REQUEST: A Scalable Framework for Interactive Construction of Exploratory Queries

Xiaoyu Ge[1], Yanbing Xue[1], Zhipeng Luo[1], Mohamed A. Sharaf[2] and Panos K. Chrysanthis[1]
[1] Department of Computer Science, University of Pittsburgh
[2] School of Information Technology and Electrical Engineering, University of Queensland
[1] {xiaoyu, yanbing, zpluo, panos}@cs.pitt.edu, [2] m.sharaf@uq.edu.au

***Abstract*—Exploration over large datasets is a key first step in data analysis, as users may be unfamiliar with the underlying database schema and unable to construct precise queries that represent their interests. Such data exploration task usually involves executing numerous ad-hoc queries, which requires a considerable amount of time and human effort. In this paper, we present REQUEST, a novel framework that is designed to minimize the human effort and enable both effective and efficient data exploration. REQUEST supports the query-from-examples style of data exploration by integrating two key components: 1) Data Reduction, and 2) Query Selection. As instances of the REQUEST framework, we propose several highly scalable schemes, which employ active learning techniques and provide different levels of efficiency and effectiveness as guided by the user's preferences. Our results, on real-world datasets from Sloan Digital Sky Survey, show that our schemes on average require 1-2 orders of magnitude fewer feedback questions than the random baseline, and 3-16× fewer questions than the state-of-the-art, while maintaining interactive response time. Moreover, our schemes are able to construct, with high accuracy, queries that are often undetectable by current techniques.**

I. Introduction

In recent years, the amount of data has been increasing tremendously. This raises the challenge for data analysts to efficiently and effectively explore large volumes of data for valuable insights. As traditional DBMSs are designed to answer well formulated and precise queries, they are rather inappropriate for exploratory discovery-oriented applications, in which queries are typically unknown in advance (e.g., [12], [19]). Particularly, in a data-driven analysis, there is a need to perform data exploration tasks, in which the user often faces two non-trivial and typically intertwined problems: 1) users are unfamiliar with the underlying database schema, and 2) users are unable to construct precise queries that represent their interests.

Motivated by the need to address the problems outlined above, recent research efforts have been directed towards designing data exploration techniques that aim to assist users in formulating and constructing their respective exploratory queries (e.g., [3], [9], [13], [21], [5], [17], [6], [8]). Examples of such techniques include *query recommendation* (e.g., [3], [9]), *query refinement* (e.g., [13], [21]), and *query-from-examples* (e.g., [5], [17], [6]). Among those techniques, query-from-examples is rapidly becoming an attractive choice for query formulation, especially for non-expert users, as it relies solely on simple forms of feedback to be provided by those users. This is opposed to query recommendation techniques which require intensive query logs and user profiles that are often unavailable when users are exploring datasets for the first time. Similarly, query refinement techniques require the user to provide some initial imprecise queries to be progressively refined into a more certain one, a process which clearly requires users to possess some good understanding of the underlying database schema as well as the ability to formulate meaningful queries.

The main idea underlying the query-from-examples techniques, which is the focus of this work, is to automatically predict and construct the user's input queries based on their feedback on a small set of sample objects [5], [17]. In particular, the user is iteratively presented by sample tuples (i.e., objects) from the database, and in turn the data exploration platform progressively constructs and refines their exploratory query. Clearly, this is an example of a "human-in-the-loop" task, in which the typical setting is to start asking the user "If you are interested in some object or not?", followed by repeatedly refining the questions until all (or most) interesting objects are discovered. For instance, in AIDE [5], [6], the user is prompted to label a set of strategically collected sample objects as relevant or irrelevant to her exploration task. Based on her feedback, AIDE generates a predictive model which is used to collect a new set of sample objects. These new samples are presented to the user and her relevance feedback is incorporated into the model. In the background, AIDE leverages the predictive model to automatically generate queries that retrieve more objects relevant to the user's task.

Clearly, the effectiveness of query-from-examples techniques, such as AIDE, relies on achieving two main objectives: 1) *maximizing the accuracy* of the employed model in predicting the users' exploratory queries, and 2) *minimizing the number of samples* that are presented to the users for their feedback. However, such two objectives are often in conflict! For instance, maximizing the accuracy of a predictive model often requires a large number of labeled samples, whereas relying on a small set of labeled samples results in a low-accuracy predictive model, and in turn imprecise query formulation that falls short in capturing the user's interests.

Consequently, in this work we propose a novel framework for query-from-examples called *REQUEST* (Data **Re**duction and **Que**ry **S**elec**t**ion), designed to address the conflicting objectives outlined above to enable both effective and efficient query-from-examples data exploration. In particular, REQUEST aims to assist users in constructing highly accurate exploratory queries, while at the same time minimizing the number of samples presented to them for labeling. To achieve that goal, REQUEST integrates the following two components:

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

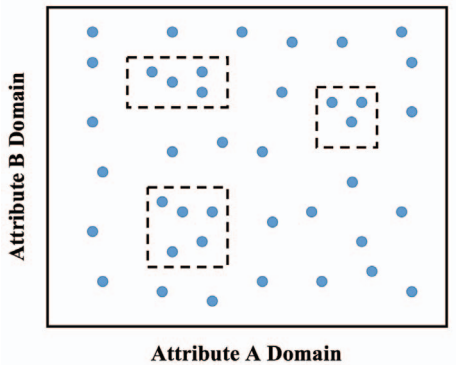

Fig. 1: 2-D Data Space.

1) *Data Reduction*: it employs efficient data reduction techniques to minimize the amount of explored data, and in turn the number of required labeled samples (e.g., [20], [11]), and 2) *Query Selection*: it employs effective active learning query selection methods to maximize the accuracy of the predictive model, and in turn the preciseness of the constructed queries (e.g., [23], [24], [22]).

Given our REQUEST general framework (Section III), we further propose several specific schemes that provide different levels of efficiency and effectiveness as guided by the user's preferences (Section IV). Our experimental evaluation on the SDSS dataset show that our schemes, can achieve the same accuracy as state-of-the-art schemes while reducing the user's efforts incurred in providing feedback (i.e., labeling samples) by up to 93% (Section V). Moreover, our schemes are also able to construct, with high accuracy, queries that are often undetectable by current techniques, even when large number of samples are explored.

II. Problem Definition & Background

Data analysts explore large volumes of data looking for valuable insights, which is typically an ad-hoc and labor-intensive process. Hence, automated data exploration solutions have emerged to effectively guide users through that challenging process. Examples of such solutions include data visualization (e.g., [27], [8]), data summarization (e.g., [7], [10]), and query formulation (e.g., [7], [14]). Our proposed REQUEST framework addresses the query formulation aspect of data exploration by means of employing a query-from-examples (e.g., [5], [17], [6]) approach to the query construction problem, which is described in the next sections.

