BUNNI: Discover Repair Actions for Data Cleaning

(Discussion Paper)

Giansalvatore Mecca[1], Paolo Papotti[2], Donatello Santoro[1] and Enzo Veltri[1,*]

[1]Università degli Studi della Basilicata (UNIBAS), Potenza, Italy
[2]EURECOM, Biot, France

Abstract

This work tackles the challenging and open problem of involving non-expert users as first-class participants in the data-repairing process. While numerous approaches have been proposed to clean data from the perspective of expert users, such as IT professionals and data scientists, there is a significant lack of studies focused on non-expert users. Given a set of predefined data quality rules, we employ machine learning techniques to guide users in identifying dirty values for each violation and repairing them. Our approach minimizes user effort while effectively producing trustworthy data repairs. Through experimental evaluation, we demonstrate that this machine-learning-driven method consistently produces a unique clean solution with high quality in scenarios where other traditional approaches fail.

Keywords

Data Cleaning, Repair Discovery, Human In The Loop, Machine Learning

1. Introduction

Data cleaning, or data repairing, is the process of detecting and removing errors from data. It is a crucial preprocessing step in many ML pipelines on different tasks, like medical predictions [1, 2], Text-To-SQL [3] or TNLI [4] applications. Over the past two decades, significant research has been focused on the data cleaning problem using *rule-based approaches* [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] and data imputation [16, 17]. In a rule-based approach, users define upfront data-quality rules, typically expressed as integrity constraints, and the system enforces these rules over the data.

Rule-based approaches usually rely on machine learning [18] or on heuristics to generate solutions [10, 11, 12, 13, 19, 14]. These may include quite strong assumptions – for example, always repair the input instance according to the right-hand-side of rules – which may lead to unsatisfactory results, especially since it is unfeasible for the user to manually check all the solutions and fix them manually.

Bunni[1] [20] aims at solving the problem of *semi-automatic repair discovery* by involving the user in the generation of a unique solution for a given set of rules. It uses a machine-learning

SEBD 2025: 33rd Symposium on Advanced Database System, June 16-19, 2025 - Ischia, Italy
*Corresponding author.
giansalvatore.mecca@unibas.it (G. Mecca); papotti@eurecom.fr (P. Papotti); donatello.santoro@unibas.it (D. Santoro); enzo.veltri@unibas.it (E. Veltri)
0000-0002-1189-1481 (G. Mecca); 0000-0003-0651-4128 (P. Papotti); 0000-0002-5651-8584 (D. Santoro); 0000-0001-9947-8909 (E. Veltri)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]Bunni stands for Bayesian Update-driven iNteractive learNIng

CEUR-WS.org/Vol-4182/paper10.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-03-13

	Title	Author	Journal	Vol	Year
t_1	Possible ... SQL	H. Kohler	VLDB J.	25	2016
t_2	Possible ... SQL	U. Leck	VLDB J.	25	2015
t_3	Possible ... SQL	S. Link	VLDB J.	25	2015
t_4	Possible ... SQL	X. Zhou	VLDB J.	25	Aug.
t_5	RDF ... survey	Z. Kaoudi	VLDB J.	25	2015
t_6	RDF ... survey	I. Manolescu	VLDB J.	24	2015
t_7	A ... Fragmentation	V. Braganholo	VLDB J.	25	2014
t_8	A ... Fragmentation	M. Mattoso	SIGMOD Rec.	43	2014

Table 1
Dataset with publications extracted from the web.

	Title	Author	Journal	Vol	Year
t_1	Possible ... SQL	H. Kohler	VLDB J.	25	2016
t_2	Possible ... SQL	U. Leck	VLDB J.	25	2016
t_3	Possible ... SQL	S. Link	VLDB J.	25	2016
t_4	Possible ... SQL	X. Zhou	VLDB J.	25	2016
t_5	RDF ... survey	Z. Kaoudi	VLDB J.	24	2015
t_6	RDF ... survey	I. Manolescu	VLDB J.	24	2015
t_7	A ... Fragmentation	V. Braganholo	SIGMOD Rec.	43	2014
t_8	A ... Fragmentation	M. Mattoso	SIGMOD Rec.	43	2014

Table 2
Dataset with cleaned publications.

interactive framework for repair discovery. Starting from a Constant Conditional Functional Dependency (CFD) [21], the system first collects some training data by asking the user to solve a few violations. After a training step, the system automatically infers when a rule should be enforced over a tuple in violation using a repair over the right-hand side (RHS), or conclusion, of the rule, or a repair over the left-hand side (LHS), or premise, for the same rule.

