LDI: Localized Data Imputation for Text-Rich Tables

Soroush Omidvartehrani
s.omidvartehrani@ualberta.ca
University of Alberta
Edmonton, Canada

Davood Rafiei
drafiei@ualberta.ca
University of Alberta
Edmonton, Canada

ABSTRACT

Missing values are pervasive in real-world tabular data and can significantly impair downstream analysis. Imputing them is especially challenging in text-rich tables, where dependencies are implicit, complex, and dispersed across long textual fields. Recent work has explored using Large Language Models (LLMs) for data imputation, yet existing approaches typically process entire tables or loosely related contexts, which can compromise accuracy, scalability, and explainability. We introduce LDI, a novel framework that leverages LLMs through localized reasoning, selecting a compact, contextually relevant subset of attributes and tuples for each missing value. This targeted selection reduces noise, improves scalability, and provides transparent attribution by revealing the dependency relations that justify each selected attribute and the evidence behind each retrieved tuple. It makes clear not only which data influenced a prediction, but also why it was chosen. Through extensive experiments on real and synthetic datasets, we demonstrate that LDI consistently outperforms state-of-the-art imputation methods, achieving up to 8% higher accuracy with hosted LLMs and even greater gains with small local models. The improved interpretability and robustness also make LDI well-suited for high-stakes data management applications.

VLDB Workshop Reference Format:
Soroush Omidvartehrani and Davood Rafiei. LDI: Localized Data Imputation for Text-Rich Tables. VLDB 2026 Workshop: Tabular Data Analysis (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/soroushomidvar/LDI.

1 INTRODUCTION

Missing values are a common characteristic of many real-world datasets, particularly those derived from open or heterogeneous sources. If left unaddressed, they can significantly undermine the reliability of downstream analysis and machine learning applications. For example, many learning algorithms operate under the assumption that all input data is fully observed [11, 21, 53], and their performance often degrades when faced with incomplete records. This not only reduces predictive accuracy but can also introduce systematic bias into analytical outcomes [16, 35, 42, 51]. As a result, predicting or imputing missing values is a critical component of the broader data wrangling process, often helping to preserve the underlying structure of the data and ensuring the validity of subsequent analysis [39].

Motivation. While the problem of data imputation has been studied for decades, a growing fraction of modern datasets are text-rich tables that mix structured fields with lengthy textual descriptions, reviews, or metadata. These tables are common in domains such as e-commerce, real estate, scientific data management, and open government data. Unlike purely numerical datasets, text-rich tables contain dependencies that are implicit, complex, and often buried within long, noisy text. Identifying which textual cues are informative for imputation remains a major open challenge.

Example. Consider a restaurant table with missing values in the `City` column (see Figure 1). Several other attributes—including `Description`, `Phone`, and `Reviews`—might help infer the correct city, but their relevance varies. The `Description` column may contain location-related hints (e.g., "Modern American dishes" and "Elegant Italian dining") that are typically too general or vague to be useful, while the `Reviews` field may offer more specific but noisy references (e.g., "Best gelato in Vegas"). The `Phone` column, in contrast, encodes reliable geographic signals via area codes, though inconsistencies in formatting can obscure them. Determining which attributes and records truly contribute to accurate imputation requires both contextual understanding and selective reasoning.

Limitations of Existing Work. Existing approaches to data imputation range from simple heuristics to advanced generative models. Heuristic techniques (e.g., mean or mode imputation) are fast but ignore inter-attribute dependencies, often leading to biased estimates [3, 17]. Model-based methods, including regression, expectation-maximization (EM), and matrix factorization, capture statistical dependencies but can be computationally expensive and rely on strong assumptions about the data distribution [24]. Recent deep learning–based approaches, such as autoencoders, GANs, and transformer models, have shown promise in capturing nonlinear patterns, especially in large or multi-modal datasets [50]. Building on this progress, large language models (LLMs) have emerged as powerful tools for data imputation, thanks to their ability to model complex dependencies and generalize across diverse contexts [23, 30, 52]. However, these approaches remain fundamentally global in scope as they tend to process entire tables or large portions thereof. This introduces three major issues that remain largely overlooked: (1) *Scalability*: Tables often exceed model context windows, forcing either truncation or splitting that loses important structure. (2) *Noise sensitivity*: Including loosely related rows and columns dilutes the signal, making imputations less accurate. (3) *Lack of transparency*: When the entire table is fed to an LLM, it becomes unclear which attributes or tuples influenced each prediction, undermining interpretability and user trust.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

	Name	Description	Menu Items	Type	Phone	Reviews	...	City
t_1	Avenue Grill	Modern American dishes ...	Burgers, Steaks, ...	American	212/333-7788	[('Rated 4.0', Amazing steaks ...		New York
t_2	Venetian	Elegant Italian dining ...	Risotto, Gelato, ...	Italian	702/876-4190	[('Rated 4.5', Best gelato in Vegas ...		Las Vegas
t_3	Felix	Cozy spot for authentic ...	Escargots, Duck, ...	French	212/431-0021	[('Rated 5.0', Best duck confit ...		New York
t_4	Tillerman	Nautical-themed seafood ...	Lobster, Crab, ...	Seafood	702/731-4036	[('Rated 5.0', Freshest lobster in ...		Las Vegas
t_5	Stefano's	Family-style Italian eatery ...	Pasta, Pizza, ...	Italian	+1702-385-7111	[('Rated 4.5', Tastes like Italy ...		Las Vegas
t_6	Tse Yang	Upscale Asian cuisine ...	Dim Sum, Peking ...	Asian	212/688-5447	[('Rated 4.5', Can't believe how ...		New York
			⋮					⋮
t_i	Moongate	Traditional Chinese dishes ...	Kung Pao, Fried, ...	Chinese	702-791-7352	[('Rated 4.0', Authentic flavors ...		N/A
			⋮					⋮

Figure 1: Example of data imputation

These limitations raise an important but underexplored question: *Can missing values be accurately imputed using only a localized subset of the table—one that preserves essential context while improving scalability and explainability?*

Our Approach. We hypothesize that, in many cases, missing values can be imputed accurately using a localized context—a compact subset of attributes and tuples most relevant to the missing value. Our approach, LDI, operationalizes this idea by decomposing the problem into two sub-tasks: (1) selecting a subset of columns that are most relevant to the column with missing values, and (2) identifying a subset of rows that provide sufficient contextual evidence for imputation. This localized reasoning enables LDI to reduce noise, scale to larger tables, and trace which data influenced each prediction. Our extensive evaluation across multiple datasets demonstrates that LDI not only improves performance but also enhances interpretability and robustness, particularly in text-rich, noisy, and heterogeneous environments.

Contributions: Our contributions can be summarized as follows: (1) We introduce LDI, a framework that performs attribute- and tuple-level selection to guide LLM-based imputation. (2) By limiting context to relevant subsets, LDI achieves both higher accuracy and transparent attribution. (3) LDI handles textual inconsistencies and heterogeneous formats more effectively than existing models. (4) The framework supports small, local LLMs, enabling efficient and privacy-preserving imputation. (5) Through analytical and experimental evaluations on real-world datasets from various domains, we demonstrate that LDI achieves superior accuracy, scalability, and explainability compared to state-of-the-art methods.

