A Novel Cost-Based Model for Data Repairing

(Extended Abstract)

Shuang Hao† Nan Tang‡ Guoliang Li† Jian He† Na Ta† Jianhua Feng†
†Department of Computer Science, Tsinghua University, Beijing, China
‡Qatar Computing Research Institute, HBKU, Qatar
{haos13, hej13, dan13}@mail.thu.edu.cn, ntang@hbku.edu.qa, {liguoliang, fengjh}@tsinghua.edu.cn

***Abstract*—Integrity constraint (IC) based data repairing is typically an iterative process consisting of two parts: detecting and grouping errors that violate given ICs; and modifying values inside each group such that the modified database satisfies those ICs. However, most existing automatic solutions treat the process of detecting and grouping errors straightforwardly (*e.g.,* violations of functional dependencies using string equality), while putting more attention on heuristics of modifying values within each group. In this paper, we propose a revised semantics of *violations* and *data consistency w.r.t.* a set of ICs. The revised semantics relies on string similarities, in contrast to traditional methods that use syntactic error detection using string equality. Along with the revised semantics, we also propose a new cost model to quantify the cost of data repairing by considering distances between strings. We show that the revised semantics provides a significant change for better detecting and grouping errors, which in turn improves both precision and recall of the following data repairing step. We prove that finding minimum-cost repairs in the new model is NP-hard, even for a single FD. We devise efficient algorithms to find approximate repairs.**

I. Introduction

Data cleaning has played an important part in data management. Although involving users in data repairing is necessary in practice, automatic repairing is still valuable, *e.g.,* they can help find possible repairs, which can alleviate the burden of users in the loop of data cleaning.

Consider an FD φ_1 : (country $\rightarrow$ capital), and three tuples t_1(China, Beijing), t_2(**China**, Tokyo), and t_3(**Chine**, Beijing), where errors are highlighted in bold font. Here, (t_1, t_2) violate φ_1. There are two main shortcomings by directly following the definitions of FDs. (i) *Unrobust error detection:* Some erroneous tuple values cannot be captured such as t_3[country] since no tuple has the same country value as t_3; and (ii) *Bad error group:* When associating t_1 with t_2 since they together violate φ_1 and trying to repair one of them, most repairing algorithms are likely to fail since no algorithm can repair t_2[country] from China to Japan.

The above observations call for a revision of data repairing problems that improves both (i) and (ii) discussed above. Intuitively, improving (ii) can increase the precision, and enhancing (i) will advance recall.

In this paper, we propose a novel cost-model for data repairing, motivated by the above two observations. Specifically, we make the following contributions. (1) We propose a new cost-based model. We show that finding the optimal solution is NP-hard. (2) We devise new data repairing algorithms, for handling both a single constraint and multiple constraints. (3) We study optimization techniques to improve the efficiency of data repairing. (4) We conduct extensive experiments, which demonstrate that our method outperforms *state-of-the-art* automatic data repairing approaches in accuracy.

II. Problem Formulation

Consider an instance $\mathcal{D}$ of relation $\mathcal{R}$. To simplify the discussion, we focus on functional dependencies (FDs).

A. Functional Dependencies and Semantics

We use a standard definition for an FD φ: $X \rightarrow Y$. We say that $\mathcal{D}$ *satisfies* φ, denoted by $\mathcal{D} \models \varphi$, if for any two tuples (t_1, t_2) in $\mathcal{D}$, when $t_1[X] = t_2[X]$, $t_1[Y] = t_2[Y]$ holds. Otherwise, we call (t_1, t_2) a violation of φ, if $t_1[X] = t_2$[X] but $t_1[Y] \neq t_2[Y]$.

Moreover, we say that $\mathcal{D}$ is *consistent* $w.r.t.$ a set Σ of FDs, denoted by $\mathcal{D} \models \Sigma$, if $\mathcal{D} \models \varphi$ for each $\varphi \in \Sigma$.

To address the two limitations of automatic repairing algorithms discussed in Section I, we use a similarity function to holistically compare both left and right hand side of constraints. More specifically, given an FD φ: $X \rightarrow Y$, we use a distance function $\mathsf{dist}_\varphi()$ to capture violations. We simply write $\mathsf{dist}()$ when φ is clear from the context. We also write t^φ for $t[X \cup Y]$.

