Interactive Data Exploration based on User Relevance Feedback

Kyriaki Dimitriadou* Olga Papaemmanouil* and Yanlei Diao†
* *Brandeis University, Waltham, MA, USA,* † *University of Massachusetts Amherst, Amherst, MA, USA,*
*{kiki,olga}@cs.brandeis.edu, †yanlei@cs.umass.edu

***Abstract*—Interactive Data Exploration (IDE) applications typically involve users that aim to discover interesting objects by iteratively executing numerous ad-hoc exploration queries. Therefore, IDE can easily become an extremely labor and resource intensive process. To support these applications, we introduce a framework that assists users by automatically navigating them through the data set and allows them to identify relevant objects without formulating data retrieval queries. Our approach relies on user relevance feedback on data samples to model user interests and strategically collects more samples to refine the model while minimizing the user effort. The system leverages decision tree classifiers to generate an effective user model that balances the trade-off between identifying all relevant objects and reducing the size of final returned (relevant and irrelevant) objects. Our preliminary experimental results demonstrate that we can predict linear patterns of user interests (i.e., range queries) with high accuracy while achieving interactive performance.**

I. Introduction

Interactive Data Exploration (IDE) refers to a class of applications in which users strive to discover interesting data objects by iteratively executing queries using varying predicates [1]. Example cases can be found in the scientific domain, such as analysis of astrophysical surveys (e.g., [2]) or of financial data, etc. Here, scientists are not always able to express their data interests precisely as they do not have upfront an understanding of the exact attributes that should be used to formulate a query that collects all relevant information. Query formulation hence becomes a highly labor intensive process where one executes numerous selection queries using iteratively different predicates. The exploration may take anywhere between days to weeks to draw conclusions since users need to examine 100s-1000s of data objects, asses their relevance to their interest and then rewrite the query to balance the trade-off between collecting all relevant information and reducing the size of returned data.

To support IDE applications, we propose a framework that eliminates the need for the user to formulate his own queries, but instead *automatically* builds these data extraction queries for him. The user engages in a "conversation" with the system by characterizing a set of strategically collected samples, as relevant or irrelevant to him. The user's feedback is incorporated into a classification model that predicts relevant to the user data objects. This model is then used to formulate an extraction query that retrieves the set of data matching the user's interest. The framework iteratively improves the effectiveness of the model by collecting in each iteration new samples to be labeled by the user and which are used to generate a new more accurate model.

While existing classification algorithms (e.g., [3]) can be used to implement the data classification and incorporate relevance feedback in the classification process (e.g., [4]), these approaches assume that training sets are available a priori or provided incrementally by a third party and hence they do not focus on identifying which data samples to acquire and label. The active learning community has also proposed solutions that maximize the learning outcome while minimizing the number of samples shown to the user but these techniques are domain specific (e.g., document ranking [5], image retrieval [6], etc.) and they sample small data sets with negligible data acquisition costs. Therefore, they cannot offer interactive performance on big data sets.

To address these challenges for relational databases, our framework closely integrates classification model learning (from existing labeled samples) and effective data exploration and sample acquisition (deciding new data areas to sample). Specifically, we introduce an *automatic* interactive data exploration framework that navigates the user through the data space by requesting his relevance feedback on data samples. We propose data exploration techniques that leverage the classification properties of decision trees to predict objects/areas of interest and progressively improve the effectiveness of these predictions, while striving to minimize the user's effort. These techniques aim to identify linear patterns of user interests (i.e., patterns that result in point or range queries).

The rest of the paper is organized as follows. In Section II we provide an overview of our framework. We introduce our space exploration approach in Section III. Section IV presents our promising preliminary experimental results, Section V discusses the related work and we conclude in Section VI.

