CleanEr: Interactive, Query-Guided Error Mitigation for Data Cleaning Systems*

Ran Schreiber
Bar-Ilan University

Yael Amsterdamer
Bar-Ilan University

Abstract

A key challenge in data cleaning is estimating which of the tuples in a given database are correct and which are not. However, the output of such systems typically includes both false positives and false negatives, i.e., incorrect tuples labeled as correct and vice versa. When queries are performed over the output of such cleaning systems, cleaning errors may further have an intricate impact on the query results.

We introduce *CleanEr*, a generic framework that is used on top of existing data cleaning systems and that assists users in identifying the impact of potential cleaning errors on query results, and in deciding accordingly whether and how to proceed with the cleaning. We introduce novel indicators reflecting the current uncertainty with respect to tuples in the query result, as well as the effect of each relevant input tuple on this uncertainty. We design and implement in CleanEr efficient algorithms for computing these indicators. Based on these indicators, CleanEr helps the data analysts decide whether to trust the query output, and guide them in further cleaning of relevant parts of the data through an interactive process. We propose to demonstrate CleanEr using NELL, a large database extracted from the Web.

1 Introduction

There is a large body of work on data cleaning, where a main challenge is estimating whether a given database tuple is correct or not. Despite great advances in this area, this classification is still far from perfect. Systems such as [7, 12] accompany their results with an estimation of uncertainty, e.g., the likelihood that correct tuples are labeled by the cleaner as incorrect and vice versa. In general, there is a tradeoff between the cost of cleaning, e.g., the amount of manually labeled data, and the uncertainty of the cleaning output.

Furthermore, the data cleaning output may be further analyzed and fed as input to queries. Already in the case of Select-Project-Join-Union (SPJU) queries, the effect of cleaning errors on query results becomes intricate. This is due to the intricate dependency of each output tuple on multiple input tuples (in turn, captured by the notion of data provenance, e.g., [6]). It may be extremely challenging to estimate to what extent one can trust query results, and to decide what further cleaning steps to perform to increase this trust. For example,

*To appear in ICDE 2024.

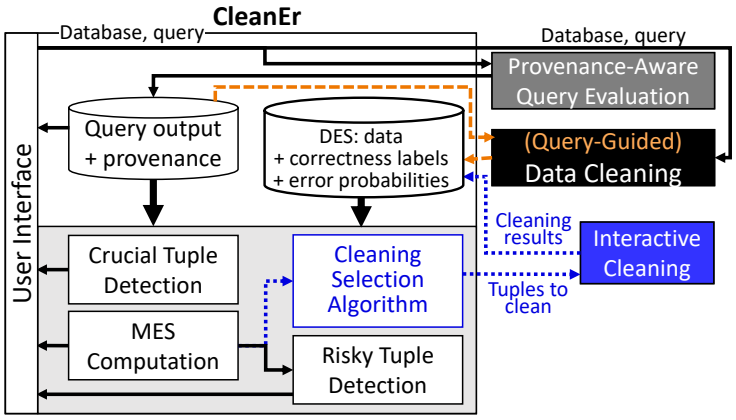

Figure 1: The architecture of CleanEr

should an additional expert be employed to assess the correctness of specific database tuples? If so, which ones?

To this end, we introduce *CleanEr*, a generic framework that is used on top of external cleaning modules and allows for the analysis and mitigation of errors in data cleaning. We now overview the architecture of CleanEr (Figure 1) and highlight its novel modules. Given a database and a query, we evaluate the query over the database along with Boolean provenance (see below) using an external Provenance-Aware Query Evaluation module (such as ProvSQL [10], denoted by a gray frame on the top right). We also use an external Data Cleaning module (such as [1,3,7,12], black frame on the right). The Data Cleaning module assigns to each database tuple a correctness label along with an estimated likelihood for this label to be erroneous.