2 RELATED WORK

Our work relates to two lines of research: data imputation, which targets the recovery of missing values, and dependency detection, which isolates the columns most predictive of a given target.

2.1 Data Imputation

Missing data imputation has been extensively studied, leading to a wide range of approaches that vary in complexity, assumptions, and applicability.

Traditional Methods. Early imputation techniques replace missing values using fixed constants, such as zero or the mean, or heuristic strategies like Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB) [12, 20, 22]. While computationally inexpensive, these methods rely on strong assumptions about missingness patterns and data ordering, often resulting in biased imputations in real-world datasets [29].

Statistical Models. More advanced approaches estimate the underlying data distribution to predict missing values. Representative methods include relational dependency networks [27], factor-graph-based models [36, 45], distance likelihood maximization [39], and low-rank matrix completion [6]. Non-linear models, such as tree-based ensembles [7, 41], further improve flexibility. However, these techniques often assume specific data distributions or linear relationships, limiting their robustness in heterogeneous or noisy datasets.

Similarity-based Approaches. Similarity-based methods infer missing values using neighboring tuples with complete information. Classic techniques include k-Nearest Neighbors (KNN) [1] and ensemble variants such as KNNE [10]. Other approaches, such as MIBOS [46], rely on iterative similarity counting, while clustering-based methods perform imputation within clusters formed using similarity measures [32, 48, 53]. Although effective in some settings, these methods depend heavily on predefined similarity functions and struggle with semantic similarity and free-text data.

Generative Models. Recent advances in deep generative models have enabled more expressive imputation techniques. GAIN [50] applies adversarial training, while MIDA [13] and HI-VAE [31] leverage autoencoders and variational inference. More recently, language-model-based approaches such as IPM [28], LakeFill [49], ZeroEC [47], UnIMP [44], and FMW [30] exploit semantic representations to improve imputation quality. Despite their strong performance, these methods generally operate as black boxes and provide limited insight into why a particular value was imputed.

2.2 Dependency Detection

Dependency discovery has long been studied in the context of relational data. Functional dependencies (FDs) capture exact deterministic relationships between attributes [33], while approximate functional dependencies (AFDs) relax this requirement to tolerate noise [19, 26]. FDs and AFDs form the basis of many data cleaning and repair techniques, including conditional FDs [4], scalable FD discovery [34], and regression-based dependency learning [14]. They have also been applied to guide imputation and candidate

selection [2, 5, 18, 38, 40]. However, these approaches are primarily designed for structured attributes and assume consistent formatting and exact or near-exact matches. They are less effective for free-text attributes, partial matches, and semantically related values. In contrast, LDI leverages transformation-based pattern discovery and LLM-based reasoning to capture dependencies that are not expressible using traditional FDs or AFDs.

3 PRELIMINARY

We formalize the problem of missing data imputation and outline the setting for LDI.

Problem Statement. Let $\mathcal{D}$ be a dataset with schema $\mathcal{R}$ consisting of attributes $A_j \in \mathcal{R}$. Each tuple $t_i \in \mathcal{D}$ contains a set of cell values, where a missing entry in the j-th attribute is denoted by $t_i[A_j] = \varnothing$. Let $\hat{\mathcal{D}}$ denote the imputed version of $\mathcal{D}$ such that, for all i, j where $t_i[A_j]$ is missing in $\mathcal{D}$, the corresponding imputed tuple $\hat{t}_i \in \hat{\mathcal{D}}$ satisfies $\hat{t}_i[A_j] \neq \varnothing$. We denote by $\mathcal{D}^*$ the ground truth dataset in which each tuple $t_i^* \in \mathcal{D}^*$ contains the true values $t_i^*[A_j]$ for all attributes $A_j \in \mathcal{R}$.

Goal. Given a dataset $\mathcal{D}$, the objective is to produce an imputed dataset $\hat{\mathcal{D}}$ by predicting missing entries of a target attribute A_T, reconstructing the ground truth dataset $\mathcal{D}^*$. LDI focuses on identifying the most relevant tuples and attributes for each imputation task and leveraging the reasoning capabilities of LLMs to achieve the following: (1) high imputation accuracy in real-world datasets, even when missing values depend on complex and unknown relationships; (2) improved interpretability by tracing which parts of the input influenced each prediction; (3) robustness to noise and formatting inconsistencies without relying on strict assumptions; and (4) minimal dependence on external data or task-specific resources, enabling broader and more efficient applicability.

Assumptions. We assume that (1) some values of A_T (the attribute with missing entries) appear more than once across tuples, allowing the model to learn associations from these repetitions, (2) one or more textual or categorical attributes are correlated with A_T through textual, structural, or contextual cues, and (3) the observed (non-missing) values are sufficiently informative to support imputation. Similar assumptions are commonly made in prior work on data imputation [5, 36, 37, 53].

4 PROPOSED METHOD

While LLMs' strong performance on a range of table reasoning tasks makes them promising candidates for data imputation, they often struggle with scalability to large tables [8, 15] and lack interpretability in their imputation decisions. To address these issues, LDI, selectively identifies the most relevant attributes and tuples, focusing the model's attention and improving explainability. In the first phase (§4.1), LDI retains only the attributes that are likely related to the target attribute A_T and informative for imputation. The second phase (§4.2) filters tuples to produce a relevant yet diverse set of records. Finally, in the third phase (§4.3), the selected attributes and tuples are fed into the LLM, which performs the imputation. An overview of this process is demonstrated in Figure 2.

4.1 Attribute Selection

To identify attributes most relevant to the imputation target, LDI performs attribute selection in two main phases: (1) detecting group-specific patterns in candidate columns, and (2) evaluating approximate dependency relationships. While traditional functional dependencies can model relationships between attributes, they are often too strict for real-world tables that contain noise and long textual values. To better handle such cases, we introduce a relaxed dependency criterion tailored to noisy, text-rich data.

Definition 1 (**(p, q)-Approximate Dependency**). *Let X and A be columns in a table, and let $p, q \in [0, 1]$. We say that a (p, q)-approximate dependency $X \rightarrow A$ holds if, for at least a fraction p of the distinct values α in column A, there exists a common substring s in column X such that:*

- *The probability that s appears in a record given the value $\alpha \in A$ is at least q, i.e., $P(s \in X \mid A = \alpha) \geq q$, and*
- *For all $\alpha' \neq \alpha$, $P(s \in X \mid A = \alpha') < q$.*

This relaxed definition captures partial and group-specific relationships between attributes. It relies on substring-level evidence, making it robust to formatting inconsistencies and local variations. The parameters p and q allow the strictness of the dependency to be tuned, enabling a balance between robustness and sensitivity to weak signals. We use this criterion to identify a subset of candidate attributes A_C that exhibit a (p, q)-approximate dependency with the target attribute A_T with the caveat that such dependencies, like traditional functional dependencies, cannot be definitively inferred from a database instance, and one can only verify that a possible dependency is not violated.

