Towards an End-to-End Human-Centric Data Cleaning Framework

El Kindi Rezig[♠] Mourad Ouzzani[◇] Ahmed K. Elmagarmid[◇] Walid G. Aref[★] Michael Stonebraker[♠]
[♠]MIT CSAIL [★]Purdue University [◇]Qatar Computing Research Institute, HBKU
elkindi@csail.mit.edu {mouzzani,aelmagarmid}@hbku.edu.qa aref@cs.purdue.edu stonebraker@csail.mit.edu

ABSTRACT

Data Cleaning refers to the process of detecting and fixing errors in the data. Human involvement is instrumental at several stages of this process such as providing rules or validating computed repairs. There is a plethora of data cleaning algorithms addressing a wide range of data errors (e.g., detecting duplicates, violations of integrity constraints, and missing values). Many of these algorithms involve a human in the loop, however, this latter is usually coupled to the underlying cleaning algorithms. In a real data cleaning pipeline, several data cleaning operations are performed using different tools. A high-level reasoning on these tools, when combined to repair the data, has the potential to unlock useful use cases to involve humans in the cleaning process. Additionally, we believe there is an opportunity to benefit from recent advances in active learning methods to minimize the effort humans have to spend to verify data items produced by tools or humans. There is currently no end-to-end data cleaning framework that systematically involves humans in the cleaning pipeline regardless of the underlying cleaning algorithms. In this paper, we present opportunities that this framework could offer, and highlight key challenges that need to be addressed to realize this vision. We present a design vision and discuss scenarios that motivate the need for this framework to judiciously assist humans in the cleaning process.

1 INTRODUCTION

Businesses often collect large volumes of data to inform key decisions. However, because data can be humongous and highly volatile, it is infeasible for humans to manually verify its accuracy. As a result, decision-makers have to deal with possibly-inaccurate data that may inherently lead to faulty business decisions. There are abundant research efforts to detect and repair the many types of data errors that one sees in the wild. This process is also known as data cleaning. Data errors include duplicates [12], violations of integrity constraints [7], and missing values [2]. While ideally we want to be able to fully automate this process, it has been widely recognized that humans have to be involved at various stages of the data cleaning pipeline [1, 11, 26]. A large spectrum of data cleaning systems involve humans. Examples include Poter's Wheel [19], GDR [26], KATARA [8], CrowdER [24], and UGuide [22]. Each of these systems involves humans to solve a particular data cleaning task. However, an end-to-end data cleaning framework that involves humans in a way that is orthogonal to the underlying cleaning algorithms is not yet available. Looking at existing data cleaning techniques, we make the following observations:

Human involvement is algorithm-driven: Humans are the ultimate authority in verifying the accuracy of the data. Because it is impractical to have humans correct the entirety of the data, many techniques strive to involve humans judiciously so as to maximize the benefit of their feedback in the cleaning process [18, 24–26]. Typically, humans are tightly coupled to the cleaning logic, i.e., humans are involved in ways that are dictated by the cleaning algorithm being used. This coupling is necessary to produce good quality results for specific data cleaning tasks. However, if we want to "plug-and-play" arbitrary tools to clean different types of data errors, then, we need a way to involve and manage humans in an algorithm-agnostic fashion in the data cleaning pipeline. This generic inclusion of humans is not meant to replace human-guided algorithms, in fact, they should go hand-in-hand to unlock various use cases in the cleaning process.

Data singularity: An assumption that is usually made in cleaning algorithms is that the data is subjected in its entirety to a given cleaning algorithm [10, 20]. However, in practice, different parts of the data are cleaned by different agents. For instance, one may use an automatic data cleaning tool to find the correct mapping of the Zip code to a City name while requesting humans to correct the Salary data (one would not trust an automatic algorithm to modify the salary data). This motivates the need for a cleaning framework that supports both human and automatic agents to holistically clean different parts of the data. In the remainder of the paper, we refer to the detection and repairing agents as *cleaning agents*.

Source of errors: There are multiple factors that can influence the quality of the computed repairs, e.g., the humans involved, the data quality rules, and the repair algorithm. However, existing techniques do not assess the effect of various factors that produce the data repairs (for example, many rule-based repair algorithms assume the rules are correct [5, 9, 15]). Understanding this effect is crucial in identifying bottlenecks in the data cleaning pipeline. Just as important as suggesting potential data errors for humans to verify, it is important to make it easy for humans to identify *faulty* factors (rules, humans, external resources, etc.) that have been involved to compute the inaccurate repairs.