A. Data Exploration Task

An example of the data exploration task handled by REQUEST is shown in Figure 1. The figure shows the data objects (i.e., tuples) in 2-dimensional database, where each data object is represented by a 2-attribute data point. Meanwhile, the dashed rectangles represent regions of the data space that are of special interest to the user's data exploration task. While those interesting regions could be formulated in the form of a selection query, the predicate values that define the ranges of such query only becomes clear in "hindsight" after spending long time exploring the underlying database. Consequently, the goal of query-from-examples techniques, including REQUEST, is to automatically and quickly formulate that query with high precision. That is, to define the predicates of a range selection query that is able to select all the objects which are of interest to the user.

To further illustrate this, lets consider a simple data exploration task in which a user is exploring a dataset to find and compare a few laptops that match his/her needs. However, the user is unfamiliar with the different specifications (i.e., dimensions/attributes) of laptops or what the range of values is for each attribute (i.e., predicates) that would be suitable for his/her needs. REQUEST will start the exploration task by presenting the user with some sample database objects (i.e., laptops), and ask the user to label them as relevant or irrelevant based on their attribute values (e.g., price, processor speed, main memory capacity, weight, etc.). REQUEST will then iteratively learn the user's interest and use all labeled samples as training data to build a predictive model to characterize and formulate the user's range selection query.

In each iteration, REQUEST will extract more samples and present them to the user for feedback. Any newly labeled objects will be incorporated with the already labeled sample together to form a new training dataset to update the predictive model. This exploration process is finished when the user reaches a set of satisfactory relevant objects or when the user does not wish to spend any more effort. Finally, the predictive model is transformed into a selection query, in which the predicate values that define each range are fully defined.

From the discussion above, it should be clear that the choice of the predictive model is a critical decision in the design of REQUEST. Similar to AIDE [5], in REQUEST we adopt a *decision tree* classifier [15] as the predictive model of choice. A decision tree classifier is well known for being easily trained using a labeled dataset to construct a model that is able to classify unlabeled data objects with high accuracy. Furthermore, a decision tree has the additional advantage of being easily transformed into a selection query. Particularly, once a decision tree is trained, it can be seen as a tree-like graph starting from a root node, which splits data to different branches/nodes according to some attribute value of the data being tested. A decision tree recursively splits data to sub-branches until a leaf node is reached, where each leaf node assigns a label to the classified labeled data. This enables a straightforward mapping from decision trees to range queries since the path to each leaf node can be interpreted as one region (or a hyper-rectangle) that represents a set of predicates which capture a region of the data space of interest to the user. An example of such range query can be: *SELECT * FROM laptops WHERE price* $>$ \$500 *AND price* $\leq$ \$860 *AND screen* ≥ 11" *AND screen* < 14".

B. Problem Settings

Given the data exploration task described in the previous section, we describe the query-from-examples problem addressed in this work as follows. Consider a d-dimensional database D of size N tuples. Further, consider L, which is a small subset of the database D of size n tuples, where $n << N$ and each tuple in L is user-labeled as relevant or irrelevant. Accordingly, the goal is to formulate a range

selection query Q, which accurately captures the regions of the data space that are of interest to the user. That is, all tuples selected by Q are relevant to the user.

As mentioned earlier, the user is actually unaware of the proper formulation of Q and can only recognize it in hindsight based on the tuples returned from executing Q. Hence, the objective is to predict the unknown query Q with high accuracy, which is naturally measured using the *F-measurement*, the harmonic mean between precision and recall. Particularly, for a database D of size N and a predicted query Q, all the tuples in the database D selected by the query Q are labeled positive, while all the remaining tuples in D are labeled negative. Accordingly, our goal is to maximize the F-measure provided by Q, which is defined as:

$$F(N) = \frac{2 \cdot \text{Precision}(N) \cdot \text{Recall}(N)}{\text{Precision}(N) + \text{Recall}(N)} \tag{1}$$

Here, precision measures the portion of true relevant tuples among all the tuples predicted as relevant (i.e., all the tuples selected by Q). Hence, true relevant, or true positive, indicates that a tuple is both: relevant to the user and has been selected by query Q. If a tuple is irrelevant but selected by Q, it is considered as false positive. Recall measures the ratio between the true relevant tuples selected by Q to all the tuples which are actually relevant to the user. The precision value of 1.0 is the gold standard, indicating that every tuple selected by Q is indeed relevant, while a good recall ensures that Q can select a good percentage of the tuples of interest to the user.

C. Active Learning

The query by example data exploration task can be viewed as a special application of *Active Learning* [23]. In the following, we explain the reasons why the related active learning approaches cannot be applied directly to our problem.

Active Learning is an interactive learning framework to achieve accurate classification with minimum human supervision. Particularly, a typical active learning approach would employ a *query selection* method to sequentially select which unlabeled example (i.e., object) in the database should be presented to the user next for labeling. A query selection method attempts to minimize the labeling costs by selecting the most informative examples for building the classification model. Different query selection methods to define the "informativeness" of examples have been proposed in the literature, two of which are *Uncertainty sampling* [16] and *query-by-committee (QBC)* [26].

Uncertainty Sampling: Uncertainty sampling is arguably the most popular query selection method employed by active learning approaches (e.g., [23], [16], [24]). The intuition underlying uncertainty sampling is that patterns with high uncertainty are hard to classify, and thus if the labels of those patterns are obtained, they can boost the accuracy of the classification models. Particularly, in binary classification models (e.g., with class labels 0 and 1), the most uncertain example $\mathbf{x}$ is the one which can be assigned to either class label $z(\mathbf{x})$ with probability 0.5.

Inspired by such idea of uncertainty, also known as least confidence, [16] proposes a measurement of uncertainty for binary classification models:

$$u^{(lc)}(\mathbf{x}) = 1 - p(\hat{y}|\mathbf{x}) \tag{2}$$

where $u^{(lc)}(\mathbf{x})$ is the uncertainty score with least confidence measurement of $\mathbf{x}$. $\hat{y}$ means the predicted class label of the unlabeled $\mathbf{x}$. Accordingly, after measuring the uncertainty of each unlabeled sample, the unlabeled sample with highest uncertainty is selected:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} u(\mathbf{x}) \tag{3}$$

where $u(\mathbf{x})$ can be any other measurement of informativeness over the unlabeled sample $\mathbf{x}$.

