GDR: A System for Guided Data Repair [*]

Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville,and Mourad Ouzzani
Purdue University, West Lafayette, IN 47907, USA
{myakout, ake, neville, mourad}@cs.purdue.edu

ABSTRACT

Improving data quality is a time-consuming, labor-intensive and often domain specific operation. Existing data repair approaches are either fully automated or not efficient in interactively involving the users. We present a demo of GDR, a *Guided Data Repair* system that uses a novel approach to efficiently involve the user alongside automatic data repair techniques to reach better data quality as quickly as possible. Specifically, GDR generates data repairs and acquire feedback on them that would be most beneficial in improving the data quality. GDR quantifies the data quality benefit of generated repairs by combining mechanisms from decision theory and active learning. Based on these benefit scores, groups of repairs are ranked and displayed to the user. User feedback is used to train a machine learning component to eventually replace the user in deciding on the validity of a suggested repair. We describe how the generated repairs are ranked and displayed to the user in a "useful-looking" way and demonstrate how data quality can be effectively improved with minimal feedback from the user.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design

Keywords

data quality, data cleaning, data repair, interactive system

1. INTRODUCTION

There is unanimous agreement among researchers and practitioners alike on the importance of data quality for many real world applications and the unimaginable consequences of making decisions based on inconsistent, inaccurate or incomplete data. For example, poor data quality in retail databases alone costs US consumers $2.5 billion annually [6]. Not to mention the importance of data quality in the healthcare domain. In such critical applications, incorrect information about patients in an Electronic Health Record (EHR) may lead to wrong treatments and prescriptions, which consequently may cause severe medical problems including death. Poor data quality is a fact of life for most organizations [1]. Despite the many existing efforts to automatically improve the quality of the data, domain users will have to be involved in interactively monitoring the repair process. This highlights the increasing need for a system that can combine the best of both; automatically suggesting repairs while efficiently involving the user to guide the cleaning process.

Existing systems for data cleaning are limited to providing tools for data exploration and transformation, where repair actions are explicitly specified by the user. For example, AJAX [9] proposes a declarative language to eliminate duplicates during data transformations. Potter's Wheel [11] combines data transformations with the detection of errors in the form of syntax and irregularities. It uses a sliding-window interface to ease data exploration. As a step forward to efficiently involve the user in the cleaning process, ALIAS and Semandaq were introduced. ALIAS [12] is a tool that is limited to identifying duplicate tuples by interactively learning domain independent similarity functions. Semandaq [7] uses an automated constrained data repairing approach while providing the user with extensive data exploration tools to track the applied repairs. If the user is not happy with the repairs, he would either specify additional constrains (or rules) for repairing or repair the data manually.

A general and recent domain independent approach for improving data quality is to (i) discover and identify some data quality rules (DQRs), and then, (ii) use these rules to derive data repairs for dirty instances that violate these rules. Various techniques have followed this approach for data repairs, e.g., [4, 2, 10]. However, in real-world scenario, this is not sufficient and domain users have to verify the applied repairs, especially, when in critical domains like healthcare.

This demo presents GDR, a *Guided Data Repair*, that tackles the problem of improving the data quality from a more realistic and pragmatic viewpoint than has commonly been the case in this area. Since automated methods for data repairs produce far more repairs than one can expect the user to comment on, techniques for selecting the *most useful* repairs for presentation to the user become of paramount importance. GDR aims at moving the data quality to a *better* state *as quickly as possible*. A machine learning component is used in GDR to learn user feedback, such that, the learning component can replace the user for similar situation, and hence, reducing user involvement.

The key novelty of GDR is a mechanism for providing automatically generated repairs ranked and displayed for the user, such that little user efforts spent on the top ranked repairs would help repairing many dirty instances in the database. To this end, GDR uses the concept of *value of information* (VOI) from decision theory to estimate the benefit of consulting the user for a group of repairs. Afterward, the repairs within a group are displayed following the *active learning* approach, i.e., repairs are order by *uncertainty*, such that user feedback for the top repairs will strengthen the learner pre-

[*]This research was supported by NSF Grant Numbers IIS 0916614 and IIS 0811954, and by Purdue Cyber Center.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.
Copyright 2010 ACM 978-1-4503-0032-2/10/06 ...$10.00.