CleanEr includes four newly developed modules that use the labeled data and the query result provenance. These modules are designed to analyze and handle cleaning errors where, unlike typical settings of uncertain databases (e.g., [8,11], we do not have an estimation of the ground truth but only of potential cleaning errors. 1. We define a notion of input tuples that are *crucial* with respect to a given output tuple, namely, incorrectly labeling them will lead to incorrectly labeling the output tuple. The first module computes the crucial input tuples per output tuple. 2. We next define a worst-case error measure called MES (stands for Maximal Error Score). Intuitively, MES reflects the maximal likelihood of an error in the label of an output tuple. The MES Computation Module implements efficient methods for computing the MES of each output tuple, distinguishing between tuples labeled as correct and incorrect, since we show the latter to be easier. 3. We then define *risky input tuples*, for which further cleaning may refute the current correctness estimation by increasing the MES. The third module efficiently identifies the risky inputs per output tuple by invoking the MES computation module. 4. The last module implements a novel algorithm for automatically selecting which tuples to clean further in order to reduce the MES of user-selected output tuples. For that, it chooses effective cleaning steps and executes them via an external Interactive Cleaning module (blue, dotted arrows).

We next briefly review related work.

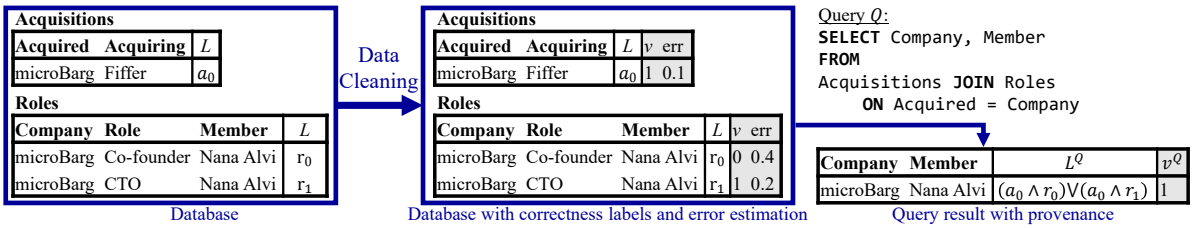

Figure 2: Example for querying the results of data cleaning

Related Work. Several cleaning systems with human-in-the-loop have been proposed (e.g., [1–3,5,7,9,12]), as well as query-guided systems (e.g., [1–3,9,12] that focus on cleaning tuples relevant to given queries. CleanEr is the only solution, to our knowledge, that can operate on top of such systems, enhancing them by letting users analyze and handle potential errors in the cleaning process. *Provenance* is a central component of our solution (see below), which has multiple definitions (e.g., [6]) and has been used, in particular, in uncertain and probabilistic databases (e.g., [8,11]). We differ from prior work on probabilistic databases, in that 1. our model captures potential errors in the cleaning process rather than a prior likelihood on the ground truth; 2. we have error likelihood only for the data labeled by the cleaning process even though unlabeled tuples may also be involved in the query; 3. due to (1) and (2) we are unable to compute the probabilities for correct query results and instead need to define new indicators; and 4. we aim to assist users in an interactive cleaning process and thus focus on potential label and likelihood changes that may result from this process.

2 Model and Indicator Definitions

We start by introducing a formal model, which will then be used in defining our three uncertainty indicators.

Database with error estimation (DES). We view data cleaning as a probabilistic process, with some likelihood of an error in each label. Therefore, we model the output of data cleaning as a database with error estimation (DES) denoted by $\overline{D} = \langle D, v, \mathrm{err} \rangle$, where D is a relational database, $v : D \to \{0, 1, \bot\}$ is a 3-value labeling function capturing the correctness labels assigned by the data cleaning process to D's tuples (where 0, 1 and $\bot$ stand for correct, incorrect and unknown, resp.) and $\mathrm{err} : D \to [0, 0.5] \cup \{\bot\}$ assigns error probabilities to the tuples of D reflecting the confidence of the data cleaner in their labels. For each $t_i \in D$, $\mathrm{err}(t_i) = \bot$ iff $v(t_i) = \bot$. For confidence computation in data cleaning see, for example [7,12]. We assume that the error probability of a Boolean label is up to 0.5; otherwise, we negate the label to achieve this. We use negation in the standard 3-valued logic sense, i.e., $\neg 1 = 0$, $\neg 0 = 1$, and $\neg \bot = \bot$.

