevance judgment: {Bad, Fair, Good}. The second dataset is the web search dataset from a commercial search engine (denoted as WEB-SEARCH hereafter), and each query–document pair is represented with 36 features. The relevance is judged with a five-level relevance scheme: {Bad, Fair, Good, Excellent, Perfect}.

Both of the two datasets are randomly divided into three disjoint parts at the query level: base training set, pool set, and test set. We use the base training set as the small labeled dataset to train the initial base ranking models. The pool set is used as a large-size unlabeled dataset to select the most informative examples. The test set $\mathcal{T}$ is used to evaluate different active learning strategies. The statistics of the two datasets are listed in Table I. In practice, the initial base training set is often collected by depth-k retrieval with randomly selected queries according to the underlying distribution. Similar to LETOR [Liu et al. 2007], we normalize the features from the WEB-SEARCH dataset to the same scale at the query level with the following function:

$$f_{(i,j)}^{\text{Norm}} = \frac{f_{(i,j)} - min_{i \in n}\{f_{(i,j)}\}}{max_{i \in n}\{f_{(i,j)}\} - min_{i \in n}\{f_{(i,j)}\}}, \tag{29}$$

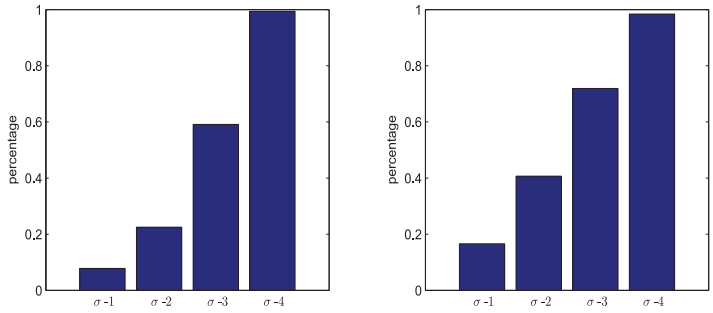
where n denotes the number of documents w.r.t. a certain query, and $f_{(i,j)}$ represents the jth feature from the ith document.

6.3. Experimental Settings

For the base ranker, we use the GBDT model, a classical pointwise ranking approach that is widely used in practice by commercial search engines such as Yahoo, to train our ranking models.

We first experiment on noise injection to determine the optimal parameters for Gaussian noise. Then, we compare the proposed RSS algorithms with several other active learning algorithms at different levels to validate the effectiveness of our methods. The algorithms select the top k informative examples. In this study, the active learning process iterates 10 rounds, which is a commonly adopted convention. In each round of active selection, 50 queries are selected at the query level and 500 documents are selected at the document level, respectively. For the two-stage active learning, we empirically fix the number of documents selected for each query to be 10 based on the results from Yilmaz and Robertson [2009].

[2]http://research.microsoft.com/en-us/um/beijing/projects/letor/.



(a) LETOR 4.0 (b) WEB-SEARCH

Fig. 9. The percentage of documents that cross the decision boundary after perturbation, where σ-$i(i = 1, 2, 3, 4)$ denotes the standard deviation for noise injection.

6.4. Evaluation Metrics

The performance of the new ranking function is evaluated on the separate test set. We use two popular IR metrics, DCG and Mean Average Precision (MAP), to measure the performance of each active learning method.

DCG is defined with respect to multilevel relevance scores, and DCG at rank n for a given query is computed as

$$\text{DCG@}n = \sum_{i=1}^{n} \frac{2^{l(r_i)} - 1}{log_2(1 + r_i)}, \tag{30}$$

where $l(r_i)$ is the relevance score of the document associated with the query q at the rank r_i.

MAP is defined to deal with binary-level relevance judgement, which is obtained by averaging the AP values for all queries:

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{n=1}^{N} P@n * rel(n)}{\#\{relevant\ docs\}}, \tag{31}$$

where P@n represents precision at position n, and $rel(\cdot)$ is a binary function on the relevance of the rank-n document. Because the datasets used in our experiments are labeled with multilevel judgment, we treat {Bad, Fair} as {irrelevant} and the other relevance levels as {relevant}. Each experiment is repeated 10 runs and we report the average DCG at the rank 10 (DCG@10) and MAP.

6.5. Noise Injection

In this subsection, we experiment with noise injection on ranking datasets to empirically determine the optimal parameters for Gaussian noise. In this study, we set $\mu = \mathbf{0}$ and $\Sigma = \sigma^2\mathbf{I}$ (as shown in Equation (8)) and experiment with different values of σ and m.

First, we simply fix the number of generated noisy copies m to be 20 and experiment with four values of σ: $\sigma = 0.000001$, $\sigma = 0.00001$, $\sigma = 0.0001$, $\sigma = 0.001$, denoted as σ-1, σ-2, σ-3, and σ-4, respectively. Note that, in this study, the σ-1 is the smallest value available since we have rounded the features to six decimal places during the normalization process. Figure 9 shows the percentage of examples (i.e., documents) that cross the decision boundary after noise injection with different standard deviations. We observe that the percentage increases monotonically with the increasing value of the

Table II. Experiments on the LETOR 4.0 Dataset with Different Standard Deviations at the Document Level

	Active Learning Round										
Std-σ	1	2	3	4	5	6	7	8	9	10	Metrics
σ-1	2.38	2.42	2.44	2.45	2.48	2.49	2.51	2.51	2.54	2.53	DCG@10
σ-2	2.37	2.40	2.42	2.43	2.45	2.45	2.48	2.50	2.52	2.52	
σ-3	2.36	2.38	2.39	2.42	2.47	2.48	2.50	2.51	2.53	2.52	
σ-4	2.32	2.35	2.36	2.42	2.44	2.45	2.47	2.49	2.49	2.51	
σ-1	0.358	0.365	0.370	0.374	0.377	0.380	0.384	0.384	0.386	0.386	MAP
σ-2	0.356	0.364	0.367	0.371	0.373	0.374	0.380	0.380	0.383	0.385	
σ-3	0.354	0.359	0.362	0.364	0.369	0.371	0.374	0.375	0.378	0.379	
σ-4	0.352	0.356	0.360	0.362	0.365	0.370	0.375	0.377	0.381	0.384	

