RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies

Bernardo Breve
bbreve@unisa.it
University of Salerno
Fisciano, Salerno, Italy

Loredana Caruccio
lcaruccio@unisa.it
University of Salerno
Fisciano, Salerno, Italy

Vincenzo Deufemia
deufemia@unisa.it
University of Salerno
Fisciano, Salerno, Italy

Giuseppe Polese
gpolese@unisa.it
University of Salerno
Fisciano, Salerno, Italy

ABSTRACT

A missing value represents a piece of incomplete information that might appear in database instances. Data imputation is the problem of filling missing values by means of consistent data with respect to the semantic of the entire database instance they belong to. To overcome the complexity of considering all possible candidates for each missing value, heuristic methods have become popular to enhance execution times, while keeping high accuracy. This paper presents RENUVER, a new data imputation algorithm relying on relaxed functional dependencies (RFDs) for identifying value candidates best guaranteeing the integrity of data. More specifically, the RENUVER imputation process focuses on the RFDs involving the attribute whose value is missing. In particular, they are used to guide the selection of best candidate tuples from which to take values for imputing a missing value, and to evaluate the semantic consistency of the imputed missing values. Experimental results on real-world datasets highlighted the effectiveness of RENUVER in terms of both filling accuracy and execution times, also compared to other well-known missing value imputation approaches.

1 INTRODUCTION

With the advent of big data, the presence of missing values inside database instances has been widely recognized as an important problem to tackle [1]. Several application contexts might require the absence of this type of inconsistencies inside their datasets. For instance, machine learning processes could not provide good accuracy scores if trained on data containing many missing values. In general, it is not possible to infer reliable knowledge using datasets with incomplete information [18]. For this reason, particular efforts have been devoted to the data imputation problem. The latter is faced through the definition of techniques for treating missing values, and it is also considered as one of the fundamental tasks in data cleaning [8]. The identification of the best values in a dataset to impute the missing ones is an extremely complex task, since it entails the evaluation of all possible combinations in the value distribution. Many approaches existing in the literature rely on the idea that a missing value can be imputed by another value belonging to the same population, aiming at preserving the overall integrity of data [12]. Moreover, heuristics have been widely employed in order to efficiently select values to be imputed, without worsening the accuracy of the imputation [11].

This paper presents RENUVER (RFD basEd NUll ValuE Repairer), a data imputation algorithm relying on Relaxed Functional Dependencies (RFDs) for imputing missing values within a relational database instance. The concept of RFD extends the FD definition by admitting the possibility to use approximate comparison methods with respect to the equality constraint, and the possibility for the dependency to hold on a subset of data [7]. In general, RFDs provide metadata suitable for detecting and repairing many types of errors, such as duplicates, outliers, and constraint violations [15]. Thus, they could be potentially used for identifying suitable candidate values for replacing missing ones in the data imputation process. Moreover, the less restrictive constraints of the RFD definition enable a broader analysis of the correlations among attributes. For this reason, RFDs might potentially suit better than canonical FDs in the data imputation process. RENUVER exploits RFDs for: i) identifying the candidate tuples useful for the imputation of missing values, ii) ranking candidate tuples based on their similarity with respect to the tuples containing missing values, and iii) evaluating each imputation aiming to guarantee the semantic consistency of the whole dataset.

RENUVER represents an heuristic approach aiming at perform an imputation process in polynomial time. In particular, the worst-case time complexity of RENUVER is $O(n^2 \cdot m \cdot |\Sigma| \cdot (k \cdot m \cdot |\Sigma| + k\log k))$, where n, m represent the number of tuples and the number of attributes of the dataset, respectively; $|\Sigma|$ indicates the number of RFDs holding on the dataset, and k represents the average number of possible candidate tuples exploitable for imputing a missing value. The effectiveness of RENUVER has been evaluated on real-world datasets in terms of precision, recall, F1-measure, and execution time. In order to extract RFDs, we relied on an existing RFD discovery algorithm [6], since the problem of discovering RFDs is out of the scope of this paper. Moreover, we introduce a novel method for the automatic evaluation of data imputation results, which permits to judge the imputed values even in case of different syntactical representations. Evaluation results demonstrate that RENUVER outperforms other data imputation approaches [14, 20, 23].

The paper is organized as follows: Section 2 reviews the literature concerning data imputation approaches. Section 3 provides preliminary notions on RFDs. Section 4 defines the data imputation problem. The RENUVER algorithm is described in Section 5, whereas an experimental evaluation measuring its effectiveness is presented in Section 6. Finally, conclusions and further research are reported in Section 7.

2 RELATED WORK

In the last decade, several solutions to the data imputation problem have been proposed. Among them, there is a class concerning the application of linear regression mechanisms. The linear regression model proposed in [26] for numerical missing values aims to solve two among the most common problems arising with data imputation mechanisms working on numerical values: *the sparsity problem*, which deals with the possibility that the number of tuples without a missing value (a.k.a. complete tuple) might not be enough to guarantee a precise imputation process, and *the data heterogeneity problem*, which deals with the fact that different tuples might not belong to the same regression model. Thus, instead of assuming the same regression model for all tuples, the authors aim to learn a model that is valid only for a single complete tuple and those most similar to it. In this way, it is no longer necessary to exclusively rely the imputation process on the values belonging to the first k-nearest complete tuples. Consequently, the process is completely dependent on the results of the regression model. Instead, a multivariate regression model (MRL) is used in REMIAN, which has been defined to address the problem of real-time missing value imputation in error-prone environments [17]. It is able to dynamically adapt the MRL model parameters in order to detect and reject anomalies in an efficient manner, and then incrementally updates the model parameters upon the arrival of new data over streams. Moreover, in [22] regression models have been employed in an alternative way to impute missing values. In this work, rather than trying to directly infer the missing values, the authors propose to first predict distances between missing values and complete ones, and then impute values by means of inferred distances. Thus, they defined an approximation algorithm to the maximum distance likelihood problem for imputing a missing value, after proving its NP-hardness.

Similarly to the approaches mentioned above, in what follows we discuss several clustering-based techniques, which can also be used for the imputation of missing values over numerical datasets. For instance, Nikfalazar *et al.* propose an imputation algorithm based on fuzzy clustering techniques [19]. In particular, it first averages attributes with no missing values to determine an initial imputation, and then identifies an appropriate number of clusters for the application of fuzzy clustering. Applying this technique to the entities within the dataset enables the acquisition of information such as the membership degree and the centroids, thanks to which the initial imputation can be updated with the most suitable value derived from the previous steps.

Instead, Wang *et al.* present a clustered adversarial matrix factorization-based framework for the imputation of structured missing values [24]. The methodology allows for the identification of a dimensional subspace consistent with the clustering structure, easing the process of transferring knowledge among data points within the same cluster. The framework encourages the imputed values to have a similarly distributed probability of the complete values so that by evaluating the joint distribution of the imputed value it is possible to evaluate the difference from the ground truth values: a small joint distribution would indicate a conspicuous difference between the imputed values and the true ones.

Another class of approaches for data imputation relies on machine learning techniques. The framework Holoclean, proposed in [20], exploits machine learning techniques for repairing errors in structured datasets. It allows for the treatment of many types of inconsistencies such as duplicate, incorrect, and missing values. This is made possible through the automatic generation of a probabilistic model, which extrapolates the features representing the uncertainty in the dataset and uses them for characterizing a probabilistic graphical model. The application of statistical learning and probabilistic inference is then responsible for repairing the errors. Moreover, the consistency of data entries can be guaranteed by specifying a set of integrity constraints.

The first two classes of approaches discussed above aim at solving the data imputation problem by means of a model inferring properties among data in a supervised/unsupervised fashion. They turn out to be very fast, even though for their own nature, they can only be applied over numerical datasets. On the contrary, RENUVER allows for the imputation of numerical, textual, and categorical data, since it treats each missing value according to the data domain it belongs to. On the other hand, the methodology used in Holoclean allows also for the imputation of textual and categorical values. Furthermore, Holoclean firstly introduces the usage of metadata, i.e., Denial of Constraints, for supporting the imputation of missing values. However, the exploitation of metadata in the imputation process is marginal in Holoclean's logic, since they are merely used as integrity constraints that imputed values have to satisfy. Instead, metadata can be exploited also to generate candidate values, as done in RENUVER with RFDs. In general, RFDs represent properties holding on the (quasi-)entire dataset, yielding the possibility of capturing the semantic structure of data.

In fact, RFDs have already been considered in the data imputation context. For instance, Bohannon *et al.* introduce a specific type of RFDs, called Conditional Functional Dependencies (CFDs), capable of capturing the overall correctness of data [3]. The general structure of CFDs yields the possibility of capturing the semantics of data, opening their usage for data cleaning purposes. Moreover, although the authors have not proposed any proper data imputation approach, they defined a set of SQL queries for detecting CFD violations, enabling the possibility to exploit them for checking the integrity of an imputed dataset. However, the acquisition of this class of RFDs from data requires an expensive process involving intensive manual effort [13], relying on this type of metadata. Instead, another data imputation algorithm relying on RFDs was introduced by Song *et al.* [23]. It relies on *differential dependencies* (DDs), another specific type of RFDs [21], which exploits similarity rules in order to offer tolerance on value variations. To enrich the data imputation process, the authors studied the problem of maximizing the number of imputed missing values. In particular, the authors presented 4 different algorithms for treating missing values, i.e. an integer linear programming, its approximation, a randomized algorithm, and its derandomized version, called Derand. The latter is identified by the authors as the ideal solution for imputing multiple missing values with a deterministic bound of approximation.

Among all the discussed proposals, Derand represents the one most similar to RENUVER, since they both make use of RFDs for the generation of candidates and for verifying the result of the imputation process. However, RENUVER exploits RFDs also for selecting the most suitable candidate for the imputation of a missing value among all the possible ones. On the contrary, Derand applies a probabilistic approach for performing such selection.