II. AIDE Framework Overview

Our IDE framework is depicted in Figure 1. An initial sample set is selected and the iterative data exploration is initiated: the user provides his feedback on the sample set by labeling data objects as either relevant or irrelevant to him (*User Relevance Feedback*). The labeled samples are the training set of a classification algorithm that generates a model to characterize the user interests (*Data Classification*). In the next iteration, new labeled samples are incorporated to the training set and a new classification model is built. Given the current classification model, we identify promising data areas

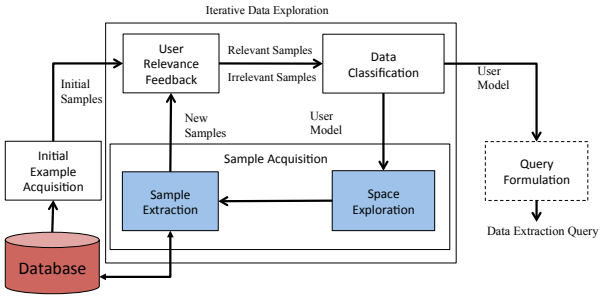


Fig. 1: Interactive Data Exploration Framework.

to be sampled further and can improve the effectiveness of our system (*Space Exploration*). Finally, we retrieve the next sample set from the new exploration data areas and we present them to the user for feedback (*Sample Extraction*).

The above steps are executed iteratively aiming to converge to a model that captures the user interests, i.e., eliminates irrelevant objects while it identifies a high percentage of relevant objects. At any time, the user can "translate" the final classification model into a query expression. This query will retrieve from the underlying database all objects characterized as relevant by the user model (*Query Formulation*). The steering process is terminated when the user terminates the process explicitly, e.g., when reaching a satisfactory output result set or when he does not wish to continue labeling samples. Therefore, users decide on the effort they are willing to invest on the exploration process (e.g., number of samples they label) while our framework tries to make the best use of these available "resources" and provide the best possible prediction for the user's interests. Intuitively, the more samples we will have available the more accurate the final user model will be. However, showing a higher number of samples to the user increases not only the user's effort but also the sample extraction time, i.e., the user's wait time.

Classification Model Our framework relies on decision tree classifiers (e.g., [3]) to identify linear patterns of user interests (e.g., range queries). Decision trees produce classification models that predict the class of an unclassified object based on input labeled training data. Their major advantage is that they provide easy to interpret models that describe the features characterizing relevant and irrelevant objects. For example, in SDSS [2], a decision tree may characterize as relevant objects satisfying the conditions: $(red \leq 13.2 \wedge 10.3 < green \leq 11.5)$ and $(13.2 < red \leq 14.1 \wedge 9.53 \leq green \leq 10)$. Hence, it is straightforward to formulate the data extraction query: `select * from galaxy where (red ≤ 13.2 and green >10.3 and green ≤ 11.5) or (red > 13.2 and red ≤ 14.1 and green ≥ 9.53 and green ≤ 10).`

III. Space Exploration

Our main research focus is to optimize the effectiveness of the data exploration while minimizing the number of samples we present to the user. We assume that user interests can be captured by *range queries*, i.e., relevant objects are clustered in one or more areas in the data space. Therefore, our goal is to discover relevant areas and propose to the user queries that select either a single relevant area (conjunctive queries) or multiple ones (disjunctive queries).

Our framework incorporates three exploration steps. First we focus on collecting samples from unexplored yet areas aiming to identify interesting objects (*Object Discovery*). In the second step we strive to identify relevant areas from already known relevant objects (*Misclassified Samples Exploitation*). Finally, given a set of predicted relevant areas, we refine their boundaries to further improve the accuracy of our predictions (*Boundary Exploitation*).

A. Relevant Object Discovery

The *object discovery phase* operates on a *hierarchical exploration grid* and it aims to identify new interesting data objects. Specifically, our system creates off-line a set of grids G and each grid has a different granularity, allowing us to "zoom in/out" into specific areas when needed. We refer to each grid as an *exploration level* and as we move to lower levels we have a higher number of smaller grid cells. Exploration-levels have d dimensions, one for each attribute the user elected to focus for his exploration and are generated by dividing the normalized domain of each attribute to equal width ranges.

