BClean+: A Bayesian Data Cleaning System with Automated Prior Generation

Ziyan Han
Shenzhen University
hanzy@szu.edu.cn

Jing Zhu
Shenzhen University
zhujing2023@email.szu.edu.cn

Jinbin Huang
Shenzhen University
jbhuang@szu.edu.cn

Rui Mao✉
SICS, Shenzhen University
mao@szu.edu.cn

Jianbin Qin✉
SICS, Shenzhen University
qinjianbin@szu.edu.cn

***Abstract*—We demonstrate BClean+, a Bayesian data cleaning system that unifies automated prior generation, data relationship modeling, and probabilistic inference into an end-to-end cleaning framework. BClean+ (1) uses large language models (LLMs) for zero-shot semantic labeling and then automatically synthesizes format patterns as user constraints, (2) constructs attribute dependency models through Bayesian networks with community detection, (3) performs maximum a posteriori (MAP)-based probabilistic inference with a compensative scoring model, and (4) supports optional synthesis of probabilistic programming language (PPL) code for reuse in PPL-based systems such as PClean. We demonstrate BClean+ for its (a) interactive interface that enables optional refinement of user constraints, Bayesian networks, and PPLs, (b) automation and interpretability, and (c) effectiveness and efficiency on real-world datasets. Extensive experiments show that BClean+ achieves an average F1-score of 0.934 and a 98.6× runtime speedup compared to competitive cleaning systems, while reducing user configuration time from hours to minutes.**

***Index Terms*—data quality, data cleaning, Bayesian networks, probabilistic inference, probabilistic algorithms**

I. Introduction

Data cleaning is essential for downstream applications such as data analysis and machine learning (ML) model development, yet existing solutions detect and repair errors by relying on expert-crafted rules, costly external knowledge, or user-annotated labels, and often struggle to learn reliable representations from noisy data (e.g., [1]). To address these limitations, probabilistic approaches treat dirty data as observations from underlying generative processes and perform probabilistic inference through probabilistic graphical models (PGMs) [2]. Probabilistic programming languages (PPLs) also emerged as powerful tools for specifying and executing probabilistic models [3]. Systems such as HoloClean [4] use probabilistic inference to evaluate denial constraints (DCs), but remain semi-supervised and depend on labeled data; other semi-supervised methods (e.g., Raha [5], Baran [6]) require nontrivial labeling and may propagate errors from detection to repair.

Among probabilistic approaches, Bayesian methods have shown promising results. A typical pipeline consists of Bayesian network (BN) construction and inference, where domain knowledge is needed to encode data distributions and BN structures. However, state-of-the-art systems such as PClean [3] require user-authored PPL programs to specify types, distributions, and noise models. This demands specialized expertise and is both labor-intensive and error-prone.

In this demo paper, we present BClean+ [7], [8], a Bayesian data cleaning system that unifies automated prior generation, data relationship modeling, and probabilistic inference into an end-to-end cleaning framework. Its main novelty consists of:
(1) Automated prior generation. BClean+ integrates a novel framework for automated prior generation, which identifies the semantic types of attributes via LLMs and synthesizes high-quality format patterns as user constraints (UCs), while maintaining a reusable template library that grows with new datasets and serves as a shared resource.
(2) Data relationship modeling. BClean+ constructs a foundational BN to capture attribute dependencies and further applies hierarchical structure discovery to identify higher-level semantic modules, providing an interpretable view to guide BN modification and subsequent automated PPL synthesis.
(3) Bayesian inference with compensatory score. Building on UCs and BN, BClean+ performs cell-level data repair through a probabilistic inference method with a compensatory strategy to select the most probable candidate value for each cell.
(4) Automated PPL synthesis. BClean+ further shows its generalizability by combining automatically learned structural dependencies (to construct data relationship models in PPLs) with heuristic analysis of data characteristics (to select and configure probabilistic distribution models). This enables automated synthesis of complex PPL code and tackles a key usability bottleneck in existing PPL-based cleaning systems.

