Guided Data Quality Improvement Through Direct/Indirect Interactions

Mohamed Yakout
Supervised by: Prof. Ahmed K. Elmagarmid
Computer Science Department, Purdue University
West Lafayette, IN 47906, USA
{myakout, ake}@cs.purdue.edu

ABSTRACT

There is unanimous agreement among researchers and practitioners alike on the importance of data quality for all applications and the unimaginable consequences of making decisions based on inconsistent, inaccurate or incomplete data (e.g., poor data quality in retail databases alone costs US consumers $2.5 billion annually). However, till recently, all data quality techniques focus on providing fully automated solutions, which are risky to count on, without efficiently and effectively consider the end user feedback.

In this thesis proposal, we present GDQI, a novel framework for Guided Data Quality Improvement that leverages direct and indirect interactions for data cleaning tasks. GDQI incorporates direct user feedback in the cleaning process to enhance and accelerate existing automatic techniques for data quality tasks (e.g., suggesting repairs, discovering data quality rules), while minimizing user involvement. User feedback can then be learned using a machine learning algorithm to minimize user’s effort. The key challenge to this approach is to determine how and in what order to solicit user feedback to confirm the suggestions with the objective of fast convergence to a better data quality. To this end, we introduce to determine which suggestions are most beneficial to the cleaning process to consult the user for verification and feedback. To rank potential suggestions for the user, we quantify the value of the suggestion in terms of (1) likely improvement in data quality using the concept of Value-Of-Information (VOI) from decision theory and (2) value to the learner using active learning.

Moreover, we consider indirect interactions in the form of generated transactions log for improving the quality by deduplication (i.e., doing record linkage). We present a novel technique for record linkage that is based on entities’ behavior. Specifically, we use entities generated transactions log to extract entities’ behavior patterns and use this behavior information to identify entities that represent in fact the same real-world entity.

1. INTRODUCTION

Data quality issues are of several kinds, e.g., inaccuracy, inconsistency, duplicates and incompleteness, and may be the consequence of several reasons, e.g., misspelling, integration from heterogeneous sources, and software bugs. Poor data quality is a fact of life for most organizations and can have serious implications on their efficiency and effectiveness [3]. For example, it is reported [11] that wrong price data in retail databases alone costs US consumers $2.5 billion annually. Not to mention the importance of data quality in the health care domain as well. In such critical applications, incorrect information about patients in an Electronic Health Record (EHR) may lead to inconsistent treatments and prescriptions, which consequently may cause severe medical problems including death. With this comes the need for effective methods to improve the quality of data.

Recent and widely studied techniques to improve the data quality by repairing dirty tuples follow the approach of constrained-based repair (e.g., [2, 4, 6, 7, 8, 15, 20]). These techniques focused more on the efficiency with less focus on the quality. The quality of the repairs is heuristically judged, such that the good repairs are those that would minimally change the data, which may not be generally applied. These techniques depend on identifying a set of integrity constrains (or data quality rules) that should not be violated by the data; They then use these rules to derive data repairs for dirty instances that violate these rules.

There are five main drawbacks for these techniques when it comes to most real-world situations:

1. Identifying the data quality rules requires extensive efforts from domain experts. Indeed, there exists techniques to automatically discover rules (e.g [14, 16, 18]), however, such techniques suffer from: (a) The confusion about selecting the best usefulness measure for the discovered rule depending on the data domain (e.g confidence, interest, conviction, . . . , etc). (b) The manual adjustment of thresholds to control the discovery process, which may lead to the following problematic situations: a set of discovered rules that have less data coverage, an inability to capture most of the existing dirty tuples, and may miss-classify some other tuples as dirty. (c) The use of a global single threshold per a quality measure, which dramatically affects the quality of the discovered rules. For example, rules that should hold on small portions of the data may not be discovered because of the specified thresholds. Such rules could be violated more than the others. Conse-
quently, these dirty portions of data instances can not be repaired.

2. In their repair generation process a heuristic is employed to pick the repairs that would minimally change the data (i.e. pick repairs that are close in distance to the original data). Despite being effective in some cases, inconsistent values could be completely different, i.e., not close in distance, because, for example, mistakes made when using a data entry tool with drop down lists.

3. It is assumed that the values derived directly from the rules are always correct. For example, if the rule is “zip=47906 → state=IN” and a dirty tuple contains “zip=47906, state=IL”, then the repair is to change the state to IN. However, it could be the case that the zip code is actually wrong.