Example 1: Consider the instance in Table 1 with publications scraped from the web. Consider CFD c_1, which states whenever the Journal is VLDB J. and the Volume is 25, the year must be 2016. Errors *w.r.t.* c_1 are highlighted in red.

$$c_1 : \text{ t[Journal]=VLDB J., t[Vol]=25} \rightarrow \text{t[Year]=2016}$$

Assume a user needs to manually enforce c_1 over the data. She inspects a few of the tuples in violation – t_2, t_3, t_4, t_5 and t_7 in our example – and updates $t_2[\text{Year}]$ to 2016 first because in her knowledge (or after some manual search) the paper was published in the VLDB J. on volume 25 in the year 2016.

Then, she turns to tuple t_3. The user has already the knowledge on how to repair the violations by updating $t_3[\text{Year}]$ to 2016. The reasons are related to the data that she is seeing: Title, Journal and Volume have the same values of the previous solved error, and moreover the Year has the same (wrong) value. In her mind, it is more likely that in tuple t_3 Year is erroneous instead of Journal or Volume. Similarly for tuple t_4.

Now the user inspects t_5, in this case, she is uncertain about how to solve it, because, even if Journal and Volume are the same as the previous repair, the Title is different and for deciding how to repair she uses again her knowledge. In this case, she decides to update the Volume to 24 instead of changing Year. Finally, the user inspects tuple t_7, in this case again she is uncertain about how to repair the data, and for removing the violations she needs to use again her knowledge. This time she decides to update the Journal and Volume because the paper was published in the SIGMOD Rec with Volume 43 in the Year 2014.

This leads to the cleaned instances in Table 2, where changes by the user are highlighted in green.

From Example 1, one may observe that there might exist different reasons for removing the errors even for the same rule. In some cases, the values within the tuple give us a clear idea of how to repair the violation – as in t_4 above. This means that the user "trusts" some of the values, and "distrusts" others; the data-quality rule, along with the trusted values provides evidence on how to correct the error. In other cases, it is less clear to identify which values to trust within

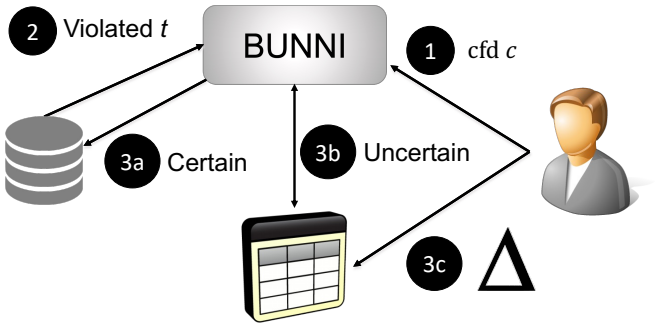

Figure 1: Bunni workflow. Starting from a cfd c (1), Bunni detects a violated tuple t *w.r.t.* c (2). If Bunni is certain about the repair it applies it directly to the data (3a), otherwise if it is uncertain (3b), it asks the user to repair the tuple with some updates Δ (3c). It then learns how to repair similar violations using Δ to train a ML model.

the tuple, and we need to rely on the user to investigate further the error and ultimately solve it, as in tuples t_2 and t_7.

Bunni mimics this form of "human thinking", trying to learn which values of the data can be trusted and which not, and involving humans only in the "hard decisions". Bunni minimizes the number of user interactions, asking only critical questions. The workflow of the system is depicted in Figure 1.

The user submits a CFD c over instance $\mathcal{I}$ (1). Bunni finds violations *w.r.t.* c, selects one of them, i.e., a tuple t, and tries to solve the violation (2). If Bunni finds out that cells of t matching the LHS of the rule can be trusted with a *confidence* above a fixed threshold, it applies the rule to repair the tuple with respect to the RHS ($3a$). Otherwise, if Bunni is uncertain, i.e. if the rule cannot be trusted, it asks the user to manually solve the violation ($3b$). If required by Bunni, the user manually updates the tuple, and Bunni uses this new evidence to refine its learning model ($3c$) . The loop in points 2 and 3 terminates when no more tuples are in violation. Then the user goes back to step 1 if she has other rules to submit.

2. Constraints and Repairs

We use the semantic of *Conditional Functional Dependencies* (CFDs) [22]. Among these, we concentrate on *constant CFDs* [21] in *normal form*, *i.e.,* such that the conclusion contains only one atom, as follows:

$$\text{cfd: } A_1{=}c_1, A_2{=}c_2, \ldots, A_i{=}c_i \rightarrow A_c{=}c_c$$

Where A_i is an attribute from a given relation of a schema $\mathcal{S}$ and c_i represents a constant value for A_i. CFDs with multiple atoms in the conclusion can be rewritten as a set of CFDs in normal form, one for each atom in the conclusion [21].