4.1.1 Pattern-Based Dependency Detection. To assess whether A_T is dependent on a candidate attribute A_C, LDI searches for group-specific patterns in A_C. Tuples are grouped by distinct values of A_T, and within each group we extract substrings from A_C using the Longest Common Substring (LCS). Intuitively, the LCS of the values in a group surfaces the substring they share, which serves as a candidate group-specific pattern (e.g., the common area-code prefix of phone numbers within a single city). Substrings appearing in at least a q fraction of tuples in a group are retained. To satisfy dependency constraints, a retained substring must be unique to a single group; substrings frequent in multiple groups are discarded. We also remove redundant substrings that are fully contained in longer ones, simplifying interpretation without affecting correctness. For example, phone numbers associated with Las Vegas frequently share prefixes such as '702/', while New York numbers share '212/' (see Phone column in Figure 1). After filtering cross-group and redundant substrings, only distinctive patterns (e.g., '702/' and '212/') remain. The LCS method is particularly effective in uncovering non-trivial patterns, even in the presence of noises such as typos or formatting inconsistencies. Its robustness stems from two key properties: (1) If a pattern is missing in some cells of a group, the LCS across the remaining cells is preserved. Adjusting the parameter q makes it possible to relax the matching requirement while maintaining the detection of meaningful dependencies. (2) Inconsistent formatting produces shorter but still relevant LCS results. For example, variations such as '+1780 ', '780/', and '780-', still reveal the underlying pattern '780'.

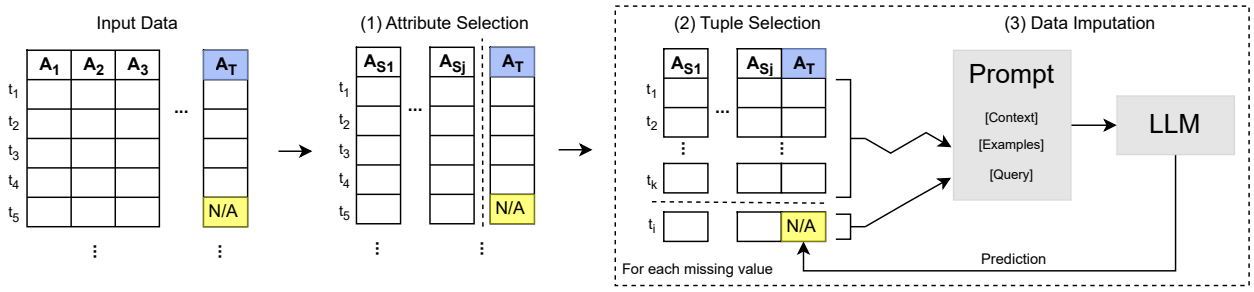


Figure 2: Overview of the LDI framework

4.1.2 Dependency Evaluation. LDI next evaluates whether a (p, q)-approximate dependency holds between A_C and A_T. A dependency is accepted if at least a fraction p of groups contain at least one unique pattern—i.e., a substring that frequently appears within that group but is uncommon across others. This ensures dependencies are supported by consistent group-level evidence rather than isolated or random occurrences. For instance, when predicting `City` from Phone, if unique patterns exist for four out of five cities, the dependency holds for any $p \leq 0.8$. Attributes satisfying this criterion are retained for subsequent phases.

4.1.3 Scalability through Stratified Group Sampling. Although our method can efficiently detect dependencies on large datasets (see §6.5 and A.2), those with hundreds of thousands or millions of records may still require additional measures to ensure scalability. To this end, we introduce an optional *group sampling* step. While LDI can operate on a random sample of tuples, naive random sampling is inadequate as it overrepresents frequent values in A_T and underrepresents rare ones, which can obscure group-level patterns. Since LDI relies on identifying such patterns, it is essential to preserve the diversity of A_T in the sample.

Group Sampling. To maintain diversity in the target attribute A_T, we partition the table by distinct values of A_T and then sample in a structured way. We randomly select m groups that each contain at least n rows, and within each group, we randomly draw n rows, producing a sample of $m \times n$ tuples. When the dataset lacks sufficient tuples for a full $m \times n$ sample, such as when some values of A_T are infrequent, we include as many complete groups as possible, prioritizing those with larger sizes. Specifically, we first select all groups with at least n rows, then add those with $n-1$ rows, continuing until no further groups can be added without exceeding the sample limit. The parameters m (number of groups) and n (tuples per group) control the breadth and depth of the sample. Increasing m covers more distinct A_T values, while increasing n captures finer-grained or noisier within-group patterns. Smaller values of (m, n) may suffice for more homogeneous datasets. This stratified approach preserves group-level diversity while keeping the sample size manageable, enabling LDI to scale effectively without sacrificing the reliability of dependency detection.

4.2 Tuple Selection

In the second phase, a small set of example tuples is selected to guide the LLM during imputation. The goal is to choose tuples that provide strong contextual evidence while remaining representative and avoiding bias. Specifically, selected tuples must: (1) have complete values for the target attribute A_T, (2) be similar to the tuple with the missing value, and (3) cover diverse values of A_T to reduce bias.

We first retain tuples with non-missing values in A_T, which are required as valid examples[1]. Similarity is computed only over the relevant attributes identified in the previous phase, avoiding noise from irrelevant fields. For a tuple t_i with missing A_T, similarity to a complete tuple t_j is measured using the average normalized Longest Common Substring (LCS) over the selected attributes S:

$$\text{similarity}(t_i, t_j) = \frac{1}{|S|} \sum_{A \in S} \frac{|\text{LCS}(t_i[A], t_j[A])|}{\max(|t_i[A]|, |t_j[A]|)} \quad (1)$$

Unlike embedding-based similarity, which can miss fine-grained patterns in long text, LCS captures exact overlapping substrings and preserves key lexical structure, making it effective for identifying closely related tuples. Tuples are ranked by similarity, and the top k tuples with distinct A_T values are selected. This balances relevance with diversity, ensuring that the LLM receives k informative yet non-redundant retrieved examples for imputation.

Suppose the relevant attribute selected in the first phase for our running example is Phone, and we want to impute the missing `City` for tuple t_i corresponding to `Moongate` in Figure 1. We compute the pairwise similarity of t_i with each complete tuple based on their Phone values as follows:

Pair	LCS	Length	Max Length	Similarity
t_i, t_1	`-7`	2	12	0.167
t_i, t_2	`702`	3	12	0.250
t_i, t_3	`02`	2	12	0.167
t_i, t_4	`702`	3	12	0.250
t_i, t_5	`702-`	4	14	0.286
t_i, t_6	`7`	1	12	0.083

Ranking by similarity yields t_5 (0.286; `City` = Las Vegas), followed by t_2 and t_4 (0.250; `City` = Las Vegas), then t_1 and t_3 (0.167; `City` = New York), and finally t_6 (0.083; `City` = New York). When $k = 1$, the most similar tuple t_5 is selected. When $k = 2$, the algorithm chooses t_5 and t_1 to preserve similarity while covering different `City` values. Although t_2 and t_4 are more similar to t_i than t_1, they share the same `City` value as t_5 (Las Vegas); the diversity constraint

[1]Tuples may have missing values in other attributes since full completeness is not required, but they are less likely to be similar to the selected tuple for imputation.

therefore skips them in favor of t_1, the most similar tuple with a different `City` value (New York).