Distance function. For attribute A in $\mathcal{R}$, $\mathsf{dist}(t_1[A], t_2[A])$ indicates how similar $t_1[A]$ and $t_2[A]$ are. There are many known distance functions, *e.g.,* Edit distance, Jaccard distance, and Euclidean distance. In this paper, by default, we use edit distance if $t_1[A]$ and $t_2[A]$ are strings and Euclidean distance if the values are numeric. Any other distance function can also be used. Formally, we have:

$$\mathsf{dist}(t_1[A], t_2[A]) = \begin{cases} Edit(t_1[A], t_2[A]) & \text{string} \\ Eucli(t_1[A], t_2[A]) & \text{numeric} \end{cases} \quad (1)$$

where $\mathsf{dist}(t_1[A], t_2[A])$ is a normalized distance in [0,1].

For t_1^φ and t_2^φ with multiple attributes, their distance is:

$$\mathsf{dist}(t_1^\varphi, t_2^\varphi) = w_l \sum_{A_l \in X} \mathsf{dist}(t_1[A_l], t_2[A_l]) + w_r \sum_{A_r \in Y} \mathsf{dist}(t_1[A_r], t_2[A_r]). \quad (2)$$

where w_l, w_r are weight coefficients in [0,1] and $w_l + w_r = 1$.

Fault-tolerant violation. Two tuples (t_1, t_2) are in a fault-tolerant (FT-) violation *w.r.t.* φ, if (1) $t_1^\varphi \neq t_2^\varphi$; and (2) the distance between them is no larger than a given threshold τ, *i.e.,* $\mathsf{dist}(t_1^\varphi, t_2^\varphi) \leq \tau$. We write $(t_1, t_2) \not\models^* \varphi$ if both (1) and (2) hold (*i.e.,* an FT-violation); otherwise, we write $(t_1, t_2) \models^* \varphi$ if $\mathsf{dist}(t_1^\varphi, t_2^\varphi) > \tau$. A possible method of deciding the threshold

τ is as follows. We first calculate the distance of each pair of tuples $(t_1^\varphi, t_2^\varphi)$ and sort them in ascending order. When the difference between the two adjacent numbers suddenly becomes large, we choose the smaller value τ as the threshold.

Fault-tolerant consistency. A database $\mathcal{D}$ is fault-tolerant (FT-) *consistent* to an FD φ, denoted by $\mathcal{D} \models^* \varphi$, if there do not exist two tuples t_1, t_2 such that $(t_1, t_2) \not\models^* \varphi$. We say that $\mathcal{D}$ is FT-consistent *w.r.t.* a set Σ of FDs, denoted by $\mathcal{D} \models^* \Sigma$, if $\mathcal{D} \models^* \varphi$ for each $\varphi \in \Sigma$.

Consistency vs FT-consistency. The FT-consistency semantics means that if two tuples on attributes $X \cup Y$ are similar but not identical, they are FT-violated. It is readily to see that in the case of $\tau \geq w_r|Y|$ (where $|Y|$ is the number of attributes in Y), if a database $\mathcal{D}$ is FT-consistent *w.r.t.* φ, it must be also consistent. Considering $w_l = 1, w_r = 0, \tau = 0$, if $\mathcal{D}$ is consistent, it must be also FT-consistent, and vice versa.

B. Problem Statement

Close-world data repair model. The repaired value for an attribute A must come from the active domain of A.

Valid tuple repair. Repairing a tuple from t to t' (t' may not be in $\mathcal{D}$ originally) is called a *valid tuple repair*, if for any FD φ, there exists a tuple t'' in $\mathcal{D}$ such that $t'^\varphi = t''^\varphi$. In other words, the whole tuple t' may be new to $\mathcal{D}$; however, the projected values t'^φ must exist in $\mathcal{D}$ originally.

Valid database repair. A database $\mathcal{D}'$ is a *valid database repair* of $\mathcal{D}$ *w.r.t.* a set Σ of FDs in FT-consistent semantics, if $\mathcal{D}'$ is FT-consistent *w.r.t.* Σ only via valid tuple repairs.

Repair cost. For each tuple t in $\mathcal{D}$, suppose t' is the corresponding tuple in the repaired database $\mathcal{D}'$. Obviously, if $t = t'$, the repair cost is 0; otherwise we use the distance functions to quantify the repair cost as below.

$$\mathsf{cost}(t, t') = \sum_{A \in \mathcal{R}} \mathsf{dist}(t[A], t'[A]). \tag{3}$$

Naturally, the repair cost of database $\mathcal{D}$ is

$$\mathsf{cost}(\mathcal{D}, \mathcal{D}') = \sum_{t \in \mathcal{D}} \mathsf{cost}(t, t'). \tag{4}$$

Problem statement. Given a database $\mathcal{D}$ and a set Σ of FDs defined on $\mathcal{D}$, the *data repairing problem* is to find a valid repair $\mathcal{D}'$ of $\mathcal{D}$ such that $\mathcal{D}'$ is FT-consistent and $\mathsf{cost}(\mathcal{D}, \mathcal{D}')$ is minimum among all valid repaired databases.