Query-by-committee: Another widely used query selection method is the well known *query-by-committee (QBC)* ([26], [22],[2]). QBC is basically a voting strategy, in which the uncertainty score is calculated based on a committee of multiple classification models, where each model is trained over a subset of the labeled set. The prediction of an unlabeled point under this voted uncertainty is calculated as:

$$p(1|\mathbf{x}) = \frac{\sum_{i \in C} \hat{y}^{(i)}}{|C|} \tag{4}$$

where $p(1|\mathbf{x})$ indicates the probabilistic prediction that $\mathbf{x}$ belongs to class 1, C indicates the committee, $|C|$ indicates the size of the committee and $\hat{y}^{(i)}$ indicates the prediction of $\mathbf{x}$ using the ith model in the committee. In this case, the uncertainty score is calculated as:

$$u^{(v)}(\mathbf{x}) = 1 - p(\hat{y}|\mathbf{x}) \tag{5}$$

where $u^{(v)}(\mathbf{x})$ is the uncertainty score with voted measurement of $\mathbf{x}$. $\hat{y}$ indicates the predicted class label of the unlabeled sample $\mathbf{x}$.

Active Learning vs. Data Exploration: On the one hand, active learning approaches have been shown to be effective in terms of: 1) maximizing classification accuracy, and 2) minimizing number of user-labeled samples. On the other hand, applying traditional active learning approaches has the drawback of introducing long delays as it requires performing multiple iterations of exhaustive search over the database. For instance, in uncertainty sampling, presenting one example to the user for labeling requires computing the uncertainty scores for all the unlabeled objects in the database to select the one with the highest uncertainty score. This process is repeated with each example selection, resulting in long waiting time before the user is able to label the next example, which in turn renders that approach impractical for many applications that access relatively large volumes of data.

D. Automatic Interactive Data Exploration (AIDE)

To circumvent the scalability shortcomings of applying active learning approaches in data exploration settings, AIDE [5] has been proposed to enable automatic interactive data exploration. As opposed to active learning approaches, AIDE emphasizes the need for providing "interactive speed" when performing data exploration tasks, and accordingly attempts to minimize the processing time required for selecting those objects to be presented to the user for labeling. In particular, like active learning approaches, AIDE employs a classifier

model for learning the user's interest from labeled examples. However, instead of exhaustively searching all the unlabeled data objects, AIDE employs some data reduction techniques to optimize the process of example selection.

Specifically, AIDE employs a decision tree, which is refined as the user labels more examples and the resulting refined decision tree is used to guide the selection of the next set of examples to be labeled. To reduce the amount of data searched for selecting those examples, AIDE performs the following operations in each iteration after refining the decision tree: *Misclassified Exploitation*: If an interesting data object that is currently being misclassified is identified, AIDE will randomly select more objects around that misclassified one within a pre-defined area and present those additional samples to the user for labeling. *Boundary Exploitation*: AIDE also randomly selects examples from around the boundaries of each relevant region that has been identified by the decision tree. Presenting those additional examples to the user for labeling allows AIDE to tune the boundaries of those relevant regions. *Relevant Object Discovery*: For the areas of the data space that are currently considered irrelevant by the decision tree, AIDE selects some examples to further ensure their (ir)relevance. Particularly, for those areas AIDE partitions the space with d-dimensional grids, each grid defines a subspace on the original domain, then randomly picks a sample from the center of each grid cell. The grid size is adjusted with each refinement of the decision tree, where a "zoom-in" operator is performed to further split each grid cell into smaller grid cells, thus generate more samples for finer granularity search.

Given the operations described above, in each iteration AIDE will present the user with a batch of examples to label which is the union of all the examples generated by those three operations. Clearly, in comparison to active learning approaches, AIDE searches through a much smaller set of unlabeled data points in order to select the new set of examples to be labeled. However AIDE has the following limitations: 1) it relies on a heavily parameterized model for navigating the search space, which makes it rather difficult to deploy in practice, and 2) reducing the search space of unlabeled data comes at the expense of requiring the user to label more samples to achieve high accuracy.

In the next section, we present our proposed REQUEST framework, which addresses the limitations outlined in above.

III. The REQUEST Framework

REQUEST is designed to achieve the following goals: 1) minimize the number of example objects that are presented to the user for labeling, 2) minimize the processing cost and delays incurred in selecting those examples, and 3) maximize the accuracy of the constructed query in capturing the user's interests. The main idea underlying REQUEST is to combine the advantages of data reduction techniques (used in data exploration platforms) with the advantages of query selection methods (used in active learning approaches).

As shown in Algorithm 1, REQUEST works in two stages, namely: *data reduction* and *query selection*. In the first stage, REQUEST reduces the original dataset D to a subset D_s, such that $|D_s| < |D|$ (line 1). In order to achieve effective data reduction, we consider different methods including traditional sampling and user-guided sampling based on the multi-instance active learning (MIAL) approach as discussed in the next section. Then REQUEST initializes two sets of data objects L and U for storing the labeled and unlabeled data objects, respectively (line 2-3).

Algorithm 1 The REQUEST Framework

Require: The raw data set D; Batch Size B
Ensure: A range query Q
1: Subset $D_s \leftarrow D$
2: Labeled set $L \leftarrow \emptyset$
3: Unlabeled set $U \leftarrow D_s$
4: $M \leftarrow$ initialize query selection method
5: **while** user continues the exploration **do**
6: **for** $i = 1$ to B **do**
7: Choose one x from U using M
8: Solicit user's label on x
9: $L \leftarrow L \cup \{x\}$
10: $U \leftarrow U - \{x\}$
11: **end for**
12: $M \leftarrow$ trained with L to update M.
13: Train a decision tree with L to obtain Q
14: **end while**
15: Return the most recently obtained Q.

REQUEST also incorporates a query selection method that is used for selecting the example objects to be presented to the user for labeling (line 4). In this work, we employ and extend variants of the well-known uncertainty query selection methods, which are described in detail in Section III-B. As long as the user is willing to label more examples (line 5), REQUEST will keep invoking the query selection method M to select a new example object x from D_s, and present it to the user to label as relevant or irrelevant (lines 6-11). Once the amount of labeled samples received from user reaches a sample batch size (denoted as B), which is a tunable parameter of the REQUEST to balance the effectiveness and efficiency, then the classifier model employed by the query selection will be updated according to the label assigned to x (line 12).

In particular, the label assigned to x will be used for retraining the classifier model, which is an essential step towards selecting the object presented to the user in the next iteration. Once the iterative labeling process is completed, a decision tree classifier is trained on all the labeled data and a range selection query Q is constructed based on that tree, as described in Section II (line 13).

Next, we describe in detail the different methods for data reduction and query selection considered under our framework.

A. Data Reduction

In active learning approaches, query selection methods are applied over the entire set of unlabeled data objects to select the next example to be presented to the user for labeling. As discussed earlier, such exhaustive search incurs high processing costs and leads to users experiencing long delays between the presented examples. To the contrary, data exploration approaches employ data reduction techniques for minimizing the number of unlabeled objects to be searched. However, current

data reduction approaches, such as the ones employed by AIDE (see Section II) have the drawback of presenting the user with a large number of examples in order to construct a query with high accuracy. In the REQUEST framework, we consider simple yet effective data reduction methods that overcome the aforementioned limitations, namely: *Data-Driven Sampling*, and *User-Driven Pruning*.

