Automatic User Steering for Interactive Data Exploration

Kyriaki Dimitriadou
Brandeis University, Waltham, MA, USA
{kiki}@cs.brandeis.edu
Expected Graduation date: 2016
Supervised by Olga Papaemmanouil

***Abstract*—The amount of data that have flooded databases during the last few years have created several new problems for the data management community to address. One of the most prominent is the discovery of new and interesting information that is hidden in the underlying big data sets. As of now, in order to explore these data sets users begin with a few general queries, study their output and iteratively issue more specific ones until they discover interesting information. This is an onerous process that requires time and effort. In this thesis we are developing an automatic data exploration framework that will make the discovery of new information in a vast data space both effective and efficient. Our system asks users for their relevance feedback on strategically collected samples in an interactive manner, steers them towards the interesting parts of the database and eventually formulates the query that retrieves their data of interest. Our preliminary results are encouraging and allow us to persevere in the development of such a system.**

I. Introduction

Traditionally, data management applications are designed around the archetype that users have a clear and precise idea of their dataset and know exactly the knowledge they want to gain out of their data. However, as the amount of data increases scientists do not have such a clear understanding of the underlying database. Thus a new type of applications has emerged, namely Interactive Data Exploration (IDE) applications, where users iteratively issue multiple queries with varying predicates until they are led to discover interesting information. These "exploratory sessions" can last many hours and even days and can be detrimental to human productivity and resource conservation.

As an example of an IDE application, the Sloan Digital Sky Survey [1] which maps the sky, has gathered up to now approximately 1328 TB of data and includes information for around 260 million stars and 208 million galaxies. To make sense of this data scientists start with a few general queries with high selectivity, study the query results, refine their queries based on those results and decide what their next query will be. They iteratively perform this long-running, multi-step process until they are satisfied with their findings, in which case they can draw their final conclusions.

There are several problems in the above process that scientists have to deal with. Due to the vast exploration space, scientists do not clearly comprehend the contents of the database and thus it becomes very difficult to identify the exact data objects they are interested in. Furthermore, writing a query to retrieve the interesting data can still be hard when the schema of the database and the data itself are very complex or when the user has poor query writing skills. In addition to that, due to the lack of proper tuning user queries have a long response time which slows down the process of exploration. Finally, manually reviewing returned query results and identifying the relevant data objects before making a decision about the next steps further increases the user effort and overall overhead of the exploration process because query replies can be too large and/or unsatisfying.

Motivated by these problems we envision a system that will address the above challenges of big data exploration and will eliminate the need for expensive ad-hoc exploratory queries. In a nutshell, our system will navigate users through the database and will steer them towards the "trajectories" of the data that are of most interest to them. To achieve that, the system will iteratively ask the users for their relevance feedback on selected tuples, will generate a model that predicts their interests using machine learning algorithms and will finally formulate a selection query that retrieves all the relevant data to their search.

Our research aims to integrate machine learning algorithms and database optimization techniques in the task of interactive data exploration. While machine learning algorithms can be used to model user interests, they are not concerned with how to steer users along interesting trajectories or how to minimize the cost of data acquisition to improve the user interactive experience. Our system addresses these drawbacks by leveraging the unique characteristics of exploration workloads. In this thesis we will develop a suite of algorithms and techniques that draw insights from machine learning algorithms to guide the exploration of a big data space, and leverage the knowledge of exploration patterns to optimize query processing inside the database.

II. Approach

Our system explores the data space by iteratively showing to the user strategically selected tuples and letting the user decide if those are interesting or not to him. After the user classifies these tuples, the system uses machine learning techniques and specifically, decision trees, to classify the rest of the data space to interesting and non-interesting areas and formulates a query that projects the user's interests based on the information

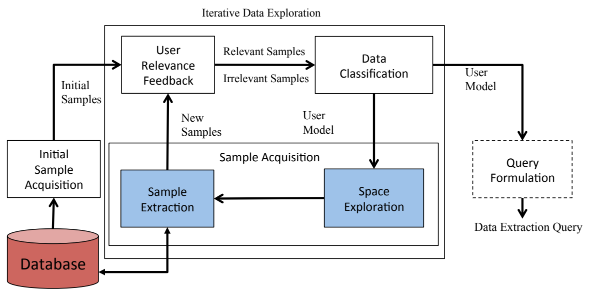

Fig. 1: Automatic Interactive Data Exploration Framework.