4.3 Data Imputation Using LLM

In the third phase of LDI, we leverage LLMs to impute missing values in a retrieval-augmented setting. The model is provided with a small set of example tuples that include the relevant attributes identified in § 4.1 and retrieved similar tuples from § 4.2. The LLM then infers the missing value by recognizing patterns across these examples. Following prior work [30], each tuple is serialized as a sequence of key–value pairs. The prompt includes a brief task description, example tuples with observed target values, and a query tuple with a missing value. The full prompt template is provided in Appendix B.2. LLMs are well suited to this final step as they leverage pretraining knowledge to ground predictions in real-world facts without requiring external lookup tables (e.g., associating area codes with cities), generalize to patterns not seen in the provided examples, and remain robust to noise and formatting variants (such as '+1-702', '702/', and '702-').

Running Example Recap. Suppose we aim to impute the missing `City` value for the tuple t_i corresponding to `Moongate` in Figure 1. The method first groups the data by `City` and samples m groups with n tuples each (Figure 3). Pattern detection identifies recurring substrings in candidate columns. Here, both `Phone` and `Reviews` show repeated substrings, but only `Phone` satisfies the group-level dependency criteria: area codes are unique to cities, whereas patterns in `Reviews` (e.g., "*[('Rated "*) appear across multiple cities and are discarded. Next, similarity search identifies k tuples most relevant to t_i based on the selected attribute (`Phone`) while ensuring diversity in `City` values. These tuples are serialized into key-value format and provided to the LLM as retrieved context for retrieval-augmented imputation. The model then imputes the missing city by relying solely on informative signals in the `Phone` column, producing predictions that are both accurate and traceable. This workflow demonstrates how LDI reduces noise, focuses on informative data, and leverages LLM reasoning to produce explainable imputations, even without predefined dependencies in the dataset.

5 PERFORMANCE ANALYSIS

The runtime of LDI depends on the size of the dataset $|\mathcal{D}|$, sampled groups m, tuples sampled per group n, candidate attributes a_c, selected attributes a_s, complete tuples c, and average string length ℓ. The following discussion breaks down the runtime of each phase. Details and proofs are provided in Appendix A.

Attribute Selection. For each candidate attribute, LDI extracts group-specific frequent patterns across the sampled groups using generalized suffix trees. This phase runs in $O(|\mathcal{D}|+a_c \cdot m \cdot \ell \cdot (n+r))$, where r denotes the extracted patterns per group. With redundancy filtering enabled, the complexity becomes $O(|\mathcal{D}|+a_c \cdot m \cdot \ell \cdot (n+r^2))$.

Tuple Selection. For each incomplete tuple, LDI computes similarity scores against the c complete tuples using GST-based longest common substrings over the a_s selected attributes and then selects the top-k diverse examples. This results in a cost of $O(a_s \cdot c \cdot \ell + c \log c)$ time per incomplete tuple.

Table 1: Dataset summary

Dataset	Domain	#Rows	#Attr.	Incomplete Attr. (Vocabulary Size)
Buy	Digital Retail	651	4	manufacturer (62)
Restaurant	Food Services	864	5	city (49)
Zomato	Food Services	51717	17	location (93)
Phone	Mobile Devices	413840	6	brand name (385)

Table 2: Average reduction in number of attributes and tokens

Dataset	#Attr. (Except Target)		Less Data
	All Attr. (FMW)	Dep. Attr. (LDI)	(#Tokens)
Buy	3	1	63.8%
Restaurant	4	2	32.7%
Zomato	16	2	96.1%
Phone	5	1	75.5%

LLM-based Imputation. LDI makes one LLM call per missing value. Each prompt contains a short fixed-length context, k examples, and a query over a_s attributes (see Figure 2). The token cost per prompt is $\lambda + k \cdot a_s \cdot \tau$, where λ is the context length and τ is the average token length per attribute.

6 EXPERIMENTS

This section presents a comprehensive evaluation of our proposed data imputation framework, LDI. The experiments aim to answer three main questions: (1) How does attribute and tuple selection affect imputation accuracy and explainability? (2) How does LDI compare to existing state-of-the-art baselines? (3) How robust and scalable is LDI across datasets and missingness conditions?

6.1 Datasets and Experimental Setup

Table 1 summarizes the datasets. They vary in size, number of attributes, and missingness characteristics, and are widely used in data imputation studies [9, 28, 30, 39], providing a diverse testbed. Experiments were conducted on a standard machine with 64 GB RAM and a 12-core CPU. Results are averaged over five runs. LLM-based imputation primarily used GPT-4o-mini, with Llama 3.2 3B used to evaluate performance in limited-resource settings. All remaining details are provided in Appendix B.1.

6.2 Data Reduction via Attribute Selection

Table 2 highlights the efficiency of our model in reducing data size by focusing only on relevant attributes. Unlike other methods that use all available attributes, such as FMW, our approach selects a smaller set of dependent attributes identified in the initial phase. This targeted selection leads to a significant reduction in the amount of data processed, as shown by the lower token counts across all datasets. The reduction in tokens is computed by comparing the average number of tokens of the selected attributes to that of all attributes. The effect is substantial across all benchmarks: in the *Zomato* dataset, token input is reduced by 96.1% through the selection of only 2 out of 16 attributes. Similarly, the *Buy*, *Restaurant*, and *Phone* achieve reductions of 63.8%, 32.7%, and 75.5%, respectively. As a result, the process becomes more explainable and the outcomes more traceable, since it is easier to understand which attributes contribute to the final results.

Name	Description	Menu Items	Type	Phone	Reviews	...	City
Avenue Grill	Modern American dishes ...	Burgers, Steaks, ...	American	212/333-7788	[('Rated 4.0', Amazing steaks ...		New York
Felix	Cozy spot for authentic ...	Escargots, Duck, ...	French	212/431-0021	[('Rated 5.0', Best duck confit ...		New York
Tse Yang	Upscale Asian cuisine ...	Dim Sum, Peking ...	Asian	212/688-5447	[('Rated 4.5', Can't believe how ...		New York
⋮							⋮
Venetian	Elegant Italian dining ...	Risotto, Gelato, ...	Italian	702/876-4190	[('Rated 4.5', Best gelato in Vegas ...		Las Vegas
Tillerman	Nautical-themed seafood ...	Lobster, Crab, ...	Seafood	702/731-4036	[('Rated 5.0', Freshest lobster in ...		Las Vegas
Stefano's	Family-style Italian eatery ...	Pasta, Pizza, ...	Italian	+1702-385-7111	[('Rated 4.5', Tastes like Italy ...		Las Vegas
⋮							⋮