4. Generally, the scalability of the existing automatic data repair techniques is still an issue [12].

5. Finally, there is no efficient way to involve a domain user to neither guide the automatic repair process nor guide the automatic rules discovery.

Another approach for repairing dirty data is to use statistical machine learning techniques. Some efforts in this direction were conducted specifically to predict missing values and detect outliers (e.g. [19]). However, for large databases, machine learning algorithms have difficulty modeling the data relationships due to, for example, over fitting problem. Moreover, sometimes the process of learning a model, as well as the model itself, can not fit in the main memory.

To guarantee the best desired quality, users (domain experts) should be involved to confirm each decision (repair action or discovered rule). This highlights the need for a framework that can combine the best of both; automatically suggesting repairs while efficiently involving the user to guide the cleaning process.

In such a framework, the data quality is improved through direct interactions with a domain user. This approach tackles the problem of improving the data quality from a more realistic and pragmatic viewpoint than has commonly been the case in this area. Automated methods to improve the data quality produce far more decisions than one can expect the user to comment on, and therefore techniques for picking out the most useful-looking ones for presentation to the user, as well as, learning user feedback are important. This will require developing a set of principled measures to estimate the improvement in quality to reason about such task, as well as, investigating machine learning techniques to minimize user’s effort.

We also consider the case where indirect interactions can help improving the data quality. Here, we focus on a novel technique to find duplicate entities (or do record linkage) using the generated transactions logs. An entities’ interactions with a system will produce a transaction log that represent their behavior vis-a-vis the system. We use the generated log to find entities that represent in fact the same entity.

In this thesis proposal, a Guided Data Quality Improvement (GDQI) framework is being developed to efficiently allow the user to guide the process of improving the data quality. GDQI aims at directly involving user feedback to achieve better data quality as quickly as possible. The intuition behind this approach is to continuously consult the user with data quality questions (e.g. suggested repairs, discovered rules) that are most beneficial for the system to improve the data quality as we go.

An architecture for the GDQI is given in Figure 1. GDQI is intended to provide a robust framework against various kinds of data problems by efficiently involving the human computing power in the automatic cleaning process. Focusing on users perspective to improve data quality raises many challenges. For example, it is important to determine how and in what order suggestions should be presented for direct user feedback and how to interpret entities generated transactions log to indirectly improve the quality. Also, learning user feedback, for data repair as well as rules discovery, is an interesting area for exploration. This should help us achieve a good trade-off between high quality and minimal user involvement.

In GDQI, there are basically three processes: (i) Guided Data Repair (GDR) Process: Given a dirty database and a set of defined (or discovered) rules, incrementally generate and consult the user for the most beneficial repairs to maximize the satisfaction of the rules. (ii) Guided Discovery of DQRs (GDQR) Process: Given a dirty database, incrementally suggest and consult the user for the most beneficial discovered rules to maximize the data coverage (i.e. the data covered by the discovered rules) and capturing the dirty tuples, as well as, minimizing the rules search space. (iii) Behavior Based Deduplication: Given entities’ transactions log generated from the usage of the system, find the entities that correspond to the same real-world entity.

2. PROPOSED RESEARCH

From the GDQI framework described in Figure 1, we can identify the following research issues:

- Guided data repair.
- A Scalable generic approach for data repair generation.
- Guided discovery of data quality rules.
- Behavior based deduplication.

In the rest of this section, we discuss challenges and solutions’ approaches for each of these issues.

2.1 Guided Data Repair

The ultimate goal of the GDR process is to combined the best of both; automatically suggesting repairs while efficiently involving the user to confirm the correctness of the repairs. GDR contains a learning component to consider the user’s feedback as training instances to learn how to predict the correctness of a discovered repair. The motivation for using a learning component in GDR stems from the existence of correlations between the original data and the correct repairs. For example, general confusion about some cities and their zip codes, or a data entry operator behavior who always commit mistakes when entering zip codes. If these correlations can be identified and represented in a classification model, then the model can be trained to predict the correctness of a suggested repair and hence replace the user in making decisions for similar (subsequent) situations.
In GDR process of Figure 1, the repair discovery component uses an automated technique to generate repairs for possibly dirty tuples in the database. Then, the grouping component finds the best grouping for the repairs that expose the structure of data relationship, and hence, makes repairs easy for visual inspection. Finally, the task of the ranking and learning components is to devise how to rank the repair groups and how to best present repairs to the user in a way that will provide the most benefit for improving the quality of the data.