The object discovery phase starts from a given exploration level and retrieves one tuple located close to the center of each grid cell in that level. Specifically, for each grid cell, we identify the "virtual" center of the cell with coordinates $(a_1, a_2, ..., a_n)$ and we retrieve a single random tuple with distance $\gamma < \beta$ along each dimension from this center. The smaller the γ parameter the more uniformly distributed our collected samples are across the domain space.

In the first iteration we show to the user one tuple for each cell of the given exploration level. By default we start with the higher exploration grid and we gradually move to more detailed levels. If the retrieved tuple in a cell is relevant to the user, then we skip this cell in the next object discovery phase. Otherwise, in the next iteration, we use the lower exploration level for this specific cell. While the user is willing to label more samples the phase will continue zooming in each cell and collecting samples around their centers.

B. Misclassified Sample Exploitation

This phase operates under the assumption that relevant objects will be clustered together, and hence its goal is to "convert" relevant objects returned from the object discovery phase to "relevant areas". To achieve that, it collects samples around relevant objects that the classifier has not yet translated to relevant areas, i.e., it has misclassified as irrelevant (false negatives).

Let us assume that in the i-th iteration the training set T_i has m false negatives based on the classifier C_i. Then, in the next iteration we collect f samples around each false negative. Our experimental results showed that f should be set to a small number since higher values will increase the user effort without improving the exploration outcome and in our experiments is set to 25 tuples. These samples are retrieved

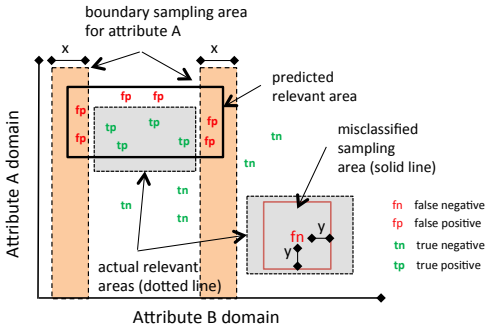


Fig. 2: Boundary exploration for the relevant areas A and D.

randomly from a normalized distance y on each dimension from each false negative sample, as shown in Figure 2 where one relevant area out of the two has been predicted (and only partially). The closer the value y is to the width of the relevant area the misclassified sample belongs to, the higher the probability to collect relevant objects than irrelevant ones.

C. Boundary Exploitation

Given a set of discovered relevant areas, the boundary exploitation phase aims to refine the boundaries of these areas and therefore to improve the effectiveness of the classification model. Figure 2 demonstrates a motivating example, where a predicted relevant area partially overlaps with the actual relevant area. These leads to false positives: irrelevant objects that the decision tree has classified as relevant.

Formally, we represent the decision tree classifier C_i as a set of hyper-rectangles in a multidimensional space and we iteratively refine the predicates that define these hyper-rectangles by collecting samples around their boundaries. Let us assume the decision tree has revealed k d-dimensional relevant areas. Our experimental results demonstrated that this phase has the smallest impact on the effectiveness measure of our model: a non discovered relevant area can reduce our accuracy more than a partially discovered relevant area. Hence, we constrain the number of samples we retrieve during this phase to α_{max}. This allows us to utilize better the user's effort as he will provide feedback mostly on samples generated from the previous two, more effective phases. Furthermore, our goal is to distribute an equal amount of user effort to refine each boundary. Since we have k relevant areas each with d boundaries, we collect $\alpha_{max}/(k \times d)$ random samples across the domain of each attribute t_i making sure that the distance of t_i's value is less than x from the boundary of the relevant area. The parameter x affects the convergence to the actual boundary and it should be adaptively set to the expected difference between the actual and predicted value for the sampling attribute. Figure 2 shows the sampling areas around the boundaries for attribute A of the predicted area.