Table 1: A summary of symbols used throughout the paper.

Symbol	Description
$\mathcal{R}$	Relational database schema
R	Relation schema
r	Relation instance
n	Number of tuples in r
m	Number of attributes in r
r'	Relation instance after the imputation process
$\hat{r}$	Relation instance of tuples with missing values
X, Y, Z	Attribute sets
A, B, C	Attributes
a, b, c	Attribute values
t	Tuple of r
$dom(A)$	Attribute domain
$t[A]$	Projection of t onto A
$t[X]$	Projection of t onto X
$t[A] = \_$	Missing value of tuple t on attribute A
φ	RFD$_c$
Σ	Set of RFD$_c$s
Σ'	Set of non-key RFD$_c$s
Φ	Set of distance constraints
ϕ	Distance constraint
δ_A	Distance function on $dom(A)$
Σ'_A	Set of RFD$_c$s for imputing $t[A] = \_$
ρ^i_A	Cluster of RFD$_c$s with RHS threshold i for imputing $t[A] = \_$
$\Lambda_{\Sigma'_A}$	Set of clusters ρ^i_A for imputing $t[A] = \_$
$\Lambda_{\Sigma'}$	Set of $\Lambda_{\Sigma'_A}$
p	Distance pattern between two tuples
$T_{candidate}$	Set of candidate tuples
k	Size of a set of candidate tuples $T_{candidate}$
$dist$	Distance value computed between two tuples according to an RFD$_c$

3 PRELIMINARIES

Before presenting the proposed approach, let us briefly introduce the definitions of FDs and RFDs. For sake of clarity, Table 1 summarizes the notations used within definitions and procedures throughout the paper.

Definition 3.1 (Functional Dependency). Given a relational database schema $\mathcal{R}$, and $R = \{A_1, \ldots, A_m\}$ one of its relation schemas, and a tuple $t \in r$, we use $t[A_i]$, with $0 \leq i \leq m$, to denote the projection of t onto A_i; similarly, for a set of attributes $X = \{A_{i_1}, \ldots, A_{i_k}\}$, with $1 \leq k \leq m$, $t[X] \in dom(A_{i_1}) \times \ldots \times dom(A_{i_k})$ represents the projection of t onto X, also denoted with $\Pi_X(t)$. An FD on $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y), with $X, Y \subseteq attr(R)$, such that, given an instance r of R, $X \rightarrow Y$ is satisfied in r if and only if for each pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. The sets of attributes X and Y are named Left-Hand-Side (LHS) and Right-Hand-Side (RHS) of the FD, respectively.

With respect to FD definition, the RFD generalizes the comparison paradigm, by including similarity/distance-based comparisons between tuple projections, also admitting the possibility for a dependency to hold only on a subset of tuples. The latter can be defined through either a *coverage measure*, quantifying the portion of the dataset on which a dependency holds or a *condition* restricting the domain on which a dependency can hold [7]. Since the proposed approach exploits only RFDs relying on a similarity/distance-based tuple comparison method, in what follows we provide only the definition of this type of RFDs, known as RFD$_c$. For a more general definition of RFD, see [7].

Definition 3.2 (RFD$_c$). Given a relational database schema $\mathcal{R}$, and $R = \{A_1, \ldots, A_m\}$ one of its relation schemas, an RFD$_c$ φ on $\mathcal{R}$

$$X_{\Phi_1} \rightarrow Y_{\Phi_2} \quad (1)$$

Table 2: A sample of the Restaurant dataset.

	Name	City	Phone	Type	Class
t_1	Granita	Malibu	310/456-0488	Californian	6
t_2	Chinois Main	LA	310-392-9025	French	5
t_3	Citrus	Los Angeles	213/857-0034	Californian	6
t_4	Citrus	Los Angeles	_	Californian	6
t_5	Fenix	Hollywood	213/848-6677	_	5
t_6	Fenix Argyle	_	213/848-6677	French (new)	5
t_7	C. Main	Los Angeles	_	French	5

where

- $X, Y \subseteq attr(R)$;
- Φ_1 contains (for each attribute $X_i \in X$) a constraint $\phi_i[X_i]$ that can be used to determine whether pair of tuples with values in $dom(X_i)$ are "similar" enough (likewise for each attribute $Y_j \in Y$ with $\phi_j[Y_j] \in \Phi_2$). More specifically, each $\phi_i[X_i]$ ($\phi_j[Y_j]$ resp.) requires the specification of a similarity/distance function defined on the domain of X_i (Y_j, resp.), an operator, and a threshold setting the boundaries for the satisfaction of the constraint;

holds on a relation instance r (denoted by $r \models \varphi$) if and only if for each pair of tuples $(t_1, t_2) \in r$ for which $t_1[X]$ and $t_2[X]$ satisfy the constraint $\phi_i[X_i]$ for each $X_i \in X$, then $t_1[Y]$ and $t_2[Y]$ satisfy the constraint $\phi_i[Y_i]$ for each $Y_i \in Y$.

For sake of simplicity, in the following examples, we apply a more compact notation for the constraints, showing only the operator and the numeric threshold associated with each attribute.

Example 3.3. Let us consider the sample relation shown in Table 2, derived from a database of restaurants in USA. Within this database, each tuple represents a restaurant providing information about its name, address, city, phone number, type of cuisine, and class. The latter is a numeric id associated to the type of cuisine. This dataset is the result of a data integration process, hence some restaurants might be duplicated. On such dataset, the following RFD$_c$ holds:

$$\varphi_4 : \mathsf{Name}_{(\leq 4)} \rightarrow \mathsf{Phone}_{(\leq 1)}$$

which states that, if two restaurants have a similar name, then they also have a similar phone number. This should be true despite the names and/or the phone numbers of restaurants being written in different ways or using different abbreviations. Thus, the boundary defining how much two tuples can be considered similar on a given attribute is represented by the threshold reported on the subscript of the attribute.

From a theoretical point of view, RFD$_c$s permit to use any type of similarity/distance functions, e.g., edit distance, abs differences, and so forth. However, they are usually inherited from the functions involved in the automatic RFD$_c$ discovery process [6]. For the scope of this proposal, without loss of generality, we can consider RFD$_c$s with a single attribute on the RHS, and the associated constraint ϕ_2. In particular, we considered ϕ_2 composed of a distance function, the operator $\leq$, and a distance threshold. Given an RFD$_c$ φ, we defined functions $LHS(\varphi)$ and $RHS(\varphi)$, returning the attributes on the LHS and the one on the RHS of φ, respectively. Furthermore, we also specify the function $RHS_{th}(\varphi)$, which returns the distance threshold associated to the constraint ϕ_2 associated to the RHS attribute of φ. A particular type of RFD$_c$ is the *key*-RFD$_c$, which is defined in the following.

Definition 3.4 (Key RFD$_c$). Given a relation schema R, and an instance r of R, an RFD$_c$ $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ is said to be *key* if and only if φ holds on r ($r \models \varphi$), but there is no pair of distinct tuples $(t_1, t_2) \in r$, for which $t_1[X]$ and $t_2[X]$ satisfy all the constraints in $\Phi_1[X]$.

4 THE PROPOSED IMPUTATION APPROACH

In this section, we formalize the data imputation problem by defining some of its underlying concepts, then describing the basics of the proposed imputation approach. Let us start defining the concept of missing value.

Definition 4.1 (Missing value). Given a relation schema R, defined over a set of attributes $attr(R)$, an instance r of R, an attribute $A \in attr(R)$, and a tuple $t \in r$, a *missing value* of tuple t on the attribute A, denoted as $t[A] = \_$, is such that $t[A]$ is null.

Here, r is said to be an *incomplete instance*, and $\hat{r} \subseteq r$ is such that each $t \in \hat{r}$ is an *incomplete tuple*.

Missing values entail the general missing value imputation problem, which is formally defined as follows.

Definition 4.2 (Missing value imputation problem). Given a relation schema R, and an instance r of R, for every tuple $t \in r$ and every attribute $A \in attr(R)$ for which $t[A] = \_$, the imputation problem consists of finding a plausible value $a \in dom(A)$, such that the database instance r' resulting from the imputation process preserves the semantic consistency of the original database.

A missing value imputation approach also requires the application of constraints for evaluating the semantic consistency at the end of the imputation process, i.e., for verifying whether the imputed values are plausible. The proposed approach exploits RFDs to both guarantee the verification of the semantic consistency, and to drive the searching of meaningful candidates for all missing values.

Definition 4.3 (Semantically consistent imputation). Given a relation schema R, defined over a set of attributes $attr(R)$, an instance r of R, and a set of RFD$_c$s, Σ, holding on r ($r \models \Sigma$), an instance r' of R resulting from an imputation process I over the instance r, denoted as $r' = I(r)$, is *semantically consistent* iff $r' \models \Sigma$.

Example 4.4. Let us consider the sample relation shown in Table 2, and the RFD$_c$

$$\varphi_0 : \mathsf{Phone}_{(\leq 0)} \rightarrow \mathsf{City}_{(\leq 10)}$$

which states that, if two restaurants have the same phone number, then they are placed in a city with a similar name. Assuming that we imputed t_7[Phone] with the value t_1[Phone], then the imputation process would not be semantically consistent, since the tuple pair (t_1, t_7) violates the previously mentioned RFD$_c$. This is due to the fact that t_1 and t_7 share the same value on the attribute Phone (i.e., edit distance = 0), but the correspondent values on the attribute City overcome the distance threshold (i.e., edit distance > 10).