Queries over DES. We next explain how 3-value labels propagate from the input DES $\overline{D} = \langle D, v, \mathrm{err} \rangle$ to the output of an SPJU query Q. Denote $D_l = \{t \mid v(t) = l\} \subseteq D$, i.e., the subset of D assigned a label l by v. We then define

a 3-value labeling function v^Q as follows: $v^Q(o) = 1$ if $o \in Q(D_1)$, $v^Q(o) = 0$ if $o \notin Q(D_1 \cup D_\perp)$, and $v^Q(o) = \perp$ otherwise, where $Q(D)$ is the standard query output of Q over D. Intuitively, output tuple o is correct if it can be derived from correct DES tuples and incorrect if it is not derived even if all the DES tuples with no labels are correct. This follows a standard 3-value logic semantics for SPJU queries.

Example 2.1. *Consider Figure 2. The database D on the left includes two relations, recording acquired companies and the roles of members in each company. Applying data cleaning to D yields a DES (middle frame) where each tuple is associated with a correctness label and an estimation of the probability that this label is erroneous, columns v and* err*, respectively. Next, we apply to the DES an SQL query (top right) returning members of acquired companies. The result (bottom right, ignore column L^Q for now) includes a single tuple. This output tuple is labeled as* correct *(column v^Q) since it can be derived using only tuples estimated to be correct (marked with a_0 and r_1).*

Crucial tuples. Errors in the correctness labels assigned to input tuples may or may not lead to errors in the labels of derived output tuples. The following definition identifies input tuples that are *crucial* in the sense that an error in their label implies an error in the label of the query output. Formally,

Definition 2.1 (Crucial tuples)**.** *Let $\overline{D} = \langle D, v, \text{err}\rangle$ be a DES, Q be a query and $o \in Q(D)$ such that $v^Q(o) \in \{0, 1\}$. For each $t \in D$, denote by v_t the labeling function that agrees with v on all tuples except that $v_t(t) = \neg v(t)$. Then t is* crucial *w.r.t. $\overline{D}, Q, o$ if $v^Q(o) \neq v_t^Q(o)$.*

For example, in Figure 2 both a_0 and r_1 are crucial, since an error in one of their labels (i.e., if one of them is in fact incorrect) means an error in the query output label. Crucial tuples resemble the *counterfactuals* of [8], except that they are defined with respect to v, which reflects the current, partial knowledge in an interactive cleaning process.

Worst-case error. We next propose a metric for measuring the uncertainty of query results. Let $\overline{D} = \langle D, v, \text{err}\rangle$ be a DES, Q a query, and $o \in Q(D)$ an output tuple such that $v^Q(o) \in \{0, 1\}$. We look at the set of possible ground truths, namely, possible worlds, with respect to D, each of which can be represented via a total labeling function $v' : D \to \{0, 1\}$ assigning the ground truth correctness to each tuple. Out of these possible worlds, we focus on those where, after performing cleaning yielding $\overline{D}$ and executing Q over $\overline{D}$, the output label $v^Q(o)$ would be erroneous. We formally denote this set of possible worlds by $\text{Inv}\,(v, o, Q) = \{v' : D \to \{0, 1\} \mid v'^Q(o) = \neg v^Q(o)\}$. Now, formally,

Definition 2.2 (Maximal Error Score)**.** *The* Maximal Error Score *(MES) w.r.t. $\overline{D} = \langle D, v, \text{err}\rangle, Q, o$ is defined as*

$$\text{MES}(\overline{D}, Q, o) \quad = \quad \max_{v' \in \text{Inv}(v,o,Q)} \prod_{\{t \in D \mid v(t) = v'(t)\}} (1 - \text{err}(t)) \cdot \prod_{\{t \in D \mid v(t) = \neg v'(t)\}} \text{err}(t) \quad (1)$$

Intuitively, out of all the possible worlds in which the output label $v^Q(o)$ is wrong, we look for the world that would maximize the likelihood of the observed

labels, assuming independence of cleaning errors. A low MES thus implies that $v^Q(o)$ is unlikely to be wrong: for every possible world in which $v^Q(o)$ is wrong, the likelihood of obtaining the DES $\overline{D}$ is low. Conversely, a high MES only implies that there exists a possible world in which $v^Q(o)$ is wrong, and in which $\overline{D}$ is a likely cleaning output.