We present the overview of BClean+ in Section II and the demonstration plan in Section III.

II. An overview of BClean+

A. Preliminaries

Given a structured dataset D with tuples $\{T_1, \ldots, T_n\}$ over attributes $\mathcal{A} = \{A_1, \ldots, A_m\}$ and domains $dom(A_j)$, our goal

✉ corresponding author

is to repair erroneous cells and produce a cleaner version D^*. For each cell $T_i[A_j]$, we estimate candidates $c \in dom(A_j)$ and select the most probable repair c^* to replace $T_i[A_j]$.

Bayesian network. We model attribute dependencies with a BN (N, E, θ) learned from D, where nodes N correspond to attributes, E the directed edges capturing dependencies, and θ the conditional probability tables (CPTs). For a tuple $T = (t_1, \ldots, t_m)$, the joint probability is

$$\mathbf{Pr}[t_1, ..., t_m] = \prod_{A_i \in \mathcal{A}} \mathbf{Pr}[T[A_i] = t_i | T[A_j] = t_j, j \in Ans(i)],$$

where $Ans(i)$ are ancestor nodes of A_i in the BN; if A_i has no ancestors, $\mathbf{Pr}[T[A_i] = t_i | T[A_j] = t_j, j \in Ans(i)]$ is determined by the prior probability of A_i, obtained directly from D.

User constraints. To ensure syntactic validity, we consider three types of UCs: (i) *attribute statistics* such as minimum/maximum lengths or value range, (ii) non-null constraints, (iii) *format constraints* via regular expressions. We denote any such validator as a binary function $UC(\cdot) \in \{0, 1\}$; candidates violating UCs are down-weighted.

B. System Architecture

Fig. 1 presents the architecture of BClean+. Stage 1 (automated prior generation) performs zero-shot type labeling with LLMs and synthesizes format patterns as UCs, maintaining a reusable template library. Stage 2 (data relationship modeling) learns a BN and further applies community detection to identify higher-level semantic module, providing an interpretable view. Their outputs are jointly fed into Stage 3 (MAP-based scoring and inference) to produce the cleaned dataset. Stage 4 (optional) compiles the learned structural dependencies and distribution models (selected from the template library by column characteristics) into PPL code for reuse in PPL-based systems (e.g., PClean). Users may optionally refine UCs, adjust BN edges, and modify synthesized PPLs.

Algorithm. BClean+ performs cell-level repair by iterating over all tuples and attributes in the observed dataset D. For each cell $T_i[A_j]$, it initializes c^* as the original value and then examines all candidate values satisfying UCs. For each candidate, its probability is estimated using the learned BN and a compensatory scoring model; c^* is iteratively updated if a higher probability is obtained. After processing all cells, the collection of final c^* values forms the cleaned dataset D^*.

C. Stage 1: Automated Prior Generation

Semantic type identification. BClean+ first assigns each attribute a semantic label (i.e., type) using the ArcheType framework [9]. Specifically, it (1) uses information-weighted sampling to select a representative subset of values, (2) serializes these samples together with a task instruction and the candidate label set into a structured prompt, (3) queries an LLM (e.g., GPT) to infer the most suitable semantic type for the attribute, and (4) normalizes the raw output to a predefined label set via string or vector similarity, ensuring standardized results.