One of the possible strategies that could guarantee the semantic consistency of the imputation process is to find candidate values for $t[A] = \_$ by considering a set $T_{candidate} \subseteq r$ of *plausible* candidate tuples from which to take values for imputing $t[A]$, such that $\forall t_k \in T_{candidate}$, $t_k[A] \neq \_$ and t_k *is similar* to t on some attributes beyond A.

In what follows we define the criteria used by RENUVER for deciding when a tuple can be considered as a plausible candidate, which is based on RFD$_c$s.

Definition 4.5 (Plausible candidate tuple). Given a missing value $t[A]$=_ over a database instance r of a relation schema R, and an RFD$_c$ $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$ holding on r, a tuple $t' \in r$ can be considered as a *plausible candidate tuple* for imputing $t[A]$ according to φ iff t and t', are similar according to the constraints in Φ_1.

Example 4.6. Let us consider the RFD$_c$ φ_0 shown in Example 3.3 and the sample relation shown in Table 2. The only candidate tuple for imputing t_6[City] according to φ_0 is t_5, since it is the only one with a phone number satisfying the LHS constraint of φ (t_6[Phone] and t_5[Phone] are equal). Thus, it is possible to use t_5[City] for imputing t_6[City], i.e., *Hollywood*.

The candidate tuple generation process performed according to Definition 4.5 has to be generalized in order to perform the imputation process on tuples containing more than one missing value, and for each $t \in \hat{r}$.

Missing value imputation for a tuple. Let R be a relational schema defined over a set of attributes $attr(R)$, r an instance of R, t a tuple of r, $Z \subset attr(R)$ a set of attributes such that for each $A \in Z$ $t[A] = \_$, and Σ a set of RFD$_c$s holding on r. An imputation process for t consists of selecting a plausible candidate tuple t_j for each $A \in Z$ such that $t[A] = \_$, so that $t[A]$ can be set equal to $t_j[A]$. However, when for a $t[A] = \_$ it is not possible to identify a plausible candidate tuple guaranteeing a semantic consistent imputation (see Definition 4.3), it is better to leave $t[A]$ unimputed. Although this strategy has been widely applied in other approaches [23], it yields to another important issue that RENUVER deals with, i.e., minimizing the number of non-imputed values.

It is worth noting that an imputed tuple t could itself become a candidate tuple for imputing another tuple $t' \in \hat{r}$.

5 THE RENUVER ALGORITHM

RENUVER takes in input a dataset r and the set of RFD$_c$s Σ holding on it, and performs the following operations:

(a) *Data pre-processing.* It extracts the set of tuples $\hat{r}$ with one or more missing values from the dataset, and it successively constructs Σ' by removing from Σ the *key*-RFD$_c$s, since they are not useful for the imputation process.

(b) *RFD$_c$ selection.* Given a tuple $t \in \hat{r}$ with a missing value on attribute A, RENUVER selects from Σ' the set of RFD$_c$s Σ'_A suitable for generating candidate values for $t[A]$, i.e., those with A as RHS attribute. The RFD$_c$s in Σ'_A are then partitioned into a set of clusters $\Lambda_{\Sigma'_A} = \{\rho_A^{RHS_{th_1}}, \ldots, \rho_A^{RHS_{th_n}}\}$ based on the RHS thresholds. The clusters define the order in which the RFD$_c$s have to be considered to generate candidate values for $t[A]$, i.e., from lowest to highest threshold values.

(c) *Imputing missing values.* Given a tuple $t \in \hat{r}$ with a missing value on attribute A, and the clusters $\Lambda_{\Sigma'_A}$, RENUVER iteratively performs the following operations:

i selects the plausible candidate tuples t' in r that satisfy the LHS constraints of an RFD$_c$ in $\rho_A^{RHS_{th_i}}$ and such that $t'[A] \neq \_$;

ii computes a distance value *dist* for each plausible candidate tuple t' as the minimum distance between t and t' on the LHS attributes of the RFD$_c$s in $\rho_A^{RHS_{th_i}}$;

iii imputes $t[A]$ with $t'[A]$, where t' is a plausible candidate tuple with the lowest distance value;

iv verifies whether the imputed value for $t[A]$ causes a violation of previously holding RFD$_c$s. In this case, RENUVER selects the next plausible candidate tuple with the lowest distance value.

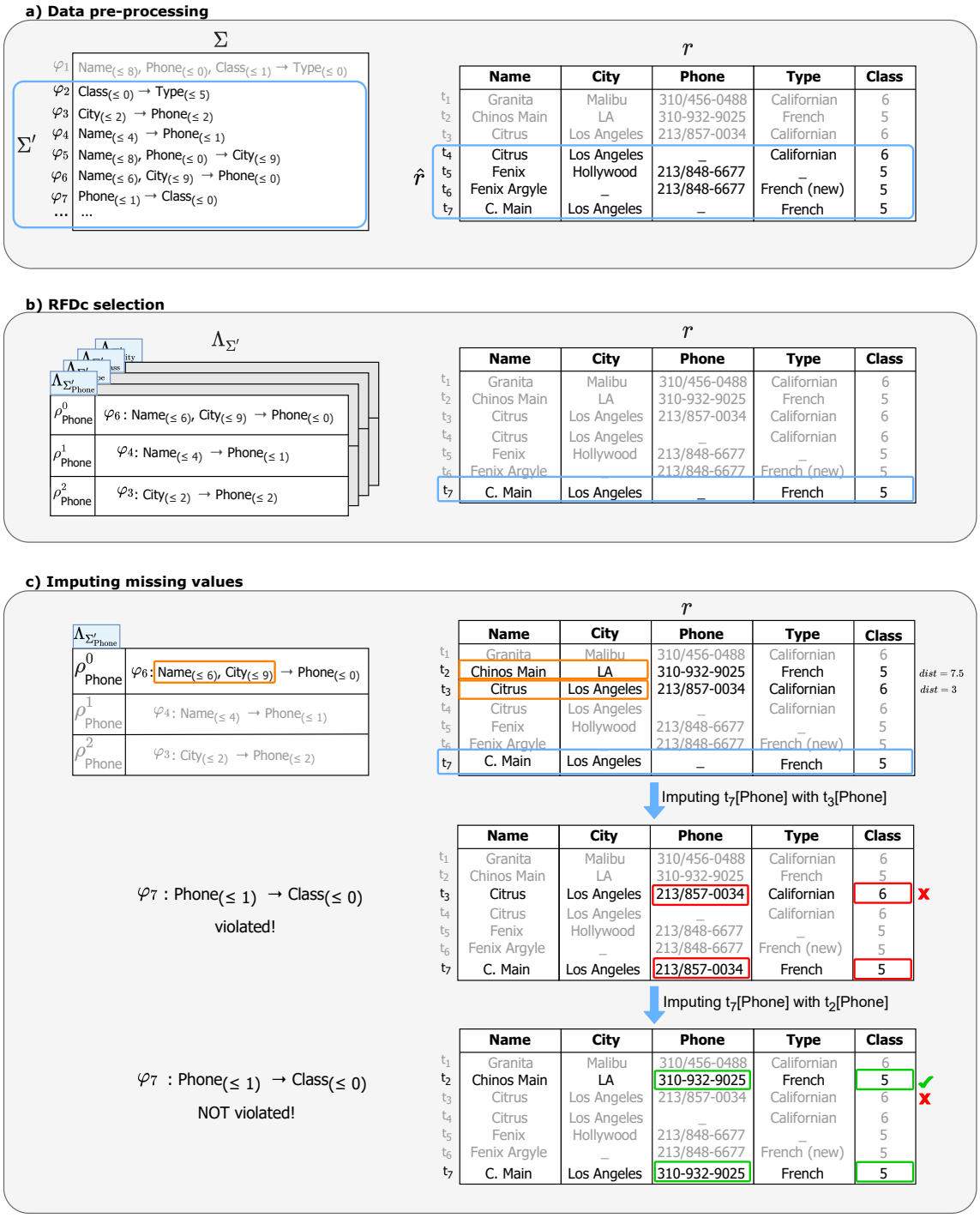


Figure 1: An example of RENUVER imputation on the Restaurant dataset of Table 2.

These operations are repeated for each cluster in $\Lambda_{\Sigma'_A}$, from th_0 to th_n, as long as the imputation is not successful.

Figure 1 shows the results of the aforementioned steps of RENUVER on the sample data in Table 2 and a subset of RFDcs holding on it. During the pre-processing step (a), RENUVER searches for all tuples containing at least one missing value, which are then added to the $\hat{r}$ set, in this case t_4, t_5, t_6, and t_7. In addition, the process of detecting *key*-RFDcs leads to the identification of φ_1, which will therefore not be used during the imputation processes. The remaining RFDcs, deemed *non-key* RFDcs, i.e., from φ_2 to φ_7, are inserted within the Σ' set.

When the RFDc selection step (b) focuses on the imputation of the missing value on the Phone attribute of tuple t_7, all RFDcs having this attribute on the RHS are selected. They correspond to φ_6, φ_4, and φ_3, which are organized in clusters based on their RHS thresholds. Thus, as shown in Figure 1, the RFDcs in the set $\Lambda_{\Sigma'_{\text{Phone}}}$ are separated into 3 clusters, labeled with ρ^0_{Phone}, ρ^1_{Phone}, and ρ^2_{Phone}, respectively. The imputation process step

(c) for t_7[Phone] starts by looking for plausible candidate tuples having a non-missing value on the Phone, and satisfying the LHS of φ_6, since it is the only RFD$_c$ in the cluster with the lowest label value, i.e., ρ^0_{Phone}. The resulting plausible candidate tuples are t_2 and t_3. Based on the LHS attributes of φ_6, i.e., Name and City, RENUVER calculates the distance values between t_2 and t_7, and between t_3 and t_7, which correspond to 7.5 and 3, respectively. Then, it first considers the tuple with lowest distance value, i.e., t_3, for the imputation process. This means that t_3[Phone] = 213/857-0034 is a candidate value t_7[Phone]. However, while verifying the semantically consistency of this imputation, it happens that the RFD$_c$ φ_7 is invalidated. Consequently, RENUVER analyzes the next candidate tuple, i.e., t_2, yielding to impute t_7[Phone] with t_2[Phone] = 310-932-9025. This is semantically consistent with Σ, yielding to the termination of the imputation for t_7[Phone].