	Title	Author	Journal	Vol	Year
t_2	Possible ... SQL	U. Leck	VLDB J.	25	2016
t_7	A ... Fragmentation	V. Braganholo	VLDB J.	25	2016

Table 3
RHS Repair.

	Title	Author	Journal	Vol	Year
t_2	Possible ... SQL	U. Leck	VLDB J.	25	2016
t_7	A ... Fragmentation	V. Braganholo	SIGMOD Rec.	43	2015

Table 4
RHS and LHS Repair.

	Title	Author	Journal	Vol	Year
t_2	Possible ... SQL	U. Leck	VLDB J.	24	2015
t_7	A ... Fragmentation	V. Braganholo	SIGMOD Rec.	43	2015

Table 5
LHS Active Domain.

	Title	Author	Journal	Vol	Year
t_2	Possible ... SQL	U. Leck	TODS	12	2015
t_7	A ... Fragmentation	V. Braganholo	JDIQ	23	2015

Table 6
LHS open world assumption.

Example 2: Consider again Table 1 and the CFD c_1: t[Journal]=VLDB J., t[Vol]=25 $\rightarrow$ t[Year]=2016. The cells for Journal, Volume and Year in tuple t_1 are not in violation with c_1, while the cells for Journal, Volume and Year in tuple t_7 are in violation. We can say that t_1 satisfies c_1, but t_7 does not, so the instance $\mathcal{I}$ in Table 1 does not satisfy c_1.

Given an instance $\mathcal{I}$ and a set of CFDs Σ, we say that $\mathcal{I}$ is *clean* if satisfies all Σ, otherwise we say that is *dirty*. It is easy to see that $\mathcal{I}$ is dirty whenever there exist some violations, i.e., tuple t (or cell values in t) such that t matches with the premise of the CFD but does not match with the conclusion. Of course, a tuple t may be in violation with one or more CFDs.

Intuitively we can use one or more CFDs to make sure that our data is clean or to find violations.

To clean a dirty instance $\mathcal{I}$ we need to define a repair strategy. Given a set of rules Σ defined over the instance $\mathcal{I}$, a repair strategy is the process of removing violation *w.r.t.* Σ in $\mathcal{I}$ such that at the end of the process $\mathcal{I}$ satisfies Σ.

We repair cells with *cell updates*, *i.e.,* we change the values of the cells in violation to clean them [13]. Despite this intuitive approach, there are multiple ways to remove a violation for a CFD *cfd*: 1) *RHS Repair*, we may change the values of cells corresponding to attributes on the RHS of the *cfd*, *i.e.,* attributes appearing in the conclusion; 2) *LHS Repair* we may change the values of cells corresponding to attributes in the LHS, *i.e.,* in the premise.

In the RHS Repair, we *trust* the values in the premise of the *cfd* and use the constant in the conclusion to change the corresponding attribute and remove the violation. In the LHS Repair, we *trust* the value in the conclusion of the *cfd*, and therefore one or more of the values appearing within the premise needs to be changed.

Example 3: Consider the instance in Table 1 and the CFD c_1 from Example 2. The cells t_2[Journal], t_2[Vol], t_2[Year], t_7[Journal], t_7[Vol] and t_7[Year], are in violation *w.r.t.* c_1. Table 3 is an example of an RHS repair. We opted for an RHS repair by changing the values related to Year since we trusted the values in the premise of the c_1. Table 4 combines RHS and LHS repair strategies. For tuple t_2 we opted for an RHS repair strategy; on the contrary for tuple t_7 we selected to use an LHS strategy by changing t_7[Journal] to SIGMOD Rec and t_7[Volume] to 43. Tables 5 and 6 are examples of LHS repairs only. In the former, we used values from the active domain of the Journal and Volume attributes, while in the latter we used values outside of the active domain.

It is clear that choosing the repair strategy $\mathcal{R}ep$ and the clean values $\mathcal{V}$ to repair violations

in a complex data-repairing task is a crucial problem because multiple repaired instances can be generated, as shown by the previous example. Making decisions about the repair process is usually difficult, both for humans and for machines. However, we could alleviate this problem.