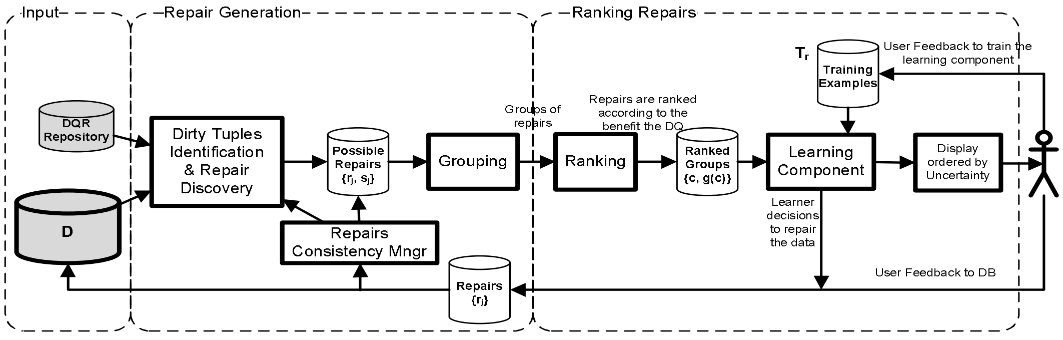

Input
Repair Generation
Ranking Repairs
DQR Repository
D
Dirty Tuples Identification & Repair Discovery
Possible Repairs {r_i, s_i}
Grouping
Repairs Consistency Mngr
Repairs {r_i}
Groups of repairs
Ranking
Repairs are ranked according to the benefit the DQ
Ranked Groups {c, g(c)}
Learning Component
Training Examples
T_r
User Feedback to train the learning component
Display ordered by Uncertainty
Learner decisions to repair the data
User Feedback to DB

Figure 1: GDR Framework.

```
Procedure 1 GDR_Process($D$ dirty database, $\Sigma$ DQRs)
 1: Identify the dirty tuples in $D$ using $\Sigma$.
 2: Generate and store suggested repairs in $PossibleRepairs$ list.
 3: Group repairs appropriately.
 4: while User is available and dirty tuples exist do
 5:     Rank groups of repairs such that the most beneficial come first and
        the user selects group $c$ from the top.
 6:     Repairs in $c$ are labeled by the learner predictions and the user in-
        teractively gives feedback on the suggested repairs, until the user is
        satisfied with the predictions or the user has labeled all repairs in $c$.
 7:     User feedback and learner decisions are applied to $D$.
 8:     Remove rejected repairs from $PossibleRepairs$ and replace as
        needed.
 9:     Check for new dirty tuples and generate repairs as necessary.
10: end while
```

diction most. The objective is to accelerate the process of learning user feedback and minimize user efforts [1].

Our experience with real-life datasets showed that GDR can accurately solve most data quality problems with little guidance from the user. Less important data quality problems which affect a small portion of the database are deferred for further interactions with the user.

2. GDR DESCRIPTION

Figure 1 shows the overall design of the GDR framework.

2.1 System Input

The primary input to the system are:

1- Database (D): The primary input to the system is a dirty database. We consider a database instance D with a relational schema S. Each relation $R \in S$ is defined over a set of attributes $attr(R)$ and the domain of an attribute $A \in attr(R)$ is denoted by $dom(A)$.

2- Data Quality Rules (DQR): The second input is a set of data quality rules Σ in the form of conditional functional dependencies (CFDs) [8]. CFDs are an extension to the standard functional dependencies FDs developed for data cleaning. They have proved to be effective in catching data inconsistencies, which triggered several efforts (e.g. [5]) to facilitate their automatic discovery. GDR allows users to specify a set of CFDs to characterize the semantics of data.

2.2 Overview on GDR Components and Process:

Procedure 1 provides an outline of the main steps of the cleaning process in GDR. GDR *guides* the user to focus his or her efforts on the repairs that would improve the quality faster, while the user *guides* the system to automatically repair the data. This is a continuous feedback process, illustrated in steps 4-10, that runs while there are dirty tuples and the user is available and willing to give feedback.

Dirty Tuples Identification and Repair Discovery: The process starts by identifying the dirty tuples which violate the CFDs. This can be done efficiently using SQL. Afterward, possible repairs for the dirty tuples are generated. To this end, we apply a technique inspired from [4] to resolve CFDs violations by repairs in the form of *value modification*. We assume that for each identified dirty tuple, all of its attribute values are wrong and proceed to find the best possible repairs for each value to satisfy the violated CFDs. When searching for a repair value, we aim at maximizing a repair evaluation function that depends on preferring repairs which (i) would minimally change the data while (ii) cutting down the number of violations. The output from this component is a list of repairs along with the repair evaluation score.