Semantic-aware pattern synthesis. Based on the identified semantic types, BClean+ automatically synthesizes format

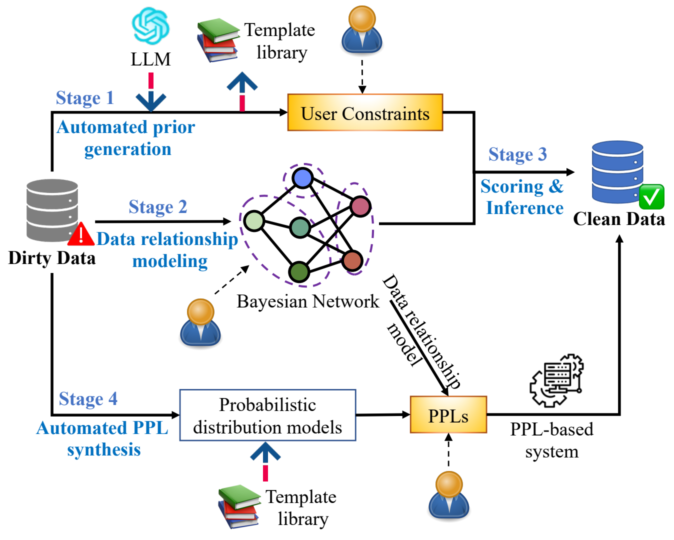


Fig. 1. The architecture of BClean+.

constraints as UCs. For each type, it first collects representative non-empty values from all related attributes, then (i) uses the tdda rexpy module [8] to infer regex-style patterns capturing common formats and (ii) augments them with patterns built from predefined atomic components to handle diversity and noise. After user confirmation, the synthesized patterns are stored and linked to their semantic types in a reusable template library, which grows with new datasets and serves as a shared resource for future cleaning tasks.

D. Stage 2: Data Relationship Modeling

Bayesian network construction. BN construction is NP-hard [10] and difficult to automate, since it requires accurate data or prior knowledge (e.g., via PPL [3]) but real-world data are often noisy and sparse. BClean+ addresses this by extending the FDX method [11] with error tolerance and using graphical lasso [12] to generate an inverse covariance matrix from which BN skeleton is derived.

Instead of hard equalities (an FD $X \to Y$ requires $T_i[X] = T_j[X] \to T_i[Y] = T_j[Y]$), we *soften* FDs by introducing *per-attribute similarities* $Sim \in [0, 1]$ between tuple pairs. For numerics: $Sim(T_i[X], T_j[X]) = 1 - \frac{2 \cdot |T_i[X] - T_j[X]|}{(|T_i[x]| + |T_j[x]|)}$, and for strings: $Sim(T_i[X], T_j[X]) = 1 - \frac{2 \cdot ED(T_i[X], T_j[X])}{len(T_i[X]) + len(T_j[X])}$, where ED denotes unit-cost edit distance and len denotes string length. We compute these pairwise attribute similarities within each tuple and treat them as observations from a multivariate Gaussian distribution. Using graphical lasso, we estimate the covariance matrix Σ of the underlying distribution and derive its inverse covariance matrix $\Theta = \Sigma^{-1}$, which is decomposed as $\Theta = (I - B)\Omega(I - B)^T$. Here, B is the model's autoregression matrix and also the exact BN's adjacency matrix; we retain only edges whose weights exceed a threshold. Given this BN skeleton, we perform parameter learning to estimate the joint distribution and derive the CPTs.

Hierarchical structure discovery. To address the flat view of BN skeletons, we adopt the Infomap algorithm [13] on the BN-derived attribute dependency graph to aggregate attributes (e.g., City, State, and ZipCode) into higher-level semantic modules (e.g., a "location module"). This design provides a more interpretable view, facilitates BN refinement by users, and supports subsequent automated PPL synthesis function.

E. Stage 3: Scoring and Inference

We formulate the data cleaning task as a MAP inference problem to determine the optimal data distribution. For each cell $T_i[A_j]$ in the dataset D, we estimate the most probable candidate c^* by applying Bayes' rule in logarithmic form:

$$c^*= \mathop{\arg\max}_{c \in dom(A_j)} \big(\log \mathbf{Pr}[c|t] + \log \mathbf{Pr}[t] - \log \mathbf{Pr}[t|c]\big) \text{ s.t. } UC(c)=1,$$

where $t = T_i[A_1, \cdots, A_{j-1}, A_{j+1}, \cdots, A_m]$ represents the remaining observed attributes of T_i excluding A_j. The objective consists of two components: $\log \mathbf{Pr}[c|t]$, which can be computed through BN inference via CPTs, and $\log \mathbf{Pr}[t] - \log \mathbf{Pr}[t|c]$, estimated through a compensatory scoring model, i.e., $\mathsf{Score}_{\mathsf{comp}} = \log \mathbf{Pr}[t] - \log \mathbf{Pr}[t|c]$.