1) Data-Driven Sampling: In traditional sampling, a small subset of data objects D_s is randomly extracted from the complete database D. As such, the query selection method for selecting examples for labeling is applied on the sample D_s to reduce the processing cost and waiting delays (see Algorithm 1). For example, if N is the size of the complete data and p is the sampling ratio, then D_s will include $p \times N$ tuples, and the remaining tuples are discarded. Note that $p = 1.0$ is a special case referred to as *None*, in which no sampling is applied and all the database tuples are considered, which is equivalent to traditional active learning approaches in which no data reduction is employed.

2) User-Driven Pruning: Inspired by the Multi-Instance Active Learning (MIAL) approach [25], we introduce another method for selectively reducing the search space of unlabeled objects. In MIAL, objects are grouped into a set of bags, and the user is requested to give their feedback on a bag of objects rather than a single object. Accordingly, MIAL assumes that a bag is irrelevant if every object in that bag is of no interest to the user; otherwise, the bag is interesting.

Such property of MIAL making it well-suited to exploration tasks for which the volume of available data objects is large but the target is represented by only a small set of data objects. In [23], it was shown with examples, that as fully labeling all data objects is expensive, it is possible to obtain labels both at the bag level and directly at the data object level, which inspires us to take advantage of coarse labelings to quickly eliminate irrelevant subspaces, then use finer query selection strategies at the object-level to polish the answer.

In REQUEST, we employ MIAL as the User-Driven Pruning (UDP) technique to prune subspaces that are completely irrelevant to the exploration task and in turn reduce the amount of data considered for labeling. To achieve this, REQUEST divides the multi-dimensional data space evenly into a number of d-dimensional hyper-rectangles. That is, each dimension of the data space is split evenly into a certain number of bins. Thus, if each dimension is split into m bins, there will be m^d hyper-rectangles in total. Then the user is presented with the ranges covered by each hyper-rectangle and is asked to label the rectangle as negative if it does not contain any relevant objects, or as positive otherwise. All negative rectangles are discarded, and all remaining positive ones are passed to the query selection method.

B. Query Selection

Query Selection is the one component of our REQUEST framework that aims to minimize the labeling effort while maximizing the accuracy of the constructed queries. It is worth mentioning that in this context, a "query" refers to the process of selecting an example object to be presented to the user for labeling. In REQUEST, we use uncertainty sampling to quickly learn the interest of the user and steer them towards the relevant data regions.

Uncertainty Sampling: As previously mentioned in Section II, uncertainty sampling is a popular active learning technique that aims to choose the data points which are most beneficial to build a classification model that precisely captures both relevant and irrelevant data regions.

According to Equation 2, to measure the uncertainty of a data object x, we need a model that would always report the probability of x being positive or negative. The decision tree classifier (that we employed for range query) is not strictly a probabilistic model, therefore in most cases, the decision tree is not an ideal choice for calculating the uncertainty score. Thus, another probabilistic classification model such as Naive Bayes Classifier is needed to determine the uncertainty score. This model will be built on the same labeled dataset as the decision tree so that they can reflect the same "knowledge" of the interest objects.

Naive Bayes Classifier: Naive Bayes classifier [18] is a classification model, which can be perceived as a mixture of multiple logistic regression models. In that sense, Naive Bayes generates similar decision boundaries to a decision tree (because multiple hyperplanes form an enclosed decision boundary similar to hyper-rectangle or hyper-ball) with probabilistic scores for calculating uncertainty. Accordingly, REQUEST trains our Naive Bayes classifier on the same set of data (all user labeled objects) at the same time it trains the decision tree classifier, therefore for a given time point T the probabilistic score given by Naive Bayes classifier reflects the uncertainty of any unlabeled data object at time T.

To be more precise, the uncertainty of a data object under Naive Bayes classifier is determined as follows: given an unlabeled data sample to be predicted, represented by $\mathbf{x}$, it assigns to this probability:

$$p(C_k|\mathbf{x}) \tag{6}$$

for each hidden class C_k. Another assumption of Naive Bayes classifier is the conditional independence: Naive Bayes classifier assumes that each feature is conditionally independent of every other feature:

$$p(\mathbf{x}_i|\mathbf{x}_{i^{\complement}}, C_k) = p(\mathbf{x}_i|, C_k) \tag{7}$$

where $\mathbf{x}_i$ is the ith feature of $\mathbf{x}$ and $i^{\complement}$ is $\mathbf{x}$ without the ith feature.

Thus, the joint probability can be expressed as:

$$\begin{aligned} &p(C_k|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \\ \propto &p(C_k, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \\ \propto &p(C_k) \prod_{i=1}^{m} p(\mathbf{x}_i) \end{aligned}$$

where i is the enumeration of all features. This means that under the above independent assumptions, the conditional distribution over the hidden variable is:

$$p(C_k|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) = \frac{1}{Z} p(C_k) \prod_{i=1}^{m} p(\mathbf{x}_i) \tag{8}$$

where $Z = p(\mathbf{x})$ is the normalization factor.

Such conditional distribution reflects the likelihood of a data object x being positive or negative (in our case relevant or irrelevant), which is essentially the uncertainty score of x.

Committee of Naive Bayes Classifiers: Following the voted uncertainty method mentioned in Section II, a small committee of Naive Bayes classifiers x can be used as an alternative way to determine the uncertainty score of an unlabeled object [22]. Under this method, the uncertainty score is calculated through Equation 5. Additionally, according to [23], [26], [24], a small committee (less than 6) work well in practice for such method.

In REQUEST, we employ both Naive Bayes Classifier and Committee of Naive Bayes Classifiers for calculating the uncertainty score. However, traditional uncertainty sampling suffers from two major drawbacks 1) *shortsightedness* (as pointed out in [4]), and 2) *low scalability*. Shortsightedness refers to the issue that the uncertainty score of unlabeled samples is based only on the information obtained from labeled samples, which usually is a tiny portion compared to the unlabeled objects, therefore, causes a biased when selecting samples for labeling. Low scalability is caused by the fact that traditional uncertainty sampling requires performing an exhaustive search over all unlabeled datasets for every sample that is presented to the user.

Randomized Uncertainty: To address the first drawback mentioned above, the work in [29] combines uncertainty with some degree of randomness. Particularly, for each query, an example is probabilistically selected from all the unlabeled objects for the user to label. The probability that an unlabeled point x is selected is proportional to its uncertainty score:

$$p(\mathbf{x} \text{ is selected}) = \frac{u(\mathbf{x})}{\sum_{\mathbf{x}_u \in U} u(\mathbf{x}_u)} \tag{9}$$

where U is the unlabeled set and $u(\mathbf{x})$ is the uncertainty score of an unlabeled sample $\mathbf{x}$.