Humans are not always right: Many human-driven data cleaning techniques assume that humans (e.g., experts) are *perfect* [26]. However, in practice, humans may make mistakes at various stages of the data cleaning pipeline (e.g., in the detection, repairing, or validation phases). Understanding how humans interact with the

HILDA'19, July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6791-2/19/07…$15.00
https://doi.org/10.1145/3328519.3329133

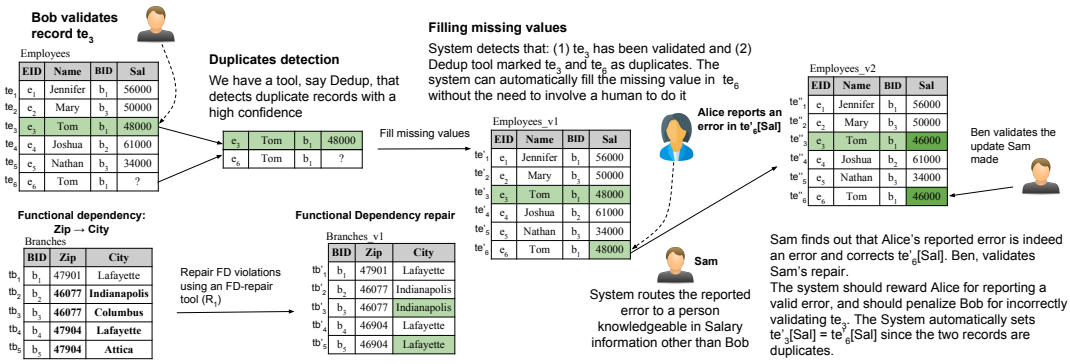


Figure 1: Example data cleaning scenario involving various cleaning tasks and agents

data is important to judiciously involve them in the cleaning process. For instance, an error reported in the *Sales* data by a person working in the *Sales* department should have more weight than one reported by a human working in another department. Therefore, there are several nuances in the human feedback that need to be dissected to effectively involve humans in the cleaning process.

Example 1.1. Refer to Figure 1. Table *Employees* contains employee data, e.g., name, salary, and the branch they belong to (BID). Table *Branches* contains the list of branches (BID) and their location (Zip, City). We assume all branches are based in the State of Indiana, thus we omit the State attribute from Table *Branches*.

Scenario 1: As we illustrate in this example, there are many use cases where it is useful to have a high-level, algorithm-agnostic understanding of how different components in the cleaning pipeline interact with each other. Because they were designed to solve a specific data cleaning task, existing human-guided algorithms do not capture those use cases. In this scenario, we present a mix of data cleaning operations performed by various agents to repair the tables *Employees* and *Branches*. The workflow is as follows:

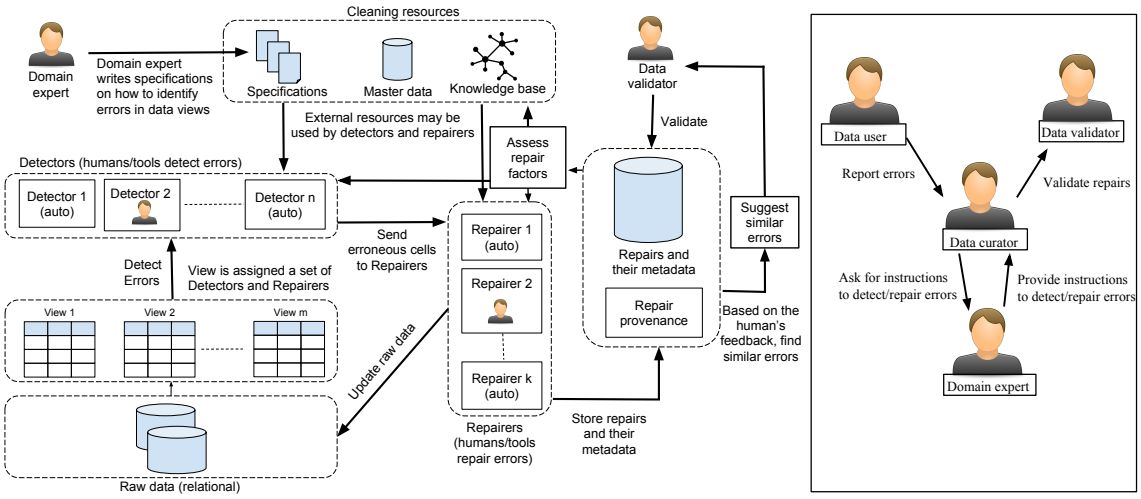
(1) *Bob* validates record te_3 as being correct.
(2) Table *Employees* is deduplicated using a tool, *Dedup*. This latter reports records te_3 and te_6 as duplicates.
(3) We would like to fill in the missing values. Instead of assigning a human to do it, the system should be able to notice that since te_3 and te_6 are duplicates, and *Bob* previously marked record te_3 as being correct, then, the missing value $te_6[Sal]$ can be set to $te_3[Sal]$. The resulting table is $Employees\_v_1$.
(4) *Alice* reports an error in cell $te'_6[Sal]$ in table $Employees\_v_1$.
(5) The system should be able to automatically ask a human, *Sam*, who is knowledgeable about the Salary data to repair the reported error. The system should be able to notice that it is better to choose a human other than *Bob* to examine the error, so that it can then compare their decisions. *Sam* then corrects the error and updates $te'_6[Sal]$. The system also updates $te'_3[Sal]$, since te'_3 and te'_6 were previously marked as duplicates.
(6) *Ben* validates the repair *Sam* made.
(7) The system should be able to capture that *Bob* is not that reliable when it comes to validating employee records, that *Alice* reports valid errors, and that *Sam* is reliable in repairing the salary data. Table $Employees\_v_2$ contains the fixed errors.
(8) We would like to enforce a functional dependency (FD) rule (ϕ_1: $Zip \rightarrow City$) on the *Branches* table. The cells marked in boldface violate ϕ_1. We use an FD repair tool, R_1, to automatically repair those violations. Table $Branches\_v_1$ is the repaired instance.