We approximate $\mathsf{Score}_{\mathsf{comp}}$ by a confidence-weighted co-occurrence correlation between context t and each candidate c:

$$\mathsf{Score}_{\mathsf{comp}}(c,t) \approx \mathsf{Score}_{\mathsf{corr}}(c,t) = \sum_{A_k \neq A_j \wedge e = t[A_k]} \mathsf{corr}(c, e, A_j, A_k),$$

where $\mathsf{corr}(c, e, A_j, A_k)$ is computed from confidence-weighted co-occurrence counts of (c, e) across D. Tuple confidence is estimated via UC satisfaction: tuples that satisfy more UCs receive higher confidence and contribute positively to the counts, while tuples with more UC violations have lower confidence and carry negative weight (penalization).

As the BN grows with large datasets, global inference becomes prohibitive. We therefore localize inference by restricting interaction to nodes within one hop of the Markov blanket. Furthermore, we introduce two pruning strategies: (i) *tuple pruning* to skip cells unlikely to be erroneous, and (ii) *domain pruning* to remove implausible candidates.

F. Stage 4 (Optional): Automated PPL Synthesis

We extend BClean+ from data repair to *automated PPL synthesis*. The automated prior generation and learned data characteristics are compiled into the reusable template library, enabling PPL-based systems such as PClean to obtain high-quality PPL programs without heavy manual authoring.

A PPL program consists of (i) a *data relationship model* and (ii) a *probabilistic distribution model*, both synthesized automatically: (i) using the constructed BN with community detection, each attribute cluster (semantically coherent group) is translated into a PPL `@class` definition; (ii) BClean+ integrates the reusable library containing candidate distribution models drawn from PClean, and automatically selects them via heuristics on inherent column characteristics (e.g., value distribution, cardinality, character composition), with parameters subsequently fit from sample data.

TABLE I
F1-SCORE (F1), USER TIME (USER), AND EXECUTION TIME (EXEC)

Method	Hospital			Beers			Facilities		
	F1	**user**	**exec**	**F1**	**user**	**exec**	**F1**	**user**	**exec**
BClean+	0.982	<5m	25s	0.951	<5m	57s	0.868	<5m	36m59s
HoloClean	0.626	15h	1m40s	0.047	15h	1m37s	0.759	15h	6m2s
Raha+Baran	0.730	30m	1m46s	0.873	30m	3m2s	0.382	30m	10m55s
PClean	0.962	≥72h	16s	0.028†	≥72h	2m55s	-†	-†	-†

† We invite people familiar with PClean to author data models in PPL.
— No repairs.

TABLE II
PRECISION (P), RECALL (R), AND F1-SCORE (F1) OF PCLEAN-AUTO

Method	Flights			Rents			Soccer			Facilities		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
PClean	0.91	0.89	0.90	0.68	0.69	0.69	0.18†	0.67†	0.29†	—†	—†	—†
PClean-auto	0.91	0.89	0.90	0.78	0.73	0.75	0.78	0.86	0.82	0.90	0.58	0.71

† We invite people familiar with PClean to author the data models in PPL.
— No repairs.