Algorithm 1 provides the main procedure of RENUVER. In particular, the pre-processing steps are described from Line 1 to Line 6. Then, the RFD$_c$ selection step is shown from Line 7 to Line 10, where a cycle iterating through each missing value $t[A] = \_$ organizes the RFD$_c$s exploitable for the imputation of each $t[A]$. Finally, the imputation step is performed from Line 11 to Line 14, where all gadgets are employed for imputing each missing value $t[A] = \_$, and verifying the correctness of such imputation (see the sub-procedure (IMPUTE_MISSING_VALUE). After the imputation of a missing value $t[A] = \_$, it is necessary to re-evaluate the set of RFD$_c$s exploitable for the imputation of some other missing values, since the new imputed value can turn a *key* RFD$_c$ into a non-*key* one (Line 14).

Example 5.1. Let us consider the tuple t_4 from Table 2 and the following RFD$_c$, previously identified as a *key*-RFD$_c$ in Example 5.2:

$$\varphi_1 : \text{Name}_{(\leq 8)}, \text{Phone}_{(\leq 0)}, \text{Class}_{(\leq 1)} \rightarrow \text{Type}_{(\leq 0)}$$

Let us suppose we imputed t_4[Phone] with the value "213/857-0034" derived from t_3[Phone]. As a consequence, φ_1 is no longer a key RFD$_c$, since upon the imputation process the tuple pair (t_3,t_4) satisfies the LHS constraints of φ_1, hence φ_1 can be used for the imputation of other missing values.

A detailed description of RENUVER procedures are provided in the following paragraphs.

5.1 Pre-processing

As said above, the first step of RENUVER consists of two data pre-processing operations: the extraction of tuples containing missing values and the identification of *key*-RFD$_c$s. The elimination of key RFD$_c$s during the pre-processing phase contributes to reduce the execution time of RENUVER since these RFD$_c$s do not contribute to the identification of plausible candidates for the imputation process.

Example 5.2. Let us consider the following RFD$_c$s,

$$\varphi_1 : \text{Name}_{(\leq 8)}, \text{Phone}_{(\leq 0)}, \text{Class}_{(\leq 1)} \rightarrow \text{Type}_{(\leq 0)}$$
$$\varphi_2 : \text{Class}_{(\leq 0)} \rightarrow \text{Type}_{(\leq 5)}$$

holding on the dataset of Restaurants whose sample is reported in Table 2. The RFD$_c$ φ_1 asserts that two restaurants having similar names, same phone, and similar classes, must also be of the same type. Referring to Table 2, it is possible to verify that there is no pair of distinct tuples satisfying the LHS constraints of φ_1, making it a *key*-RFD$_c$. Conversely, for φ_2 there are at least two tuples (e.g., t_3 and t_4) satisfying the LHS constraint in Table 2.

Algorithm 1 RENUVER

INPUT: a relation instance r, a set of RFD$_c$s Σ
OUTPUT: an imputed relation instance r

1: **let** $\Sigma' \leftarrow \{\varphi \mid \varphi \in \Sigma \wedge$ *is not a Key-RFD$_c$*$\}$
2: **let** $\hat{r} \leftarrow \emptyset$
3: **let** $\Lambda_{\Sigma'} \leftarrow \emptyset$
4: **for each** $t \in r$ **do**
5: **if** t *has at least one missing value* **then**
6: $\hat{r} \leftarrow \hat{r} \bigcup \{t\}$
7: **for each** *missing value* $A \in t$ **do**
▷ RFD$_c$ selection procedure
8: **let** $\Sigma'_A \leftarrow \{\varphi \mid \varphi \in \Sigma' \wedge \varphi : X_{\Phi_1} \rightarrow A_{\phi_2}\}$
9: **let** $\Lambda_{\Sigma'_A} \leftarrow \{\rho^i_A \mid \rho^i_A$ *holds all* φ with $RHS_{th}(\varphi) = i\}$
10: $\Lambda_{\Sigma'} \leftarrow \Lambda_{\Sigma'} \bigcup \Lambda_{\Sigma'_A}$
11: **for each** $t \in \hat{r}$ **do**
12: **for each** *missing value* $A \in t$ **do**
13: r = IMPUTE_MISSING_VALUE$(r, A, t, \Sigma', \Lambda_{\Sigma'_A})$
14: $\Sigma' \leftarrow \{\varphi \mid \varphi \in \Sigma \wedge$ *is not a Key-RFD$_c$*$\}$
15: **return** r

Pre-processing steps are included in the main procedure of RENUVER (Algorithm 1). In particular, the definition Σ' represents a set of RFD$_c$s where key dependencies have been filtered-out (Line 1). Moreover, by iteratively analyzing all tuples in the instance r, RENUVER stores all tuples having at least a missing value in the set $\hat{r}$ (Lines 4-6). In this way, the whole imputation process can start and it can properly focus the attention on RFD$_c$s and data properly filtered.

5.2 RFDc selection

For each a missing value $t[A] = \_$ (Algorithm 1 - Line 7), RENUVER collects all RFD$_c$s that can be used to impute $t[A]$, allowing for the identification of all plausible candidates for its imputation. In particular, the RFD$_c$s are selected based on their RHS attribute, i.e., all RFD$_c$s with attribute A as RHS are added to a set Σ'_A (Line 8).

Successively, $\Lambda_{\Sigma'_A}$ is defined as a set of clusters ρ^i_A, each grouping all RFD$_c$s having the same threshold on the RHS attribute (Line 9).

Example 5.3. Let us consider tuple t_4 extracted from Table 2. Since t_4 has a missing value on the attribute Phone, it is possible to consider the following two RFD$_c$s for its imputation:

$$\varphi_3 : \text{City}_{(\leq 2)} \rightarrow \text{Phone}_{(\leq 2)}$$
$$\varphi_4 : \text{Name}_{(\leq 4)} \rightarrow \text{Phone}_{(\leq 1)}$$

Consequently, RENUVER adds φ_3 and φ_4 to the same set of RFD$_c$s exploitable for the imputation of attribute Phone. However, the two RFD$_c$s belong to two different clusters because of their RHS threshold, i.e., 2 and 1, respectively.

5.3 Imputing missing values

Algorithm 2 summarizes the procedure for imputing a missing value of a tuple. It takes as input a database instance r of a relation schema R, an attribute $A \in attr(R)$, a tuple t having a missing value on A, and the set of RFD$_c$ clusters $\Lambda_{\Sigma'_A}$ (see Algorithm 1). The procedure iteratively selects a cluster in descending order of their associated RHS thresholds (Line 1). Based on the currently examined RFD$_c$s cluster, it retrieves all plausible candidate tuples

Algorithm 2 IMPUTE_MISSING_VALUE

INPUT: an instance r, an attribute A, a tuple $t \in \hat{r}$ with $t[A] = \_$, a set of RFD$_c$s Σ', a set of clusters $\Lambda_{\Sigma'_A}$ containing RFD$_c$s exploitable for imputing $t[A]$
OUTPUT: an imputed relation instance of r for the missing value $t[A] = \_$ or the same instance of r passed as input if the imputation for $t[A] = \_$ is not performed

1: **for each** $\rho^i_A \in \Lambda_{\Sigma'_A}$ in descending order of RHS threshold **do**
2: **let** $T_{candidate} \leftarrow$ FIND_CANDIDATE_TUPLES(r, t, ρ^i_A)
3: *Sort* $T_{candidate}$ *in ascending order of distance values*
4: **for each** $(t_j, dist_j) \in T_{candidate}$ **do**
5: $t[A] \leftarrow t_j[A]$
6: **if** IS_FAULTLESS$(r,\ A,\ t,\ \Sigma')$ **then**
7: **return** r
8: **else**
9: $t[A] \leftarrow \_$
10: **return** r

Algorithm 3 FIND_CANDIDATE_TUPLES

INPUT: an instance r, a tuple $t \in \hat{r}$ with $t[A] = \_$, a set of RFD$_c$s ρ^i_A with A as RHS attribute and i as RHS threshold.
OUTPUT: A set of pairs $(t_j, dist_j)$, where t_j is a candidate tuple valid to impute A and d_j is the computed distance between t and t_j

1: **let** $T_{candidate} \leftarrow \emptyset$
2: **for each** $t_j \neq t \in r \wedge t_j[A] \neq \_$ **do**
3: **let** p *be a distance pattern computed between* t, t_j
4: **let** $dist_{min} \leftarrow +\infty$
5: **for each** $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2} \in \rho^i_A$ **do**
6: **if** p *satisfies* Φ_1 *on* LHS(φ) **then**
7: **let** $dist \leftarrow \frac{\sum_{B \in X} p[B]}{|X|}$
8: **if** $dist < dist_{min}$ **then**
9: $dist_{min} \leftarrow dist$
10: $T_{candidate} \leftarrow T_{candidate} \bigcup (t_j, dist_{min})$
11: **return** $T_{candidate}$

from r (Line 2). At this point, all candidate tuples have already been evaluated according to a *distance value* with respect to t (see Algorithm 3), and they are sorted in ascending order based on this distance value (Line 3) and singularly examined for the imputation process (Line 4). More specifically, the procedure tries to use a candidate tuple t_j previously selected from the set of all candidate tuples $T_{candidate}$; but before concluding the imputation process, it is necessary to verify whether the semantic consistency of the whole relation instance is preserved after imputing $t[A]$ with the value $t_j[A]$ (Line 6), and if the evaluated value for $t[A]$ turned out to be fault for $t[A]$, the procedure returns the missing value flag, i.e., _ (Line 9).