Scenario 1 shows that it could be useful to have a holistic strategy to deal with various cleaning agents acting on different parts of the data. Such a framework has the potential to facilitate the human involvement in the cleaning pipeline at a high-level. However, realizing such a framework poses several challenges. Particularly, Scenario 1 raises several questions:

- How do we know how confident *Bob* is about the validation he has performed on te_3? Asking another human to verify *Bob's* validation is expensive. How can we model *Bob's* knowledge on different parts of the data?
- How do we assess the quality of automatic tools, such as *Dedup* and R_1 so that we know if a human validation is required after these tools are executed?
- How do we assess *Alice's* knowledge on the data?
- How can the system automatically route reported errors to humans with the right expertise to examine them?
- What if a certain repaired cell, say, $tb'_3[City]$ is deemed incorrect. Should we examine R_1 or the FD rule ϕ_1, or both? In other words, how can the system isolate the culprits in the data cleaning pipeline so that humans can easily debug cleaning decisions?

The above questions are among many others that need to be addressed to effectively involve humans in the data cleaning pipeline. To this end, we propose a vision for an end-to-end data cleaning system that supports the following features:

- **Heterogeneity:** The system should be able to simultaneously support cleaning agents of different types, i.e., human, automatic or semi-automatic. Since different parts of the data can be cleaned by different agents, each agent receives part or all of the data as input (Section 2).
- **Isolation:** The system should treat cleaning agents as black boxes while still enabling humans to detect, repair, and verify errors or bottlenecks (e.g., cleaning agents that are associated with wrong repairs) in the cleaning process. Thus, humans



(a) Architecture

(b) Example Human Interaction Model

Figure 2: Architecture (vision) and an example Human Interaction Model

are isolated from the specific cleaning logic of a specific cleaning algorithm (Section 2, 6).

- **Human Cost Optimization:** The system should be able to reason about the expertise of different humans when assigning cleaning tasks. It should also account for the cost and expertise when involving a given human in a cleaning task (Section 3- 5).
- **Accountability:** There are many factors involved in computing a given repair. Based on human feedback (e.g., human reports an error in a repaired cell), the system should automatically assess the reliability of different factors (e.g., agents, rules, etc.) that were involved in computing a repair over time. This assessment is crucial for humans to identify bottlenecks, i.e., factors associated with inaccurate repairs, in the data cleaning process (Section 6).

2 ARCHITECTURE OVERVIEW

2.1 Terminology

Consider a relational database D containing relations R_1, R_2, ..., R_n. Every relation R_i ($1 \leq i \leq n$) contains a set of attributes A_1^i, A_2^i, ..., A_k^i with domains $dom(A_1^i)$, $dom(A_2^i)$, ..., $dom(A_k^i)$ respectively. For the instance I_i of R_i containing tuples T, a cell c is the value of a tuple $t \in T$ in attribute $A \in R_i$, denoted $t[A]$.

Detector: Detectors are humans or programs that, given a set of cells as input, provide a set of cells that are potentially erroneous as output. Example detector programs are those that use data quality rules (e.g., Denial Constraints) to identify the cells that violate those rules.

Repairer: Repairers are humans or programs that update the input cells in a way that "fixes" the data errors.

Repair: We refer to an update to a set of cells C made by a Repairer R as a *repair*.