User Involvement: from Naïve to Interactive. Several studies have been devoted to consider the presence of humans in the repairing process [23, 24, 25, 26, 27, 28]. Early proposals have mainly relied on what we might call a *"naïve"* way to involve users in the rule-based repairing process, as follows.

The user first uses her domain knowledge to write the constraints. Then defines a strategy for repair discovery $\mathcal{R}ep$ – i.e., an algorithm to decide which cells are dirty for each tuple in violation with Σ –, and a strategy for value discovery $\mathcal{V}$ – i.e., an algorithm to replace cells that are deemed dirty with clean values, in some cases, multiple choices are available. At the end of the cleaning process one or multiple solutions depending on $\mathcal{R}ep$ and $\mathcal{V}$ are generated and the user will manually inspect all the generated solutions. If $\mathcal{V}$ uses variables [29], i.e., placeholders, to remove violations, then the user needs to change each placeholder into the correct constant value.

In the end, the user selects the best solution among those generated. By examining the solutions, she might find out that some of the errors in the data were not detected or repaired In this case, she needs to write additional rules that capture these errors and execute the process again.

In this *"naïve"* form, the user is involved exclusively at the beginning of the process, to identify the constraints, and at the end, to inspect the generated solutions.

Instead, Bunni uses a different protocol to involve the user in the repair process. It requires the user to $a)$ inspect cells wrong *w.r.t.* a constraint and $b)$ resolve the violations by submitting a *value update*.

These value updates are then used by our learning algorithms to infer the intended repair strategy, and the strategy to select clean values. In fact, by providing an update, the user $1)$ indicates which cells need to be repaired, and indirectly indicates which cells should be trusted, and $2)$ indicates the values to use to fix the errors. As a result, the system and the user generate a single *trusted repair*, in which all choices taken during the repair process are validated directly or indirectly by user actions.

Framework for Repair Discovery. Given a constant CFD *cfd* and a violated tuple t we say that the set of cells in t corresponding to the attribute in the premise of the *cfd* can be trusted if the probability of each cell is above a threshold τ_{clean}. More specifically, we fix two user-defined thresholds, τ_{clean} and τ_{dirty}. Then, for each cell c_i involved in a violation, we estimate the probability p_i that is clean. We rule that c_i can be trusted if $p_i > \tau_{clean}$; we rule that the c_i is dirty if $p_i < \tau_{dirty}$. We ask the user otherwise.

In light of this, solving the Repair Discovery problem amounts to calculating the probability that a cell is clean, and more specifically it consists of solving a classification problem where the classifier outputs both the prediction of whether a cell is clean or dirty and also the probability about the prediction that is then compared against τ_{clean} and τ_{dirty}.

To handle the case in which more than one cell of a tuple t may be dirty, we extend the binary classification model to a *multilabel classification problem*. Practically, a multilabel classifier is based on a set of *n* binary classifiers, one for each outcome, where *n* represents the number of

possible values for the outcome. Then, given a threshold τ and a tuple t to classify, the response consists of all the predicted outcomes of each binary classifier such that the probability that the hypothesis is true is above the threshold τ. In our case, we generate a classifier for each attribute in the premise of the CFD, and we use τ_{clean} as the threshold.

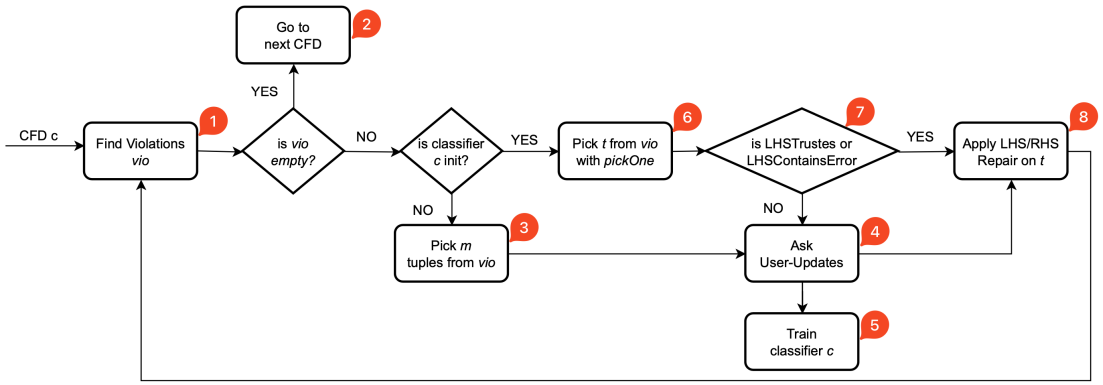

Figure 2: Bunni starts from a CFD c and finds the set of violated tuples. If there are no violations, it goes to the next CFD (2). If there are violations, it selects m tuples to be manually cleaned by the user (3-4). Such updates are used to train a classifier c (5), and to repair the m tuples. After the initialization step, if there are violations, tuple t to be repaired is selected using *pickOne* function (6). The classifiers predicts if the LHS could be trusted (7), either repair is applied (8) or it asks the user to manually repair the tuple t (4), and also retrains the classifier c with the new user updates (5). The process continues until the set of violated tuples is empty.