III. DEMONSTRATION OVERVIEW

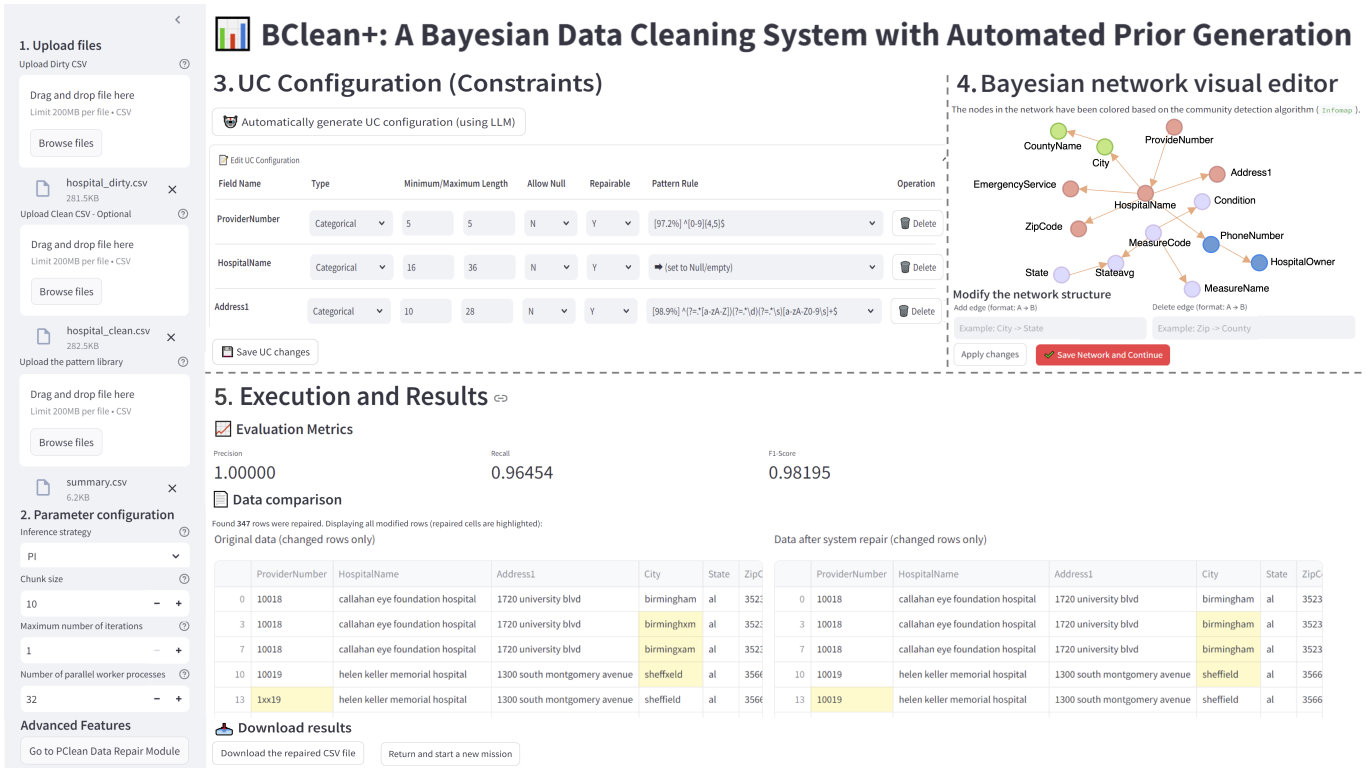
A. Demonstration Setting

Datasets. We will use six real-world datasets (see [7], [8]) to demonstrate the effectiveness and efficiency of BClean+. [1]
Baselines. We will compare BClean+ with HoloClean [4], Raha+Baran [5], [6], PClean [3], and its variant PClean-auto with PPLs automatically synthesized by our system.
Platform. We will conduct the experiments on a single machine with 128GB RAM and 32 Intel(R) Core(TM) Xeon(R) 6326 CPUs at 2.90GHz.