In the following paragraphs, we provide the details of the sub-procedures mentioned above.

Generation of candidate tuples. Algorithm 3 defines the procedure for finding the candidate tuples used to impute the missing value of $t[A]$. The goal here is to process all tuples in r, by evaluating their distance with respect to t. To this end, it uses two underlying concepts: *distance pattern* and *distance value*, whose definitions are provided in the following.

Definition 5.4 (Distance pattern). Given a relation schema $R = \{A_1, \ldots, A_m\}$, a relation instance r of R, a tuple pair (t, t_j), and $\delta_{A_1}, \ldots, \delta_{A_m}$ a list of distance functions on $dom(A_1), \ldots, dom(A_m)$, respectively. A *distance pattern* p for the tuple pair (t, t_j) is a vector of m elements $[p_1, \ldots, p_m]$ such that for each $1 \leq i \leq m$, $p_i = \_$ if $t[A_i] = \_$ or $t_j[A_i] = \_$, and $p_i = \delta_{A_i}(t[A_i], t_j[A_i])$ otherwise.

In other words, the distance pattern defines the distance between t and t_j for each attribute in R, which is computed through a distance function suitable to the domain of each attribute. In particular, RENUVER uses the absolute difference for numerical values, the edit distance for string values [25], and the equality constraint for boolean values, as distance functions, so it is able to work with string, int, float, double, and boolean attributes.

Example 5.5. Let us consider the tuple pair (t_5, t_6) of Table 2. The distance pattern is a vector of 5 elements whose values can be computed using the edit distance for attributes Name and Phone, and the absolute distance for attribute Class. Since the attributes City and Type have a missing value on both t_5 and t_6, it is not possible to calculate a distance value for them. Thus, the corresponding value on the distance pattern is flagged with _, and the pattern is the vector: $[7, \_, 0, \_, 0]$.

Given a distance pattern p and an RFD$_c$ $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$, it is possible to state that p satisfies the constraint Φ_1 iff for each attribute $B \in X$, we have $p[B] \neq \_$ and $p[B]$ contains a value less or equal than the distance threshold associated to B.

Definition 5.6 (Distance value). Given a relation schema R, an instance r of it, a tuple pair (t, t_j) of r, an RFD$_c$ $\varphi : X_{\Phi_1} \rightarrow A_{\phi_2}$, and a distance pattern p satisfying the constraint Φ_1. A *distance value* $dist$ between tuples t and t_j according to φ can be computed as

$$dist = \frac{\sum_{B \in X} p[B]}{|X|} \quad (2)$$

Example 5.7. Let us consider tuple t_6 from Table 2. Suppose we are following the imputation process for t_6[City] by considering tuple t_5 as a plausible candidate tuple. As stated on Example 5.5, the distance pattern p between t_5 and t_6 is $[7, \_, 0, \_, 0]$. Among the RFD$_c$s exploitable for imputing t_6[City] the following one is considered:

$$\varphi_5 : \mathsf{Name}_{(\leq 8)}, \mathsf{Phone}_{(\leq 0)} \rightarrow \mathsf{City}_{(\leq 9)}$$

Since p satisfies the constraints on the LHS of φ_5, the distance value between t_5 and t_6 can be computed according to Equation 2:

$$dist_{(t_5,t_6)} = \frac{p[\mathsf{Name}] + p[\mathsf{Phone}]}{|\{\mathsf{Name}, \mathsf{Phone}\}|} = \frac{7+0}{2} = 3.5$$

By considering the definitions provided above, for each tuple $t_j \neq t$ having a non-missing value on attribute A, Algorithm 3 computes their distance pattern (Line 3). Then, a cycle iterates through each RFD$_c$ φ (Lines 5-9), and analyzes its LHS constraints (Φ_1) to check if the distance pattern p satisfies them (Line 6). The actual distance between t_j and t is computed according to Equation 2, yielding a distance value $dist$. In particular, the iteration over the RFD$_c$s (Lines 5-9) aims at finding the minimum possible distance value $dist_{min}$ achievable for t_j with respect to the considered RFD$_c$s. The candidate tuple t_j and the minimum distance value $dist_{min}$ are then included in the candidate set $T_{candidate}$ (Line 10).

Algorithm 4 IS_FAULTLESS

INPUT: a database instance r, an attribute A, a tuple t imputed on attribute A, a set of RFD$_c$s Σ'
OUTPUT: **true** if imputation is semantically consistent, **false** otherwise

1: **for each** $\varphi : X_{\Phi_1} \rightarrow B_{\phi_2} \in \Sigma' \quad s.t.\ A \subseteq X$ **do**
2: **for each** $t_i \in r$ **do**
3: **let** p *be a distance pattern computed between* t, t_j
4: **if** p *satisfies* Φ_1 *on* LHS(φ) **then**
5: **if** p *does not satisfy* Φ_1 *on* RHS(φ) **then**
6: **return false**
7: **return true**

Example 5.8. Let us consider the imputation process for t_7[Phone]. Suppose that during the identification of plausible candidate tuples t_2 and t_3 are selected through the following RFD$_c$:

$$\varphi_6 : \mathsf{Name}_{(\leq 6)}, \mathsf{City}_{(\leq 9)} \rightarrow \mathsf{Phone}_{(\leq 0)}$$

Let the distance patterns of the tuple pairs (t_2, t_7) and (t_3, t_7) be $p_{(t_2,t_7)} = [6, 9, \_, \_, 0]$ and $p_{(t_3,t_7)} = [6, 0, \_, \_, 1]$, respectively. Then, by applying Equation 2, the distance value between the tuples of each tuple pair are:

$$dist_{(t_2,t_7)} = \frac{p_{(t_2,t_7)}[\mathsf{Name}] + p_{(t_2,t_7)}[\mathsf{City}]}{|\{\mathsf{Name}, \mathsf{City}\}|} = \frac{6+9}{2} = 7.5$$

$$dist_{(t_3,t_7)} = \frac{p_{(t_3,t_7)}[\mathsf{Name}] + p_{(t_3,t_7)}[\mathsf{City}]}{|\{\mathsf{Name}, \mathsf{City}\}|} = \frac{6+0}{2} = 3$$

According to the computed distances, we can establish the order in which the plausible candidate tuples will be considered, i.e., first t_3, and if the verification of the imputation process fails, then t_2.

Verifying the imputation process. Algorithm 4 shows the procedure to verify whether the imputation of a tuple t on attribute A introduces semantic inconsistencies. To this end, the algorithm selects all available RFD$_c$s φ having the imputed attribute A on the LHS (Line 1), and searches for any tuple t_i satisfying LHS constraints of φ, but violating the RHS one when compared to t (Lines 3-5) (see Definition 4.3).

Example 5.9. Let us consider two tuples, t_3 and t_7 extracted from Table 2, and suppose we imputed t_7[Phone] with the value "213/857-0034", derived from t_3[Phone], having selected t_3 as the first candidate tuple in Example 5.8. During the verification process, the following RFD$_c$ is considered:

$$\varphi_6 : \mathsf{Phone}_{(\leq 1)} \rightarrow \mathsf{Class}_{(\leq 0)}$$

The evaluation of the imputed tuple t_7 according to φ_6 allows to verify that the value pair (t_3[Phone], t_7[Phone]) satisfies the LHS of φ_6. However, when evaluating the RHS of φ_6 concerning the value pair (t_3[Class], t_7[Class]), the similarity/distance function (in this case, the absolute distance) returns a value *1*, which is above the threshold provided by φ_6 on attribute Class. φ_6 no longer holds on the entire dataset, and the value "213/857-0034" is not suitable for t_7[Phone].

6 EVALUATION

As shown in [23], the problem of imputing missing values is NP-compete. To this end, RENUVER exploits RFD$_c$s in order to find the best candidates for each missing value in polynomial time. In particular, the *pre-processing* - step (a) requires $O(n + |\Sigma|)$ time, with n tuples analyzed to extract the ones containing at least one missing value, and $|\Sigma|$ RFD$_c$s evaluated to determine and remove all key RFD$_c$s. The *RFD$_c$ selection* - step (b) requires $O(n \cdot m \cdot |\Sigma|)$ time, since in the worst case $n \cdot m$ missing values are considered (with n number of tuples and m number of attributes), and for each of them, all RFD$_c$s should be analyzed (with $|\Sigma|$ number of RFD$_c$s) in order to consider the ones useful to extract possible candidate values. Finally, the most complex step is the one for *imputing missing values* - step (c), which requires to extract candidate tuples by pairing tuples containing a missing value with all the remaining ones (i.e., $O(n^2 \cdot m)$). Moreover, for each value, it is necessary to analyze all RFD$_c$s collected in the previous step by computing the distance value (i.e., $|\Sigma| \cdot m$ for each missing value, in the worst case), yielding a $O(n^2 \cdot m^2 \cdot |\Sigma|)$) time complexity for finding candidate tuples. Furthermore, by considering a set of candidate tuples for each missing value and each RFD$_c$ in Σ (i.e., $O(n^2 \cdot m|\Sigma|)$), having size $k << n$, entails sorting them in $O(k \log k)$ according to the computed distance value, using the first one that does not produce a violation for any RFD$_c$ (k candidate tuples in the worst case, each of them requiring the computation of distance values for each RFD$_c$, i.e, $|\Sigma| \cdot m$ in the worst-case). Thus, the whole complexity of step (c) is $O(n^2 \cdot m \cdot |\Sigma| \cdot (k \cdot m \cdot |\Sigma| + k \log k))$, where n represents the number of tuples, and m represents the number of attributes. With this in mind, in what follows, we present the experimental results obtained for evaluating the imputation efficiency and accuracy of RENUVER. We also compared RENUVER with other approaches exploiting different imputation strategies.