Example 2.2. *Continuing Example 2.1, let W' be the possible world where only a_0 is correct. I.e., the cleaning process was right about a_0, r_0 but wrong about r_1. In W', the likelihood of attaining the DES in Figure 2 as a cleaning result is $(1 - \mathrm{err}(a_0)) \cdot (1 - \mathrm{err}(r_0)) \cdot \mathrm{err}(r_1) = 0.9 \cdot 0.6 \cdot 0.2 = 0.108$. In fact, W' is the most probable world for which the result of the query over the DES is wrong, i.e., incorrect in contrast with its label. Therefore, the MES of the output tuple is 0.108.*

Risky tuples. MES is an indicator of lack of confidence in a query result, based on which analysts may decide whether to apply additional cleaning steps to the database. To assist them further in choosing *which* tuples to clean, we classify each of the database tuples based on whether cleaning them further, namely, reducing the likelihood of an error in their label, may in fact refute the current label assigned to the query result in question. Those that may refute the label are called *risky*:

Definition 2.3 (Risky input tuples)**.** *Let $\overline{D} = \langle D, v, \mathrm{err}\rangle$ be a DES, Q a query and $o \in Q(D)$ an output tuple such that $v(o) \in \{0, 1\}$ and let $t \in D$. We say that $\overline{D_t} = \langle D, v', \mathrm{err}'\rangle$ is a possible cleaning result for $\overline{D}, t$ if it agrees with $\overline{D}$ on all tuples except t, and $\mathrm{err}'(t) < \mathrm{err}(t)$. Then, $\overline{D_t}$* refutes *$\overline{D}, Q, o$ if $\mathrm{MES}(\overline{D}', Q, o) > \mathrm{MES}(\overline{D}, Q, o)$. Finally, t is a* risky tuple *w.r.t. $\overline{D}, Q, o$ if it has a possible refuting cleaning result.*

Example 2.3. *Reconsider Example 2.2. If we ask additional experts and reduce the likelihood of an error in r_0 from 0.4 to 0.2, the worst-case world is still W', but the likelihood of attaining the DES increases to* 0.144*. If, instead, we reduce the likelihood of a mistake in r_1 from 0.2 to 0.1, the likelihood of obtaining the DES given W' (which is still the worst-case world) decreases to* 0.054*. Hence, r_0 is risky but r_1 is not.*

3 Analysis and Algorithms

We start by recalling the notion of Boolean provenance, and then use it in our algorithms.

Provenance. In an *annotated database* $\hat{D} = (D, X, L)$, tuples are uniquely labeled by Boolean variables in X using a function $L : D \to X$. The result of a query Q over $\bar{D}$ is $Q(\bar{D}) = (Q(D), \mathrm{Bool}[X], L^Q)$, where L^Q annotates each query output tuple with a Boolean expression over variables in X. Provenance computation commutes with truth assignments [6]: applying the assignment to the input database and then evaluating the query is equivalent to applying it directly to the provenance expressions. E.g., in Figure 2, tuple annotations appear in column L and the query output provenance appears in column L^Q. Indeed, the provenance reflects that the result is correct either if the input tuples annotated by a_0 and r_0 or those annotated by a_0 and r_1 are correct.

3.1 Computing the Indicators

We next analyze the complexity and show algorithms for computing the three indicators defined in Section 2.

Crucial tuples (Def. 2.1). Given an output tuple o and its provenance $L^Q(o)$, determining whether an input tuple t is crucial amounts to checking whether flipping the assignment of its annotation $L(t)$ changes the value of $L^Q(o)$, which can be done in time linear in the size of $L^Q(o)$.

MES computation (Def. 2.2). We next characterize the complexity of MES computation.

Proposition 3.1. *Let $\overline{D} = \langle D, v, \mathrm{err}\rangle$ be a DES, $Q(\bar{D}) = (Q(D), \mathrm{Bool}[X], L^Q)$ be the result of an SPJU query, and let $o \in Q(D)$ be an output tuple.*

1. If $v^Q(o) = 0$, then $\mathrm{MES}\left(\overline{D}, Q, o\right)$ can be computed in polynomial time in the size of D.