B. Demonstration Plan

(1) BClean+ data cleaning workflow. The demo first walks users through an end-to-end BClean+ pipeline. [2] A snapshot of the system is shown in Fig. 2.
Automated UC generation. Users first upload the dirty dataset (and optionally a clean version only for displaying evaluation metrics and visual comparison of repaired values), and then set the parameters for inference (left panel). BClean+ then automatically infers semantic types and generates UCs, which are populated into an interactive table (top-left). Users can review, edit, delete, or approve the suggested UCs, effectively shifting their role from "pattern creator" to "pattern validator".
BN Construction. The system next automatically constructs a Bayesian network (top-right) and visualizes it with communities (via Infomap), where nodes in the same color belong to the same cluster. This gives users an intuitive, modular view. Participants can refine the BN using the "Add edge" or "Delete edge" operations and confirm the final BN.
Inference and Repair. With the approved UCs and BN, BClean+ performs MAP-based inference and displays the cleaning results (bottom). All repaired cells are highlighted in yellow, enabling side-by-side comparison and validation. The cleaned table can then be exported as a CSV file by clicking the "Download the repaired CSV file" button.
Automated PPL synthesis. We also demonstrate that BClean+ support automatically synthesizing PPL code, which removes

[1]BClean+ is available at https://github.com/DSEG-Group/BCleanplus.git.
[2]A companion video is available at https://youtu.be/oSh5OQXr9zE and https://b23.tv/KPxIbuN.



1. Upload files

Upload Dirty CSV

Drag and drop file here
Limit 200MB per file • CSV
Browse files

hospital_dirty.csv 281.5KB

Upload Clean CSV - Optional

Drag and drop file here
Limit 200MB per file • CSV
Browse files

hospital_clean.csv 282.5KB

Upload the pattern library

Drag and drop file here
Limit 200MB per file • CSV
Browse files

summary.csv 6.2KB

2. Parameter configuration

Inference strategy: PI

Chunk size: 10

Maximum number of iterations: 1

Number of parallel worker processes: 32

Advanced Features

Go to PClean Data Repair Module

BClean+: A Bayesian Data Cleaning System with Automated Prior Generation

3. UC Configuration (Constraints)

Automatically generate UC configuration (using LLM)

Edit UC Configuration

Field Name	Type	Minimum/Maximum Length		Allow Null	Repairable	Pattern Rule	Operation
ProviderNumber	Categorical	5	5	N	Y	[97.2%] ^[0-9]{4,5}$	Delete
HospitalName	Categorical	16	36	N	Y	➡ (set to Null/empty)	Delete
Address1	Categorical	10	28	N	Y	[98.9%] ^(?=.*[a-zA-Z])(?=.*\d)(?=.*\s)[a-zA-Z0-9\s]+$	Delete

Save UC changes

4. Bayesian network visual editor

The nodes in the network have been colored based on the community detection algorithm (`Infomap`).

Modify the network structure

Add edge (format: A → B): Example: City -> State

Delete edge (format: A → B): Example: Zip -> County

Apply changes

Save Network and Continue

5. Execution and Results

Evaluation Metrics

Precision	Recall	F1-Score
1.00000	0.96454	0.98195

Data comparison

Found **347** rows were repaired. Displaying all modified rows (repaired cells are highlighted):

Original data (changed rows only)

	ProviderNumber	HospitalName	Address1	City	State	ZipC
0	10018	callahan eye foundation hospital	1720 university blvd	birmingham	al	3523
3	10018	callahan eye foundation hospital	1720 university blvd	birminghxm	al	3523
7	10018	callahan eye foundation hospital	1720 university blvd	birmingxam	al	3523
10	10019	helen keller memorial hospital	1300 south montgomery avenue	sheffxeld	al	3566
13	1xx19	helen keller memorial hospital	1300 south montgomery avenue	sheffield	al	3566

Data after system repair (changed rows only)

	ProviderNumber	HospitalName	Address1	City	State	ZipC
0	10018	callahan eye foundation hospital	1720 university blvd	birmingham	al	3523
3	10018	callahan eye foundation hospital	1720 university blvd	birmingham	al	3523
7	10018	callahan eye foundation hospital	1720 university blvd	birmingham	al	3523
10	10019	helen keller memorial hospital	1300 south montgomery avenue	sheffield	al	3566
13	10019	helen keller memorial hospital	1300 south montgomery avenue	sheffield	al	3566

Download results

Download the repaired CSV file

Return and start a new mission

Fig. 2. A snapshot of the data cleaning panel of BClean+.

the manual-coding bottleneck in PPL-based systems such as PClean. By clicking the "Go to PClean Data Repair Module" button (left-bottom), users can view and edit the generated PPL program; once finalized, it is executed to run inference and repair, and the cleaned results can be exported.