6.1 Evaluation settings

Implementation details and hardware. We benchmarked RENUVER against a kNN-based approach [14], a holistic-machine learning-based approach [20], and a differential dependencies guided approach [23], respectively. All evaluations were performed under the same conditions. The sessions have been executed separately on an iMac Pro with an Intel Xeon W 8-core @3.2 GHz, 32GB RAM.

Datasets. The compared algorithms have been evaluated on four real-world datasets whose statistics are reported in Table 3. In order to perform an accurate comparison between the imputed values and the expected ones, missing values have been artificially injected. This process is achieved by randomly selecting a certain percentage of values in the dataset to be turned into missing values. In particular, we performed the evaluation sessions by varying the percentage of missing values in the range [1%, 5%] on each considered dataset. Furthermore, to avoid an arrangement of missing values in favor of one algorithm over another, for each missing rate we produced five injected datasets, yielding a total of twenty-five variants of the same dataset. The metrics adopted for the comparison are then averaged over each missing rate. Finally, in order to evaluate time and memory limits, we also considered the Physician dataset[1], since it contains a suitable mix of textual and numerical attributes, and has a sufficiently high number of attributes to stress the performances of the considered imputation approaches.

Relaxed functional dependencies. For evaluating how much the imputation process varies accordingly to the selected RFD$_c$s, both the RFD$_c$ based imputation algorithms RENUVER and Derand were executed on different sets of RFD$_c$s, extracted by setting the threshold limits for attribute comparison to set of values

[1] https://data.medicare.gov/data/physician-compare

Table 3: Details of the considered real-world datasets.

Dataset	Attributes [#]	Tuples [#]	#RFD$_c$s thr=3	thr=6	thr=9	thr=12	thr=15	# missing values [1%]	[2%]	[3%]	[4%]	[5%]
Restaurant	6	864	25	124	445	1262	1961	52	104	155	206	259
Cars	9	406	802	3696	10454	23902	38522	37	73	110	146	183
Glass	11	214	1511	5763	11648	18748	27445	24	47	71	94	118
Bridges	13	108	1830	5381	8961	14711	18844	14	28	42	56	70

{3, 6, 9, 12, 15}, respectively. Notice that, a greater threshold value admits a less restrictive similarity correlation among the attribute values. The sets of RFD$_c$s for the considered datasets have been obtained by executing the discovery algorithm presented in [6].

Evaluation process. In order to provide an accurate assessment of the capabilities offered by the various imputation algorithms, the achieved results were validated not only on the basis of strict equality with the original value, but also of their semantic similarity (e.g., despite not being equal, values "213/848-6677" and "213-848-6677", do represent the same semantic value). Carrying out this type of analysis on the imputation results would entail a considerable manual effort, making it extremely difficult to perform on large datasets. For this reason, we designed a rule-based framework for the automatic verification of the imputation results, which has been applied with the same configurations to evaluate results of all compared approaches. The methodology exploits a rule file that specifies the admissible values for each attribute in the dataset. Plausible values can be specified in three different ways:

- *Value set*: it is possible to specify what are the values that have the same meaning, e.g., "new york", "new york city", and "ny" all express the same concept, so they are all placed in the same set. If both the imputed value and the expected value belong to the same set, then the imputation will be considered correct.
- *Custom designed regex*: it is possible to specify which structural variations are admissible in order to consider the imputation still valid. It is the case of the aforementioned attribute Phone, for which it is possible to specify that, although the numerical part must be the same, it is admissible to employ different separators. Thus, the imputed value only needs to preserve the same characters (or digits) as the expected value, according to what is expressed in the regex.
- *Delta variation*: it is possible to specify for a numerical attribute the delta from the expected value, e.g., attribute Horsepower on the *cars* dataset admits a delta of 25 horsepower both in positive and in negative. If the imputed value falls within the admissible delta variation from the expected value, then the imputation will be considered correct.

The rule files for each dataset have been manually defined after a painstaking evaluation of each attribute value distribution. This methodology guarantees a completely automatic evaluation process, allowing for the verification of a missing value imputation also in terms of syntactical variations. Furthermore, this methodology helps to save time in the analysis of results derived from different executions and configurations. To the best of our knowledge, this is the first time that this type of approach for validation has been applied in the data imputation context.

Evaluation metrics. The effectiveness of the data imputation approaches have been evaluated by considering three different metrics: *precision*, *recall*, *F1-measure*.

In this context, *precision* defines how many missing values are correctly imputed with respect to the number of imputed missing values. This parameter resembles a reliability score, indeed it keeps track of how the algorithm performs when it has to decide whether to impute with an uncertain value or leave the missing value. In details, let *true* be the correctly imputed missing values and *imputed* be all the imputed missing values, then: $precision = \frac{|\text{true} \bigcap \text{imputed}|}{|\text{imputed}|}$

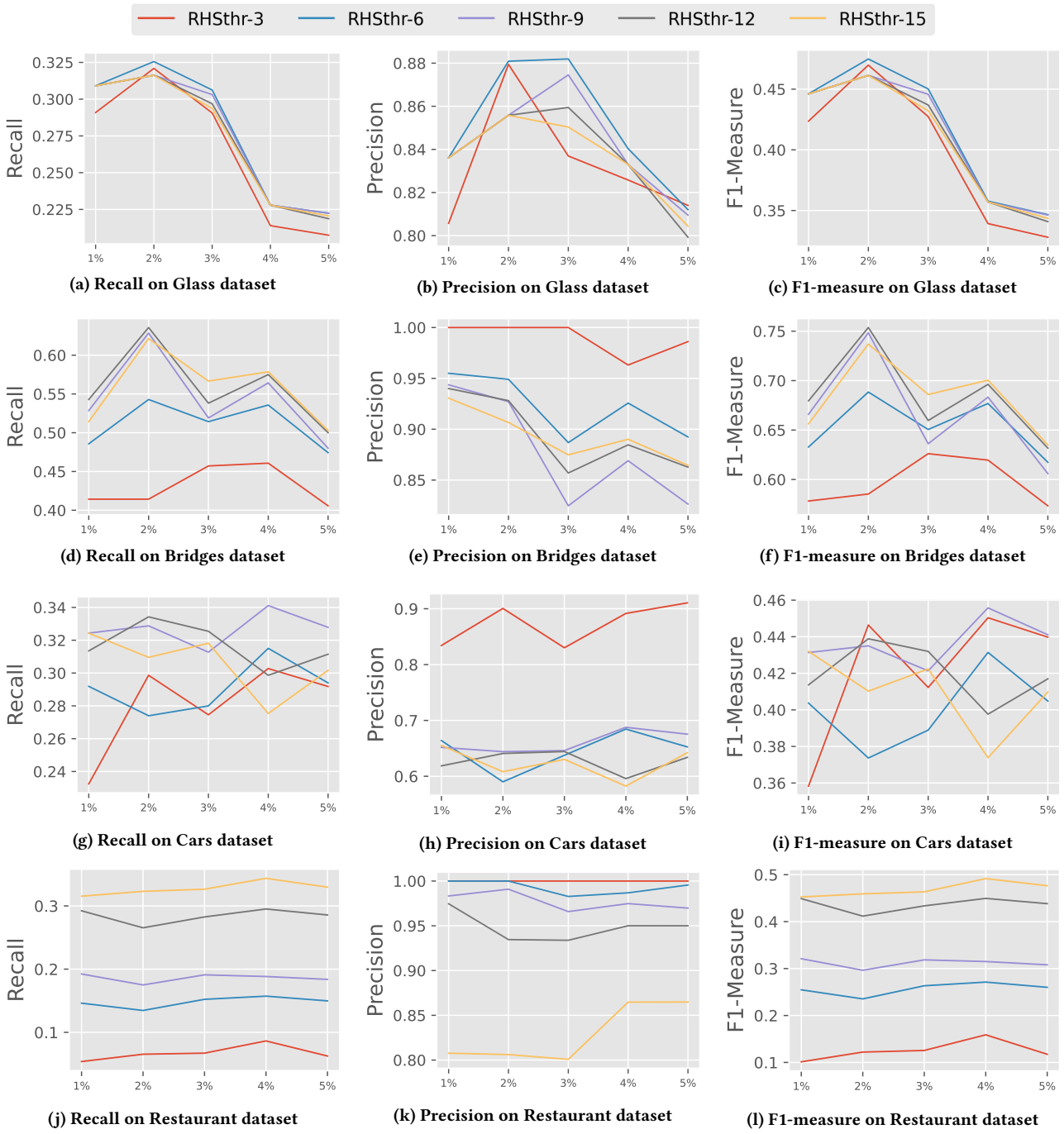
The *recall* represents the fraction correctly imputed missing values. In detail, let *missing* be the missing values in the dataset and *true* the correctly imputed missing values at the end of the imputation process, then: $recall = \frac{|\text{true} \bigcap \text{missing}|}{|\text{missing}|}$

The *F1-measure* is computed by combining the *precision* and *recall* according to the following formula:
$F1\text{-}measure = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

6.2 Qualitative evaluation

We first evaluated how many missing values were correctly imputed by RENUVER considering different sets of RFD$_c$s. In particular, we carried out different tests by varying the maximum RHS distance threshold that the RFD$_c$s set should provide. Considering five threshold limits, we wanted to verify whether by varying the thresholds also the scores based on the adopted metrics underwent the stated previously expected change. In fact, as said before, by adopting higher thresholds, an increase in the number of imputed missing values is expected, to the detriment of a decrease in accuracy, due to a less restrictive constraint provided by the RFD$_c$s.