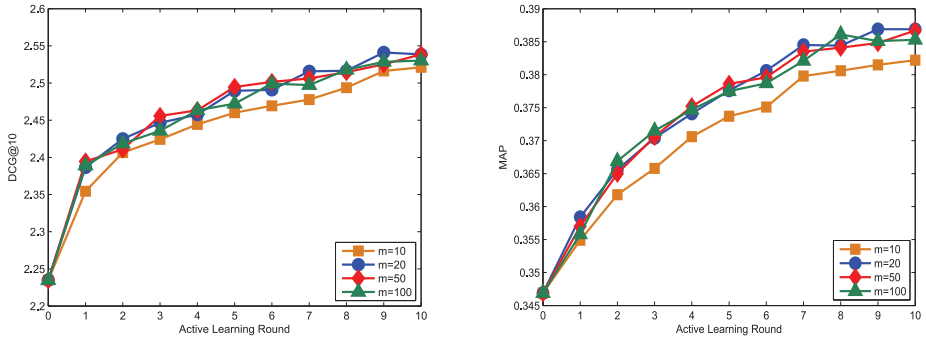
Table III. Experiments on the WEB-SEARCH Dataset with Different Standard Deviations at the Document Level

	Active Learning Round										
Std-σ	1	2	3	4	5	6	7	8	9	10	Metrics
σ-1	13.79	13.85	13.90	13.93	13.94	13.96	14.00	14.03	14.04	14.05	DCG@10
σ-2	13.79	13.84	13.87	13.90	13.92	13.94	13.98	13.99	13.99	13.99	
σ-3	13.78	13.80	13.82	13.85	13.87	13.90	13.90	13.95	13.96	13.95	
σ-4	13.78	13.84	13.90	13.92	13.93	13.93	13.97	13.99	14.01	14.02	
σ-1	0.658	0.663	0.666	0.668	0.668	0.669	0.669	0.670	0.672	0.672	MAP
σ-2	0.656	0.661	0.662	0.663	0.665	0.665	0.666	0.668	0.669	0.669	
σ-3	0.653	0.654	0.657	0.659	0.660	0.661	0.662	0.665	0.665	0.666	
σ-4	0.654	0.660	0.663	0.663	0.664	0.665	0.667	0.667	0.667	0.669	

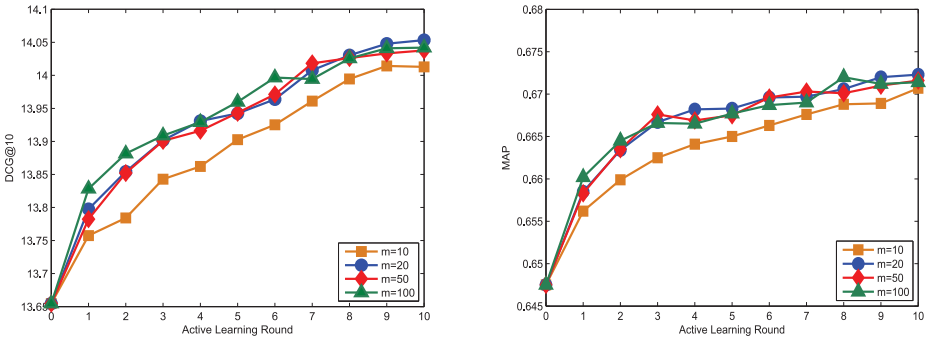
standard deviation σ. The results agree with the explanation discussed previously. If an example is very close to the decision boundary, a small noise perturbation will make it cross the decision boundary. Otherwise, a larger distortion is needed. Moreover, we experiment on the proposed RSS algorithms with different standard deviations. Table II and Table III show the experimental results at the document level on the LETOR 4.0data set and the WEB-SEARCH dataset, respectively. For each experiment, the values of best performance are underlined. We observe that σ-1 consistently outperforms the other three σ values for both datasets. A possible explanation is that using a small σ noise can select the examples that are very close to the decision boundary. We further see that the performance with σ-4 is better than σ-2 and σ-3 in some evaluation points. This phenomenon may be explained as follows. As shown in Figure 9, it is observed that nearly 100% of the examples cross the boundaries after injecting the noise with σ-4, and thus the examples chosen with σ-4 are more likely to capture the data distribution, resulting in a relatively good performance. Similar results are obtained when experimenting on the query level and two stage. Based on the experimental results, we empirically set σ to be 0.000001 in the remainder of our experiments.

Then, we fix the standard deviation σ to be 0.000001 and experiment with four different values of m: $m = 10$, $m = 20$, $m = 50$, $m = 100$. Figure 10 presents the empirical results at the document level on the LETOR 4.0 dataset and the WEB-SEARCH dataset, respectively. As shown in the figure, the RSS algorithm with $m = 10$ certainly performs the worst, which may be attributed to the reason we discussed in the previous section. Moreover, the performance of RSS with the other three values of m are comparable. Similar results can be obtained at other sampling levels. As a consequence, we empirically set $m = 20$ in the following.

The process to determine the optimal parameters is similar to previous studies [Zhu et al. 2010b]. In real-world applications in which we have an amount of initial labeled set, the optimal parameters could be similarly determined with this procedure, that is, experimenting with the parameter within a certain range by randomly splitting the



(a) LETOR 4.0 (b) LETOR 4.0



(c) WEB-SEARCH (d) WEB-SEARCH

Fig. 10. Experiments on the LETOR and WEB-SEARCH datasets with a different number of generated noisy copies m at the document level.