Since the probability that an unlabeled point x is chosen to be presented to the user is equal to its normalized uncertainty score, therefore, less uncertain samples may still have a small chance of being accepted, which has the effect of reducing the bias introduced by the labeled samples.

Randomized Accept/Reject Uncertainty (RARU): When exploring large datasets, randomized uncertainty still suffers from low scalability for the fact that each example selection must go over all the unlabeled object to compute their normalized score. Thus, to address the second drawback of uncertainty sampling, we propose an accept/reject example selection approach. Particularly, to choose an unlabeled sample to be presented to the user, we first randomly pick an unlabeled object x, then calculate the uncertainty score of x. To decide whether x can be presented to the user we use the uncertainty score of x as the way to determine its acceptance, such that the probability of an unlabeled data sample $\mathbf{x}$ being accepted under RARU is:

$$p(\mathbf{x} \text{ is selected}) = \underset{k \in 0,1}{\operatorname{argmin}} \frac{Pr(C_k|\mathbf{x})}{0.5} \tag{10}$$

where $Pr(C_k|\mathbf{x})$ is the probability of $\mathbf{x}$ being assigned a binary class label C_k, and 0.5 is a normalizing factor since a prediction score of 0.5 indicates the classifier is most uncertain about an object.

If x is accepted, it will be presented to the user for labeling. Otherwise, RARU will continue to select other unlabeled objects randomly until one object is accepted (according to Equation 10) and presented to the user.

Such Randomized Accept/Reject Uncertainty strategy provided an early termination to prevent the enumeration of all unlabeled data items, while still preserving the feature of randomized uncertainty that mitigates the shortsightedness of the traditional uncertainty sampling.

IV. The REQUEST Schemes

In this section we propose four specific schemes to support the query-from-examples style of data exploration that are based on our REQUEST framework. These schemes are: 1) The None+RARU Scheme, 2) The None+VotedRARU Scheme, 3) The MIAL+RARU Scheme, and 4) The MIAL+VotedRARU Scheme.

A. The None+RARU Scheme

This scheme focuses on the Randomized Accept/Reject Uncertainty (RARU) as the query selection method with no data reduction techniques used. In None+RARU, a single Naive Bayes Classifier is employed to compute the uncertainty scores. An issue of this scheme is the initial sample acquisition, since initially no relevant objects are discovered. Therefore no uncertainty score can be computed. To solve this issue, we randomly sample unlabeled data objects from the database for labeling until the first relevant object is discovered, thus, enables Naive Bayes classifier to compute the uncertainty score of unlabeled objects. Once the first relevant object is discovered, the choice of selecting subsequent samples for labeling would be determined according to Equation 10.

B. The None+VotedRARU Scheme

The None+VotedRARU scheme is similar to the None+RARU Scheme except the way how uncertainty score is computed. In None+RARU Scheme, a single Naive Bayes classifier is used to determine whether to reject/accept a sample from being presented to the user. In the case of None+VotedRARU Scheme, this decision is co-decide by a committee of Naive Bayes classifiers. In particular, given the labeled data L, we use k-fold cross-training method to train k different classifiers and the uncertainty score of an unlabeled object is computed according to Equation 5.

C. The UDP+RARU Scheme

For this scheme, we employ the User-Driven Pruning (UDP) technique based on MIAL (see Section III) as the data reduction technique. We start with a partition of d-dimensional data space into multiple d-dimensional grids with a tunable parameter δ, such that each of the attributes is split into δ equal width ranges, and each grid covers a range of $100/\delta$ of the domain for each attribute. We then query user on the ranges covered by each grid and prune those grids that are labeled as irrelevant. For each of the relevant grids, we would visit them in a round-ribbing fashion and apply the RARU strategy (with Naive Bayes classifier) to select samples from each grid for labeling. Later, each labeled samples from any positive grids would be feed into a decision tree model to capture relevant regions.

TABLE I: PARAMETERS

Total number of data objects	10×10^6
Number of runs per result	10
Number of dimensions considered	2, 3, 4, 5
Number of relevant regions	1, 3, 5
Cardinality of the relevant regions	0.1% (S), 0.4% (M), 0.8% (L)
Data Reduction Strategies	Sampling, UDP
Number of grids	3^D (D = Number of Dimensions)
Query Selection Methods	RARU, VotedRARU
Uncertainty Sampling algorithm	Naive Beyes
Decision Tree Algorithm	J48 (C4.5)
Considered Sample Batch Sizes	50
Number of Committees	5
Baseline Schemes	AIDE, Random
Our Schemes	None+RARU, None+VotedRARU UDP+RARU, UDP+VotedRARU

D. The UDP+VotedRARU Scheme

This scheme is the same as the UDP+RARU Scheme, except that the VotedRARU strategy is used to compute the uncertainty score.

V. Experimental Evaluation

In this section, we will present the results of our experiments that compare the performance of our schemes to AIDE [5], which is the state-of-the-art (see Section II-D).

A. Experiment Setup

SDSS Dataset In our experiments we used 40 GiB of real-world dataset from Sloan Digital Sky Survey (SDSS) [1] that consists of 10×10^6 tuples.

Environment Since we are unable to obtain the original implementation of AIDE, we faithfully implemented all algorithms with Java JRE 1.7 and all the experiments were run on an Intel Core i7 4-core CPU with 24GiB RAM. We used the Weka [28] library for executing the J48 decision tree algorithm. All experiments reported are averages of 10 complete runs. We have considered five numerical attributes rowc, colc, ra, field and fieldID of the PhotoObjAll table.

Target Interest Regions The exploration task characterizes user interests and eventually predicts the interest regions by iteratively gathering user labeled tuples. As mentioned before, we focus on predicting range queries (the user interest regions) and in our experiments, we experiment with three different interest region amounts $\{1, 3, 5\}$. Further, we vary the single region complexity based on the data space coverage of the relevant regions. Specifically, we categorize relevant regions to small, medium and large. Small regions have cardinality with an average of 0.1% of the entire experimental dataset, medium regions have a cardinality of 0.4%, and large regions have a cardinality of 0.8%.

User Simulation Given a target interest region, we simulate the user by executing the corresponding range query to collect the exact target set of relevant tuples. We rely on this "oracle" set to label the tuples we extract in each iteration as relevant or irrelevant depending on whether they are included in the target region. We also use this set to evaluate the accuracy (F-measure) of our final predicted range queries. Further, we consider each sample as one question to the user. Thus, the question asked by the data reduction strategy (e.g., UDP) are also counted as one sample.

Parameters Table I shown a list of all settings and shames of the experiment. By default we use 1×10^6 distinct tuples, a batch size of 50, a committee size of 5 and attributes rowc and colc, unless otherwise specified. The words "f-measure" and "accuracy" are interchangeable in the text below.