IV. Preliminary Experimental Results

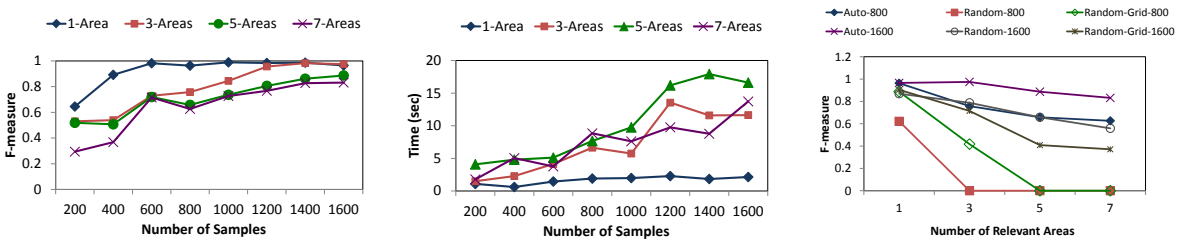
We implemented our framework on JVM 1.7 and our underlying data space was drawn from the SDSS database [2]. We used a database size of 11GB and our exploration space had two attributes. Specifically, we used two uniformly distributed numerical attributes `rowc` and `colc` of the `PhotoObjAll` table. A covering index was also used on both of these attributes. Finally, all our experiments were run on an Intel PowerEdge R320 server with a 32GB RAM. The desicion tree algorithm we used is CART [3].

We focused on predicting range queries (*target queries*) and we varied their complexity based on the number of disjunctive predicates they include (*number of relevant areas*). The diversity of our target query set is driven by the query characteristics we observed in the SDSS benchmark [2]. Specifically, 90% of their queries select a single area, while 10% selects only four (4) relevant areas. Our experiments cover even more complex queries of 3, 5 and 7 relevant areas.

Given a target query, we simulate the user by executing the query to collect the exact set of relevant tuples. We rely on this set to label the collected samples depending on whether they are included in it. We also use this set to evaluate the accuracy of our final predicted extraction query. We measure accuracy using the F-measure of our final prediction. This metric calculates the harmonic mean of precision and recall. Achieving a good precision will eliminate irrelevant objects from the retrieved data set while a good recall will ensure that our final query can retrieve a good percentage of the relevant to the user objects. We assume that the user is willing to label N samples in total and we report the F-measure we converge to when we reach this number. Our approach achieves high accuracy for the most common queries of one area while it can predict, given a reasonable number of samples, complex queries with a higher number of relevant areas.

Effectiveness Figure 3(a) shows the accuracy when we increase the query complexity and the number of relevant areas we are trying to discover ranges between one (1) and seven (7). Naturally, labeling more samples improves in all cases the accuracy. Our approach performs very well for common conjunctive queries: for one relevant area we need only 200 samples to reach an accuracy higher than 60%. To accurately predict highly complex disjunctive queries more samples are needed. However, even for complex queries of seven (7) areas we achieve an accuracy of 63% or higher when the user labels at least 600 samples. This is an improvement over the 1000s of objects users have to review if they have to manually assess the quality of their exploration queries (e.g., the output size of our target queries ranges between $13,957 - 99,671$ objects).

Efficiency Figure 3(b) shows the time overhead of our exploration process (seconds per iteration) as the user labels more samples. This includes the time of the space exploration, data classification and sample extraction steps. In all cases, extracting more samples increases the exploration time. Furthermore, the complexity of the target queries affects the overhead of the system. It takes less time per iteration to discover fewer relevant areas, since searching for more relevant areas leads to more false negative samples and thus to more sample extraction queries in the misclassified exploitation phase. Our time overhead is acceptable: for an average of 10 iterations we had 2.1 secs in average per iteration for common queries of 1 area, while the average time is 9.2 secs for the most complex



(a) Effectiveness (F-measure). (b) Efficiency (time per iteration). (c) Comparison to random exploration.

Fig. 3: Evaluation results for increasing query complexity (number of relevant areas) and varying number of samples.

queries of 7 areas.

Random Exploration We compared our approach (*Auto*) with two alternatives. *Random* does a random selection of samples in the data space, receives user feedback and then builds a classification model. In *Random-Grid* the sample selection is done randomly within each cell of our exploration grid (and not around the cell center as Auto). This provides a better distribution of samples across the exploration space than Random. We also set the number of samples to be $N = 800$ and $N = 1600$ to ensure that both random techniques can predict complex queries. Figure 3(c) shows that we deliver better results in all cases and our accuracy is close to 95% for 1600 samples and 80% for 800 samples in average. In all cases (except AUTO-1600) increasing the query complexity reduces the accuracy since it becomes more difficult to discover objects of interest within all areas while keeping the number of samples constant. Having more samples (i.e., 1600) allows all techniques to achieve a higher F-measure.