2. If $v^Q(o) = 1$, then unless P=NP, $\mathrm{MES}\left(\overline{D}, Q, o\right)$ may not be computed in polynomial time in the size of D.

Proof (Sketch). For part 1, we can show that the possible world that maximizes the likelihood of v is similar to v except that it satisfies one term (i.e., conjunction) in the DNF provenance of o. Therefore, we can compute the MES by computing the probability of v for every such possible world. Part 2 is proved by a reduction from Independent Set. □

The above proof yields a PTIME algorithm for computing the MES of output tuples labeled 0. For output tuples labeled 1, we implement two alternatives: first, for some subclasses of SPJU, the provenance may be transformed in PTIME into conjunctive normal form (CNF) [3]. In such cases, a PTIME algorithm analogous to the one in the above proof exists. For the general case, we formulate the problem as an integer linear program (ILP) that can be solved using state-of-the-art solvers.

Risky tuples (Def. 2.3). Finally, we study the detection of risky tuples. Here, we can prove that interestingly, if reducing the error probability of input tuple t to some p_j increases the MES of output tuple o, the same is true for *every* $p < p_j$. We can thus detect risky tuples by 3 MES computations, as follows. Given a DES $\overline{D} = \langle D, v, \mathrm{err}\rangle$, let v' be identical to v except that $v'(t) = \neg v(t)$, and err' identical to err except that $\mathrm{err}'^t_0 = 0$. Tuple t is risky with respect to $\overline{D}, Q, o$ iff $\mathrm{MES}(\langle D, v, \mathrm{err}'\rangle, Q, o) > \mathrm{MES}(\overline{D}, Q, o)$ or $\mathrm{MES}(\langle D, v', \mathrm{err}'\rangle, Q, o) > \mathrm{MES}(\overline{D}, Q, o)$, the latter capturing t's label being inverted by additional cleaning.

3.2 Selecting Further Cleaning Steps

So far, we have focused on computing indicators both with respect to the uncertainty of each output tuple and for the effect of each input tuple on this uncertainty. The analysts may use this information to select input tuples which

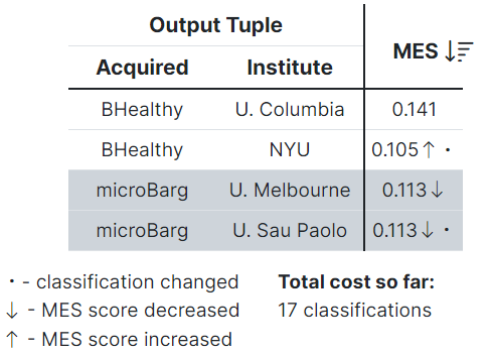

Figure 3: Query results screen

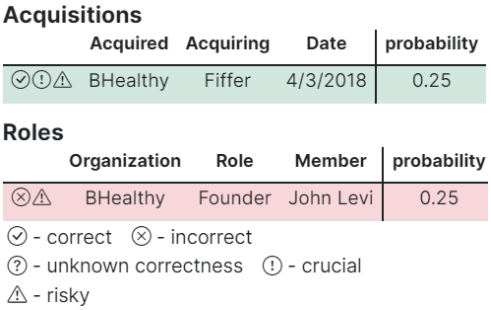

Figure 4: Input tuples indicators screen

they wish to clean further, e.g., asking additional experts whether they are correct. To assist the analyst further in this choice, we propose an algorithm that automatically selects these tuples.

The input to the algorithm is a set of output tuples of interest, a desired MES level $p*$, set, by default, to the minimal error probability of any input tuple, and optionally maximal execution time. The algorithm iteratively selects tuples to clean further until the MES of all the selected output tuples is at most $p*$ or until a timeout is reached. In each iteration, a batch of input tuples is selected and is passed to the external Interactive Cleaning module, and this module returns as output the tuples with their updated error probability and possibly updated correctness labels (blue, dotted arrows in Figure 1).