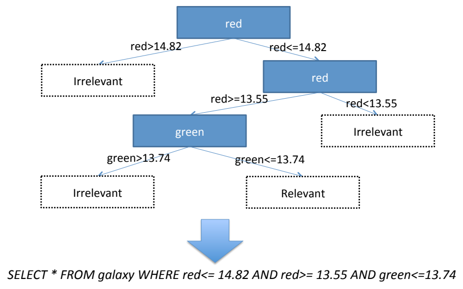

Fig. 2: Translation of a decision tree to a selection query.

that he has provided. In each iteration, the system strives to select the appropriate tuples to show to the user aiming to improve the accuracy of the formulated query. The user can explicitly stop this process whenever he feels satisfied with the query that the system has formulated. In the next section we provide a more detailed description of the various steps of our framework (Figure 1).

A. Interactive Automatic Data Exploration

Initially, the system chooses a first batch of samples from the entire data space to show to the user *(Initial Sample Acquisition)* and the user is asked to provide his feedback on this set of samples *(User Relevance Feedback)*. Our relevance feedback system is binary; an item can only be classified as relevant or irrelevant.

The labeled samples are used as an input (training set) to our data classification algorithm. Our system uses a decision tree algorithm to predict a model that classifies the whole database as relevant/irrelevant to the user based on that training set *(Data Classification)*. The model that the decision tree generates is expressed in simple classification rules which can be then interpreted to a query that selects the objects that are relevant to the user's search. In Figure 2 we can see an example decision tree based on the SDSS data and its translation to a selection query. This decision tree is generated for a user that is interested in galaxies with brightness in the red wavelength between 13.55 and 14.82 and in the green wavelength smaller than 13.74. For the moment we assume numerical attributes, while future work will focus on categorical data as well.

At this point, the user could possibly stop the process if he was satisfied with the query that the system formulated in this iteration. If that is not the case, then the system needs to build a more accurate query to reflect the user's interest. To achieve that the system needs more feedback from the user on a different set of samples.

Selecting this next batch of samples is crucial; these samples need to be the most informative and accuracy yielding samples in the whole data space. In order to find them, the system launches a series of space exploration steps (explained below) with each step returning the appropriate samples from the database *(Space Exploration)*. After the system identifies the new sampling set, it presents it to the user for labeling, at which point a decision tree is generated based on all of the samples that the user has labeled up to this iteration.

When the iterative process completes, the system translates the final model built by the decision tree classifier into the user's extraction query that retrieves all the data that are characterized as relevant to him.

B. Space Exploration Phases

In order to efficiently and effectively explore the data space the system deploys three phases which build upon each other; the *Object Discovery* phase, the *Misclassified Exploitation* phase and the *Boundary Refinement* phase.

Object Discovery Phase The objective of this phase is to identify new interesting data objects from unexplored areas of the data space. These objects need to be well distributed in the data space so that the user can examine representative tuples from as many different areas as possible.

To achieve this we divide the data space into g equal width grids where each grid defines a different data area and we select one random tuple close to the center of each grid to represent this area to the user for labeling. The number of grids we are going to construct depends on the number of samples we want to spend on this phase. If our exploration space consists of d attributes and the number of samples we have for this phase is n then we want to break down the domain of each attribute into $b = \sqrt[d]{n}$ equal width ranges to build the g grids (here $g = n$ since we select one tuple from each grid).

For our system, we keep a pre-built set of grids with different values for b to satisfy users that wish to invest different levels of effort in the system. If a user has labeled all the samples for the grids built with value b_0 and he wishes to further explore in more depth the data space then we carry on to present the user with the next set of tuples drawn from the grids where $b_1 > b_0$.

Misclassified Exploitation Phase The previous phase identifies single points of interests, one per each grid explored. The *Misclassified Exploitation* phase leverages this information in order to turn these points of interest into areas of interest by increasing the relevant samples in our training set.

Typically, the classification rules that a decision tree outputs contain a prediction error, meaning that a subset of the training samples will be misclassified, i.e., the decision tree assigns them to the opposite class. It is usually the case that points of interest discovered from the previous phase get misclassified as irrelevant because the irrelevant class dominates our sample set. Therefore, we need to reinforce our training set with more tuples that are similar to the interesting misclassified tuples (*false negatives*) in order to provide enough information to the decision tree to formulate the interesting areas.

In our framework, we measure similarity in terms of distance; we assume that interesting tuples will be located close to each other in the data space forming interesting areas. Thus, this phase selects a number of random tuples around each false negative sample to present to the user. In this fashion, we strive to increase in the next iteration our training set with samples from the relevant class by having the user examine and classify objects that are similar to his interesting ones.

Boundary Refinement This last step adds samples to our training set that improve the accuracy of an already built decision tree by striving to refine the boundaries that separate the two different classes of data (relevant and irrelevant).

To achieve this we represent boundaries as hyperectangles in a multidimensional space defined by the classification rules of the decision tree and we aim to iteratively refine these boundaries by selecting samples that belong to both classes. Such samples are located around the boundaries of the formulated relevant hyperectangles. Therefore, given a set of boundaries we collect random samples across the domain of each attribute in our data space making sure that each sample is in close proximity to the boundary of the relevant area. This approach is applied across all the boundaries of the relevant hyperectangles, allowing us to shrink/expand each dimension of the relevant areas.