Accurate Repair: A repair is accurate if it contains cells with values that match the ground truth.

2.2 Architecture

Figure 2a illustrates the proposed architecture to implement our system vision. In a nutshell, there are four main components: Detectors, Repairers, Cleaning resources and Validators. All the Detectors and Repairers are treated as pluggable black boxes. One could use any number of detection and repairing algorithms to clean the data. Since different agents can be involved to detect/repair different parts of the data, Detectors and Repairers are applied to different data views. Furthermore, Detectors and Repairers may use *cleaning resources* such as rules, masterdata, etc., to detect and/or repair the data. Cleaning resources are commonly produced by humans. We explore in Section 6 how the envisioned system should make it easy for humans to identify agents or cleaning resources that produce inaccurate repairs. Finally, in addition to detecting and fixing errors, humans are also able to validate the computed repairs, and based on their feedback, the system assesses the reliability of different factors that were involved in computing the repairs.

Data Cleaning job: The envisioned system allows humans to declaratively specify a data cleaning operation as a function of several parameters. Specifically, a data cleaning job is represented as the quadruplet $\langle C, D, R, V \rangle$ where: C is the set of input cells (cannot be empty), D is the set of Detectors to be used to detect errors in C, R is the set of Repairers to repair the errors found in C, V is the set of humans to validate the produced repairs. Using this representation, we can capture most of the cleaning scenarios. For example, if D and R are empty and V is not empty, then, the job will be a validation task on the cells in C.

Example 2.1. In Example 1.1 (scenario 1), the cleaning jobs are represented as:
$job_1 : \langle C = *, D = \emptyset, R = \emptyset, V = \{\text{“}Bob\text{”}\} \rangle$

$job_2 : \langle C = *, D = \{\text{“}Alice\text{”}\}, R = *, V = * \rangle$
$job_3 : \langle C = \{tb[Zip] = *, tb[City] = *\}, D = \{\phi_1\}, R = \{R_1\}, V = \emptyset \rangle$

job_1 states that “Bob” can validate any cell in the data.

job_2 states that “Alice” can report errors in any data cell, and if she does, the system should automatically assign a repairer to update the data, and a validator to verify the update.

job_3 states that we are using ϕ_1 (in practice, there is a program that projects ϕ_1 on the data to extract violations, but for simplicity, we are only including the rule) to detect errors in all the Zip and City cells. The errors are then repaired using the FD-repair tool R_1. The repairs are not subjected to validation ($V = \emptyset$).

3 HUMANS IN THE CLEANING PROCESS

While several research efforts involve the human in specific cleaning problems (e,g,. Entity Resolution [24], Integrity Constraints [26], Data Fusion [18]), there is no proposal that involves humans for general data cleaning (regardless of the cleaning problem at hand). Furthermore, characterizing human expertise for the purpose of general data cleaning remains unexplored. Particularly, data cleaning efforts that use crowdsourcing [8, 24] assume that crowd workers are non-experts. On the other end of the spectrum, we have data cleaning methods [22, 26] that assume humans are experts whose feedback is assumed to be always correct. In practice, humans can have different degrees of expertise on different parts of the data. We shed some light to highlight key challenges that need to be addressed to realize this characterization.

3.1 Characterizing Human Expertise

Cleaning tasks: Humans interact in various ways in the cleaning process. Based on our vision, we list four human-driven tasks, referred to in this paper by *cleaning tasks*, that a human-centric data cleaning system needs to support: (1) **Detection:** Humans should be able to report errors in a given set of cells; (2) **Repairing:** When errors are reported, humans should be able to fix those errors by updating the data to reflect accurate values; (3) **Validation:** Humans should be able to verify a repair that has been made by another cleaning agent (human or automatic); and (4) **Specification:** Humans should be allowed to write specifications (e.g., FD rules) to detect data errors.

Human roles: We distinguish four human roles (Figure 2b): (1) Data User: person using the data; (2) Data Curator: person fixing data errors; (3) Data Validator: validates fixes made by the Data Curator; and (4) Domain Expert: the person writing specifications (e.g., in the form of rules).

Data Expertise: Humans have different knowledge about different parts of the data. For example, a person working in the Sales department is probably more aware of the Sales data than someone working in the Marketing department. When assigning humans to cleaning tasks, it is important the system makes sure the assigned humans are knowledgeable enough in the data they are asked about. In a human-centric data cleaning system, every human has a history of the data cells they helped clean. Through the *Validation* task, the system can learn how *good* a given human is for a certain cleaning task and for a given cell. We believe that the framework should (1) support rules to assign data items to human groups (e.g., sales data is reviewed by sales people); and (2) score the human expertise for a given data cleaning task. For example, a simplistic measure to quantify the expertise of a human h on data cells C, for a task T, is the following (there are several sophisticated models in the literature to capture human expertise, and the following equation is only presented to illustrate a naive way to capture human expertise):

$$Expertise(h, C, T) = \frac{\#(h, C, T)}{\#validated(C, T)} \quad (1)$$

Equation 1 calculates the ratio of cells in which a human h performed a task T over the number of cells in C that were subject to validation.