Figure 2 illustrates the detailed workflow of Bunni. The process begins with a CFD c, the system identifies all violations vio *w.r.t.* c. If no violations are found, it goes to the next CFD. Otherwise, it selects m tuples to be manually labeled by the user. These manually annotated tuples are then used to train a classifier, which helps repair other tuples that violate c.

Once the classifier is initialized, if there are still tuples in vio, Bunni selects a tuple t using the *pickOne* function, which we will discuss next. If the classifier determines that the cells in t related to the premise of c can be trusted, i.e., their probability exceeds τ_{clean}, it applies the RHS update using the value in c's conclusion ($LHSTrusted$). If the classifier detects that some cells in t related to the premise of c cannot be trusted, i.e., their probabilities fall below τ_{dirty}, Bunni applies an LHS update ($LHSContainsError$). If neither an RHS nor LHS repair is feasible, meaning the classifier is in an *indecision area*, Bunni asks the user to manually clean the affected cells. These manual updates are then used to retrain the classifier. This process repeats until vio is empty, then the system moves on to the next CFD.

The *pickOne* function uses *Active Learning* [30] to minimize user interactions. For each tuple in violation, we calculate a score that measures the *uncertainty* of the classifier. Intuitively, submitting to the user the tuple with the highest score, *i.e.,* with the highest uncertainty, will increase the quality of the model in a faster way. The score for each cell is $score_c = 2 \cdot p_{clean} \cdot p_{dirty}$. The score for a tuple is the sum of each score for each cell in the premise of the CFD. This strategy is also used to select the tuples that should be manually labeled by the user.

3. Experiments

We use three real-world datasets and a synthetic dataset. BUS: a UK government public dataset with 15 attributes and 250K tuples. DBLP: is based on the collection of authors, publications and venues with 15 attributes and 1M tuples. FLIGHT [31] is a real-world dataset related to flight departures and arrivals. It contains real errors and is provided with both clean and dirty versions. Synth is a dataset we designed starting from the original Soccer dataset with 10 attributes and 100k tuples.

Errors and Metrics. To compute the quality of a data repair algorithm we need two versions of the same dataset: a *dirty* version with the errors and its *clean* version to compare with the repaired generated solution. We use the BART error generation tool [32], which allows us to introduce detectable errors w.r.t. a set of input CFDs by changing the cell values. BART allows configuring how errors are injected, i.e., how many errors are introduced in the LHS/RHS of the CFDs and thus we also control how many errors are introduced for each tuple. We manually define six normalized constant CFDs for each dataset and we generate for each dataset two different versions of dirty instances. We generate an *easy* (*hard*) scenario, where we inject approximately 75% (50%) of the errors on the RHS of every CFD using random values (typo, random value, value from active domain) and the remaining on the LHS using values from active domain; As metrics we use: 1) the Quality of the repaired instances by using the F-Measure. We do not measure the similarity [29] of the repaired instance *w.r.t.* the clean version since we do not introduce Nulls or LLUNS [10]. 2) The Benefit *w.r.t.* to manually solve all the errors, i.e. the percentage of saved user interaction. 3) The Effectiveness, i.e., the F-Measure between the Quality and the Benefit. Using the effectiveness we can compare different strategies considering both Quality and Benefit. And 4) the Time spent to clean all the instance excluding the think time of the user.