We next overview the operation of the algorithm for a single output tuple o, and the extension to multiple output tuples is straightforward. The candidate tuples for further cleaning are those who participated in the derivation of o (which can be read from the provenance). We select in each iteration a batch of such tuple according to the provenance structure and the current correctness label of the output, $v^Q(o)$, as follows. If $v^Q(o) = 1$, we choose one satisfied term in $L^Q(o)$ with the smallest number of variables whose error probabilities are above $p*$, and add its variables to the batch. Otherwise, we choose from each term, among the variables set to 0, the one with minimal probability and add it to the batch. The chosen tuples are passed on to the Interactive Cleaning module, that can process them in parallel. We repeat this iteratively until the MES of o reaches the desired level.

4 Implementation and Demo Scenario

We now describe our prototype implementation, whose code and screenshots can be found in `https://github.com/ransch/CleanEr`. Recall Figure 1. The MES Computation and Risky Tuple Computation modules are implemented in Python 3.9 using SymPy (`https://www.sympy.org`) for processing Boolean formulae and OR-Tools library (`https://developers.google.com/optimization/introduction/python`) for solving ILPs. The Crucial Tuple and Cleaning Selection Algorithms are implemented in Javascript, and the User Interface is built using Next.js (`https://nextjs.org`), accessing the python modules through a Flask REST API (`https://flask.palletsprojects.com`). We use an existing query-guided data cleaning solution for data management and for the three

external tools [3]. We also provide our own implementation, with expert interaction incorporated in the display, allowing participants to play the role of human experts and control the errors in the cleaning process.

We will demonstrate the use of CleanEr over NELL [4], a general-purpose database constructed by information extraction from the Web that therefore contains many errors. NELL also includes many readily available real expert correctness labels. To illustrate the initial cleaning process, we will choose a query, and play the role of data experts by entering correctness labels (some of which will be deliberately erroneous) for input tuples on request by the Data Cleaning module. We will then show the query result screen (Figure 3), displaying the output tuples estimated to be correct and incorrect in white and gray resp., both in decreasing MES order by default.

We will then ask participants to choose output tuples of interest. By clicking them they will be able to see which input tuples contributed to their derivation (Figure 4). Each input tuple is presented with its correctness label, its error probability, and whether it is crucial or risky with respect to the selected output tuple. The input tuples may further be filtered by the user according to their properties (e.g., whether they are crucial, risky, etc.). We will explain to the audience the intuition on how MES is computed. Based on the presented indicators, we will choose input tuples and perform further cleaning steps on them. We will observe the effect of these cleaning steps on the labels and MES of the output tuples (Figure 3), e.g., via arrows showing for which tuples the MES increased or decreased. We will also invite participants to choose which tuples to clean and observe the effect.

Finally, we will demonstrate the execution of the algorithm from Section 3.2 on output tuples of the audience's choice, observing the effect on the output tuples in comparison with the effect of manually selected tuples.

Acknowledgments. This work was partly funded by the Israel Science Foundation (grant No. 2015/21) and by the Israel Ministry of Science and Technology.

References

[1] M. Bergman, T. Milo, S. Novgorodov, and W.-C. Tan. Query-oriented data cleaning with oracles. In *SIGMOD*, 2015.

[2] R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. *PVLDB*, 1(1), 2008.

[3] O. Drien, M. Freiman, A. Amarilli, and Y. Amsterdamer. Query-guided resolution in uncertain databases. *Proc. ACM Manag. Data*, 1(2), 2023.

[4] T. M. et al. Never-ending learning. In *AAAI*, 2015.

[5] A. Fariha, A. Tiwari, A. Meliou, A. Radhakrishna, and S. Gulwani. CoCo: Interactive exploration of conformance constraints for data understanding and data cleaning. In *SIGMOD*, 2021.

[6] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *SIGMOD*, 2007.

[7] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. ActiveClean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12), 2016.

[8] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1), 2010.

[9] L. Mo, R. Cheng, X. Li, D. W. Cheung, and X. S. Yang. Cleaning uncertain data for top-k queries. In *ICDE*, 2013.

[10] P. Senellart, L. Jachiet, S. Maniu, and Y. Ramusat. ProvSQL: Provenance and probability management in postgresql. *PVLDB*, 11(12), 2018.

[11] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.

[12] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD*, 2014.