For example, we would like to measure the expertise of a human h in the detection task for a set of cells C. Assume h correctly reported errors in two of these cells. The total number of cells that were validated by a Data Validator in C is 4. Therefore: $Expertise(h, C, T = \text{“}Detection\text{”}) = \frac{2}{4} = 0.5$

Cost Model. Involving humans is generally expensive. It is important to be able to characterize the cost of involving a human to perform a certain cleaning task. For example, involving domain experts is generally more costly than involving ordinary data users. This cost should also take into consideration the availability of different human roles. For instance, if we have very few data curators, we would want to make sure they are assigned the most critical tasks only. Furthermore, it would be interesting to incorporate the cognitive effort of looking at the data to perform a cleaning task.

Human Budget. The human budget for a data cleaning job j could be expressed as a combination of many factors including the maximum number of humans available to perform a certain task, the total money cost to spend to perform a task, time limit, etc.

We are now ready to formally define a Human characterization of a human h.

Definition 3.1. Human Characterization. A human h in a data cleaning scenario is represented as h: $\langle Role, Data, Cost, Expertise \rangle$ where $Role$ is the role of the human, $Data$ is the set of cells h is knowledgeable about, $Cost$ is the cost of involving h, $Expertise$ is a score that reflects how good h is for the role $Role$ in cells $Data$.

Suggesting possible errors: Given a set of initial labels (correct/wrong) on data items, the system should be able to benefit from active learning classifiers [6] to interactively suggest interesting data items to label by the data validator (Figure 2a) and refine a classification model to predict possibly faulty data items.

4 TASK ALLOCATION

The envisioned system should allow users to define data cleaning jobs without explicitly stating the humans involved. Specifically, the system should be able to select from a pool of humans H with different characterizations, the right human for the right task. An example job is defined as follows:

$job_4 = \langle C = *, D = \{\text{“}Alice\text{”}\}, R = *, V = \emptyset \rangle$

In job_4, The set of repairers includes all the humans H available for this task. This makes the system responsible for assigning a repairer for the errors that *Alice* detects. In our system, we are only interested in automatically assigning humans to cleaning tasks. Assigning automatic agents to cleaning tasks is outside the scope of the proposed vision.

Given a set of humans with their characterizations, the proposed system should be able to automatically assign cleaning tasks to them. We now discuss key building blocks that are needed to effectively assign cleaning tasks to humans.

4.1 Human-to-Human Interactions

It is crucial to develop an interaction model between different human roles to optimize the cleaning effort. Ideally, the interaction model should produce the best cleaning results with the least human cost. In particular, a "good" interaction model should (1) minimize the communication overhead between different human roles; and (2) account for all possible human-to-human interaction cases in the cleaning scenario. We understand that in the real world, interactions between humans are often complex and not well-defined, however, we would like to explore some key human-to-human scenarios. For instance, as illustrated in Figure 2b and using the roles we defined previously, the possible human-to-human interaction scenarios are the following: Data User reports errors to the Data Curator; Domain Expert provides specifications (e.g., rules, etc.) to the Data Curator to enforce on the data; Data Curator reports errors found in specifications to the Domain Expert; and the Data Validator validates fixes performed by the Data Curator.

4.2 Task Assignment

Given a data cleaning job j for cells C, a pool of humans, say H, and a budget, say B, the framework should assign automatically cells in C to humans in H (e.g., job_4 defined above). The assignment should guarantee the following properties: (1) **Coverage:** If the job is to be performed by humans only, every cell in C should be covered by at least one human; (2) **Maximize expertise**: The assigned humans should have good knowledge about cells in C; (3) **Minimize cost:** The human cost should not exceed Budget B.

Example 4.1. Assume that we have a validation task on all the *Sal* cells of Table *Employees* in Figure 1 defined as the data cleaning job job_5 as follows:

$job_5 = \langle C = Employees[Sal], D = \emptyset, R = \emptyset, V = * \rangle$

Consider a pool of humans $H = \{Alice, Bob, \text{and } Sam\}$, and a human budget ($B = 1$) for job_5 expressed (for simplicity) as the maximum number of humans involved in the task. *Alice*, *Bob*, and *Sam* have good knowledge on the following sets of cells $\{te'_1[Sal], te_2[Sal], te_3[Sal], te_4[Sal], te_5[Sal]\}$, $\{te_3[Sal], te_4[Sal]\}$, and $\{te_5[Sal]\}$, respectively. In this scenario, the system should assign job_5 to *Alice* only (since $B = 1$ and *Alice* covers all the cells of *Sal*).