Figure 3: Example of data imputation (after applying our approach)

6.3 Performance Compared to Baselines

We compare LDI with both traditional and LLM-based baselines: KNNE [10] and MIBOS [46] (nearest-neighbor-based), CMI [53] (clustering-based), ERACER [27], Baran [25], and HoloClean [36] (statistics-based), along with two approaches that use language models: IPM [28] and FMW [30]. Baran, originally an error correction model, is adapted for data imputation by replacing each missing value with a null token and predicting its correction. The number of labels (examples) is set to 20, consistent with prior studies [25, 28]. To simulate missing data, we apply a *missing completely at random* mechanism to originally complete datasets, removing 10% of the values. Table 3 reports the accuracy of LDI compared to these baselines across four datasets. Our method achieves the highest accuracy on three datasets (*Buy*, *Restaurant*, and *Phone*) and the second-highest on *Zomato*. LDI, IPM, and FMW all utilize language models and demonstrate high performance across all datasets. This creates a clear gap between them and traditional methods, which may perform reasonably on a single dataset but fail to generalize. For instance, Baran and HoloClean perform well only on *Zomato*, with substantially lower results on the other datasets.

IPM treats imputation as a classification task and introduces two variants: IPM-Multi and IPM-Binary[2]. IPM-Multi formulates the problem as multiclass classification over the candidate set, and IPM-Binary ranks each candidate using a binary classification model. Both require fine-tuning to capture the relation between context and candidate values, making the model resource-intensive and less flexible. Moreover, IPM performance is sensitive to the quality of training data and the chosen modeling strategy. In practice, large and inconsistent gaps are often observed between IPM-Multi and IPM-Binary across datasets. In contrast, LDI avoids the need for training data by dynamically selecting informative attributes and constructing retrieval-augmented prompts using relevant examples. This allows LDI to adapt to new datasets more easily and generate more explainable localized prompts that are better aligned with the LLM's strengths.

FMW performs imputation by prompting a language model with tuples from the dataset, where examples can be randomly or manually selected. All attributes are included in the prompt, regardless of their relevance, which can introduce noise and distract the model. While FMW avoids training, is simple to apply, and allows manual selection of informative examples, this selection does not scale to large datasets[3] and their method does not distinguish between informative and irrelevant attributes. In contrast, LDI narrows the context to relevant tuples and attributes only, reducing prompt noise and enabling imputations that are both more accurate and easier to explain through clearly identifiable supporting signals.

6.4 Robustness to Different Missingness Levels

LDI is designed to maintain high accuracy regardless of the missing value rate, as long as it is provided with a few well-chosen examples. For best results, these examples should be sufficient to detect dependencies and also provide contextual information to frame the imputation task. This behavior is similar to Programming by Examples (PBE) methods, where the model generalizes well from a small, informative set of examples.

While the imputation task is defined over a single target column, additional missing values may exist in other attributes for both the query tuple and the candidate pool. Given that similarity is evaluated on a per-column basis, the presence of missing values diminishes the set of attributes contributing to the similarity computation. This reduction leads to lower similarity scores for such tuples, decreasing their chance of being selected during example retrieval.

To evaluate the robustness of LDI to changes in missing rate, we vary the missing rate from 10% to 50%, with a *missing completely at random* mechanism applied to complete tuples. As shown in Figure 4, even after imputing 50% of the data, the accuracy does not drop significantly, indicating that the model can still identify attribute dependencies and use similar examples to guide the LLM. This robustness is expected to extend to different missingness mechanisms—*missing at random* and *missing not at random*—because the model can uncover attribute dependencies even from limited samples, as reflected in the results, and a small set of informative examples is sufficient to drive accurate imputations.

6.5 Runtime and Scalability

Our results show that LDI is highly efficient and scalable. The attribute selection phase completes in negligible time across all datasets, and the tuple selection phase scales linearly with the number of imputations. Runtime grows moderately with increasing tuple counts, string lengths, and number of selected attributes, but remains practical even for thousands of tuples and long strings.

[2]Under the IPM column, we report the better result of these two for each dataset.

[3]We therefore use random selection for comparison.

Table 3: LDI accuracy compared to baselines at 10% missing rate

Dataset	KNNE	MIBOS	CMI	ERACER	Baran	HoloClean	IPM	FMW (k=10)	LDI (k=10)
Buy	0.355±0.016	0.016±0.011	0.653±0.029	0.003±0.006	0.227±0.173	0.162±0.043	0.965±0.015	0.928±0.033	**0.974±0.026**
Restaurant	0.206±0.019	0.093±0.025	0.560±0.049	0.185±0.031	0.315±0.044	0.331±0.056	0.772±0.023	0.766±0.122	**0.832±0.054**
Zomato	0.581±0.005	0.024±0.001	0.746±0.006	0.001±0.000	0.938±0.023	0.956±0.005	**0.995±0.001**	0.946±0.028	0.974±0.011
Phone	0.466±0.000	0.000±0.000	0.404±0.000	0.175±0.000	0.162±0.073	0.130±0.055	0.867±0.002	0.944±0.031	**0.970±0.016**

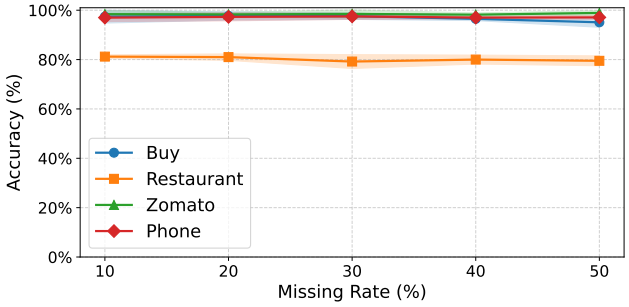


Figure 4: Impact of varying missing rates on accuracy

Table 4 summarizes execution times on real-world datasets. Attribute selection, executed once to identify dependent columns, is consistently fast with slightly higher times for *Zomato* and *Phone* due to their larger number of attributes and longer average string lengths. Tuple selection must be performed for each missing value, and its runtime depends mainly on the number of tuples compared during similarity search as well as the number and length of selected attributes.

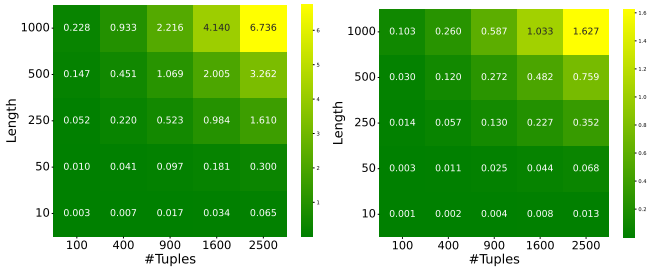
To further assess scalability with respect to input characteristics—the number of groups (m), tuples per group (n), and attribute value length in characters (l)—we conducted controlled experiments on synthetic data. We varied l from 10 to 1000 and both m and n from 10 to 50. As shown in Section 5, these parameters have the largest impact on the runtime of attribute selection. The number of returned substrings was controlled by setting $p = q = 0.5$. The results are shown in Figure 5. As expected, runtime increases with both the number of tuples and the average string length during the attribute selection phase (Figure 5a), but remains practical even for lengthy strings and thousands of tuples. Group sampling can further bound this cost by operating on a small yet representative subset of the data; for instance, sampling only 100 tuples ($m = n = 10$) was sufficient in our experiments to detect reliable patterns. The runtime of the tuple selection phase exhibits a similar trend (Figure 5b), growing moderately as the number and length of tuples increase. Since this phase must be repeated for every missing entry, its cost scales linearly with the number of imputations. When multiple attributes are selected in the first phase, the runtime increases proportionally, as similarity is computed pairwise across attributes. Nevertheless, even in these settings, runtime remains well within practical limits, supporting the scalability of our approach.