Our optimizations include dynamically regulating the number s of samples we gather around each boundary and progressively adapting our sampling areas in each iteration. As the percentage of change of a boundary increases from one iteration to the other so does s since this reveals that the boundary is not correctly defined yet and we need more samples to refine it. Moreover, we avoid sampling areas that have high overlap in between iterations in order to present the user varying samples from different data areas. In this way we increase the accuracy of the final query we formulate for him.

III. Preliminary Results

Queries We performed a preliminary set of experiments on a real data set drawn from the Sloan Digital Sky Survey [1]. For our experimental purposes we studied the query workload issued to the SDSS database over a period of time. We observed that 90% of the queries issued to the database selected one single area of interest and that the predicates used in these queries had an average coverage of 3.4% of their domain. Thus in our target query set we included queries that select one area with domain coverage (i.e., relevant area size) between 1-9%. Specifically, we experimented with *small*, *medium* and *large* relevant areas. Small areas have attribute ranges with average width of 1-3% of their normalized domain, while medium areas have width 4-6% and large ones have 7-9%.

Experimental Setup We implemented our framework on JVM 1.7 and run our experiments on an Intel PowerEdge R320 server with 32GB RAM. The database size we used is 11GB and our exploration space had 2 attributes. A covering index

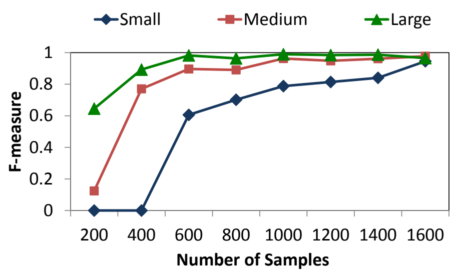

Fig. 3: Predicted query accuracy for increasing complexity of one relevant area.

was also used that included these attributes. The database we used is MySQL 5.5.32.

Evaluation Metrics In our experiments we measure effectiveness using the F-measure metric which is the harmonic mean of the recall (i.e., percent of relevant objects we retrieve) and precision (i.e., percent of irrelevant objects we retrieve) of our final extraction query. In our experiments we report the F-measure we converge to when the use has classified up to N number of tuples.

User Simulation Given a target query, we simulate the user by executing the query to collect the exact target set of relevant tuples. We use this set to label the new sample set we extract in each iteration as relevant or irrelevant depending on whether they are included in it or not. Moreover we use this set to evaluate the accuracy of our final predicted extraction queries.

Results Our preliminary results are really encouraging. Figure 3 reveals the F-measure of our predicted query when we increase the target query complexity by varying the size of the relevant area from *large* to *medium* and *small*. Naturally, labeling more samples improves in all cases the accuracy. As we can see with only 200 tuples we achieve an F-measure higher than 60% and with labeling 400 samples we reach an F-measure higher than 89% for the less complex case of a large relevant area. As the query complexity increases the user needs to label more samples (at least 800 for small areas and 600 for medium areas) to get highly accurate predictions (greater than 82%) but even an acceptable set of 400 samples offers accuracy higher than 60% in all cases. These numbers show a great improvement over the thousands of objects scientists need to review before ending up with their final query that selects their data of interest (in our experiments our target query selectivities range between $2,957 - 26,817$ objects).

The results regarding the user's wait time in between iterations are also motivating. Our system performs better when the size of the relevant area is small with an average of 1.2 secs wait time per iteration. As the size of the relevant area increases to medium the average time per iteration increases to 1.5 secs and finally when the size of the relevant area is large the overhead raises to an average of 1.8 secs. This small extra time when we are dealing with less complex queries leads to a higher accuracy. Due to space limitation we omit this graph.

IV. Challenges & Future Directions

In this section we briefly sketch some of the future directions we plan to undertake in our research.

Interactivity Providing an interactive experience to the user is a prerequisite for our system. Since each of our three exploration phases (Section II-B) increases our labeled sample set with new tuples using queries to retrieve them from the database, these queries need to return results as fast as possible in order for the user's wait time in between iterations to be minimal. One direction we plan to take to achieve that is sampling. The queries for all of our three phases select random tuples from different areas of the data space. The amount of data that exist in those areas is not important to our cause as long as a few exist for the user to label. In fact, the less data that exist in the data space the faster our queries are going to run. Therefore, this offers us a great chance to study various sampling techniques to minimize the volume of the data and expedite our queries but without a loss in the accuracy of the final query we formulate for the user. Although sampling has been studied before in order to approximately answer queries ([2], [3], [4]) it presents a new challenge in our setting since the query workload that our phases generate is unique and different from what previous works in the field have studied.