B. Experimental Results

Accuracy Comparison Figures 2-10 shown the number of samples needed to reach an accuracy (f-measure) of 60% and 80% of all participating algorithms with different range queries. Here we very the target region size from small to large, and change the target region numbers with three different settings 1, 3 and 5. From these figures, we observed that our scheme UDP+VotedRARU consistently demonstrate the highest effectiveness when compare to other alternatives. Such that, to reach an accuracy of 60% UDP+VotedRARU only 150-200 samples for small regions, less than 80 samples for medium regions and less than 70 samples for large regions on average. To reach an accuracy of 80%, UDP+VotedRARU requires on average on 350 samples for small regions, 140 samples for medium regions, and 120 samples for large regions. Further our UDP+RARU also shown an excellent performance and is only slightly behind the UDP+VotedRARU.

Although, None+RARU and None+VotedRARU without using UDP do not achieve as effective as working with UDP. However, they still demonstrate an acceptable performance due to the efficiency of RARU, and are fully usable when facing medium and large regions, as both of they only require less than 800 samples and 370 samples to achieve an accuracy of at least 80% for medium and large regions.

AIDE also showed a good performance for medium and large regions especially for 60% of accuracy, as in most case it requires 400 samples for medium and 330 samples for large regions to achieve an accuracy of 60%. For large regions, it requires less than 360 samples to achieve an accuracy of 80%. But AIDE fails to discover really small regions as it requires 1450-3500 samples to achieve a 60% of accuracy and would require more than 5500 samples to achieve 80% accuracy for small regions. The reason for such behavior is when discovering very small regions, AIDE would generate a large amount of samples due to its costly "zoom-in" operations. Comparing AIDE to our UDP+VotedRARU, our scheme requires 9x-16x fewer samples for small target regions, 5x-10x fewer samples for medium target regions, and 3x-5x fewer samples for large target regions than AIDE.

We also compare all algorithms with Random which is a baseline algorithm that randomly (based on uniform distribution) selects samples from the exploration space, present them to the user for feedback and then builds a decision tree classifier based these samples. The result shows that Random fails to discover small regions. Even when we increase the number of samples to 9000, it still fails to reach an accuracy of 60%. Random can discover medium and large regions but with a great cost, such that it requires $2000-3000$ samples for medium regions and $1200-1800$ samples for a large region to reach an accuracy of 80%.

Runtime Comparison Figures 11-13 shown the runtime (in logarithmic scale) of each algorithm to reach an accuracy of at least 60% and 80%. In all cases, smaller target region