6.6 Ablation Study

To analyze the contribution of each phase, we evaluate performance by first using all attributes and randomly selected tuples, then adding attribute and tuple selection step by step. This layered

Table 4: Runtime on real-world datasets (in seconds)

Dataset	Attr. Selection	Tup. Selection
Buy	0.022	0.058
Restaurant	0.012	0.116
Zomato	0.338	0.375
Phone	0.795	0.162



(a) Attr. Selection **(b) Tup. Selection**

Figure 5: Runtime trends on synthetic data (in seconds)

setup shows how each phase improves results. Unlike typical ablation studies that remove one component at a time, we do not remove the first phase entirely, since the second phase depends on its selected attributes, and evaluating similarity over all attributes would obscure the effect of relevant signals due to noisy and redundant fields. We evaluate performance using two metrics: exact match accuracy and ROUGE-1 F1 score. The former measures whether the model prediction is exactly correct, while the latter provides a softer measure of correctness, crediting partial matches. Typically, ROUGE is used to evaluate summarization and generation tasks, and ROUGE-1, specifically, calculates the overlap of unigrams between the prediction and the reference. In our case, using ROUGE-1 F1 helps distinguish between predictions that are completely incorrect and those that contain part of the correct value or provide a more verbose version of it. We report ROUGE-1 instead of higher-order ROUGE-N metrics, as most target values are short (1-2 tokens), making unigram overlap a more reliable measure.

6.6.1 Impact of Attribute Selection Phase. Table 5 reports results for a variant of LDI where tuples are randomly sampled, but only attributes in (p, q)-approximate dependency with the target attribute are retained. This setup isolates the effect of removing irrelevant attributes. Overall, by narrowing the input to the most informative data, the model can better focus on the patterns that truly matter. The results show that keeping only dependent attributes preserves, and in some cases improves, performance, confirming that our attribute selection phase successfully filters out distracting or redundant features. An especially notable improvement appears on the *Zomato* dataset, which contains a larger number of attributes. For example, with $k = 3$, exact match accuracy increases from

Table 5: Impact of attribute selection phase on accuracy with randomly selected tuples for $k = 3$ (and $k = 10$ in parentheses)

Dataset	Exact Match			ROUGE-1 F1		
	All Attr.	Dep. Attr.	Gain	All Attr.	Dep. Attr.	Gain
Buy	0.923 (0.928)	0.923 (0.923)	0.0% (-0.5%)	0.985 (0.985)	0.985 (0.985)	0.0% (0.0%)
Restaurant	0.747 (0.766)	0.761 (0.772)	+1.9% (+0.8%)	0.894 (0.890)	0.885 (0.890)	-1.0% (0.0%)
Zomato	0.846 (0.946)	0.958 (0.964)	+13.2% (+1.9%)	0.866 (0.966)	0.988 (0.988)	+14.1% (+2.3%)
Phone	0.950 (0.944)	0.952 (0.942)	+0.2% (-0.2%)	0.980 (0.978)	0.980 (0.980)	0.0% (+0.2%)

0.846 to 0.958 and ROUGE-1 F1 from 0.866 to 0.988 after removing irrelevant attributes. The gains are more evident when $k = 3$, highlighting the importance of reducing input noise when only a few in-context examples are available. Beyond accuracy, this phase improves traceability and explainability. Limiting the input clarifies which attributes support each prediction. For instance, when imputing a missing city using only a phone number, any correct or incorrect prediction can be directly attributed to the LLM's understanding of phone number patterns. This makes the model's decisions easier to validate and debug.

6.6.2 Impact of Tuple Selection Phase. Table 6 compares two settings: random tuple selection and our proposed diverse similarity-based method. In our approach, similarity is computed explicitly over the attributes identified in phase one, ensuring that selected examples are similar in the most informative dimensions and avoiding irrelevant features that could dilute the closeness of the examples. The results show that our similarity-based tuple selection leads to substantial improvements in exact match accuracy across all datasets and values of k. For example, in the *Buy* dataset with $k = 3$, accuracy increases from 0.923 (random) to 0.974 (diverse similarity). Similar trends are observed in the *Phone* dataset (0.952 to 0.976) and *Zomato* dataset (0.958 to 0.976). Likewise, in the more difficult *Restaurant* dataset, the model achieves better performance, improving from 0.761 to 0.816 at $k = 3$, and from 0.772 to 0.832 at $k = 10$. Interestingly, while the accuracy improvements are often substantial, the changes in ROUGE scores are minimal with only minor differences across datasets. This suggests that the similarity-based selection method helps the model make more precise and fully correct predictions, as opposed to simply improving surface-level similarity. This precision stems from selecting relevant examples that reinforce correct specificity. For instance, when imputing brand names in the *Buy* dataset, random examples may lack an LG product, causing the model to predict the generic "LG". In contrast, our method is more likely to retrieve "LG Electronics", yielding a more accurate output. Similar cases occur with "Sony" vs. "Sony Ericsson" in *Phone*, "HSR" vs. "HSR Layout" in *Zomato*, and "New York" vs. "New York City" in *Restaurant* datasets. Moreover, the tuple selection strategy implicitly leverages redundancy and repetition in the data. Many datasets contain repeated patterns, such as multiple tuples referring to the same location, brand, or product variant. By selecting a diverse yet similar set of examples, our method helps the LLM generalize more effectively from minimal context. This is particularly valuable in retrieval-augmented settings with limited in-context examples, where each example has a strong influence on the model's output. The ablation results confirm that both attribute and tuple selection are critical to the effectiveness of LDI. Phase one improves the model traceability and paves the way for effective tuple selection in the next phase. Phase two then selects diverse and similar examples that help the model better capture subtle variations in the data. Together, these phases guide the model with focused, representative context, leading to improved accuracy and more interpretable behavior [4].