Past User Interactions Past user interactions with the system create a wealth of information that if studied and used carefully could possibly help with the "exploratory sessions" of new users. For example, if we deem that the exploration paths (queries that are generated in each iteration) of two users are similar then we can use the knowledge we have gained from the first user to jump ahead in the exploratory session of the other user and predict data trajectories or more attributes that could be interesting to him. Furthermore, user interaction histories could be used to proactively fetch and cache samples from the database and in that way speed up the data exploration of new users. Multiple aspects of query workloads as a sequence have been studied in the past ([5], [6]) however within our framework we have the chance to develop new optimization algorithms that harness interaction histories to facilitate interactive data exploration.

Understanding the Query Output A significant part in the exploration process is to help the user understand the data that the predicted query is selecting for him. This is important in order to assist the user in drawing conclusions regarding the selected data. Furthermore, it would help him label the next set of samples if he is not satisfied with his conclusions and he wishes to continue the process. Towards this goal the system should provide insightful statistics about the areas that the user is interested in such as the average of specific columns, min, max, sum etc., based on indications by the user or previous similar queries related to these areas, helping him in that way to get a clearer view of the data the formulated query has selected for him. Although query suggestions based on previous queries have been studied before ([7]), we believe that turning those previous queries to useful information about the predicted areas and doing so interactively, offers a promising research direction. Furthermore, retrieving statistics regarding the formulated areas means issuing more queries to the database, which will be a challenge to achieve without affecting the user's wait time in between iterations.

Data Visualization Data visualization is a natural extension of an exploration system such as the one described here. When coming to graphical interface, graphs and images are easier to mentally grasp than raw text and numbers. Therefore, a research challenge would be to try to visualize samples and present them for labeling to the user in an easily comprehensible format. For example, data that deal with geographical information could be presented to the user as points on a map or data that contain chronological information could be directly visualized as points on a time chart. Data visualization has seen significant support by the data management community ([8]), however the main challenge in our setting would be to try and integrate data visualization into our system given its interactive and multi-dimensional character.

V. Related Work

A related body of work focuses on the development of "Query-By-Example" [9] and front-end interfaces that assist in query formulation [10]. These systems do not attempt to understand user interests and to retrieve "similar" data tuples. Query relaxation techniques have also been proposed for database exploration [11], [12], [13], [14]. These solutions focus on adjusting the query parameters to reach a cardinality goal and therefore cannot characterize user interests or reduce the number of tuples shown to the user for labeling. Moreover, relevance feedback techniques have been studied in the field of active learning but they target different types of data (ranking of documents (e.g., [15]), image retrieval [16]) and usually examine each sample in the database [17] before presenting it to the user which is not feasible in the case of an interactive system. In [18] they facilitate query formulation using past SQL query templates. Idreos et al. [19] present a vision towards a system for interactive data processing tasks that reduces the time spent on data analysis. These works do not focus on predicting user interests or "similar" data tuples. In [20] they require the user to specify in advance the attributes and value assignments used to learn queries, which are assumptions we cannot make in our work. Finally, our vision for automatic, interactive navigation in databases was first introduced in [21].

VI. Conclusions

We described an automatic data exploration system that is crucial for identifying interesting data objects in huge and complex datasets that one encounters in many big data applications. Our system significantly reduces human effort on data exploration as users are methodically steered through the data by providing their feedback on selected samples. Such automated steering, fully exploiting user interests while grounded in rigorous learning theory, assists users in discovering new interesting data patterns. It also eliminates expensive ad-hoc exploratory queries, leading to further improvements in user productivity and resource conservation.

References

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

[2] B. Babcock et al, "Dynamic sample selection for approximate query processing," in *SIGMOD*, 2003.

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

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

[5] S. Agrawal, E. Chu, and V. R. Narasayya, "Automatic physical design tuning: workload as a sequence," in *SIGMOD*, 2006, pp. 683–694.

[6] I. T. Bowman and K. Salem, "Semantic Prefetching of Correlated Query Sequences," in *ICDE*, 2007, pp. 1284–1288.

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

[8] C. Stolte et al, "Polaris: A System for Query, Analysis and Visualization of Multi-dimensional Relational Databases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 52–65, 2002.

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

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

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

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

[13] N. Koudas, C. Li, A. Tung, and R. Vernica, "Relaxing join and selection queries," in *VLDB*, 2006.

[14] A. Kadlag, A. V. Wanjari, J. Freire, and J. R. Haritsa, "Supporting Exploratory Queries in Databases," in *DASFAA*, 2004.

[15] 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.

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

[17] Y. Chen and S. Mani, "Active learning for unbalanced data in the challenge with multiple models and biasing." *Journal of Machine Learning Research - Proceedings Track*, vol. 16, pp. 113–126, 2011.

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

[19] 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.

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

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