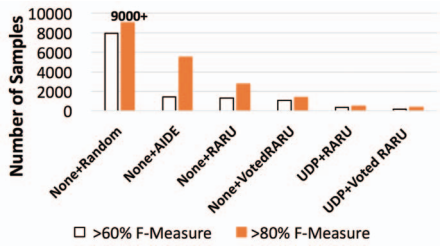

Fig. 2: Accuracy, 2D, 1 Small Region

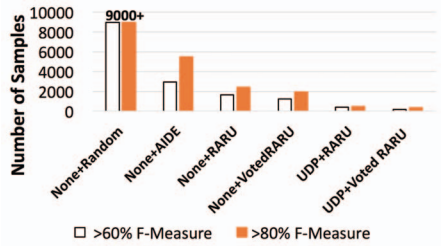

Fig. 3: Accuracy, 2D, 3 Small Regions

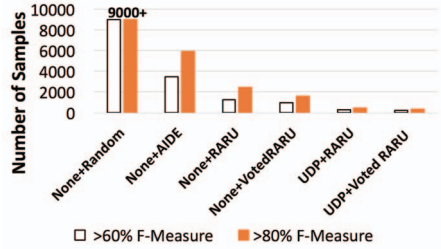

Fig. 4: Accuracy, 2D, 5 Small Regions

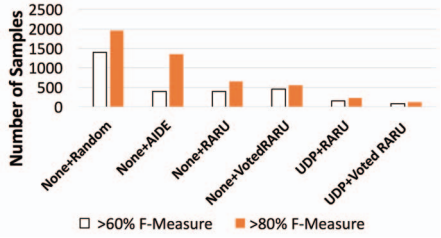

Fig. 5: Accuracy, 2D, 1 Medium Region

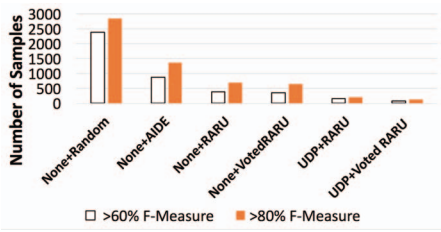

Fig. 6: Accuracy, 2D, 3 Medium Regions

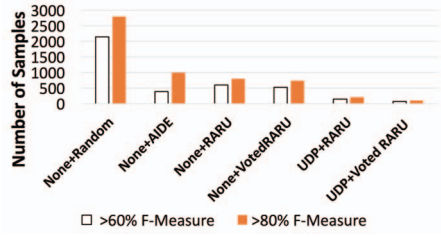

Fig. 7: Accuracy, 2D, 5 Medium Regions

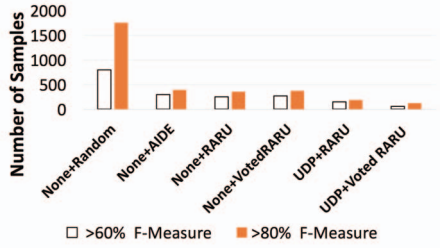

Fig. 8: Accuracy, 2D, 1 Large Region

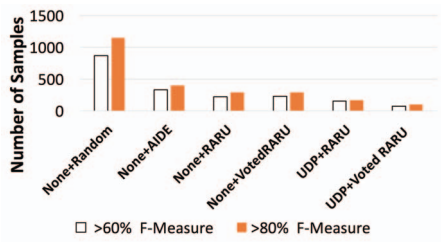

Fig. 9: Accuracy, 2D, 3 Large Regions

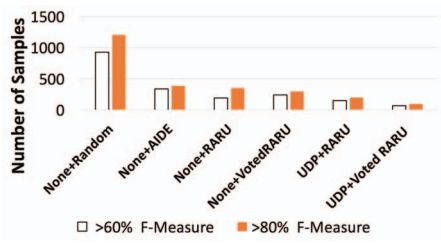

Fig. 10: Accuracy, 2D, 5 Large Regions

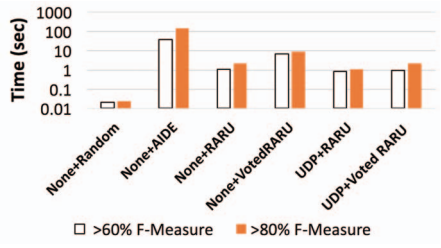

Fig. 11: Runtime, 2D, 1 Small Region

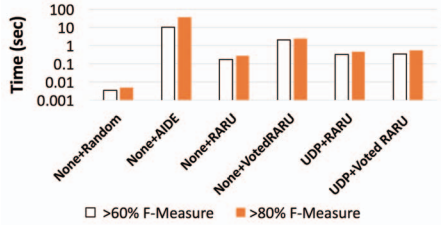

Fig. 12: Runtime, 2D, 1 Medium Region

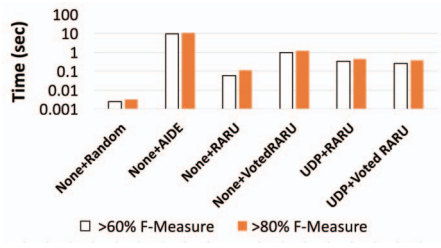

Fig. 13: Runtime, 2D, 1 Large Region

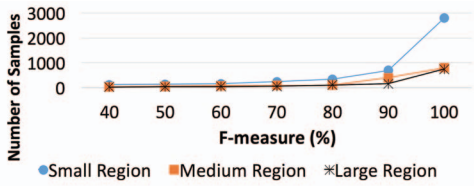

Fig. 14: UDP+VotedRARU, Increase Regions Sizes (1 Region)

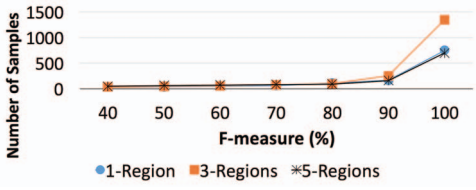

Fig. 15: UDP+VotedRARU, Increase Regions Numbers (Large Region)

would increase the runtime. Naturally, random is overall the fastest scheme as it requires almost no computation. Other than random, the runtime is acceptable for all of our schemes, such that for UDP+RARU and UDP+VotedRARU the runtime to achieve 80% of accuracy is only 1-2 seconds for small regions and less than 0.5 seconds for both medium and large region, which is expected as our schemes only generate a small number of query samples.

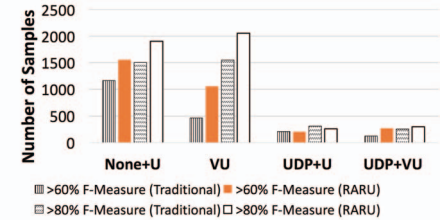

Fig. 16: Accuracy,1 Small Region, Small Dataset

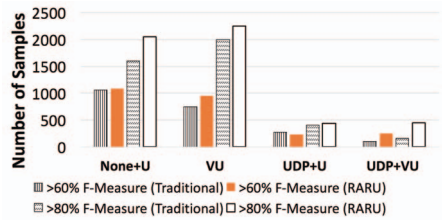

Fig. 17: Accuracy, 3 Small Regions, Small Dataset

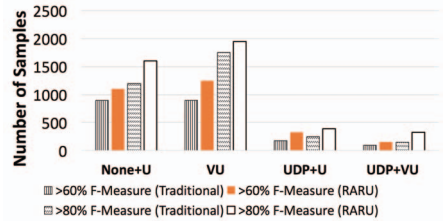

Fig. 18: Accuracy, 5 Small Regions, Small Dataset

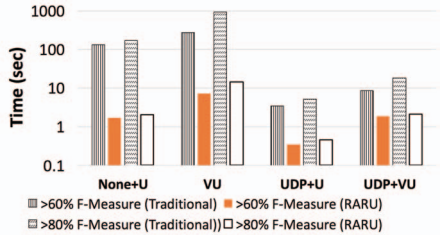

Fig. 19: Runtime, 1 Small Region, Small Dataset

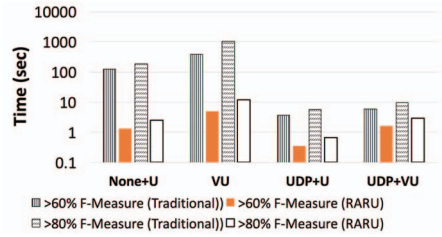

Fig. 20: Runtime, 3 Small Regions, Small Dataset

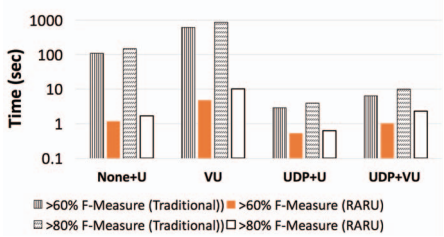

Fig. 21: Runtime, 5 Small Regions, Small Dataset

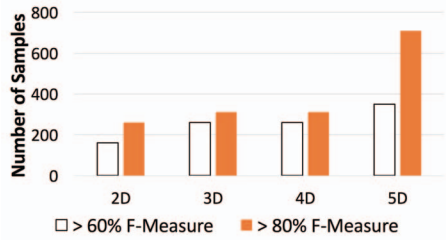

Fig. 22: F-Measurement of UDP+VotedRARU, 1 small region, different dimensions

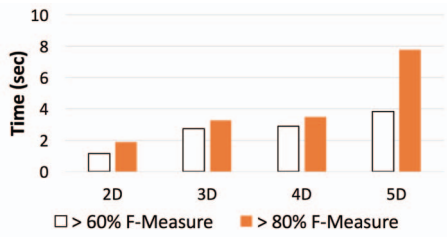

Fig. 23: Runtime of UDP+VotedRARU, 1 small region, different dimensions

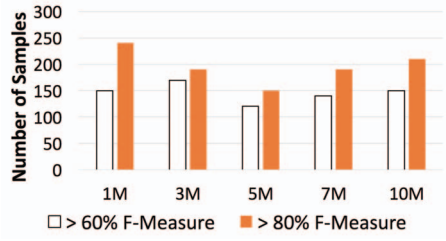

Fig. 24: F-Measurement of UDP+VotedRARU, 1 small region, different dataset sizes

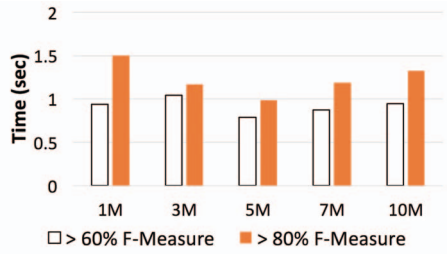

Fig. 25: Runtime of UDP+VotedRARU, 1 small region, different dataset sizes

None+RARU, None+VotedRARU and AIDE would require higher runtime than UDP+RARU and UDP+VotedRARU as they require more samples to reach 80% accuracy. But the runtime for both None+RARU and None+VotedRARU are still acceptable as they require less than 10 seconds for small and less than 1 seconds for medium and large regions to achieve 80% accuracy. Note that although we have implemented AIDE accurately and faithfully, some optimization may be missed, as the result, the runtime may be two times slower than a fully optimized version (according to the runtime reported in [5]). In all runtime experiments, we only report the real runtime that we have measured based on our implementation.

