# The Case of Data Cleaning with Spatial and Temporal Awareness

Yuchuan Huang
huan1531@umn.edu
University of Minnesota, USA
(Supervised by Mohamed F. Mokbel)

## ABSTRACT

Though data cleaning systems have earned great success and wide spread in both academia and industry, they fall short when the data contains spatial and temporal information that affects functional dependencies. In particular, state-of-the-art data cleaning systems yield sub-optimal accuracy when cleaning attributes (e.g., census tract) that are in dependency of spatial attributes (e.g., latitude and longitude). Moreover, they cannot handle the case where the groundtruth of a functional dependency is changing over the time (e.g., the census tract of a location updates every 10 years). In this paper, I discuss two case studies I have done/am doing in my PhD time, aiming to inject the spatial and temporal awareness into data cleaning systems. In the first study, we propose Sparcle; a novel framework that injects spatial awareness into the core engine of rule-based data cleaning systems, that significantly boosts their accuracy when dealing with spatial data. Then for the second case, I show the motivating example where temporal awareness is in need for data cleaning, and justify why the problem is worth studying.

**VLDB Workshop Reference Format:**
Yuchuan Huang. The Case of Data Cleaning with Spatial and Temporal Awareness. VLDB 2024 Workshop: VLDB Ph.D. Workshop.

## 1 INTRODUCTION

Motivated by the imperfection of real data sets, along with the huge efforts carried by data scientists to manually clean their data, efforts have been dedicated to develop various approaches and systems for automated data cleaning. The large majority of such approaches (e.g., see [4, 6, 8, 14, 20, 24]) and systems (e.g., see [9, 12, 13, 18, 21, 25, 26, 31]) are rule-based, where functional dependencies between various attributes guide the data cleaning process. The success and immense need of such data cleaning systems made it widely adopted by industry [2, 11, 16, 22] and commercial startups [17, 28, 29].

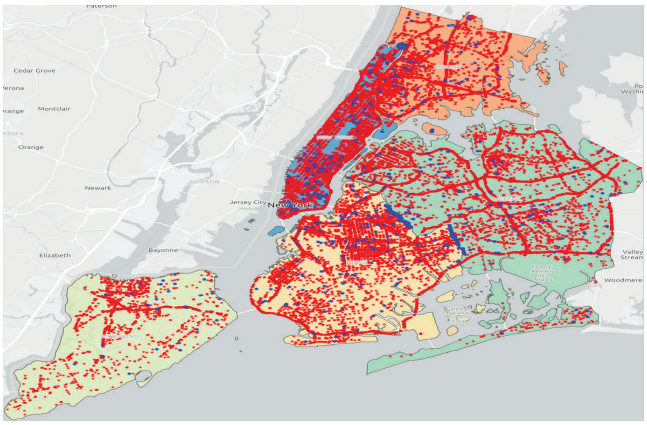
Unfortunately, with all its success and wide spread, we found state-of-the-art data cleaning systems fall short when the data contains spatial or temporal attributes. In particular, we found them yield sub-optimal accuracy when trying to clean spatial data with functional dependencies such as (*Latitude*, *Longitude*) → *Census Tract*. The main reason is that state-of-the-art data cleaning systems mainly rely on functional dependencies where there are sufficient co-occurrence of value pairs to learn that a certain value of an attribute leads to a corresponding value of another attribute. However,

This work is supported by NSF under grants IIS-2203553 and OAC-2118285.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

| ID | Latitude | Longitude | Borough | ... |
|---|---|---|---|---|
| 4486519 | 40.751441 | -73.973974 | Manhattan | ... |
| 4486555 | 40.690183 | -73.956825 | | ... |
| 4486604 | 40.681582 | -73.974638 | Brooklyn | ... |
| 4486609 | 40.683304 | -73.917274 | Queens | ... |
| 4486660 | 40.868165 | -73.831487 | Bronx | ... |
| ... | ... | ... | ... | ... |

(a) Part of Table of NYC Motor Vehicle Collision Data




(b) Map of NYC Motor Vehicle Collision Data

**Figure 1: NYC Motor Vehicle Collision Data**

for spatial attributes that represent locations, there is very little chance that two records would have exact same coordinates, and hence co-occurrence is unlikely to exist. Moreover, we observe the fact that the groundtruth with respect to a functional dependency may change throughout the dataset. For example, *Census Tract* of the above-mentioned dependency updates every 10 years. However, existing systems would only try to enforce a single version of groundtruth, which apparently results in sub-optimal accuracy.