5 CROSS-AGENT COST OPTIMIZATION

Minimizing the human cost to repair the data has been the cornerstone of numerous research efforts [26]. However, when the human is not aware of the cleaning algorithm's logic, it becomes hard to achieve this goal. For instance, consider a Detector *Dedup* that detects duplicate records using a clustering algorithm. *Dedup* uses some similarity measure *Sim* to decide if a set of data points belong to the same cluster. Knowing how *Dedup* works, if we want to validate its output, we could ask a human to verify if the closest points (using *Sim*) between the clusters are indeed not duplicates. In the case of our envisioned system, *Dedup* would simply provide its output as clusters expressed in terms of records (duplicate records would share the same value of a designated attribute). In this case, involving the human usefully becomes more challenging. In general, the optimization problem is intractable and we only shed some light on key questions to address when building such an optimizer.

We need to answer the following questions: (1) When we have human and automatic cleaning agents, what are the consequences of involving one over the other on human cost and data quality? (2) Given multiple humans that are assigned the same set of cells to repair, which human do we choose? (3) How do we schedule different cleaning jobs in order to achieve an optimal human cost and data quality?

5.1 Quantitative Cost Optimization

When we have humans and automatic agents that are assigned overlapping input data, which one should we prioritize? and what are the consequences for each choice? For example, in Example 1.1, what if *Sam* has also been assigned to repair cells $tb_2[City]$, $tb_3[City]$, $tb_2[Zip]$, $tb_3[Zip]$. In this case, the input to R_1 (automatic agent) overlaps with the input to *Sam*. This overlap is possible in practice. For example, one may want an automatic agent to clean a large amount of data while requiring the human to repair only a small subset of it. If we want to minimize human intervention, we can simply prioritize automatic agents over humans for a given set of cells. As a result, because *Sam* is assigned cells that are part of the input to R_1, we can simply save cost by not asking *Sam* to repair $tb_2[Zip]$, $tb_2[City]$, $tb_3[Zip]$, $tb_3[Zip]$, but we would still ask him to fix the Salary value in $te'_6[Sal]$ because this cell is not input to an automatic agent. While human intervention is minimized in this strategy, we note: (1) The human cost is minimized at the expense of data quality. That is, humans generally perform better repairs than automatic agent. (2) This strategy can be suitable if the automatic agents provide high repair accuracy.

5.2 Qualitative Cost Optimization

This strategy gives preference to humans over automatic agents. As a result, the human cost will be higher compared to the previous strategy. In this strategy, when the input cells for an automatic agent overlap with those for a human agent, the system first invokes the automatic agent, and then asks the human to correct the overlapping cells. This way, the human updates will be ordered last and will not be undone by the automatic agent. Using this strategy, we note: (1) Because humans are prioritized over automatic agents, it is expected that this strategy results in better data quality compared to the previous one. (2) Human cost is high in this strategy. This strategy is suitable when invoking the automatic agents would result in a low repair quality.

6 IDENTIFICATION OF BOTTLENECKS

There are several factors involved in repairing a cell, say c. We refer to these factors as $factors(c)$, and include: (1) Detectors: Human or automatic agents that have flagged c's old value as erroneous; (2) Repairers: Agents (humans or automatic) that have computed the repair in c; (3) Cleaning resources: Resources used to compute c, which include Rules, Metadata, etc. (4) Data Validators: Humans that validated c as a correct repair (if c has been subject to validation). Therefore, the framework has to keep track of the provenance of

every computed repair expressed in terms of all the factors that were involved to compute the repair for given cells.

After human validation, if a repair for c is deemed accurate (respectively, inaccurate), then every factor in $factor(c)$ should be rewarded (respectively, penalized). Providing this accountability will help identify factors that are commonly associated with inaccurate repairs. This assessment is crucial for humans as it helps them identify bottlenecks in the cleaning pipeline as a whole.

Scoring factors: One way to capture the reliability of different factors is to compute a score for each one of them that reflects how "good" each factor is. A simple way to capture the quality of a given factor $f \in factor(c)$ is the following:

$$Quality(f) = \frac{\#correct(f)}{\#validated(f)} \quad (2)$$

Equation 2 calculates the ratio of correct cells (as validated by a human) where f was involved over the total number of validated cells where f was involved.