We propose a novel ranking mechanism for the generated repairs that applies a combination of decision theory and active learning in the context of data quality. First in the ranking component, we use the concept of value-of-information (VOI) [22] from decision theory to develop a mechanism to estimate the data quality repair benefit from consulting the user on a group of repairs. We quantify the data quality loss by the degree of violations to data quality rules. The benefit of a group of repairs can be then computed by the difference between the data quality loss before and after user feedback. Since we do not know the user feedback beforehand, we develop a set of approximations that allow efficient estimations. Second in the learning component, we apply active learning to order the repairs within a group such that the repairs that can strengthen the learned model prediction capabilities the most come first. To this end, we assign to each repair instance an uncertainty score that quantifies the benefit to the prediction model, or learning benefit, when the repair is labeled.

In GDR, the user is involved in an interactive active learning sessions with the learning component. Specifically, GDR trains a classification model on predicting user’s feedback on the repairs within a group \( c \). The learner orders the repairs such that the repairs that would most benefit from labeling come first. The repairs are displayed to the user along with the learner prediction for each single repair. The user will then give feedback on the top \( n_s \) repairs that he/she is sure about and inherently correct any mistakes made by the learner. The newly labeled instances in \( n_s \) are added to the training dataset and the active learner is retrained. If the user is not satisfied with the learner predictions so far, the learner refreshes the displayed order of the repairs and the user provides feedback on another \( n_s \) repairs from \( c \). This interactive process continues until the user is either satisfied with the learner predictions and thus delegates the remaining decisions on suggested repairs in \( c \) to the model, or the repairs within \( c \) are all labeled by the user.

All decisions on suggested repairs, either made by the user or the learner, are applied to the database. GDR maintains the consistency of the suggested repairs by removing the incorrect repairs and replacing those whose corresponding tuples are still dirty. Since a repair may introduce new violations, GDR always search for new dirty tuples, and new repairs are generated accordingly.

GDR guides the user to focus his/her efforts on the repairs that would improve quality faster, while the user guides the system to automatically repair the data. This feedback process, illustrated in the top part of Figure 1, runs continuously while there are dirty tuples and the user is available and willing to give feedback. Particularly, the repairs group
ranking component provide the repairs such that the most important groups comes first, and moreover, the learning component uses active learning ranking mechanism to guide the user to help strengthen the learner with the objective of automating the process of deciding on the repairs correctness. On the other hand, the user provide the necessary guidance to GDR through the learning component, which learns how would the user correct the data.

We have already discussed the notion of ranking data repairs in [24] with a focus on performance issues, and an implementation for the GDR has already been accepted for demonstration in SIGMOD 2010 [25]. We discuss the details and principals upon which GDR was built in a paper that is under submission.

2.2 A Scalable generic approach for data repair generation

In the repair discovery component of the GDR, we can use a mechanism to generate repairs that is based on any existing fully automated techniques for repairing dirty databases. However as mentioned earlier, the existing techniques that follow the constraints-based repair approach heuristically judge the quality of the repairs, such that the good repairs are those that would minimally change the data, which may not be generally applied. Moreover, these techniques requires the identification of rules that must not be violated. Finally, the scalability to large databases is still an issue.

On the other hand, correlations between data values can be well captured using machine learning models, and moreover, it can directly predict the most likely correct repairs for possibly dirty values. However, the quality and efficiency of these techniques degrades dramatically when applied to large databases. To overcome these limitations, we observe that blocking techniques, which are widely used to improve the efficiency of the well known duplicate detection problem [10], can help in solving the scalability problem when applying machine learning techniques. Blocking techniques are used to partition the tuples into possibly overlapping blocks such that the tuples within one block are close (or similar) to each other. Given this property, applying a machine learning algorithm to learn a model from a given block can be quickly done and fit in the main memory. Moreover, the constructed model from a single block will be more accurate, because it is modeling a small portion of the data with some common properties due to the blocking criterion.