Table 6: Impact of the tuple selection phase on accuracy using only the dependent attributes for $k = 3$ (and $k = 10$ in parentheses)

Dataset	Exact Match			ROUGE-1 F1		
	Rand.	Div. Sim.	Gain	Rand.	Div. Sim.	Gain
Buy	0.923 (0.923)	0.974 (0.974)	+5.5% (+5.5%)	0.985 (0.985)	0.991 (0.991)	+0.6% (+0.6%)
Restaurant	0.761 (0.772)	0.816 (0.832)	+7.2% (+7.8%)	0.885 (0.890)	0.887 (0.889)	+0.2% (-0.1%)
Zomato	0.958 (0.964)	0.976 (0.974)	+1.9% (+1.0%)	0.988 (0.988)	0.983 (0.983)	-0.5% (-0.5%)
Phone	0.952 (0.942)	0.976 (0.970)	+2.5% (+3.0%)	0.980 (0.980)	0.986 (0.980)	+0.6% (0.0%)

6.7 Varying p and q

To assess the robustness of LDI under different noise levels in approximate dependencies, we varied p and q from 0 to 0.75 in steps of 0.25 and evaluated performance over five independent runs per setting. We used the *Zomato* dataset, whose larger number of columns allows clearer observation of how attribute selection behaves under different noise levels, and we exclude settings where p or q exceed 0.75 because the dataset is highly noisy; stricter thresholds (e.g., $p, q = 1$) would remove nearly all dependencies, often leaving no attributes selected. When either parameter is set to 0, no constraint is applied and all attributes are retained. Specifically, $q = 0$ removes the frequency threshold for substring co-occurrence, and $p = 0$ removes the requirement on the number of distinct target values supporting a dependency. Table 7 reports imputation performance when $k = 3$ examples are selected, comparing (a) random selection and (b) diverse similarity-based selection. For both strategies, performance improves once $p, q \geq 0.25$, compared to the unfiltered baseline ($p = q = 0$), and the improvement is more pronounced for random selection, where attribute filtering plays a stronger role. With random selection (Table 7a), the baseline setting ($p = q = 0$) achieves 0.846 exact-match accuracy and 0.883 ROUGE-1, matching FMW results. Increasing both thresholds to 0.25 raises performance to roughly 0.960 accuracy and 0.978 ROUGE-1. Performance remains stable as p and q grow, even when the model uses only two attributes at $p = q = 0.75$, reducing over 96% of the input size while maintaining similar accuracy. Using diverse similarity (Table 7b) further improves performance across all parameter settings. Without attribute filtering ($p = q = 0$), accuracy reaches 0.944 and

[4] We further analyze the zero-shot setting to compare against the LLM's internal knowledge; results are provided in Appendix B.3

Table 7: Effect of varying (p, q) on imputation performance in the Zomato dataset for $k = 3$, with Exact Match accuracy (and ROUGE-1 score in parentheses)

(a) Example selection: Random

q↓ p→	0	0.25	0.50	0.75
0	0.846 (0.883)*	0.846 (0.883)	0.846 (0.883)	0.846 (0.883)
0.25	0.846 (0.883)	0.960 (0.978)	0.960 (0.977)	0.954 (0.974)
0.50	0.846 (0.883)	0.966 (0.982)	0.960 (0.973)	0.960 (0.975)
0.75	0.846 (0.883)	0.954 (0.972)	**0.970 (0.985)**	0.960 (0.980)

(b) Example selection: Diverse similarity

q↓ p→	0	0.25	0.50	0.75
0	0.944 (0.951)	0.944 (0.951)	0.944 (0.951)	0.944 (0.951)
0.25	0.944 (0.951)	0.966 (0.979)	0.968 (0.978)	0.966 (0.977)
0.50	0.944 (0.951)	0.964 (0.976)	0.966 (0.978)	**0.978 (0.990)**
0.75	0.944 (0.951)	0.964 (0.976)	**0.978 (0.986)**	0.976 (0.983)†

* indicates the FMW result, and † indicates the LDI result.

Table 8: Effect of varying (p, q) on the average # selected Attr. in the Zomato dataset

q↓ p→	0	0.25	0.50	0.75
0	16	16	16	16
0.25	16	11.8	10.2	6.8
0.50	16	11.4	7.6	3.4
0.75	16	7.4	5	2

ROUGE-1 0.951, demonstrating the benefit of selecting informative tuples alone. The best results occur at $p = 0.75, q = 0.50$ and $p = 0.50, q = 0.75$, achieving 0.978 accuracy and ROUGE-1 scores of 0.990 and 0.986. However, these configurations select more attributes than the stricter setting $p = q = 0.75$, which uses only two attributes (see Table 8) and achieves nearly identical performance. This highlights the flexibility of our method that tighter thresholds can greatly reduce input size with minimal impact on accuracy.

Table 8 further analyzes the effect of p and q on attribute reduction. When either is 0, all 16 attributes are selected, and as the thresholds increase, the selection becomes more restrictive, decreasing the average number of attributes from 7.6 at $p = q = 0.50$ to 2 at $p = q = 0.75$. This confirms that stricter dependency thresholds successfully remove weak and noisy attributes while preserving the key information needed for imputation.

6.8 Limited-Resource Setting

To assess the effectiveness of LDI in resource-constrained environments, we evaluate its performance using Llama 3.2 3B, a smaller, open-source language model with significantly fewer parameters than GPT-4o-mini. This experiment compares LDI with FMW, which uses all attributes and randomly selected examples for imputation, without any filtering.

As shown in Table 9, LDI consistently outperforms the baseline across all datasets. These results show that LDI's improvements are not dependent on large-scale language models; rather, its design effectively enhances performance through better input selection by providing the model with cleaner, more focused input. When smaller LLMs such as Llama 3.2 3B are used, focusing on localized input becomes especially important as these models have limited capacity to process and reason over lengthy inputs. If irrelevant attributes or uninformative examples are included, they can dilute the clues and distract the model from important patterns.

Table 9: Performance comparison when deployed with a smaller language model (Llama 3.2 3B) for $k = 3$ (and $k = 10$ in parentheses)

Dataset	Exact Match			ROUGE-1 F1		
	FMW	LDI	Gain	FMW	LDI	Gain
Buy	0.918 (0.923)	0.954 (0.944)	+3.9% (+2.3%)	0.954 (0.959)	0.980 (0.973)	+2.7% (+1.5%)
Restaurant	0.701 (0.738)	0.775 (0.770)	+10.6% (+4.3%)	0.802 (0.833)	0.837 (0.839)	+4.4% (+0.7%)
Zomato	0.766 (0.128)	0.894 (0.252)	+16.7% (+96.9%)	0.826 (0.286)	0.928 (0.404)	+12.3% (+41.3%)
Phone	0.910 (0.854)	0.966 (0.964)	+6.2% (+12.9%)	0.941 (0.890)	0.977 (0.977)	+3.8% (+9.8%)

A particularly interesting case is the *Zomato* dataset with $k = 10$, where there is a large drop in performance for both LDI and FMW. This dataset contains many lengthy attributes, and including 10 examples results in a large input prompt. In such cases, the model struggles to extract the patterns. Interestingly, using only 3 examples yields much better performance, suggesting that more examples do not always lead to better results when the context is noisy or overloaded. Even in this challenging setting, LDI performs considerably better than the baseline (0.252 vs. 0.128 in accuracy, and 0.404 vs. 0.286 in ROUGE F1), demonstrating its advantage in isolating informative content. These findings illustrate that guiding the model with concise, high-quality input, rather than simply more input, is key to achieving high performance.