## 2 CASE 1: SPATIAL-AWARE DEPENDENCY

In this case, we investigated the NYC Motor Vehicle Collision data [23], which includes 1,751,624 collision records that took place in the New York City since 2014. A snapshot of this dataset is in Figure 1(a) for five collision records and only four attributes of each collision (*ID*, *Latitude*, *Longitude*, *Borough*). The snapshot shows two kinds of errors: (1) the second record is missing the *Borough* information, and (2) the fourth record has the wrong *Borough* information. To get an idea of the scale of the problem, Figure 1(b) plots all the erroneous records over NYC map (421,013 records), where 418,896 records have a missing borough (plotted in red) and 2,117 records have incorrect borough (plotted in blue). We fed this data

**Table 1: Error Repairing of NYC Motor Vehicle Collision Data**

| | HoloClean | SPARCLE |
|---|---|---|
| **Total** | 58.7% | **99.4%** |
| Errors at duplicated location | 99.6% | **99.7%** |
| Errors at new location | 30.3% | **99.1%** |

along with the functional dependency: (*Latitude*, *Longitude*) → *Borough* to HoloClean [25] system as a state-of-the-art rule-based data cleaning framework. It is important to note that in this particular example, we do not rely on any external knowledge of borough boundaries. HoloClean only repaired 58.7% of the errors, which is a pretty low accuracy compared to its ability in cleaning non-spatial data with more than 95% accuracy [25]. To understand such poor accuracy, we distinguish between: (a) erroneous records that took place in the *same exact* location of at least one other correct record, and (b) erroneous records that took place in *new* locations where there is no other correct records. As depicted in Table 1, HoloClean was able to correct 99.6% of the former, but only 30.3% of the latter.

The main reason behind such poor performance of HoloClean, as a representative of rule-based data cleaning systems, is twofold: (1) Cleaning with functional dependencies relies on sufficient co-occurrence of value pairs to learn that a certain value of an attribute leads to a corresponding value of another attribute. However, for spatial attributes, there is very little chance that two records have the exact same coordinates, mainly due to the inherent inaccuracy of location-detection devices. Hence, a rule-based system will not be able to find sufficient *spatial* co-occurrence to be used to detect and repair erroneous entries. (2) The outcome of whether a certain record satisfies a rule is binary (*True* or *False*). However, in spatial rules, such outcome needs to be fuzzy, as a certain record may satisfy the rule in stronger terms than other records.

The closest related works to solve this problem would be the set of relaxed functional dependencies [5], including matching dependency [10], metric dependency [19], differential dependency [27], and ontology dependency [3]. However, these dependencies are mainly proposed to tolerate marginal syntactic difference for entities that are actually considered the same, e.g., the words "Ave." and "Avenue" should mean the same thing. While in spatial data, two records with nearby coordinates are truly two different records. As a result, they still yield sub-optimal cleaning accuracy (i.e., over 0.2 in terms of precision) compared with our proposed approach [15].

## 2.1 Proposed Approach: SPARCLE

To address on the problem, we propose SPARCLE (SPatially-AwaRe CLEaning) [15]; a novel framework that injects spatial awareness into the core engine of rule-based data cleaning systems as a means of boosting their accuracy. A key idea behind SPARCLE is that it goes beyond the traditional functional dependency rules of the form: *"Two records with the **same** location **should** have the same borough"* to support the more relaxed functional dependency form: *"Two records with **more similar** locations are **more likely** to have the same borough"*. To do so, SPARCLE injects two main spatial concepts into its host data cleaning system: (1) *Spatial Neighborhood*. To support going from the *"same"* predicate to the *"similar"* predicate, records with spatial attributes satisfying some spatial neighborhood