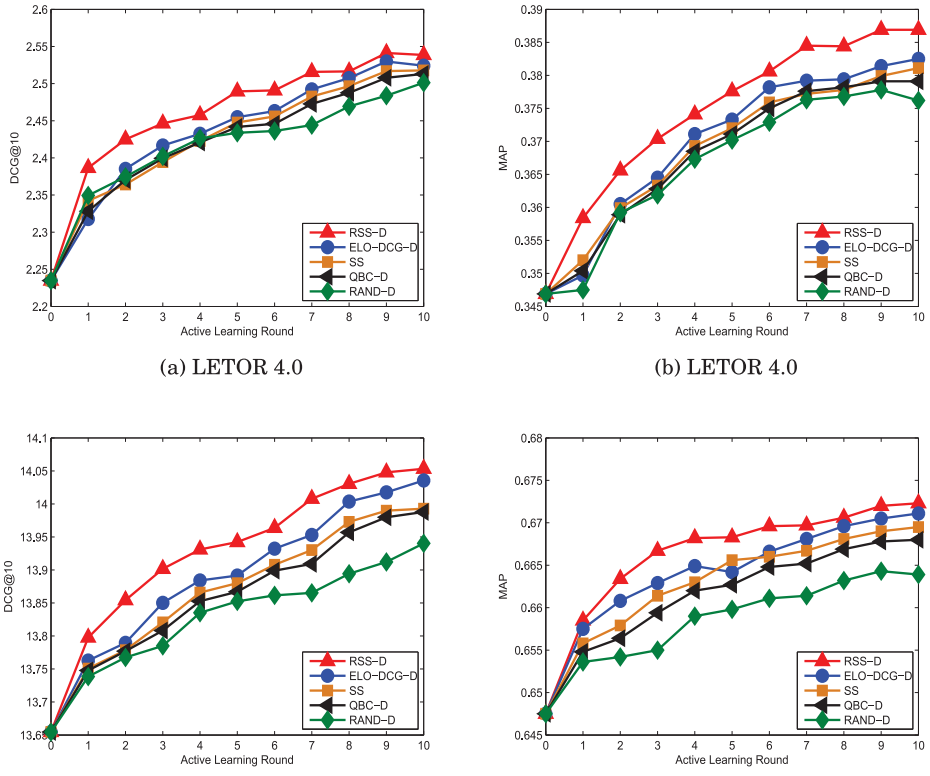
labeled set as training-pool-test sets and choosing the value having the best averaged performance. More complicated methods such as cross-validation can also be used, with increasing computation and complication.

6.6. Comparison Results and Discussion

In this subsection, we compare our proposed RSS algorithms with several other active learning methods to test the effectiveness in data selection at different levels. We denote the proposed query level, the document level, and the twostage RSS algorithms as RSS-Q, RSS-D, and RSS-QD, respectively.

6.6.1. Document-Level Active Learning. We first perform the active learning experiments at the document level and interpret the results. The document-level sampling is similar to the traditional active learning framework, which ignores the query–document structure in the data and chooses the examples independently. We compare our RSS-D algorithm against the following four competitors:

(1) SS: The proposed score-based sensitivity sampling algorithm, which individually chooses the documents with the highest sensitivity in predicted scores.
(2) QBC-D: The classical QBC for regression method is to choose the data points having the highest variance among the members' predictions (note that pointwise is regression in fact). In this study, the committee is constructed on bootstrap examples



(a) LETOR 4.0 (b) LETOR 4.0

(c) WEB-SEARCH (d) WEB-SEARCH

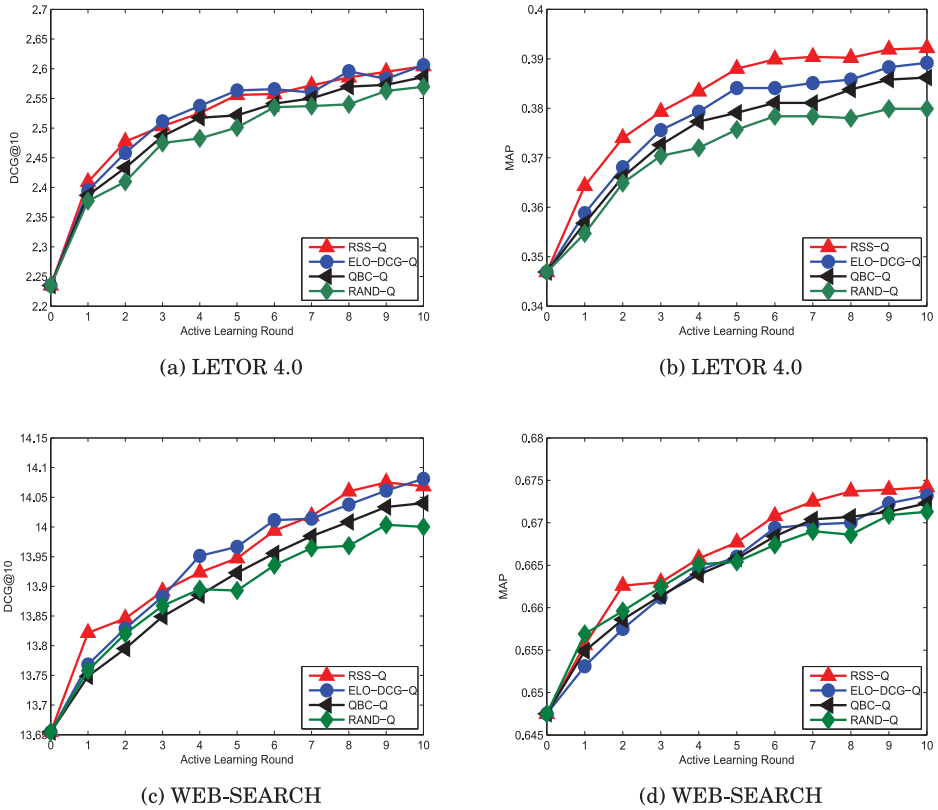
Fig. 11. Comparison results of RSS-D, ELO-DCG-D, SS, QBC-D, and RAND-D for the document-level active learning. Two datasets are used for the comparison: LETOR 4.0 and WEB-SEARCH. The evaluation metrics used are DCG@10 and MAP.

with Bagging type, which refers to the Query by Bagging [Abe and Mamitsuka 1998].