Since they would result in inaccurate repairs, the "bad" factors would elicit more human feedback than the "good" ones. Therefore, identifying them is crucial to minimize the human cost in the cleaning process.

Scenario 2: Consider Example 1.1. We want to perform a new cleaning iteration with an additional FD rule that will be enforced on table $Branches\_v_1$ (Figure 1). Let us add an *incorrect* FD rule: $\phi_2 : City \rightarrow Zip$. This rule states that records that share the same City should have the same Zip code. This is in reality not correct because a city can have multiple zip codes. We now create a new data cleaning job: $job_6 : \langle C = \{tb'[Zip] = *, tb'[City] = *\}, D = \{\phi_1, \phi_2\}, R = \{R_1\}, V = \emptyset\rangle$

The set of violating cells will be $C^{\not\models} = \{tb'_1[Zip], tb'_1[City], tb'_4[Zip], tb'_4[City], tb'_5[Zip], tb'_5[City]\}$. Let us assume that R_1 lifts the violation by setting $tb'_1[Zip] = 47904$. Let us call the repaired cell $tb''_1[Zip]$.

If we want to ask *Jen*, a human validator about the repairs computed by R_1, which violating cells should we ask her to validate? More importantly, how does the choice of cells we choose to validate affect the ability to isolate troublesome factors? Furthermore, how can we adjust the choice of cells to validate to our available human budget? To shed some light on answering those questions, we discuss the following key cases:

(1) If we validate cells that were computed using many factors, we get an *aggregate* feedback on all the involved factors. For instance, asking *Jen* to validate $tb'_5[City]$ would provide a feedback about ϕ_1, ϕ_2 and R_1 (that cell was involved in two violations across different cleaning iterations). This is useful to get a feedback about many factors at once, however, it may not be good at isolating factors to identify the bottlenecks causing inaccurate repairs. This strategy is suitable when the cost of involving human validators is high. Therefore, this strategy allows us to have an idea about as many factors as possible using the least number of cells to hopefully identify a combination of factors that produced inaccurate repairs.

(2) If we validate cells that were computed using few factors, we get a more fine-grained feedback about the involved factors. For example, asking *Jen* to validate $tb''_1[City]$ and $tb''_1[Zip]$ (which represent the repairs for cells $tb'_1[City]$ and $tb'_1[Zip]$ respectively) would isolate ϕ_2 as a problematic FD rule (since $tb'_1[Zip]$ and $tb'_1[City]$ violated ϕ_2 only). While this strategy provides a better isolation of factors, it involves more cells to be validated which translates into spending a higher human cost.

7 RELATED WORK

There is a rich literature on Data Cleaning techniques and theory [1, 4, 13, 14, 16, 21]. We discuss a few papers in two areas: general data cleaning systems and human-assisted data cleaning techniques.

General Data Cleaning Systems: A strongly related system to our proposal is the data cleaning system NADEEF [10]. Like our envisioned system, NADEEF adopts a system-approach to realize an end-to-end data cleaning framework that supports a number of data cleaning tasks (rule-based repairing, deduplication, etc.). NADEEF offers a programming interface so that users can implement detection and repairing components. As opposed to NADEEF, our framework not only supports automatic agents, but supports involving humans in all the stages of the data cleaning pipeline. Another related system is *KATARA* [8] which leverages the crowd and knowledge bases (KB) to clean dirty tables. *KATARA* is not rule-driven and can repair any cells in the input tuples (hence its categorization as a general data cleaning system). *KATARA* jointly uses the KB and the crowd feedback to identify correct and erroneous data in the input dirty table. Our vision is different from *KATARA* as we are considering humans with different expertise and roles (as opposed to using non-expert crowd workers). Furthermore, our envisioned system supports any number of cleaning agents for different cleaning tasks (integrity constraints, deduplication, etc.).

Human-Guided Data Cleaning: The idea of assisting humans in specific data cleaning problems (Entity Resolution, Schema transformations, etc.) has been widely studied [3, 11, 19, 22, 23, 26]. However, human involvement in these proposals is coupled to the underlying cleaning logic. Our proposal complements this line of work by enabling any number of cleaning tools to co-exist in the same platform while supporting multi-tool, algorithm-agnostic use cases that are otherwise not captured when using a tool to perform a specific data cleaning task.

The crowdsourcing literature [17] focuses mainly on scenarios that involve non-expert humans. However, in our framework, humans have multiple roles with varying degrees of expertise. Furthermore, our setup supports non-human agents as well.

8 ACKNOWLEDGMENT

Walid Aref acknowledges the support of the National Science Foundation under Grant Number III-1815796.