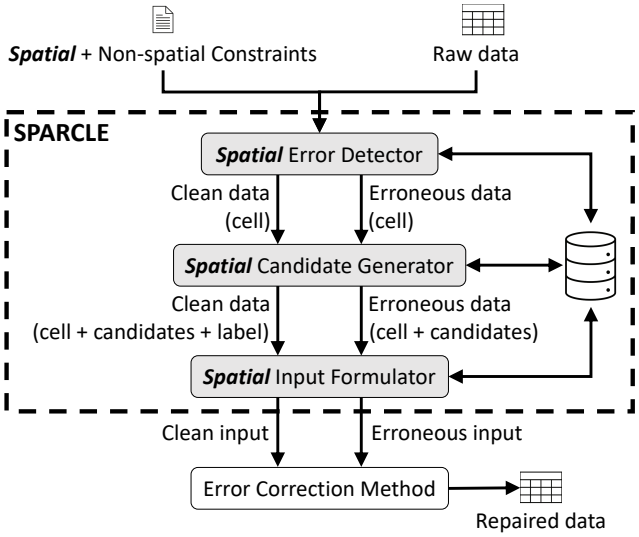



**Figure 2: SPARCLE Architecture**

(similarity) criteria should be considered as relatively equivalent with respect to the spatial functional dependencies; (2) *Distance Weighting*. To support going from *"should"* to *"likely"* and to have the keyword *"more"* in the relaxed functional dependency, records will be given a weight of how much they satisfy each rule, where the weight will be based on the distance between records satisfying the functional dependency. With this, the last column of Table 1 shows that, for NYC collision data, SPARCLE was able to correct 99.4% of all errors and 99.1% of the errors with new locations.

A main contribution of SPARCLE is that it proves that spatial awareness of *Spatial Neighborhood* and *Distance Weighting* could be and should be injected into most data cleaning systems. Our goal in SPARCLE is *not* to come up with a new data cleaning system. Instead, it is to *boost* the accuracy of current systems. In particular, SPARCLE lives inside a *host* data cleaning system, making it spatially-aware. To this end, its architecture follows the common architecture of most rule-based data cleaning systems. In particular, such systems (e.g., [9, 12, 18, 21, 25, 31]) are typically composed of four back-to-back components, *error detector*, *candidate generator*, *input formulator*, and *error corrector*. The first three modules are mainly for error detection and preparing the data in some format that can be repaired using a statistical method in the last module, which is very system-specific. Hence, SPARCLE focuses on modifying the first three modules to be: *spatial error detector*, *spatial candidate generator*, and *spatial input formulator*, while leaving the fourth module intact as it is system-specific. These modified modules will mainly support spatial dependencies. Multiple constraints are handled separately and in parallel. Non-spatial dependencies are still supported through the host data cleaning system. The outcome of the third module (*spatial input formulator*) is forwarded as is to the downstream *error correction* module to be combined with other non-spatial constraints to find out the final repaired value.

We implemented SPARCLE inside two state-of-the-art data cleaning systems HoloClean [30] and Baran [21]. Experiments on four real datasets, each with multiple spatial and non-spatial dependencies, show that SPARCLE boosts 0.3-0.9 cleaning accuracy (in terms of F1 Score) of the host system.

## 2.2 Sparcle Architecture

Figure 2 depicts Sparcle architecture, deployed inside a host data cleaning system. Sparcle takes two types of inputs, the raw data to be cleaned and the constraints that define functional dependencies. The output of Sparcle is the detected erroneous *cells*, where a *cell* is a certain attribute of a certain record, along with a weighted list of suggested correct values for each cell. Internally, Sparcle follows similar architecture to that of rule-based data cleaning systems, mainly composed of three modules, *spatial error detector*, *spatial candidate generator*, and *spatial input formulator*. A brief description of Sparcle input, modules, and output is below:

Sparcle **Input: Constraints.** As Sparcle is injected into a host rule-based data cleaning system (e.g., [12, 21, 25, 30]), it is automatically triggered only when there are spatial constraints, and it only takes care of such constraints. Non-spatial constraints over the same input data will still be supported by the host data cleaning system without any interference from Sparcle.