(3) ELO-DCG-D [Long et al. 2010]: The document-level ELO-DCG algorithm aims to select the documents with the largest expected DCG loss, representing the state-of-the-art method.

(4) RAND-D: The random document selection, representing a baseline.

The experimental results of the five document-level active learning algorithms on the LETOR 4.0 dataset are plotted in Figures 11(a) and 11(b). As shown in the figures, we observe that both RSS-D and ELO-DCG-D perform better than the other three methods in most cases, demonstrating that active learning-for-ranking algorithms (i.e., RSS-D and ELO-DCG-D) are more effective in selecting the most informative examples in the ranking task than score-based sensitivity sampling (i.e., SS), regression-oriented QBC-D, and the random document selection. Furthermore, the RSS-D consistently outperforms ELO-DCG-D during the entire active learning process. For the comparison results between SS and QBC-D, we observe that SS works slightly better than QBC-D in most of the cases. Similar results are obtained when comparing the five algorithms on the WEB-SEARCH dataset (Figures 11(c) and 11(d)). Here, we are particularly interested in the performance gap between RSS-D and SS since both algorithms aim to select the examples with the highest sensitivity. We observe that RSS-D significantly outperforms SS on both datasets. A possible explanation of the results is as follows.



(a) LETOR 4.0 (b) LETOR 4.0

(c) WEB-SEARCH (d) WEB-SEARCH

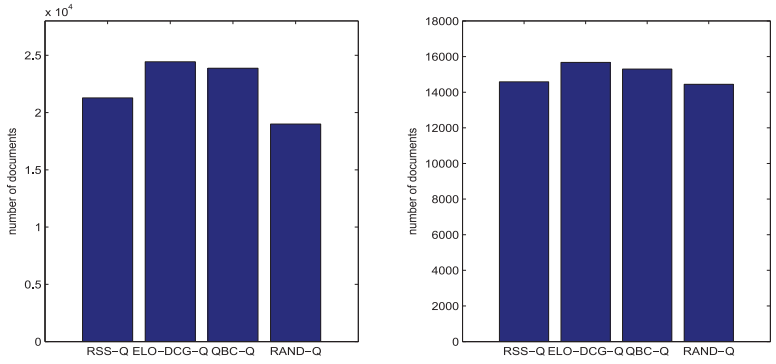
Fig. 12. Comparison results of RSS-Q, ELO-DCG-Q, QBC-Q, and RAND-Q for the query-level active learning. Two datasets are used for the comparison: LETOR 4.0 and WEB-SEARCH. The evaluation metrics used are DCG@10 and MAP.

While the SS simply selects the documents with the highest sensitivity in predicted ranking scores, the RSS-D chooses the documents with the highest sensitivity in ranking, which are more likely to contribute positively to the ranking model.

6.6.2. Query-Level Active Learning. Query-level sampling chooses all documents related to the selected query. In this subsection, we compare the proposed RSS-Q algorithm with the other three query-level data selection algorithms.

(1) ELO-DCG-Q [Long et al. 2010]: The query-level ELO-DCG algorithm that aims at selecting the queries with the largest expected DCG loss.
(2) QBC-Q [Cai et al. 2011]: The QBC algorithm for active query selection that chooses the queries that have the largest inconsistency of rankings predicted by committee members.
(3) RAND-Q: The baseline random query selection.

Figure 12 presents the results of the four query-level active learning algorithms on the LETOR 4.0 dataset and the WEB-SEARCH dataset. We observe that both RSS-Q and ELO-DCG-Q perform better than QBC-Q and RAND-Q in terms of DCG@10 and MAP. A possible reason for these results may be as follows. The ELO-DCG-Q selects the queries with the largest expected DCG loss that is directly related to the objective function DCG@10 used to evaluate the ranking function, and the RSS-Q chooses the most sensitive queries to improve the ranking model performance effectively. For the



(a) LETOR 4.0 (b) WEB-SEARCH

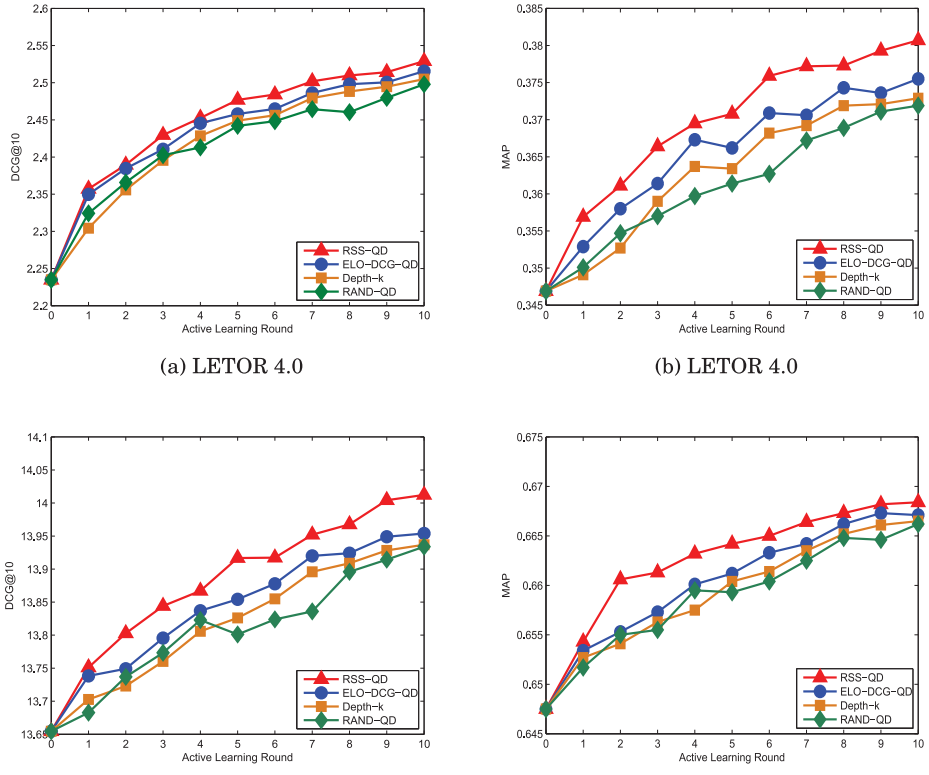
Fig. 13. The number of documents chosen by RSS-Q, ELO-DCG-Q, QBC-Q, and RAND-Q after active sampling for the LETOR 4.0 dataset and the WEB-SEARCH dataset.