In this proposal, we present a systematic approach to repair dirty tuples by value modification. The approach is composed of two phases: the repair generation and the repair selection phases. The first phase has three main steps: (1) The tuples are partitioned using the blocking technique. (2) For each block, a set of machine learning models are learned, one for each possibly dirty attribute to predict its value. We call the set of constructed models from a block as a committee of value predictors. (3) Each committee is used to predict the attributes’ values of each of the original tuples within the corresponding block to get a possible new tuple repair.

A single blocking criterion may fail to divide the tuples into groups that are accurate enough to separate between the possible correlations between the tuples. Therefore, the above steps are repeated few times, each with different blocking criterion, to allow tuples to be grouped is various ways. Each iteration will provide a possible repair for each of the original tuples, i.e., each constructed committee will predict (or vote) for a tuple repair.

To this end in the repair selection phase, given a set of possible repairs for a single tuple, a repair selection strategy is required to pick the best repair. We introduce a repair selection strategy that considers both the values predicted (or voted) for a single attribute, while taking into account the dependency between the committee members on predicting the other attribute values. We show that this problem is NP-hard and provide an approximate solution that is empirically proved to be accurate.

When there are data quality rules available (either approximate rules automatically discovered or manually defined) our approach can benefit from this advantage. Our approach can help improve the quality of the constrained-based repair techniques, as well as, gaining efficiency and quality improvement.

Given the available rules, dirty tuples can be directly identified as those tuples that violate the rules. Given this information about the existing dirty tuples, the efficiency of our approach can be improved because: (i) only the blocks that contain dirty tuples will be processed, and (ii) repairs will be generated only for the dirty tuples (i.e. the repair selection strategy will run for the dirty tuples only). Moreover, the quality will be improved because: (a) we can avoid the dirty tuples when learning the machine learning models — the models will be learned with less noisy tuples leading to better prediction accuracy, and (b) the directly implied values from the rules will be considered as a possible repair to be taken into consideration in the repair selection phase.

Generally, this repair approach is systematic in the sense that a typical process is applied on each block of tuples to generate tuple repairs. Also the repair selection is another typical process that is applied for each tuple. The computations in the repair generation on each block is independent of any other block, and moreover, the repair selection for a tuple is independent from any other tuple. This leads to an extensively parallelizable solution that is able to scale up for very large databases and benefit from massive parallel frameworks (e.g. MapReduce [9]). The details of our approach is currently in a paper that is under submission.

2.3 Guided Discovery of Data Quality Rules

In the GDQR process of Figure 1, the rules discovery component mines the dirty database and explores the rules search space to generate possible rules. We consider rules in the form of attributes and values dependencies (e.g. Conditional Functional Dependency (CFD) [13] and conditional inclusion dependency (CIND) [5]). Then the rules are grouped for easy visual inspection and it is ranked according to three main objectives; (i) maximize rules data coverage, (ii) maximize the captured dirty instances, and (iii) minimize the rules search space for mining.

The learning component employs user feedback to help automatically classifying useful rules. By computing several usefulness measures (e.g. support, confidence, interest, $\chi^2$, conviction, . . . , etc) for the rules, user feedback provides an opportunity to allow a machine learning algorithm to identify the best combination of measures for the usefulness of the rules, as well as, avoiding manual adjustment of thresholds. The learning component should eventually replace the user in similar situations.

The key novelty here is the ranking of the discovered rules
for user feedback. The main objective in this process is to converge quickly to a set of useful rules that well represent the data semantic to help in decision making, and most importantly, to help identifying dirty tuples and hence help in suggesting repairs in the GDR process.

Along the same line followed before in the GDR for ranking, decision theory as well as active learning can be applied in the context of discovering the rules. The same rational behind the active learning mechanism in the GDR will apply here as well. The difference is that we will need different classification model with different data representation.

For the decision theory ranking mechanism, the concept of VOI can be applied to estimate the usefulness (or benefit) of the rules “to the data quality”. To do this, it is possible to quantify the utility of the discovered set of rules by the degree of coverage to the attribute values and capturing existing dirty instances, as well as, the portion of rules search space that can be trimmed. The benefit can then be computed as the difference between the discovered rules utility before and after user feedback. Again, we do not know neither the user feedback nor the existing dirty instances in the database beforehand, therefore, efficient estimations need to be developed. This research issue is part of our future work.

2.4 Behavior based deduplication

Here, we provide an example for an approach to improve the data quality via indirect interactions. Specifically, we leverage entities’ transactions log for the purpose of deduplication, or finding entities that represent the same real-world entity. In the following, we assume that the database is a result of integrating two data sources, where it is expected to have entities that are shared by the two data sources.