**Spatial Error Detector.** The input to this module is the input to Sparcle. The output is two sets of cells, erroneous and clean cells. It injects spatial-awareness into existing error detection modules. Hence, instead of detecting errors based on exact co-occurrence, it relaxes the co-occurrence criteria to consider records within spatial proximity. It also assigns weights to all detected errors based on the distance between co-occurred records.

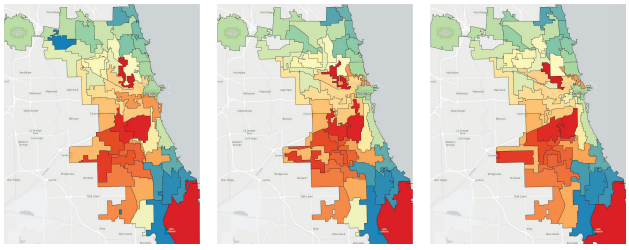
**Spatial Candidate Generator.** The input to this module is the two sets of erroneous and clean cells coming out of the *spatial error detector*. The output is two similar sets of cells, considering the following: (1) There will be more cells in the clean set, as some erroneous cells will be cleaned, (2) Each cell will have a set of weighted candidate values, where Sparcle believes that one of these values in the correct value, (3) Clean cells will be labeled with the value that Sparcle believes it is the correct one.

**Spatial Input Formulator.** This module mainly injects spatial-awareness into existing input formulator modules, where the input is the output of the *spatial candidate generator*, while the output is the output of Sparcle. As these modules are very specific to the host cleaning systems, Sparcle has to have various versions of such module to match its host system. The goal is to score each possible candidate value and prepare the output in a certain format to match the requirements of the host error correction module.

Sparcle **Output: Interaction with Host System.** The output of Sparcle is the output of the *Spatial Input Formulator*, which is the detected erroneous *cells* and their weighted suggested values. If we only have spatial constraints, then the output of Sparcle is the completely corrected input data. When having non-spatial constraints, the output of Sparcle is sent to the *error correction* module of its host data cleaning system. Then, it will be integrated with other suggested values from the non-spatial constraints for *error correction method* to statistically come up with the **final** correct value. As the correction module is very system-specific, the *Spatial Input Formulator* has to customize the output of Sparcle based on its host system.

# 3 CASE 2: TEMPORAL-AWARE DEPENDENCY

Another practical yet unstudied challenge of functional dependencies-based data cleaning is that the groundtruth with respect to a certain dependency may change throughout the dataset. For example, in a dataset [7] that contains building permits issued in Chicago from 2006 to present, there are attributes such as *Ward*, *Census Tract* and *Police District* that are in dependencies with spatial attributes *Latitude* and *Longitude*. And the boundaries of these attributes are changing every few years. Figure 3 shows three versions of ward map during the time span of the dataset. As a result, an area belongs to ward $A$ in 2010 may belong to ward $B$ in 2020.



(a) 2003-2015 (b) 2015-2023 (c) 2023-Present

**Figure 3: Chicago Ward Map**

This characteristic brings extra difficulties for current data cleaning systems. In particular, for a system working with dependency (*Latitude*, *Longitude*) → *Ward*, it will detect both records as erroneous if they have the same location but different wards. However, both entries might be just correct; they have different wards only because the ward map changed. As for error repairing, solving all records together will confuse the system as it only intends to enforce a single version of groundtruth. Hence an entry might be repaired to an out-of-date value, which clearly hurts the accuracy. The closest related works are the ones on *temporal functional dependencies* [1], which restrict the rule on the temporal dimension, for example, person cannot be reported in two countries at the same time (*Person* → *Destination* in a 1-hour window). Since those works mainly focus on time window while our challenge is about time period, the applicability is questionable.