comparisons between RSS-Q and ELO-DCG-Q, we see that RSS-Q performs as well as ELO-DCG-Q measured by DCG@10 during the sampling process. In terms of MAP, RSS-Q consistently outperforms ELO-DCG-Q. A possible explanation is that the expected loss optimized by ELO-DCG-Q is based on DCG, which is not directly correlated with MAP. Moreover, we observe that ELO-DCG-Q performs even worse than the weak baseline RAND-Q on the WEB-SEARCH dataset measured by MAP when the size of the training set is small. This is likely because the ELO algorithm uses the function ensemble to calculate the expected loss. When the base training data is restricted (e.g., at the early stage of the active learning process), the ELO algorithm may have very low prediction accuracy in expected loss and result in inferior performance. With the size of training set increasing, ELO starts to perform well.

The query-level active learning algorithms select all documents associated with the selected queries. Figure 13 shows the comparison of the number of documents selected by these four query-level algorithms after active sampling. We observe that all active learning algorithms tend to select queries with more documents than passive learning. A possible explanation is that active learning aims to select the most informative samples and queries with more documents are more likely to contain more information. Here, we focus on the comparison between RSS-Q and ELO-DCG-Q since their ranking performances are comparable in terms of DCG@10, and ELO-DCG-Q is observed to select more documents than RSS-Q (e.g., more than 3,000 documents on the LETOR 4.0 dataset). In terms of MAP, RSS-Q performs even better than ELO-DCG-Q while RSS-Q selects less documents than ELO-DCG-Q. Therefore, RSS-Q can reduce more labeling cost than ELO-DCG-Q in practice.

6.6.3. Two-Stage Active Learning. The two-stage active learning framework first selects the most informative queries at the query level and then selects the most informative documents related to the selected queries at the document level. In this subsection, we compare our proposed two-stage RSS active learning algorithm (denoted as RSS-QD) against the following three algorithms. As mentioned before, we simply fix the number of documents selected for each query to be 10 for all algorithms.

(1) ELO-DCG-QD [Long et al. 2010]: The two-stage ELO-DCG algorithm first selects the queries at the query level and then selects the documents related to the selected queries.



(a) LETOR 4.0 (b) LETOR 4.0

(c) WEB-SEARCH (d) WEB-SEARCH

Fig. 14. Comparison results of RSS-QD, ELO-DCG-QD, Depth-k, and RAND-QD for the two-stage active learning. Two datasets are used for the comparison: LETOR 4.0 and WEB-SEARCH. The evaluation metrics used are DCG@10 and MAP.

(2) Depth-k [Aslam et al. 2009]: The Depth-k retrieval approach, which first randomly selects queries and then selects the top k relevant documents for each query, is widely used in practical search engines. In our experimental setting, this approach corresponds to random query selection followed by selecting k documents with the highest predicted relevance scores within each selected query.
(3) RAND-QD: The two stage random selection can be viewed as a variant of Depth-k, that is, random query selection followed by random document selection for each query.

Figure 14 presents the comparison results of the four two-stage sampling algorithms on the LETOR 4.0 dataset and the WEB-SEARCH dataset. We observe that among the four sampling methods, RSS-QD achieves the highest DCG@10 and MAP scores, and ELO-DCG-QD performs the second best, demonstrating that both the RSS-QD and ELO-DCG-QD could select more informative queries and more informative documents than Depth-k and RAND-QD. Moreover, Depth-k performs better than RAND-QD in general. This is likely because Depth-k tends to select more relevant examples than RAND-QD, and those examples are more helpful in improving the ranking performance than irrelevant examples.

6.6.4. Significance Test. To better test the effectiveness of the proposed algorithms, we conduct the significance test on the comparisons. Table IV reports the results of the

Table IV. Win% of RSS Versus the Other Strategies in One-Tailed Paired T-Test at 95% Significance Level ($p < 0.05$)

	LETOR 4.0		WEB-SEARCH		
Paired t-Test	DCG@10	MAP	DCG@10	MAP	Sampling Level
RSS-D vs. ELO-DCG-D	70%	70%	90%	60%	Document level
RSS-D vs. SS	80%	100%	100%	90%	
RSS-D vs. QBC-D	100%	100%	100%	100%	
RSS-D vs. RAND-D	100%	100%	100%	100%	
RSS-Q vs. ELO-DCG-Q	0%	10%	10%	30%	Query level
RSS-Q vs. QBC-Q	20%	50%	40%	30%	
RSS-Q vs. RAND-Q	70%	90%	70%	60%	
RSS-QD vs. ELO-DCG-QD	10%	30%	70%	40%	Two stage
RSS-QD vs. Depth-k	40%	60%	90%	90%	
RSS-QD vs. RAND-QD	80%	80%	100%	90%	

one-tailed paired t-test of RSS versus the competitors at different sampling levels on both of the two datasets. We compare the DCG@10 and MAP over 10 runs at each evaluation point and present the percentage of evaluation points at which RSS statistically outperforms the competitors, denoted as Win%. The results show that RSS performs significantly better than the compared algorithms in most cases.