Zoom-in to the Best Scheme Figures 14-15 shown a closer look at the effectiveness of the best scheme UDP+VotedRARU as we increase the complexity of range query by varying the number of relevant regions and the size of the relevant region. We noticed that UDP+VotedRARU achieves a remarkable performance as it only requires 110 samples in medium and large regions, and 350 samples (out of 10×10^5 tuples) for small regions to reach 80% accuracy. Further, the performance of UDP+VotedRARU only decreases slightly when the number of relevant regions is increased.

Traditional Uncertainty VS. RARU We compare our RARU with the traditional Uncertainty Sampling, both strategies employ Naive Bayes Classifier to compute uncertainty. Figures 16-21 show the effectiveness and efficiency of both schemes. In these figures, we used U to denote Uncertainty and VU to denote Voted Uncertainty. These experiments are based on a small dataset that contains only 100k tuples (due to the time complexity of traditional uncertainty sampling) and small target regions. Note that, for Figures 19-21 the runtimes are in logarithmic scale. As expected, the traditional Uncertainty Sampling overall reaches the same level of accuracy with less number of samples than RARU. Such that compare to RARU; the traditional uncertainty saves up to 35% of samples when UDP is employed, and up to 50% of samples when UDP is not employed. However, traditional Uncertainty Sampling can be up to 60 times slower than RARU when UDP is employed and up to 100 times slower than RARU when UDP is not employed. As the efficiency of traditional Uncertainty Sampling is extremely low, it is still unfeasible to apply it directly on the modern database systems for real-world interactive data exploration tasks.

Impact of the Data Reduction Further, the effectiveness of data reduction techniques is demonstrated with UDP, such that apply UDP as data reduction on average requires 3×-10× fewer samples than no data reduction for both traditional Uncertainty Sampling and RARU. We also observed the same amount of reduction in runtime (to achieve 60% of accuracy) when UDP is employed, which is as expected, since both the data space and the number of samples generated are reduced.

Dimensionality Figures 22-23 demonstrate the effective and efficiency of UDP+VotedRARU as we increase the dimensionality of our exploration space from 2-D to 5-D with one small target region. Our target range query have conjunctions on two attributes. Our solution has correctly identified the two attributes that define the target region, thus, able to discard not related attributes from the decision tree to obtain the target range query. As expected, high dimensions would require more samples to reach the same accuracy as low dimensions. However, the number of samples only increased slightly from 2D to 3D and 4D. Even for 5D, the number of samples needs to achieve 60% of accuracy only increased 35% compare to 4D, and the number of samples needed for 80% is still within 700 samples even for 5-dimensional space.

Database Size Figures 24-25, illustrates the scalability of UDP+RARU as we increasing the dataset size from 1 millions to 10 millions. The result shown that our method is highly scalable as both the effectiveness and efficiency are independent of the size of the data set. This is because in RARU the time taken to generate one sample to present to the user is depended on the distribution of the uncertainty score of all objects in the dataset and is independent of the size of the dataset.

VI. Conclusion

Motivated by the challenge of reducing human effect in exploring large datasets, in this paper we proposed REQUEST, a novel framework for the query-from-examples style of data exploration. The REQUEST framework consists of two key components, namely, data reduction and query selection.

We show the applicability of REQUEST by proposing an efficient user-guided data reduction technique with Multi-Instance Active Learning (MIAL), and a novel query selection method, called Randomized Accept/Reject Uncertainty (RARU), that aims to provide high scalability. Specifically, we designed and experimented with four schemes: None+RARU, None+VotedRARU, MIAL+RARU, and MIAL+VotedRARU as solutions to the query-from-examples data exploration.

Our experimental results have shown that our proposed schemes achieve much higher performance in both effectiveness and efficiency when compared to the state-of-the-art. Further, the human efforts incurred in providing feedback was reduced by up to 93%.

References

[1] SDSS samples queries - http://cas.sdss.org/dr4/en/help/docs/realquery.asp.
[2] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *ICML*, 1998.
[3] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *WSDM*, 2010.
[4] R. J. Brachman, W. W. Cohen, and T. Dietterich. *Synthesis Lectures on Artificial Intelligence and Machine Learning*. 2012.
[5] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *SIGMOD*, 2014.
[6] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: An active learning-based approach for interactive data exploration. In *IEEE TKDE*, volume 28, pages 2842–2856, Nov 2016.
[7] M. Drosou and E. Pitoura. Disc diversity: Result diversification based on dissimilarity and coverage. In *PVLDB*, 2012.
[8] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis. Muve: Efficient multi-objective view recommendation for visual data exploration. In *ICDE*, 2016.
[9] H. Feild and J. Allan. Task-aware query recommendation. In *SIGIR*, 2013.
[10] X. Ge, P. K. Chrysanthis, and A. Labrinidis. Preferential diversity. In *ExploreDB*, 2015.
[11] D. Haas, J. Wang, E. Wu, and M. J. Franklin. Clamshell: Speeding up crowds for low-latency data labeling. In *VLDB*, 2015.
[12] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, 2015.
[13] S. Islam, C. Liu, and R. Zhou. A framework for query refinement with user feedback. In *J. Syst. Softw., 86(6):15801595*, 2013.
[14] H. A. Khan and E. A. Sharaf. Progressive diversification for column-based data exploration platforms. In *ICDE*, 2015.
[15] L.Breiman, J.H.Friedman, R.A.Olshen, and C.J.Stone. Classificationand regression trees. 1984.
[16] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, 1994.
[17] H. Li, C.-Y. Chan, and D. Maier. Query from examples: An iterative, data-driven approach to query construction. In *VLDB*, 2015.
[18] D. Lowd and P. Domingos. Naive bayes models for probability estimation. In *ICML*, 2005.
[19] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, 2009.
[20] B. Mozafari, P. Sarkar, M. J. Franklin, and M. I. Jordan. Scaling up crowd-sourcing to very large datasets:a case for active learning. In *VLDB*, 2014.
[21] B. Qarabaqi and M. Riedewald. User-driven refinement of imprecise queries. In *ICDE*, 2014.
[22] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *ACM SIGKDD*, 2002.
[23] B. Settles. Active learning literature survey. Technical report, University of WisconsinMadison, 2009.
[24] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 2008.
[25] B. Settles, M. Craven, and S. Ray. Multipleinstance active learning. In *NIPS*, 2008.
[26] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *ACM Workshop on Computational Learning Theory*, 1992.
[27] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *PVLDB*, 2015.
[28] I. H. Witten and et al. Weka: Practical machine learning tools and techniques with java implementations. In *Workshop: Emerging Knowledge Engineering and Connectionist-Based Information Systems*, 1999.
[29] Y. Xue and M. Hauskrecht. Robust learning of classification models from noisy soft-label information. In *ECCV*, 2016.