Figure 2 shows the results achieved for Glass, Bridges, Cars, and Restaurant datasets, respectively. For the Glass dataset, the variation of RFD$_c$ threshold limits does not produce a serious impact on all metrics as the missing rate increases. This is due to the fact that the values of the dataset are closed decimal numbers, and in some cases, the RFD$_c$ threshold values do not capture the correlation among data. Thus, as the missing rate increases, the amount of imputed and erroneously imputed values increases. Instead, the Bridges dataset highlights the aforesaid theoretical assumptions. In fact, despite providing high precision scores, a low RHS threshold limit severely afflicts the recall score, and consequently, the F1-measure. On the contrary, a higher threshold limit improves the recall score, since more missing values can be imputed. However, the precision scores are lower, since with a higher threshold a broader set of candidate tuples are admissible, which does not always correspond to a correct imputation. For this dataset, the F1-measure score provides a good overview of the threshold limits trade-off. These metrics highlights that a high RHS threshold limit still provides better overall results than a lower limit. However, despite the application of threshold limits, on the Bridges dataset, a higher missing rate yields lower

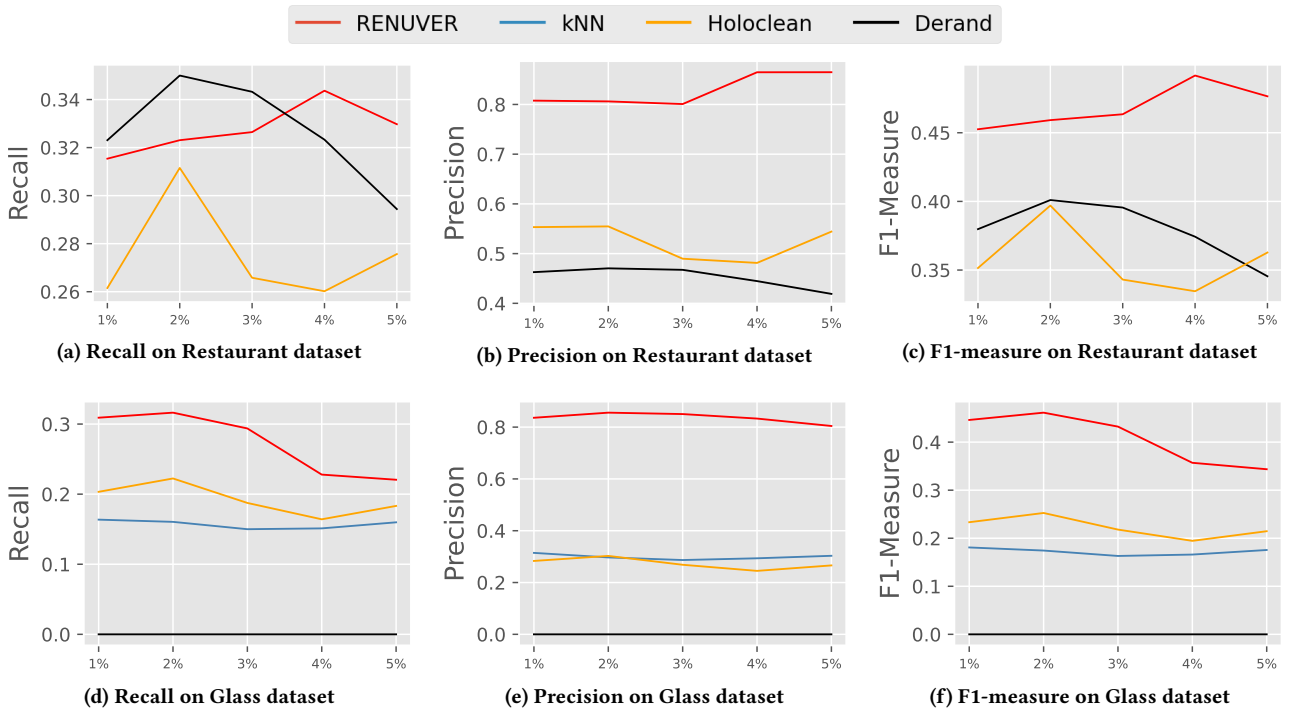


(a) Recall on Glass dataset
(b) Precision on Glass dataset
(c) F1-measure on Glass dataset
(d) Recall on Bridges dataset
(e) Precision on Bridges dataset
(f) F1-measure on Bridges dataset
(g) Recall on Cars dataset
(h) Precision on Cars dataset
(i) F1-measure on Cars dataset
(j) Recall on Restaurant dataset
(k) Precision on Restaurant dataset
(l) F1-measure on Restaurant dataset

Figure 2: Performances of RENUVER by varying the max RHS distance thresholds and with different missing rates.

scores. Concerning the Cars dataset, results are comparable in terms of precision scores with the Bridges dataset. In fact, also in this case, lower RHS distance threshold limits provide higher precision scores. However, unlike the Bridges dataset, the application of RFD$_c$s with a lower maximum RHS threshold does not cause an excessive drop in recall scores. Thus, this test proved that running RENUVER on the Cars dataset, by exploiting RFD$_c$s with lower RHS threshold limits, yields better results in terms of the trade-off between the number of the imputed missing values and the correctly imputed ones. Finally, results for the Restaurant dataset follow the same trends of the ones collected for the Bridges dataset. In fact, Figures 2j-2l strongly highlight the increase of the Recall as the RHS threshold increases, but yielding to a decrease of the Precision. Nevertheless, since Precision scores are high enough, higher RHS thresholds appear to be the best performing ones, in terms of F1-measure, for the Restaurant dataset.

In general, for all considered qualitative metrics it is not unusual to record a non-monotonic trend. In fact, a slight variability in missing rates does not impact on the amount of available candidate tuples. Thus, the overall achievable score strictly depends on the attributes exhibiting the missing values.



(a) Recall on Restaurant dataset

(b) Precision on Restaurant dataset

(c) F1-measure on Restaurant dataset

(d) Recall on Glass dataset

(e) Precision on Glass dataset

(f) F1-measure on Glass dataset

Figure 3: Comparison among RENUVER, Derand, Holoclean and kNN on real-world datasets by varying the missing rates.

6.3 Comparative evaluation

We performed a comparative evaluation of RENUVER respecting to three other solutions, whose methodologies cover most of the classes of approaches for data imputation discussed in Section 2. In particular, RENUVER has been compared to the Derand algorithm presented in [23], which also exploits RFD_cs during the imputation process. Furthermore, we also compared RENUVER to Holoclean, a Holistic machine-learning solution that has set the state-of-the-art in the data imputation context [10]. Finally, we also compared RENUVER to the kNN-based imputation approach presented in [14]. Both RENUVER and Derand have been implemented in Java, while kNN and Holoclean[2] have been implemented in Python.

Concerning the metadata, for the comparison between RENUVER and Derand, we used the same set of RFD_cs (i.e., by using the following threshold limits: 15 for the Restaurant and Glass datasets, and 3 for the Physician dataset). As for Holoclean, its execution requires a set of Denial Constraints, which were also retrieved through an automatic discovery process [2, 9].

RENUVER, Derand, and Holoclean were tested on the Restaurant, Glass, and Physician datasets. Instead, the comparison with kNN has been performed on the Glass dataset, since it contains only numerical values. In fact, kNN solutions are specifically designed to perform imputation on this type of dataset. All experimental sessions were performed on the same sets of missing values.

Figure 3a shows the obtained results in terms of recall, by varying the missing rate over the x scale. Notice that RENUVER provides almost a monotonous trend, with the recall score improving as the missing rate increases, but for the 5% missing rate, where RENUVER has recorded a slight decrease of recall score. On the contrary, both Derand and Holoclean follow a non-monotonous trend. Derand, initially obtained a higher recall score, but considerably dropping as the missing rate increases. Instead, Holoclean obtained recall scores that are always lower than both RENUVER and Derand, following an unpredictable trend.

Figure 3b shows the precision scores obtained by RENUVER, Derand, and Holoclean. The results highlight the ability of RENUVER to accurately identify a suitable imputation, outperforming the compared solutions with a precision score always above the 0.8 value. On the contrary, Derand and Holoclean achieved worse precision scores, with maximum precision values of 0.55 and 0.47, respectively. Figure 3c summarizes the results in terms of F1-measure. Results further emphasize the overall better ability of RENUVER to perform a more coherent imputation process, achieving better scores on all missing rates.

Figures 3d, 3e, and 3f provide the results of the comparison between RENUVER and all considered approaches (including kNN) on the Glass dataset. As opposed to the previous comparison, here the improvements are remarkably higher in this evaluation since RENUVER outperforms the compared approaches on all considered metrics. The worst results are obtained by Derand, since it was not able to fill any missing value for this dataset. Finally, also for this evaluation, the precision scores proved the superiority of RENUVER in accurately identifying whether to impute or restore the missing value, achieving an overall score always above 0.8.

Time and memory requirements. As a final evaluation, we performed stress tests on RENUVER and all compared imputation approaches, aiming to determine their time and memory requirements. To this end, we stopped the execution of algorithms exceeding 48 hours of execution time and/or 30GB of memory consumption, respectively. To do this, we performed two additional evaluation sessions. The first one is focused on the Restaurant dataset, but by considering greater missing rates, i.e., [5%, 10%, 20%, 30%, 40%], whose results are shown in Table 4. We can notice that the fastest approach is Holoclean, whereas

[2]Publicly available on https://github.com/HoloClean/holoclean

Table 4: Performance limits on the Restaurant dataset by varying the missing rates, i.e. [5%, 10%, 20%, 30%, 40%].