7. EXTENSIONS

The proposed noise-injection-based active learning strategy is flexible and can be generalized to a wide range of learners. This section discusses a number of possible extensions to show its applicability to other base learners.

In the following, we first generalize the noise injection technique for sample selection to RankSVM, a classical pairwise ranking approach. Then, we apply it to classification tasks and accordingly derive a new active learning algorithm for classification tree.

7.1. Noise Injection for RankSVM

Following previous work on RankSVM-oriented active learning [Yu 2005], we are interested in document retrieval here, that is, document-level active learning.

Given the training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where x_i is a data point (document) and y_i is an integer indicating the relevance of x_i to its related query, the goal of RankSVM is to learn a ranking function $f(x) = \langle w, x \rangle$ from partial orders that satisfies

$$\begin{aligned} x_i \succ x_j &\Leftrightarrow f(x_i) > f(x_j) \\ &\Leftrightarrow \langle w, x_i \rangle > \langle w, x_j \rangle \\ &\Leftrightarrow \langle w, x_i - x_j \rangle > 0, \end{aligned} \tag{32}$$

where $x_i \succ x_j$ denotes that x_i is ranked higher than x_j. Constructing the RankSVM model is to solve the following Quadratic Programming (QP) problem:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}||w||^2 + C \sum \xi_{ij} \\ \text{s.t.} \quad & \langle w, x_i - x_j \rangle \geq 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0, \qquad \forall i, j, \end{aligned} \tag{33}$$

where ξ_{ij} is a slack variable.

Here, we generalize the noise injection technique to RankSVM, which is employed as the base ranking function. Considering that the decision boundary is constructed with the pairwise orderings that are generated from $\mathcal{D}$, we accordingly add the noise

on the data pairs, that is, $(x_i, x_j), \forall i, j$, which can be formulated as

$$(x_i - x_j)^k = (x_i - x_j) + \epsilon^k, \quad k = 1, 2, \ldots m. \tag{34}$$

Because the pairwise approach is a classification, the document pair crossing from one decision region to another will receive a different class label. In this study, we use a simple and natural function to define the sensitivity w.r.t. a given pair:

$$\mathrm{S}(x_i, x_j) = \sum_{k=1}^{m} \mathbf{1}\{\mathrm{sign}\langle w, (x_i - x_j)^k\rangle \neq \mathrm{sign}\langle w, x_i - x_j\rangle\}, \tag{35}$$

where $\mathbf{1}(.)$ is the indicator function.

The intuitive explanation for Equation (35) is similar to earlier. If the document pair is close to the decision boundary, its related noisy copies are more likely to cross the boundary, resulting in a large value of sensitivity. On the contrary, the sensitivity score will be small or even 0 for the example pair staying far away from the decision boundary. Thus, the pair-based sensitivity sampling (PSS), which aims to choose the close-to-boundary data pair, can be expressed as

$$(x_i, x_j)^*_{\mathrm{PSS}} = \arg\max_{x_i, x_j \in \mathcal{U}} \sum_{k=1}^{m} \mathbf{1}\{\mathrm{sign}\langle w, (x_i - x_j)^k\rangle \neq \mathrm{sign}\langle w, x_i - x_j\rangle\}, \tag{36}$$

where $(x_i, x_j)^*_{\mathrm{PSS}}$ stands for the selected data pair.

Discussion. Here we discuss several related issues w.r.t. the proposed PSS algorithm.

(1) For SVM models, the distance between the data point x and the decision boundary $\langle w, x\rangle = 0$ can be directly computed as $\mathrm{Dist}(w, x) = |\langle w, x\rangle|/||w||$. One of the most widely used active learning algorithms for SVM is called simple margin, which aims to choose the examples that are closest to the boundary [Tong and Koller 2001].

(2) Similarly, for RankSVM, we can directly calculate the distance from the data pair (x_i, x_j) to the boundary as $\mathrm{Dist}(w, x_i, x_j) = |\langle w, x_i - x_j\rangle|/||w||$. In this sense, the noise injection technique aiming to identify the close-to-boundary data pairs could be regarded as a variant of simple margin in the context of RankSVM.

(3) As discussed in Section 2, a classical active learning targeting on RankSVM as the base ranking model (denoted as SEL) is to choose the document pairs with similar predicted relevance scores [Yu 2005]:

$$(x_i, x_j)^*_{\mathrm{SEL}} = \arg\min_{x_i, x_j \in \mathcal{U}} |\langle w, x_i\rangle - \langle w, x_j\rangle|. \tag{37}$$

On the other hand, the proposed PSS algorithm can be treated as an approximation of choosing the close-to-boundary pairs, and we formulate this as

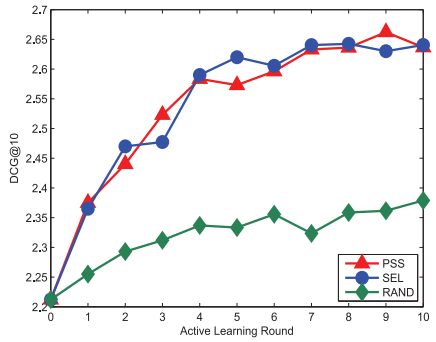
$$(x_i, x_j)^*_{\mathrm{PSS}} \approx \arg\min_{x_i, x_j \in \mathcal{U}} \frac{|\langle w, x_i - x_j\rangle|}{||w||} = \arg\min_{x_i, x_j \in \mathcal{U}} |\langle w, x_i - x_j\rangle|. \tag{38}$$

Putting together Equations (37) and (38), PSS can be viewed as an approximation of SEL:

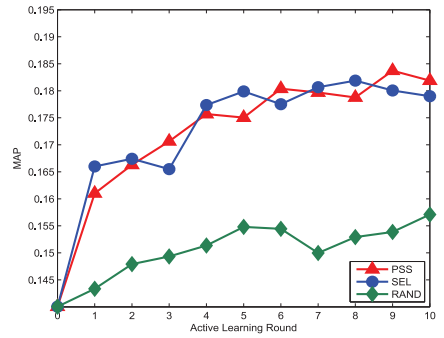
$$(x_i, x_j)^*_{\mathrm{PSS}} \approx (x_i, x_j)^*_{\mathrm{SEL}}. \tag{39}$$