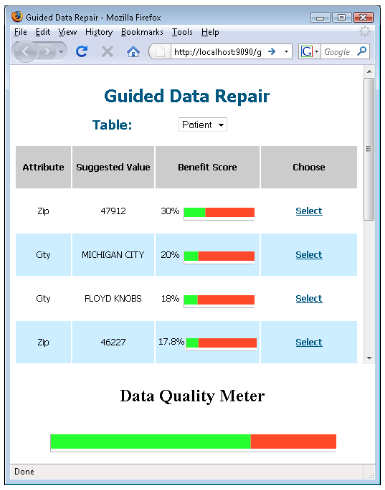
Grouping: Once repairs are generated, they are grouped such that repairs that suggest the same value to a given attribute come together. This grouping helps in exposing the structure of data relationships. There are two reasons for the grouping: (i) Providing a *useful-looking* set of repairs with some common relationships which will be easier for the user to handle and process. (ii) Feeding a machine learning algorithm with a set of instances that have some correlations due to the grouping will increase its prediction accuracy within these groups compared with just providing random, not related instances.

Ranking: Once the repairs are grouped, the ranking component's role is to devise how the repair groups are best presented to the user in a way that could provide the *most benefit* in improving the quality of the data. The ranking component uses the concept of value of information from decision theory to principally reason about this ordering. We quantify the data quality loss (DQLoss) using the amount of violations with respect to the CFDs. Then, we assign a *benefit score* to each repair group by computing the difference between the DQLoss values obtained before and after consulting the user, respectively. Since the user feedback is not known before hand, we leverage the repair evaluation score computed by the repair discovery component and use it as an estimation probability for the possible user feedback. Finally, the user picks one of the top ranked groups c to work on.

Learning: In step 6, the user is involved in an interactive active learning session to train a classification model in predicting user's feedback on the repairs within a group c. The learning component includes a machine learning algorithm, or simply a learner, to learn a classification model. The learner orders the repairs such that the repairs that would most benefit from labeling (in the form of user feedback) come first. More precisely, we order the repairs by their model uncertainty, because labeling the most uncertain repairs will be more beneficial to the learner. We used *random forest* [3] model, where a set of decision trees are learned from random variations of the instances. The uncertainty of a repair is then computed using the entropy of the predicted labels fraction among the trees.

The repairs are displayed to the user along with the learner pre-

[1]Details on the mechanism is in a paper under submission


Guided Data Repair - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:9090/g
Google
Done

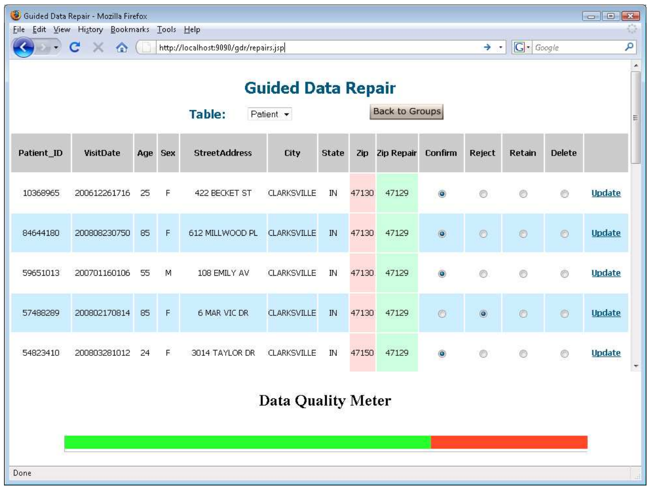
Guided Data Repair

Table: Patient

Attribute	Suggested Value	Benefit Score	Choose
Zip	47912	30%	Select
City	MICHIGAN CITY	20%	Select
City	FLOYD KNOBS	18%	Select
Zip	46227	17.8%	Select

Data Quality Meter

(a) Screen containing the repair groups displayed for the user. Each repair group represent changing the value of a given attribute for some tuples.