Simply having the temporal attribute on the left-hand side of the dependency (e.g., (***Time***, *Latitude*, *Longitude*) → *Ward*) is not enough to solve the problem. The main reason is again the insufficient co-occurrence as it now needs records also to have the same timestamp to learn the dependency. Relaxing on the temporal attributes, just like how Sparcle relaxes spatial attributes, is a possible solution. This spatio-temporal relaxation requires a spatio-temporal similarity measurement, which is untrivial. Another possible solution is to detect the change of dependency groundtruth directly. In an ideal world, we could split the data by different versions of groundtruth and clean them separately. However, it is often unclear when a new groundtruth become effective. The intuition is that if an area has many records with ward as $A$ and ward as $B$ before and after a certain time $T$, then there is a great chance that a new groundtruth becomes effective at the time $T$. Efficiency is a foreseeable challenge here to identify and verify all the cases.

Last but not least, this temporal awareness is not only limited to spatial boundaries. It is valid in many traditional data cleaning cases. Taking a human resource data for example, an employee may change their telephone or address at some time. In general, there should be no eternal groundtruth; instead, a groundtruth is only valid for a certain period of time.

## 4 FUTURE WORKS

Besides Section 2 and 3, there are more future works along the path of data cleaning with spatial and temporal awareness.

**Learn the parameters and fit the weight function.** The two key ideas in Sparcle (Section 2) introduces two parameters: (1) distance $d$ for *Spatial Neighborhood*, which means we consider the area within distance $d$ to be the neighborhood of a location; and (2) weight function $\mathcal{W}(d)$ for *Distance Weighting*, which means the a closer record will have a higher weight. In Sparcle [15], we use a specified set of distance and weight function for all data after trying many combinations, and we show that the choice significantly affect the cleaning accuracy. As for future, we could learn the distance threshold and fit the weight function from the dataset, which would not only improve the cleaning accuracy per dataset, but also saves the efforts to find the optimal parameters. This should be feasible given the assumption that the majority of a dataset is correct.

**Fix the latitudes and longitudes.** An importance thing in Sparcle [15] is that we assume the *Latitude* and *Longitude* attributes are correct. This assumption is reasonable as the latitudes and longitudes are often directly read from devices and are not meant to be repaired in a functional dependency manner. However, there are cases the locations seem suspicious. For example, if all attributes of a record such as *Borough*, *Zipcode*, *Police Ward*, *Community Neighborhood*, *Census Tract* are detected erroneous, it is reasonable to suspect that maybe the actual incorrectness is in *Latitude* and *Longitude*. Also, the *Latitude* and *Longitude* could be missing and need to be imputed. In these cases, we could make the best guess of the locations (for example, in the intersection of the multiple boundaries) that preserves the dependencies best. Fixing and imputing the locations will benefit downstream applications like spatial analysis.

**Reconstruct the boundary.** In Sparcle [15], what we are trying to do with dependency (*Latitude*, *Longitude*) $\rightarrow$ *Borough* is to figure out the mappings from point location to borough name, such as

$$POINT(-73.960077, 40.784859) \rightarrow Manhattan$$

$$POINT(-73.816996, 40.720750) \rightarrow Queens$$

. However, given the fact that each borough is an area rather than a collection of points, it would be beneficial to reconstruct the boundary, which means to figure out the mapping such as

$$POLYGON((-73.951793\ 40.824084, -73.916165\ 40.811506, ..., -73.951793\ 40.824084)) \rightarrow Manhattan$$

$$POLYGON((-73.853140\ 40.739160, -73.777560\ 40.729019, ..., -73.853140\ 40.739160)) \rightarrow Queens$$

. The benefits are at least twofold: (1) it constructs the knowledge base that can be easily applied to new data. (2) A special consideration of the boundary might improve the cleaning accuracy of the records around the boundary, which are the hardest ones to clean according to our experiments [15].

## REFERENCES


[1] Ziawasch Abedjan, Cuneyt Gurcan Akcora, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. Temporal Rules Discovery for Web Data Cleaning. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 9(4):336–347, 2015.

[2] Prepare data for machine learning - Amazon SageMaker Data Wrangler - Amazon Web Services. https://aws.amazon.com/sagemaker/data-wrangler/.