Interactive System Evaluation. We evaluate Bunni against HoloClean (HC) [18], Raha and Baran, and LLunatic [10]. Raha [33] is an automatic error detection system that uses user updates to find cells with errors, while Baran [31] is an automatic data repair system that uses user updates to infer cell repairs. We combine both systems (Raha + Baran) to infer errors in the cells and repair such cells using user updates. LLunatic is a rule-based data-cleaning system that can be configured with different strategies. To reduce the number of user updates, we configured LLunatic with a similarity-based strategy to repair the data. It applies an RHS repair if the value in the conclusion is similar to the one constrained by the CFD (using the Levenshtein distance), otherwise, it applies a LHS repair. In case of multiple possible values in the LHS, it makes use of placeholders (LLUNS [10]) that are interactively updated by the user.
For HoloClean, LLunatic and Bunni, we use the same CFDs to detect errors. We mine CFDs for Flight using Metanome [34] with the CFDFinder discovering algorithm [22, 7]. We measure the quality (Q), benefit (B), effectiveness (E), and total execution time (T) in seconds. For Raha and Baran we use different budgets, i.e., number of user updates, and we report the best results in terms of Effectiveness. We report the results in Table 7. For Raha and Baran, we had memory errors for BUS and DBLP and we do not report such results. We also report the average of all the metrics over all datasets. The best results for each metric and the lowest total time are shown in bold.

	Raha + Baran				HoloClean				LLunatic				Bunni			
dataset	**Q**	**B**	**E**	**T(s)**	**Q**	**B**	**E**	**T(s)**	**Q**	**B**	**E**	**T(s)**	**Q**	**B**	**E**	**T(s)**
BUS — Easy	-	-	-	-	0.97	**1.00**	**0.98**	1k	0.65	0.69	0.67	12	**1.00**	0.75	0.86	**7**
BUS — Hard	-	-	-	-	0.94	**1.00**	**0.97**	1k	0.71	0.39	0.51	29	**1.00**	0.47	0.64	**6**
DBLP — Easy	-	-	-	-	0.06	**1.00**	0.11	1.5k	0.46	0.81	0.59	18	**1.00**	0.55	**0.71**	**3.6**
DBLP — Hard	-	-	-	-	0.07	**1.00**	0.13	1.5k	0.36	0.62	0.46	18	**0.93**	0.33	**0.49**	**12.7**
Synth — Easy	0.48	0.67	0.56	15k	0.05	**1.00**	0.09	415	0.83	0.8	0.81	18	**1.00**	0.78	**0.87**	**3**
Synth — Hard	0.53	0.67	0.59	16k	0.07	**1.00**	0.13	521	0.82	0.6	0.7	18	**1.00**	0.61	**0.76**	**3**
Flights	0.66	1.00	0.79	6k	0.76	**1.00**	0.86	70.6	**1.00**	**1.00**	**1.00**	1523	**1.00**	0.92	0.96	**4**
AVG	0.55	0.77	0.65	12.3k	0.42	**1.00**	0.47	858	0.69	0.70	0.67	233	**0.99**	0.63	**0.76**	**5.6**

Table 7
For each system and dataset, we report the (Q)uality, (B)enefit, (E)ffectiveness and total (T)ime in seconds. Cells without the number correspond to memory errors. We also report the average of the measures over the dataset. In bold we report the highest metric for each dataset, and the lowest total time execution. Bunni shows a good trade-off between high quality and user effort as reported by the average of the Effectiveness.

For Raha and Baran, datasets with a large number of columns had out-of-memory errors on our machine. This is due to the number of clustering algorithms trained for each column. Another issue is the initialization of the strategies for error detection, which takes considerable time for an interactive system; however, such initialization is executed once and is skipped in the other executions. HoloClean does not require any user update, so the benefit is always 1.00, but the quality for DBLP and Synth is low, and thus the effectiveness is also low. Moreover, like Raha and Baran, also HoloClean takes considerable time to repair the dataset. LLunatic can reach a good quality without requiring too much user effort, and it also requires a reasonable time for an interactive system (ignoring Flights dataset). Finally, Bunni represents the good trade-off when we require high-quality results (on avg. we have 0.99), while it also reduces the user effort (benefit on avg. 0.63). The effectiveness of Bunni is the highest in this experiment, while the avg. time is the lowest, proving that is suitable for an interactive system.

4. Conclusions

We presented Bunni, an interactive system for repair discovery. We demonstrated how the system is able to interact with users in real-time to generate a solution with a user-defined quality. We used a well-known machine learning technique, namely Naïve Bayes, to solve the repair-discovery problem while minimizing the number of user interactions and computing time. We showed how the Naïve Bayes method outperforms other approaches in terms of quality, benefit, effectiveness, and execution times. A promising research direction in this respect consists of using other bayesian approaches with graphical models, like Bayesian Networks. This would allow us to consider other factors like causality in the attributes, or semantic correlation. Indeed, our Naïve Bayes classifier is a special case of a Bayesian Network. Another aspect to investigate is the integration of Bunni with Language Models (LMs) such as BERT [35] and T5 [36], or even tabular LMs [37], to automatically suggest the values for the LHS repair.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] P. Lapadula, G. Mecca, D. Santoro, L. Solimando, E. Veltri, Humanity is overrated. or not. automatic diagnostic suggestions by greg, ml (extended abstract), Communications in Computer and Information Science 909 (2018) 305 – 313. doi:10.1007/978-3-030-00063-9_29.