III. Algorithm Framework

A. Single FD

Graph model. Given a database $\mathcal{D}$ and a single FD φ, we first transform it to a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Each vertex $v \in \mathcal{V}$ is a tuple, and an undirected edge between two vertices u and v indicates that (u, v) is an FT-violation *w.r.t.* φ. Each edge is associated with a weight, denote by $\omega(u, v) = \mathsf{cost}(u^\varphi, v^\varphi)$.

Repairing based on a maximal independent set. A maximal independent set $\mathcal{I}^*$ of $\mathcal{G}$ has some salient properties. (1) Tuples in $\mathcal{I}^*$ have no FT-violations and are thus FT-consistent. (2) Any tuple that is not in $\mathcal{I}^*$ must have FT-violations with at least one tuple in $\mathcal{I}^*$. Based on these two features, we can repair the database using a maximal independent set as follows. Given a maximal independent set $\mathcal{I}^*$, for any tuple x not in $\mathcal{I}^*$, let $\mathcal{N}(x) = \{v | v \in \mathcal{I}^* \text{ and } (v, x) \in \mathcal{E}\}$ denote the neighbor set of x in $\mathcal{I}^*$. We can repair x to any tuple $v \in \mathcal{N}(x)$ (by modifying x^φ to v^φ in order to resolve the violation between them) and the repairing cost is $\omega(x, v)$. Naturally we want to repair x to v with the minimal cost, *i.e.,* $\omega(x, v) \leq \omega(x, u)$ for any $u \in \mathcal{N}(x)$, and the cost of repairing x given $\mathcal{I}^*$ is $\mathsf{cost}(x|\mathcal{I}^*) = \omega(x, v)$. We enumerate every tuple not in $\mathcal{I}^*$, and iteratively repair all tuples in $\mathcal{D}$ using $\mathcal{I}^*$ and get a repaired database $\mathcal{D}'$. The total cost of repairing $\mathcal{D}$ given $\mathcal{I}^*$ is $\mathsf{cost}(\mathcal{D}, \mathcal{D}') = \mathsf{cost}(\mathcal{D}|\mathcal{I}^*) = \sum_{x \in (\mathcal{V} \setminus \mathcal{I}^*)} \mathsf{cost}(x|\mathcal{I}^*)$.

Optimal Repairing. We enumerate every maximal independent set, select the independent set ($\mathcal{I}^B$) with the minimal repairing cost, *i.e.,* $\mathsf{cost}(\mathcal{D}|\mathcal{I}^B) \leq \mathsf{cost}(\mathcal{D}|\mathcal{I}^*)$, called the best maximal independent set, and use $\mathcal{I}^B$ to repair the database.

Repairing Algorithms. We prove that using the best maximal independent set $\mathcal{I}^B$ to repair $\mathcal{D}$ is optimal and NP-hard, and propose an expansion-based algorithm to find the optimal result and a greedy algorithm to find an approximate result.

B. Multiple FDs

For multiple FDs with common attributes, we need to repair them jointly to avoid repairing conflicts on the common attributes. We prove that this problem is also NP-hard. We also propose an algorithm to find the optimal result and two heuristic algorithms to find approximate results [4].

IV. Experimental Study

Datasets. We used one real-world dataset HOSP and one synthetic dataset TAX. (1) HOSP was taken from the US Department of Health Services (http://www.hospitalcompare.hhs.gov/). We used 20k records with 19 attributes and 9 FDs. (2) TAX was generated by a generator http://www.cs.utexas.edu/users/ml/riddle/data.html. Each record represented an individual's address and tax information. It had 9 FDs. Errors were produced by adding noises with a certain rate $e\%$, *i.e.,* the percentage of dirty cells over all data cells *w.r.t.* all FDs.

Baselines. We compared with NADEEF [2], Unified Repair Model (URM) [1] and Llunatic [3] for FD repairs. Experiments show that our approach significantly outperforms existing automatic repair algorithms in both precision and recall.

Acknowledgment

Guoliang Li was supported by 973 Program of China (2015CB358700), NSF of China (61422205, 61373024, 61632016, 61472198, 61661166012), Shenzhou, Tencent, FDCT/116/2013/A3, and MYRG105 (Y1-L3)-FST13-GZ.

References

[1] F. Chiang and R. J. Miller. A unified model for data and constraint repair. In *ICDE*, 2011.

[2] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: a commodity data cleaning system. In *SIGMOD*, 2013.

[3] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The llunatic data-cleaning framework. *PVLDB*, 6(9), 2013.

[4] S. Hao, N. Tang, G. Li, J. He, N. Ta, and J. Feng. A novel cost-based model for data repairing. In *TKDE*, 2016.