[3] Sridevi Baskaran, Alexander Keller, Fei Chiang, Lukasz Golab, and Jaroslaw Szlichta. Efficient Discovery of Ontology Functional Dependencies. In *Proceedings of the International Conference on Information and Knowledge Management, CIKM*, 2017.

[4] Philip Bohannon, Michael Flaster, Wenfei Fan, and Rajeev Rastogi. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2005.

[5] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. Relaxed Functional Dependencies - A Survey of Approaches. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 28(1), 2016.

[6] Fei Chiang and Renée J. Miller. A unified model for data and constraint repair. In *Proceedings of the International Conference on Data Engineering, ICDE*, 2011.

[7] Chicago Data Portal. Building Permits. https://data.cityofchicago.org/Buildings/Building-Permits/ydr8-5enu.

[8] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *Proceedings of the International Conference on Data Engineering, ICDE*, 2013.

[9] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. NADEEF: a commodity data cleaning system. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2013.

[10] Wenfei Fan. Dependencies revisited for improving data quality. In *Proceedings of the ACM Symposium on Principles of Database Systems, PODS*, 2008.

[11] Dataprep by Trifacta, Google Cloud. https://cloud.google.com/dataprep.

[12] Congcong Ge, Yunjun Gao, Xiaoye Miao, Bin Yao, and Haobo Wang. A Hybrid Data Cleaning Framework Using Markov Logic Networks. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 34(5):2048–2062, 2022.

[13] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. Cleaning data with Llunatic. *VLDB Journal*, 29(4):867–892, 2020.

[14] Stella Giannakopoulou, Manos Karpathiotakis, and Anastasia Ailamaki. Cleaning Denial Constraint Violations through Relaxation. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2020.

[15] Yuchuan Huang and Mohamed F. Mokbel. Sparcle: Boosting the Accuracy of Data Cleaning Systems through Spatial Awareness. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 17(9):2349 – 2362, 2024.

[16] Data Quality Tools and Solutions, IBM. https://www.ibm.com/data-quality.

[17] Inductiv. https://cs.uwaterloo.ca/news/waterloo-based-ai-start-up-inductiv-acquired-apple.

[18] Zuhair Khayyat, Ihab F. Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. BigDansing: A System for Big Data Cleansing. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2015.

[19] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. Metric Functional Dependencies. In *Proceedings of the International Conference on Data Engineering, ICDE*, 2009.

[20] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing Optimal Repairs for Functional Dependencies. *ACM Transactions on Database Systems, TODS*, 45(1):4:1–4:46, 2020.

[21] Mohammad Mahdavi and Ziawasch Abedjan. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 13(11):1948–1961, 2020.

[22] Data Cleansing - Data Quality Services (DQS) in Microsoft SQL Server. https://learn.microsoft.com/en-us/sql/data-quality-services/data-cleansing?view=sql-server-ver16.

[23] NYC Open Data. Motor Vehicle Collisions - Crashes. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.

[24] Nataliya Prokoshyna, Jaroslaw Szlichta, Fei Chiang, Renée J. Miller, and Divesh Srivastava. Combining Quantitative and Logical Data Cleaning. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 9(4):300–311, 2015.

[25] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 10(11):1190–1201, 2017.

[26] El Kindi Rezig, Mourad Ouzzani, Walid G. Aref, Ahmed K. Elmagarmid, Ahmed R. Mahmood, and Michael Stonebraker. Horizon: Scalable Dependency-driven Data Cleaning. *Proceedings of the International Conference on Very Large Data Bases, PVLDB*, 14(11):2546–2554, 2021.

[27] Shaoxu Song and Lei Chen. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems, TODS*, 36(3):16:1–16:41, 2011.

[28] Tamr. https://www.tamr.com/.

[29] Trifacta. https://www.trifacta.com/.

[30] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. Attention-based Learning for Missing Data Imputation in HoloClean. In *Proceedings of Machine Learning and Systems, MLSys*, 2020.

[31] Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2013.