9 CONCLUSIONS

Scaling the generation and processing of big data often comes at the cost of its quality. We presented our vision for a framework that assists humans in all the major stages of the cleaning pipeline. We proposed several properties that need to be met to unlock the full potential of mixing humans and cleaning tools in the data cleaning pipeline. We raised key questions that need to be addressed to bring the vision to life. There are still many other questions that were not addressed in this vision such as data privacy, i.e., how can humans clean the data in the presence of privacy constraints? But we believe the vision raises central questions that we need to answer to realize a human-centric data cleaning system.

REFERENCES

[1] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *PVLDB* 9, 12 (2016), 993–1004. http://www.vldb.org/pvldb/vol9/p993-abedjan.pdf

[2] Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. 2016. DataXFormer: A robust transformation discovery system. In *ICDE*.

[3] A. Assadi, T. Milo, and S. Novgorodov. 2017. DANCE: Data Cleaning with Constraints and Experts. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 1409–1410. https://doi.org/10.1109/ICDE.2017.199

[4] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noel Novelli, and Saravanan Thirumuruganathan. 2018. Discovery of Genuine Functional Dependencies from Relational Data with Missing Values. *PVLDB* 11 (2018), 880–892.

[5] Philip Bohannon, Wenfei Fan, Michael Flaster, and Rajeev Rastogi. 2005. A Cost-based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*. ACM, New York, NY, USA, 143–154. https://doi.org/10.1145/1066157.1066175

[6] Q. Gu J. Han P. Yu C. Aggarwal, X. Kong. 2014. Active Learning: A Survey, Data Classification: Algorithms and Applications. In *CRC Press*.

[7] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*. 458–469. https://doi.org/10.1109/ICDE.2013.6544847

[8] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *SIGMOD Conference*.

[9] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. 2007. Improving Data Quality: Consistency and Accuracy. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. 315–326. http://www.vldb.org/conf/2007/papers/research/p315-cong.pdf

[10] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: A Commodity Data Cleaning System. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*. ACM, New York, NY, USA, 541–552. https://doi.org/10.1145/2463676.2465327

[11] Sanjib Das, Paul Suganthan G. C., AnHai Doan, Jeffrey F. Naughton, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra, and Youngchoon Park. 2017. Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services. In *SIGMOD Conference*.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2007).

[13] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers.

[14] Ihab F. Ilyas and Xu Chu. 2015. Trends in Cleaning Relational Data: Consistency and Deduplication. *Found. Trends databases* 5, 4 (Oct. 2015), 281–393. https://doi.org/10.1561/1900000045

[15] Solmaz Kolahi and Laks V. S. Lakshmanan. 2009. On Approximating Optimum Repairs for Functional Dependency Violations. In *Proceedings of the 12th International Conference on Database Theory (ICDT '09)*. ACM, New York, NY, USA, 53–62.

[16] Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, and Eugene Wu. 2017. BoostClean: Automated Error Detection and Repair for Machine Learning. *CoRR* abs/1711.01299 (2017). arXiv:1711.01299 http://arxiv.org/abs/1711.01299

[17] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. 2016. Crowdsourced Data Management: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (Sept 2016), 2296–2319.

[18] Romila Pradhan, Siarhei Bykau, and Sunil Prabhakar. 2017. Staging User Feedback toward Rapid Conflict Resolution in Data Fusion. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. 603–618.

[19] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 381–390. http://dl.acm.org/citation.cfm?id=645927.672045

[20] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1190–1201. https://doi.org/10.14778/3137628.3137631

[21] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Zhan Xu. 2013. Data Curation at Scale: The Data Tamer System. In *CIDR*.

[22] Saravanan Thirumuruganathan, Laure Berti-Equille, Mourad Ouzzani, Jorge-Arnulfo Quiane-Ruiz, and Nan Tang. 2017. UGuide: User-Guided Discovery of FD-Detectable Errors. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM, New York, NY, USA, 1385–1397. https://doi.org/10.1145/3035918.3064024

[23] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. 2014. Crowdsourcing Algorithms for Entity Resolution. *Proc. VLDB Endow.* 7, 12 (Aug. 2014), 1071–1082. https://doi.org/10.14778/2732977.2732982

[24] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *Proc. VLDB Endow.* 5, 11 (July 2012), 1483–1494. https://doi.org/10.14778/2350229.2350263

[25] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proc. VLDB Endow.* 6, 8 (June 2013), 553–564. https://doi.org/10.14778/2536354.2536356

[26] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F. Ilyas. 2011. Guided Data Repair. *Proc. VLDB Endow.* 4, 5 (Feb. 2011), 279–289. https://doi.org/10.14778/1952376.1952378