Dataset	#Tuples	#Attributes	#Missing val.	#RFD$_c$s	#DCs
Restaurant	864	6	259 (**5%**) 518 (**10%**) 1037 (**20%**) 1555 (**30%**) 2074 (**40%**)	1961	9

Dataset	Approach	Recall	Precision	F1-Meas.	Time	Mem.
Restaurant (varying the missing rate)	RENUVER	0.329	0.864	0.476	14m 29s	1.38 GB
		0.296	0.832	0.437	23m 21s	1.31 GB
		0.294	0.845	0.436	33m 20s	1.36 GB
		0.258	0.828	0.394	36m 37s	1.37 GB
		0.232	0.726	0.349	30m 23s	1.38 GB
	Derand	0.295	0.419	0.345	47h 13m	7.21 GB
		-	-	-	TL	-
		-	-	-	TL	-
		-	-	-	TL	-
		-	-	-	TL	-
	Holoclean	0.275	0.544	0.362	14s	0.99 GB
		0.099	0.218	0.131	15s	0.99 GB
		0.071	0.153	0.095	14s	0.99 GB
		0.064	0.192	0.095	11s	0.78 GB
		0.165	0.419	0.237	10s	0.79 GB

TL: time limit of 48 hours exceeded – ML: memory limit of 30 GB exceeded

Table 5: Performance limits on the Physician dataset by varying the percentage of tuples, i.e. [1%, 2%, 3%, 4%, 5%].

Dataset	#Tuples	#Attributes	#Missing val.	#RFD$_c$s	#DCs
Physician	104 (**1%**) 208 (**2%**) 1036 (**3%**) 2072 (**4%**) 10359 (**5%**)	18	13 (1%) 27 (1%) 135 (1%) 269 (1%) 1319 (1%)	1430 2553 3895 5708 6137	74

Dataset	Approach	Recall	Precision	F1-Meas.	Time	Mem.
Physician (varying the number of tuples)	RENUVER	0.338	1	0.505	470ms	1.48 GB
		0.328	0.547	0.410	3s	1.79 GB
		0.326	0.607	0.424	1m 19s	0.71 GB
		0.254	0.483	0.333	15m 1s	1.30 GB
		-	-	-	TL	-
	Derand	0.121	0.210	0.151	1h 10s	1.25 GB
		0.125	0.190	0.150	9h 49m	3.32 GB
		0.110	0.121	0.115	25h 40m	8.21 GB
		-	-	-	TL	-
		-	-	-	TL	-
	Holoclean	0.230	0.300	0.599	7s	3.95 GB
		0.115	0.120	0.117	12s	5.15 GB
		0.097	0.114	0.104	1m 8s	6.16 GB
		0.156	0.167	0.161	8m 21s	26.89 GB
		-	-	-	-	ML

TL: time limit of 48 hours exceeded – ML: memory limit of 30 GB exceeded

Derand registered severely higher execution times, exceeding the 48h time limit starting from the 10% of missing rate. The faster execution times of Holoclean can be justified by the conspicuously lower number of metadata to be processed during the imputation process, i.e., 9 Denial of Constraints, compared to 1961 RFD$_c$s. Nevertheless, RENUVER still registered the best performances on all the considered qualitative metrics.

The second evaluation session is focused on the Physician dataset, by fixing the missing rate and by varying the number of tuples to be considered. This dataset is particularly complex to analyze, since it also contains a high number of attributes (i.e., 18 attributes). In fact, this dataset allowed us to catch a time and/or memory limit for all considered approaches (i.e., RENUVER, Derand, and Holoclean), as shown in Table 5. In particular, we can notice that, on average, both RENUVER and Holoclean registered faster execution times than Derand. In fact, the latter exceeds the time limit of 48h on the datasets having 2072 and 10359 tuples, respectively. On the other hand, Holoclean manages to achieve reasonable executions times, but the huge amount of consumed memory makes it exceed the 30GB memory limit on the dataset having 10359 tuples. Finally, RENUVER also exceeds the time limit on the largest dataset, despite a more reasonable memory consumption. This evaluation session also proved the capability of RENUVER to outperform the compared approaches on the considered qualitative metrics.

7 CONCLUSION

In this paper, we proposed RENUVER, a data imputation algorithm that exploits attribute correlations expressed in terms of relaxed functional dependencies. The latter enables RENUVER to select and evaluate tuple candidates to be used during the imputation process. The whole imputation process preserves the semantic consistency of the data, by guaranteeing that no imputation can violate any RFD$_c$. Evaluation results demonstrate the effectiveness of RENUVER on both filling rate and qualitative metrics, i.e. Recall, Precision, and F1-Measure. In particular, the results highlighted that RENUVER works particularly well on the *precision* measure. Finally, a comparative evaluation demonstrated that RENUVER outperforms recent approaches using different imputation strategies, classification models (kNN), machine learning-based (Holoclean), and dependency-based (Derand).

In the future, we would like to evaluate RENUVER with RFD$_c$s whose thresholds have associated an upper bound dependent from attribute domains and value distributions. Moreover, to increase the number of imputed values, we would like to extend RENUVER with the possibility of selecting plausible candidate tuples among multiple datasets. Finally, we would like to study the applicability of RENUVER over incremental scenarios, like for example those related to the imputation of time series [16], which would require the usage of incremental RFD$_c$ discovery algorithms [4, 5].

REFERENCES

[1] Gustavo Batista and Maria Carolina Monard. 2003. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence* 17, 5-6 (2003), 519–533.

[2] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment* 11, 3 (2017), 311–323.

[3] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE '07)*. IEEE Computer Society, 746–755.

[4] Loredana Caruccio and Stefano Cirillo. 2020. Incremental discovery of imprecise functional dependencies. *Journal of Data and Information Quality (JDIQ)* 12, 4 (2020), 1–25.

[5] Loredana Caruccio, Stefano Cirillo, Vincenzo Deufemia, and Giuseppe Polese. 2019. Incremental Discovery of Functional Dependencies with a Bit-vector Algorithm. In *Proceedings of Italian Symposium on Advanced Database Systems (SEBD '19)*, Vol. 2400. CEUR-WS.org, 1–12.

[6] Loredana Caruccio, Vincenzo Deufemia, Felix Naumann, and Giuseppe Polese. 2021. Discovering relaxed functional dependencies based on multi-attribute dominance. *IEEE Transactions on Knowledge and Data Engineering* 33, 9 (2021), 3212–3228.

[7] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2016. Relaxed functional dependencies—A survey of approaches. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 147–165.

[8] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: overview and emerging challenges. In *Proceedings of the International Conference on Management of Data (SIGMOD '16)*. ACM, 2201–2206.

[9] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1498–1509.

[10] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: putting violations into context. In *Proceedings of the 29th International Conference on Data Engineering (ICDE '13)*. IEEE Computer Society, 458–469.

[11] Pinar Cihan and Zeynep Banu Ozger. 2019. A new heuristic approach for treating missing value: ABCimp. *Elektronika ir Elektrotechnika* 25, 6 (2019), 48–54.

[12] A Rogier T Donders, Geert JMG Van Der Heijden, Theo Stijnen, and Karel GM Moons. 2006. A gentle introduction to imputation of missing values. *Journal of clinical epidemiology* 59, 10 (2006), 1087–1091.

[13] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2010. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering* 23, 5 (2010), 683–698.

[14] Chi-Chun Huang and Hahn-Ming Lee. 2004. A grey-based nearest neighbor approach for missing attribute value prediction. *Applied Intelligence* 20, 3 (2004), 239–252.

[15] Ihab F Ilyas, Xu Chu, et al. 2015. Trends in cleaning relational data: consistency and deduplication. *Foundations and Trends® in Databases* 5, 4 (2015), 281–393.

[16] Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux. 2020. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series. *Proceedings VLDB Endowment* 13, 5 (2020), 768–782.

[17] Qian Ma, Yu Gu, Wang-Chien Lee, Ge Yu, Hongbo Liu, and Xindong Wu. 2020. REMIAN: Real-time and error-tolerant missing value imputation. *ACM Transactions on Knowledge Discovery from Data* 14, 6 (2020), 1–38.

[18] Besay Montesdeoca, Julián Luengo, Jesús Maillo, Diego García-Gil, Salvador García, and Francisco Herrera. 2019. A first approach on big data missing values imputation. In *Proceedings of 5th International Conference on Internet of Things, Big Data and Security (IoTBDS)*. SciTePress, 315–323.

[19] Sanaz Nikfalazar, Chung-Hsing Yeh, Susan Bedingfield, and Hadi A Khorshidi. 2017. A new iterative fuzzy clustering algorithm for multiple imputation of missing data. In *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '17)*. IEEE Computer Society, 1–6.

[20] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: holistic data repairs with probabilistic inference. *Proceedings of VLDB Endowment* 10, 11 (2017), 1190–1201.

[21] Shaoxu Song and Lei Chen. 2011. Differential dependencies: reasoning and discovery. *ACM Transactions on Database Systems* 36, 3 (2011), 16.

[22] Shaoxu Song and Yu Sun. 2020. Imputing various incomplete attributes via distance likelihood maximization. In *Proceedings of the 26th International Conference on Knowledge Discovery & Data Mining (SIGKDD '20)*. ACM, 535–545.

[23] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2020. Enriching data imputation under similarity rule constraints. *IEEE Transactions on Knowledge and Data Engineering* 32, 2 (2020), 275–287.

[24] Qi Wang, Pang-Ning Tan, and Jiayu Zhou. 2018. Imputing structured missing values in spatial data with clustered adversarial matrix factorization. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '18)*. IEEE Computer Society, 1284–1289.

[25] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.

[26] Aoqian Zhang, Shaoxu Song, Yu Sun, and Jianmin Wang. 2019. Learning individual models for imputation. In *Proceedings of 35th International Conference on Data Engineering (ICDE '19)*. IEEE Computer Society, 160–171.