[2] P. Lapadula, G. Mecca, D. Santoro, L. Solimando, E. Veltri, Greg, ml – machine learning for healthcare at a scale, Health and Technology 10 (2020) 1485 – 1495. doi:10.1007/s12553-020-00468-9.

[3] E. Veltri, G. Badaro, M. Saeed, P. Papotti, Data ambiguity profiling for the generation of training examples, in: 39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023, IEEE, 2023, pp. 450–463. URL: https://doi.org/10.1109/ICDE55515.2023.00041. doi:10.1109/ICDE55515.2023.00041.

[4] J.-F. Bussotti, E. Veltri, D. Santoro, P. Papotti, Generation of training examples for tabular natural language inference, Proc. ACM Manag. Data 1 (2023). URL: https://doi.org/10.1145/3626730. doi:10.1145/3626730.

[5] X. Chu, I. F. Ilyas, P. Papotti, Discovering denial constraints, Proc. VLDB Endow. 6 (2013) 1498–1509. URL: http://www.vldb.org/pvldb/vol6/p1498-papotti.pdf. doi:10.14778/2536258.2536262.

[6] S. Song, L. Chen, Efficient discovery of similarity constraints for matching dependencies, Data Knowl. Eng. 87 (2013) 146–166. URL: https://doi.org/10.1016/j.datak.2013.06.003. doi:10.1016/j.datak.2013.06.003.

[7] W. Fan, F. Geerts, J. Li, M. Xiong, Discovering conditional functional dependencies, IEEE Trans. Knowl. Data Eng. 23 (2011) 683–698. URL: https://doi.org/10.1109/TKDE.2010.154. doi:10.1109/TKDE.2010.154.

[8] F. Chiang, R. J. Miller, Discovering data quality rules, Proc. VLDB Endow. 1 (2008) 1166–1177. URL: http://www.vldb.org/pvldb/vol1/1453980.pdf. doi:10.14778/1453856.1453980.

[9] L. Golab, H. J. Karloff, F. Korn, B. Saha, D. Srivastava, Discovering conservation rules, in: IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012, IEEE Computer Society, 2012, pp. 738–749. URL: https://doi.org/10.1109/ICDE.2012.105. doi:10.1109/ICDE.2012.105.

[10] F. Geerts, G. Mecca, P. Papotti, D. Santoro, Cleaning data with llunatic, VLDB Journal 29 (2020) 867 – 892. doi:10.1007/s00778-019-00586-5.

[11] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J. Quiané-Ruiz, N. Tang, S. Yin, Bigdansing: A system for big data cleansing, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015, ACM, 2015, pp. 1215–1230. URL: https://doi.org/10.1145/2723372.2747646. doi:10.1145/2723372.2747646.

[12] J. Wang, N. Tang, Towards dependable data repairing with fixing rules, in: International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014, ACM, 2014, pp. 457–468. URL: https://doi.org/10.1145/2588555.2610494. doi:10.1145/2588555.2610494.

[13] P. Bohannon, M. Flaster, W. Fan, R. Rastogi, A cost-based model and effective heuristic for repairing constraints by value modification, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005, ACM, 2005, pp. 143–154. URL: https://doi.org/10.1145/1066157.1066175. doi:10.1145/1066157.1066175.

[14] W. Fan, J. Li, S. Ma, N. Tang, W. Yu, Towards certain fixes with editing rules and master data, VLDB J. 21 (2012).

[15] M. Buoncristiano, G. Mecca, D. Santoro, E. Veltri, Detective gadget: Generic iterative entity resolution over dirty data, Data 9 (2024). doi:10.3390/data9120139.

[16] B. Breve, L. Caruccio, V. Deufemia, G. Polese, RENUVER: A missing value imputation algorithm based on relaxed functional dependencies, in: Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022, OpenProceedings.org, 2022. doi:10.5441/002/EDBT.2022.05.

[17] S. Song, Y. Sun, A. Zhang, L. Chen, J. Wang, Enriching data imputation under similarity rule constraints, IEEE Trans. Knowl. Data Eng. 32 (2020) 275–287. doi:10.1109/TKDE.2018.2883103.