For evaluation, the SVM$^{\mathrm{rank}}$ is used for training the base ranker.[3] Figure 15 presents the learning curves with different document selection methods on the LETOR 4.0 and

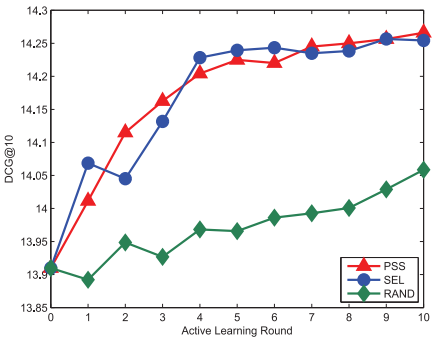
[3]http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html.



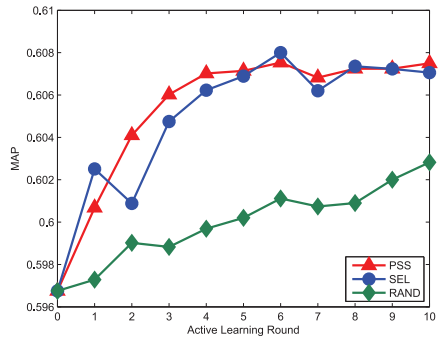
(a) LETOR 4.0



(b) LETOR 4.0



(c) WEB-SEARCH



(d) WEB-SEARCH

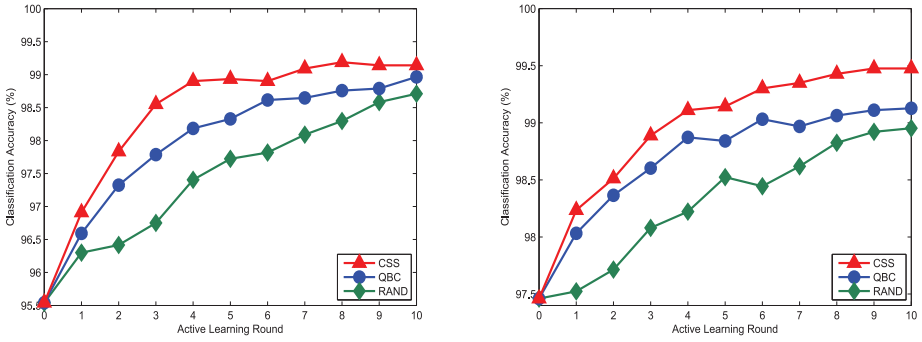
Fig. 15. Comparison results of PSS, SEL, and RAND at the document level with RankSVM. Two datasets are used for the comparison: LETOR 4.0 and WEB-SEARCH. The evaluation metrics used are DCG@10 and MAP.

WEB-SEARCH datasets. As shown in the figure, the performance of the proposed PSS algorithm is comparable to SEL, which is specifically designed for RankSVM. This is because PSS is actually an approximation of SEL as we discussed earlier. In addition, PSS significantly outperforms RAND on both of the two datasets, indicating the effectiveness of our algorithm.

For the listwise learning-to-rank approaches, applying noise-injection-based active learning to them is more complex. First,the constructed decision boundary is complicated. Take SVM-MAP for example [Yue et al. 2007]: the decision boundary is built in a transformed space using combined features, which are mapped from query–document pairs and their relevance judgments, making the geometric property of noise injection incomprehensible. Another issue is that the listwise approach takes ranked lists of documents (all possible queries) as the training instances. Hence, it has the potential to directly apply the query-level data sampling, but the document-level sampling needs to be thoroughly justified. To this end, we need to carefully design the noise injection technique and leave this as our future work.

7.2. Noise Injection for Classification Tree

Compared to the regression tree model, where its predicted score is a continuous value, the output of a classification tree is discrete. Here, we focus on the binary classification



(a) M-vs-N (b) U-vs-V

Fig. 16. Comparison results of CSS, QBC, and RAND for letter recognition. The evaluation metric used is classification accuracy.

problem; that is, the class labels are represented as {0, 1}. It can be easily generalized to multiclass problems.

Noise injection for classification tree proceeds in a similar fashion to that shown in Equation (7). Given the noisy copies generated with Gaussian noise, the sensitivity with respect to a given unlabeled example x is defined with the function that follows, which is similar to Equation (35):

$$\mathrm{S}(x) = \sum_{k=1}^{m} \mathbf{1}\{f(x^k) \neq f(x)\}, \tag{40}$$

where $f(.)$ denotes the classification tree model learned with the current training set. As a consequence, the classification-based sensitivity sampling (CSS) can be formulated as

$$x^*_{\mathrm{CSS}} = \arg\max_{x \in \mathcal{U}} \sum_{k=1}^{m} \mathbf{1}\{f(x^k) \neq f(x)\}, \tag{41}$$

which is derived in a similar manner as shown in Equation (36).

For evaluation, we test the CSS algorithm with application to letter recognition from the UCI benchmark dataset.[4] We select two pairs of letters (M-vs.-N and U-vs.-V) that are relatively difficult to distinguish and construct a binary-class dataset for each pair. The classification tree is employed as the base learner, and the performance metric is classification accuracy. Figure 16 shows the comparison results. We observe that the CSS algorithm performs better than the other two competitors, demonstrating the effectiveness of our method.