(2) Performance. Users are invited to interact with BClean+ and see how it outperforms state-of-the-art data cleaning systems in both effectiveness and efficiency (Tables I and II).

Effectiveness. BClean+ consistently outperforms the baselines; it achieves an average F1 of 0.934 and exceeds them by 0.39 on average. Compared with PClean, PClean-auto achieves comparable or better performance, confirming the effectiveness of the automatically synthesized PPLs.

Efficiency. BClean+ achieves a breakthrough in efficiency by reducing user configuration time to $\leq$5m, as opposed to $\geq$72h for PClean and 30m for Raha+Baran, owing to its automated prior generation framework. As a result, it achieves an average speedup of 98.6X in total runtime compared to the baselines.

Acknowledgment

This work was supported by NSFC 62472289, 62532007, 62502325, and Guangdong Province Key Laboratory of Popular High Performance Computers 2017B030314073.

The authors confirm that no main content (text, figures, tables, or code) was generated by AI tools. Minor assistance was limited to grammar checking and writing polish using GPT-5. This statement complies with the ICDE 2026 policy on AI-generated content disclosure.

References

[1] J. Wang and N. Tang, "Towards dependable data repairing with fixing rules," in *SIGMOD*, 2014, pp. 457–468.
[2] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[3] A. K. Lew, M. Agrawal, D. A. Sontag, and V. Mansinghka, "PClean: Bayesian data cleaning at scale with domain-specific probabilistic programming," in *AISTATS*, 2021, pp. 1927–1935.
[4] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "HoloClean: Holistic data repairs with probabilistic inference," *PVLDB*, vol. 10, no. 11, pp. 1190–1201, 2017.
[5] M. Mahdavi, Z. Abedjan, R. Castro Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Raha: A configuration-free error detection system," in *SIGMOD*, 2019, pp. 865–882.
[6] M. Mahdavi and Z. Abedjan, "Baran: Effective error correction via a unified context representation and transfer learning," *PVLDB*, vol. 13, no. 11, pp. 1948–1961, 2020.
[7] J. Qin, S. Huang, Y. Wang, J. Zhu, Y. Zhang, Y. Miao, R. Mao, M. Onizuka, and C. Xiao, "BClean: A bayesian data cleaning system," in *ICDE*, 2024, pp. 3407–3420.
[8] Z. Han, J. Zhu, J. Huang, S. Huang, Y. Wang, R. Mao, and J. Qin, "BClean+: A bayesian data cleaning system with automated prior generation," *TKDE*, 2026.
[9] B. Feuer, Y. Liu, C. Hegde, and J. Freire, "Archetype: A novel framework for open-source column type annotation using large language models," *PVLDB*, vol. 17, no. 9, p. 2279–2292, 2024.
[10] M. Chickering, D. Geiger, and D. Heckerman, "Learning bayesian networks: Search methods and experimental results," in *AISTATS*, 1995.
[11] Y. Zhang, Z. Guo, and T. Rekatsinas, "A statistical perspective on discovering functional dependencies in noisy data," in *SIGMOD*, 2020, pp. 861–876.
[12] S. Wu, S. Sanghavi, and A. G. Dimakis, "Sparse logistic regression learns all discrete pairwise graphical models," in *NeurIPS*, 2019, pp. 8069–8079.
[13] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *PNAS*, vol. 105, pp. 1118–1123, 2007.