Guided Data Repair - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:9090/gdr/repairs.jsp
Google
Back to Groups
Done

Guided Data Repair

Table: Patient

Patient_ID	VisitDate	Age	Sex	StreetAddress	City	State	Zip	Zip Repair	Confirm	Reject	Retain	Delete	
10368965	200612261716	25	F	422 BECKET ST	CLARKSVILLE	IN	47130	47129	●	○	○	○	Update
84644180	200808230750	85	F	612 MILLWOOD PL	CLARKSVILLE	IN	47130	47129	●	○	○	○	Update
59651013	200701160106	55	M	108 EMILY AV	CLARKSVILLE	IN	47130	47129	●	○	○	○	Update
57488289	200802170814	85	F	6 MAR VIC DR	CLARKSVILLE	IN	47130	47129	○	●	○	○	Update
54823410	200803281012	24	F	3014 TAYLOR DR	CLARKSVILLE	IN	47150	47129	●	○	○	○	Update

Data Quality Meter

(b) After choosing one of the repair groups, this screeen display the tuples along with the suggested repairs. Here, we selected to repair the Zip code to 47129. The old and suggested value are displayed along with the learner prediction as radio button. The user may correct the prediction by choosing a different radio button.

Figure 2: Screen shots for the repair groups display and the repairs within a group display. The data quality meter is always displayed.

dictions. The user will then give a feedback on the top n_s repairs and inherently correct any mistakes made by the learner. The newly labeled instances in n_s are added to the training dataset T_r 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 user feedback is provided to 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 learned model or the repairs within c are all labeled by the user.

Repair Consistency Manager: In step 7, all decisions on suggested repairs, either made by the user or the learner, are enacted. In step 8, the suggested repairs consistency is maintained by removing the incorrect repairs and replacing those whose corresponding tuples are still dirty. Since a repair may introduce new violations, in step 9, we search for new dirty tuples, and new repairs are generated accordingly.

3. DEMONSTRATION OVERVIEW

In the demo, we use (anonymized) personal address information of patients collected from 74 hospitals. We construct a list of CFD rules to clean address information. These rules have been collected from zip lookup websites. We used a batch uploading feature in the system to upload the list of the rules. Users can also specify new rules. We demonstrate the progress in improving the data quality as the user is providing feedback for the displayed repairs.

Once the system is launched, an initial set of repairs is generated for those tuples that violate the specified rules. We show how the repairs are displayed into groups in Figure 2(a). The groups are ranked using an estimated benefit score. This score represents the percentage increase in the data quality when the user helps GDR with some feedback for the corresponding group.

Every repair group is identified using the name of the attribute being repaired and the suggested value. The user selects a repair group which will show up with the original tuples along with suggested repair values. The second screen in Figure 2(b) shows the tuples displayed along with the repairs. Also, for each row, the predictions of the learning component on the repairs are displayed as radio buttons, where the actual predictions are the ones that are turned on.

GDR expects from the user one of four labels for each suggested repair : (i) *confirm* the suggested repair, (ii) *reject* the repair and allow GDR to find another repair for the attribute, (iii) *retain* the value in the tuple as it is already correct, or (iv) *delete* the whole tuple. The user may also suggest new value as a repair and GDR will consider such a suggestion as a confirm feedback for the user suggested value.

In this screen (Figure 2(b)), we demonstrate how the user verifies and corrects the learner predictions for the top-k repairs as well as how the learner re-orders repairs on-the-fly and refreshes the predictions for the rest of the repairs. GDR asks the user to provide click away answers on the radio buttons instead of explicitly typing the repair values.

At the lower part of the repair screen, the user will see the progress of improving the data quality through a *data quality meter*. This meter displays the percentage of rule satisfactions to the rule violations. After finishing the work in each repair group, GDR refreshes this meter allowing users to immediately feel the effect of their feedback on the data quality.

4. REFERENCES

[1] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Addison-Wesley.

[2] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *ACM SIGMOD*, 2005.

[3] L. Breiman. Random forests. In *Machine Learning*, 2001.

[4] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: consistency and accuracy. In *VLDB*, 2007.

[5] G. Cormode, L. Golab, F. Korn, A. McGregor, D. Srivastava, and X. Zhang. Estimating the confidence of conditional functional dependencies. In *ACM SIGMOD*, 2009.

[6] L. English. Information quality management: The next frontier. *Information Management Magazine*, 2000.

[7] W. Fan, F. Geerts, and X. Jia. Semandaq: A data quality system based on conditional functional dependencies. In *VLDB*, 2008.

[8] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. In *TODS*, 2008.

[9] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: an extensible data cleaning tool. In *SIGMOD*, 2001.

[10] A. Lopatenko and L. Bravo. Efficient approximation algorithms for repairing inconsistent databases. In *ICDE*, 2007.

[11] V. Raman and J. M. Hellerstein. PotterŠs wheel: An interactive data cleaning system. In *VLDB*, 2001.

[12] S. Sarawagi, A. Bhamidipaty, A. Kirpal, and C. Mouli. Alias: An active learning led interactive deduplication system. In *VLDB*, 2002.