In contrast to most existing techniques, we are considering entity behavior as a new source of information to enhance the record linkage quality. We observe that by interpreting transaction logs, we can discover behavior patterns and identify entities based on these patterns. Various applications such as retail stores, web sites, and surveillance systems, maintain transaction logs that track the actions performed by entities over time. Entities in these applications will usually perform actions, e.g., buying a specific quantity of milk at a specific point in time or browsing specific pages within a web site, which represent their behavior vis-à-vis the system.

A seemingly straightforward strategy to match two entities is to measure the similarity between their behaviors. However, a closer examination shows that this strategy may not be useful, for the following reasons. It is usually the case that the complete knowledge of each entity’s behavior is not available to both sources, since each source is only aware of the entity’s interaction with itself (the source). Hence, the comparison of entities’ “behaviors” will in reality be a comparison of their “partial behaviors”, which can easily be misleading. Moreover, even in the rare case when both sources have almost complete knowledge about the behavior of a given entity (e.g., a customer who did all his grocery shopping at Walmart for one year and then at Safeway for another year), the similarity strategy still will not help. The problem is that many entities do have very similar behaviors, and hence measuring the similarity can at best group the entities with similar behavior together (e.g., [21, 17, 1]), but not find their unique matches.

Fortunately, we developed an alternative strategy that works well even if complete behavior knowledge is not known to both sources. The key to our proposed strategy is that we merge the behavior information for each candidate pair of entities to be matched. If the two behaviors seem to complete one another, in the sense that stronger behavioral patterns become detectable after the merge, then this will be a strong indication that the two entities are, in fact, the same. The problem of distinct entities having similar overall behavior is also handled by the merge strategy, especially when their behaviors are split across the two sources with different splitting patterns (e.g., 20%-40% versus 60%-40%). In this case, two behaviors (from the first and second sources) will complete each other if they indeed correspond to the same real world entity, and not just two distinct entities who happen to share a similar behavior (which is one of the shortcomings of the similarity strategy).

In Figure 2, we conceptually illustrated the idea of merging the behaviors to match entities. The behavior can be represented in a matrix where a cell contains a value indicating that the entity performed a specific action at a specific point of time. It contains zero otherwise.

In this proposal, we develop principled computational algorithms to detect those behavior patterns which correspond to latent unique entities in merged logs. We compute the gain in recognizing a behavior before and after merging the entities transactions and use this gain as a matching score. In our empirical studies with real world data sets, the behavior merge strategy produced much better results than the behavior similarity strategy in different scenarios of splitting the entities’ transactions among the data sources. Our approach for behavior based record linkage can be summarized as follows:

Phase 0: In the initial pre-processing and behavior extraction phase, we transform raw transaction into a standard format showing the action being performed and its time. Next, we extract the behavior data for each single entity. Behavior data is initially represented in a matrix format similar to those given in Figure 2, which we refer to as Behavior Matrix (BM).
Phases 1: Similar to most record linkage techniques, we start with a candidate generation phase that uses a “quick and dirty” matching function. When matching a pair of entities, we follow the merge strategy described in the introduction. Moreover, in this phase, we map each row in the BM to a 2-dimensional point resulting in a very compact representation for the behavior with some information loss. This mapping allows for very fast computations on the behavior data of both the original and merged entities.

Phase 2: Once the candidate matches are generated, the following phase, which is the core of our approach, is to perform the accurate (yet more expensive) matching of entities. Accurate matching of the candidate pair of entities \((A, B)\) is achieved by first modeling the behavior of entities \(A\) and \(B\) and \(AB\) using a statistical generative model, where \(AB\) is the entity representing the merge of \(A\) and \(B\). The estimated models’ parameters are then used to compute the matching score.

In addition to the above statistical modeling technique, we also propose an alternative heuristic technique that is based on information theoretic principles for the accurate matching phase. This alternative technique relies on measuring the increase in the level of compressibility as we merge the behavior data of pairs of entities. While, to some extent, it is less accurate than the statistical technique, it is computationally more efficient.

Phase 3: The final filtering and conflict resolution phase is where the final matches are selected. In our implementation, a simple filtering threshold, \(t_f\), is applied to exclude low-scoring matches.

This approach has already been accepted for publication [23].

3. REFERENCES