V. Related Work

A related body of work proposed the"Query-By-Example" framework [7] and query formulation interfaces [8]. These systems do not model user interests and cannot retrieve "similar" objects. Query relaxation techniques have also been proposed for supporting data exploration [9], [10]. These solutions adjust the query parameters to reach a cardinality goal and therefore do not characterize user interests.

Various sampling techniques have been proposed for approximate query processing [11], [12]. Instead, our sampling aims to improve our prediction of the user interests. In [13] they facilitate query formulation using past SQL query templates. In [14] they use collaborative filtering to provide query recommendations. These systems do not focus on predicting "similar" data objects. Idreos et al. [15] propose a system for interactive data processing tasks aiming to improve data analysis tasks. [16] explores the data space based on statistical properties of the data and provides suggestions for further exploration. Our system is different since we rely on the expert user's feedback to provide query suggestions. Finally, [17] learns conjunctions of quantified Horn expressions over nested relations, which differ from our target SQL queries.

VI. Conclusions & Future Work

In this paper we proposed an iterative framework that assists users in discovering interesting linear patterns and eliminates expensive ad hoc exploratory queries. Our solution relies on a seamless integration of classification and data management algorithms that collectively strive to match the user's interests while minimizing the number of samples presented to him. Our results are positive: we can deliver highly accurate query predictions within acceptable wait time.

We have built an initial prototype and we aim to deploy it as a public service operating on scientific data [2]. This will help us better debug our system as well as collect real usage scenarios for further experimentation. We also plan to expand our research agenda towards two dimensions. First, incorporate optimizations that reduce the user's feedback effort especially for complex disjunctive queries. Second, develop techniques that allow for high scalability with the data space size as well as the dimensionality of the exploration space.

References

[1] U. Çetintemel et all, "Query steering for interactive data exploration," in *CIDR*, 2013.

[2] "Sloan Digital Sky Survey," http://www.sdss.org/.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, 1984.

[4] X. S. Zhou and T. Huang, "Relevance Feedback in Image retrieval: A comprehensive review," *Multimedia System*, vol. 8, no. 2, 2003.

[5] I. Ruthven et al, "A survey on the use of relevance feedback for information access systems," *The Knowledge Engineering Review*, vol. 18, no. 2, 2003.

[6] N. Panda, K.-S. Goh, and E. Y. Chang, "Active learning in very large databases," *Multimedia Tools Appl.*, vol. 31, no. 3, 2006.

[7] M. M. Zloof, "Query-by-example: operations on hierarchical data bases," in *National Computer Conference and Exposition*, 1976.

[8] C. Ahlberg et al, "Dynamic queries for information exploration: an implementation and evaluation," in *CHI '92*, 1992.

[9] S. Chaudhuri, "Generalization and a framework for query modification," in *ICDE*, 1990.

[10] C. Mishra and N. Koudas, "Interactive query refinement," in *EDBT*, 2009.

[11] S. Agarwal et al, "Blink and it's done: interactive queries on very large data," in *VLDB*, 2012.

[12] L. Sidirourgos, M. Kersten, and P. Boncz, "SciBORQ: Scientific data management with Bounds On Runtime and Quality," in *CIDR*, 2011.

[13] N. Khoussainova et al, "A Case for A Collaborative Query Management System," in *CIDR*, 2009.

[14] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query Recommendations for Interactive Database Exploration," in *SSDBM*, 2009.

[15] M. L. Kersten et al, "The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds," *PVLDB*, vol. 4, no. 12, 2011.

[16] T. Sellam and M. L. Kersten, "Meet Charles, big data query advisor," in *CIDR*, 2013.

[17] A. Abouzied et al, "Learning and verifying quantified boolean queries by example," in *PODS*, 2013.