8. CONCLUSIONS AND FUTURE DIRECTION

In this article, we propose a novel active learning for ranking strategy called ranking-based sensitivity sampling (RSS), which is tailored for Gradient Boosting Decision Tree (GBDT). The strategy relies on noise injection to perturb the original data examples and selects the examples with the largest sensitivity in the predicted scores. Moreover, we theoretically analyze the proposed strategy by exploring the connection between the sensitivity used for sample selection and model regularization to provide a potentially theoretical support. Since the performance metrics of ranking are based on the resulted

[4]http://archive.ics.uci.edu/ml/.

ranking list focusing on the top-ranked documents rather than the actual value of predicted ranking scores, the RSS approach transforms the score distribution to the rank distribution and then uses the expected variation of a DCG-like gain to measure the sensitivity of each example. Considering the query–document structure in web search ranking, the proposed RSS strategy is applied at both the query level and the document level, and the two-stage algorithm is further derived. Experimental results on both the LETOR 4.0 dataset and a real-world web search ranking dataset have demonstrated that the proposed active learning algorithms can achieve better performance than the state-of-the-art methods.

The proposed noise-injection-based active learning strategy is flexible, which can be further extended to other base models. We generalize it to RankSVM and classification tree. Empirical studies are performed on several benchmark datasets, and the results show the effectiveness of our proposed algorithms.

The proposed algorithm performs active learning in batch mode, that is, selecting top-k informative data examples in each sampling iteration. The correlation or similarity among the selected examples at each batch is not considered. A possible extension of this work is to incorporate the diversity of the selected dataset to further minimize labeling cost.

ACKNOWLEDGMENTS

This research was supported by the High Technology Research and Development Program of China (2012AA011702), STCSM # 14511107500, and National Natural Science Foundation of China (No. 61003107 & No. 61221001).

REFERENCES

N. Abe and H. Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*. 1–10.

N. Ailon. 2011. Active learning ranking from pairwise preferences with almost optimal query complexity. In *Advances in Neural Information Processing Systems (NIPS'11)*. 810–818.

J. A. Aslam, E. Kanoulas, V. Pavlu, S. Savev, and E. Yilmaz. 2009. Document selection methodologies for efficient and effective learning-to-rank. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 468–475.

M. Bilgic and P. N. Bennett. 2012. Active query selection for learning rankers. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12)*.

C. Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural Computation* (1995), 108–116.

P. Cai, W. Gao, A. Zhou, and K. F. Wong. 2011. Relevant knowledge helps in choosing right teacher: Active query selection for ranking adaptation. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11)*. 115–124.

W. Cai and Y. Zhang. 2012. Variance maximization via noise injection for active sampling in learning to rank. In *Proceedings of the 21st Conference on Information and Knowledge Management (CIKM'12)*. 1809–1813.

Y. B. Cao, J. Xu, T. Y. Liu, and H. Li. 2006. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*. 186–193.

O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. 2010. Multi-task learning for boosting with application to web search ranking. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. 1189–1198.

O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. 2011. Boosted multi-task learning. *Machine Learning* 85, 1–2 (2011), 149–173.

D. A. Chon, Z. Ghahramani, and M. I. Jordan. 1996. Active learning with statistical models. *Journal of Machine Learning Research* (1996), 129–145.

D. Cossock and T. Zhang. 2006. Subset ranking using regression. In *Proceedings of the 16th International Conference on Learning Theory (COLT'06)*. 605–619.

P. Donmez and J. G. Carbonell. 2008. Optimizing estimated loss reduction for active sampling in rank learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 248–255.

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

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning* 28, 2–3 (1997), 133–168.

J. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning*. Springer.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*. MIT Press.

K. Jarvelin and J. Kekalainen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*. 41–48.

D. D. Lewis and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*. 3–12.

T. Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. 2007. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*.

B. Long, O. Chappelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng. 2010. Active learning for ranking through expected loss optimization. In *Proceedings of the 33th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*. 267–274.

K. Matsuoka. 1992. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics* 22, 3 (1992), 436–440.

H. Nguyen and A. Smeulders. 2004. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*. 623–630.

B. Qian, X. Wang, J. Wang, H. Li, N. Cao, W. Zhi, and I. Davisdon. 2013. Fast pairwise query selection for large-scale active learning to rank. In *Proceedings of the 13th International Conference on Data Mining (ICDM'13)*. 607–616.

N. Roy and A. McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*. 441–448.

B. Settles. 2012. *Active Learning*. Morgan & Claypool.

K. Shen, J. Wu, Y. Zhang, Y. Han, X. Yang, L. Song, and X. Gu. 2013. Reorder users tweets. *ACM Transactions on Intelligent Systems and Technology* 4, 1 (2013), Article No. 6.

R. Silva, M. A. Gonçalves, and A. Veloso. 2011. Rule-based active sampling for learning to rank. In *Proceedings of European Conference on Machine Learning and Principles and Practise of Knowlege Discovery in Databases (ECML-PKDD'11)*. 240–255.

M. Taylor, J. Guiver, S. Robertson, and T. Minka. 2008. SoftRank: Optimizing non-smooth rank metrics. *Proceedings of the 1st ACM International Conference on Web Search and Data Mining (WSDM'08)*. 77–86.

S. Tong and D. Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2 (2001), 45–66.

G. Valentini and T. G. Dietterich. 2004. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. *Journal of Machine Learning Research* 5 (2004), 725–775.

F. Xia, T. Y. Liu, J. Wang, W. Zhang, and H. Li. 2008. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 1192–1199.

L. Yang, L. Wang, B. Geng, and X. Hua. 2009. Query sampling for ranking learning in web search. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 754–755.

E. Yilmaz and S. Robertson. 2009. Deep versus shallow judgments in learning to rank. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 662–663.

H. Yu. 2005. SVM selective sampling for ranking with application to data retrieval. In *Proceedings of the 11st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)*. 354–363.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 271–278.

J. Zhu, H. Wang, E. Hovy, and M. Ma. 2010a. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing* 6, 3 (2010), Article No. 3, 1–24.

J. Zhu, H. Wang, B. Tsou, and M. Ma. 2010b. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech and Language Processing* 18, 6 (2010), 1323–1331.

Received January 2014; revised October 2014; accepted October 2014