[18] T. Rekatsinas, X. Chu, I. F. Ilyas, C. Ré, Holoclean: Holistic data repairs with probabilistic inference, Proc. VLDB Endow. 10 (2017) 1190–1201. URL: http://www.vldb.org/pvldb/vol10/p1190-rekatsinas.pdf. doi:10.14778/3137628.3137631.

[19] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, N. Tang, Interactive and deterministic data cleaning, in: Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, ACM, 2016, pp. 893–907. URL: https://doi.org/10.1145/2882903.2915242. doi:10.1145/2882903.2915242.

[20] G. Mecca, P. Papotti, D. Santoro, E. Veltri, BUNNI: learning repair actions in rule-driven data cleaning, ACM J. Data Inf. Qual. 16 (2024) 12:1–12:31. URL: https://doi.org/10.1145/3665930. doi:10.1145/3665930.

[21] W. Fan, F. Geerts, Foundations of Data Quality Management, Synthesis Lectures on Data Management Morgan & Claypool Publishers, 2012. URL: http://dx.doi.org/10.2200/S00439ED1V01Y201207DTM030. doi:10.2200/S00439ED1V01Y201207DTM030.

[22] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, ACM Trans. Database Syst. 33 (2008) 6:1–6:48. URL: https://doi.org/10.1145/1366102.1366103. doi:10.1145/1366102.1366103.

[23] V. Raman, J. M. Hellerstein, Potter's wheel: An interactive data cleaning system, in: VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy, Morgan Kaufmann, 2001, pp. 381–390. URL: http://www.vldb.org/conf/2001/P381.pdf.

[24] J. Heer, J. M. Hellerstein, S. Kandel, Predictive interaction for data transformation, in: Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings, www.cidrdb.org, 2015. URL: http://cidrdb.

org/cidr2015/Papers/CIDR15_Paper27.pdf.

[25] B. Wu, C. A. Knoblock, An iterative approach to synthesize data transformation programs, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 1726–1732. URL: http://ijcai.org/Abstract/15/246.

[26] S. Kandel, A. Paepcke, J. M. Hellerstein, J. Heer, Wrangler: interactive visual specification of data transformation scripts, in: Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011, ACM, 2011, pp. 3363–3372. URL: https://doi.org/10.1145/1978942.1979444. doi:10.1145/1978942.1979444.

[27] S. Sarawagi, A. Bhamidipaty, Interactive deduplication using active learning, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada, ACM, 2002, pp. 269–278. URL: https://doi.org/10.1145/775047.775087. doi:10.1145/775047.775087.

[28] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, I. F. Ilyas, Guided data repair, Proc. VLDB Endow. 4 (2011) 279–289. URL: https://doi.org/10.14778/1952376.1952378. doi:10.14778/1952376.1952378.

[29] B. Glavic, G. Mecca, R. J. Miller, P. Papotti, D. Santoro, E. Veltri, Similarity measures for incomplete database instances, in: Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28, OpenProceedings.org, 2024.

[30] B. Settles, Active Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012. URL: https://doi.org/10.2200/S00429ED1V01Y201207AIM018. doi:10.2200/S00429ED1V01Y201207AIM018.

[31] M. Mahdavi, Z. Abedjan, Baran: Effective error correction via a unified context representation and transfer learning, Proc. VLDB Endow. 13 (2020) 1948–1961. URL: http://www.vldb.org/pvldb/vol13/p1948-mahdavi.pdf.

[32] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, D. Santoro, Messing up with BART: error generation for evaluating data-cleaning algorithms, Proc. VLDB Endow. 9 (2015) 36–47. URL: http://www.vldb.org/pvldb/vol9/p36-arocena.pdf. doi:10.14778/2850578.2850579.

[33] M. Mahdavi, Z. Abedjan, R. C. Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, N. Tang, Raha: A configuration-free error detection system, in: Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, ACM, 2019, pp. 865–882. URL: https://doi.org/10.1145/3299869.3324956. doi:10.1145/3299869.3324956.

[34] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, F. Naumann, Data profiling with metanome, Proc. VLDB Endow. 8 (2015) 1860–1863. URL: http://dx.doi.org/10.14778/2824032.2824086. doi:10.14778/2824032.2824086.

[35] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics., Association for Computational Linguistics, 2019, pp. 4171–4186. doi:10.18653/v1/n19-1423.

[36] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu,

Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 140:1–140:67.

[37] G. Badaro, M. Saeed, P. Papotti, Transformers for Tabular Data Representation: A Survey of Models and Applications, Transactions of the Association for Computational Linguistics 11 (2023) 227–249. URL: https://doi.org/10.1162/tacl_a_00544. doi:10.1162/tacl_a_00544.