7 CONCLUSION AND FUTURE WORK

We proposed a localized imputation method that improves both accuracy and explainability of LLM-based data imputation by selecting informative attributes and representative tuples for each missing value. This focused approach reduces noise, handles inconsistencies, and guides the LLM toward more accurate predictions. Experiments on four real-world datasets show state-of-the-art performance with better explainability than existing methods. Future work includes developing more advanced selection strategies where dependencies are conditional or involve more complex patterns, exploring transfer learning across domains, and providing richer explanations of model reasoning.

ACKNOWLEDGMENTS

We sincerely thank the reviewers for their comments and suggestions, which improved the quality of this work. This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). Soroush Omidvartehrani was supported by the Alberta Innovates Graduate Student Scholarship (AIGSS).

REFERENCES

[1] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[2] Yael Amsterdamer, Susan B Davidson, Tova Milo, Kathy Razmadze, and Amit Somech. 2023. Selecting sub-tables for data exploration. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2496–2509.

[3] Amanda N Baraldi and Craig K Enders. 2010. An introduction to modern missing data analyses. *Journal of school psychology* 48, 1 (2010), 5–37.

[4] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2006. Conditional functional dependencies for data cleaning. In *2007 IEEE 23rd international conference on data engineering*. IEEE, 746–755.

[5] Bernardo Breve, Loredana Caruccio, Vincenzo Deufemia, Giuseppe Polese, et al. 2022. RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies.. In *EDBT*. 1–52.

[6] Emmanuel Candes and Benjamin Recht. 2012. Exact matrix completion via convex optimization. *Commun. ACM* 55, 6 (2012), 111–119.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[8] Wenhu Chen. 2023. Large Language Models are few (1)-shot Table Reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*. 1120–1130.

[9] Z Chen, L Cao, S Madden, T Kraska, Z Shang, J Fan, N Tang, Z Gu, C Liu, and M Cafarella. 2024. SEED: Domain-Specific Data Curation With Large Language Models. arXiv 2023. *arXiv preprint arXiv:2310.00749* (2024).

[10] Carlotta Domeniconi and Bojun Yan. 2004. Nearest neighbor ensemble. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 1. IEEE, 228–231.

[11] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. 2021. A survey on missing data in machine learning. *Journal of Big data* 8 (2021), 1–37.

[12] Jean Mundahl Engels and Paula Diehr. 2003. Imputation of missing longitudinal data: a comparison of methods. *Journal of clinical epidemiology* 56, 10 (2003), 968–976.

[13] Lovedeep Gondara and Ke Wang. 2018. Mida: Multiple imputation using denoising autoencoders. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*. Springer, 260–272.

[14] Zhihan Guo and Theodoros Rekatsinas. 2019. Learning functional dependencies with sparse regression. *arXiv preprint arXiv:1905.01425* (2019).

[15] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys* 55, 12 (2023), 1–38.

[16] Hyun Kang. 2013. The prevention and handling of the missing data. *Korean journal of anesthesiology* 64, 5 (2013), 402–406.

[17] Shahidul Islam Khan and Abu Sayed Md Latiful Hoque. 2020. SICE: an improved missing data imputation technique. *Journal of big Data* 7, 1 (2020), 37.

[18] Ronald S King and James J Legendre. 2003. Discovery of functional and approximate functional dependencies in relational databases. *Journal of Applied Mathematics and Decision Sciences* 7, 1 (2003), 49–59.

[19] Jyrki Kivinen and Heikki Mannila. 1995. Approximate inference of functional dependencies from relations. *Theoretical Computer Science* 149, 1 (1995), 129–149.

[20] Verena Klamroth-Marganska, Javier Blanco, Katrin Campen, Armin Curt, Volker Dietz, Thierry Ettlin, Morena Felder, Bernd Fellinghauer, Marco Guidali, Anja Kollmar, et al. 2014. Three-dimensional, task-specific robot therapy of the arm after stroke: a multicentre, parallel-group randomised trial. *The Lancet Neurology* 13, 2 (2014), 159–166.

[21] Arun Kumar, Matthias Boehm, and Jun Yang. 2017. Data management in machine learning: Challenges, techniques, and systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1717–1722.

[22] John M Lachin. 2016. Fallacies of last observation carried forward analyses. *Clinical trials* 13, 2 (2016), 161–168.

[23] Xue Li and Till Döhmen. 2024. Towards efficient data wrangling with llms using code generation. In *Proceedings of the Eighth Workshop on Data Management for End-to-End Machine Learning*. 62–66.

[24] Roderick JA Little and Donald B Rubin. 2019. *Statistical analysis with missing data*. John Wiley & Sons.

[25] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[26] Panagiotis Mandros, Mario Boley, and Jilles Vreeken. 2017. Discovering reliable approximate functional dependencies. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 355–363.

[27] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: a database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 75–86.

[28] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing semantics for imputation with pre-trained language models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 61–72.

[29] Frank J Molnar, Brian Hutton, and Dean Fergusson. 2008. Does analysis using "last observation carried forward" introduce bias in dementia research? *Cmaj* 179, 8 (2008), 751–753.

[30] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment* 16, 4 (2022), 738–746.

[31] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* 107 (2020), 107501.

[32] Sanaz Nikfalazar, Chung-Hsing Yeh, Susan Bedingfield, and Hadi A Khorshidi. 2017. A new iterative fuzzy clustering algorithm for multiple imputation of missing data. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 1–6.

[33] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.

[34] Thorsten Papenbrock and Felix Naumann. 2016. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*. 821–833.

[35] Yongsong Qin, Shichao Zhang, Xiaofeng Zhu, Jilian Zhang, and Chengqi Zhang. 2007. Semi-parametric optimization for missing data imputation. *Applied Intelligence* 27, 1 (2007), 79–88.

[36] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proceedings of the VLDB Endowment* 10, 11 (2017).

[37] El Kindi Rezig, Mourad Ouzzani, Walid G Aref, Ahmed K Elmagarmid, Ahmed R Mahmood, and Michael Stonebraker. 2021. Horizon: Scalable dependency-driven data cleaning. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2546–2554.

[38] Roee Shraga and Renée J Miller. 2023. Explaining dataset changes for semantic data versioning with explain-da-v. *Proceedings of the VLDB Endowment* 16, 6 (2023).

[39] Shaoxu Song and Yu Sun. 2020. Imputing various incomplete attributes via distance likelihood maximization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 535–545.

[40] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2020. Enriching Data Imputation under Similarity Rule Constraints. *IEEE Transactions on Knowledge and Data Engineering* 32, 2 (2020), 275–287.

[41] Daniel J Stekhoven and Peter Bühlmann. 2012. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (2012), 112–118.

[42] Julia Stoyanovich, Bill Howe, and Hosagrahar Visvesvaraya Jagadish. 2020. Responsible data management. *Proceedings of the VLDB Endowment* 13, 12 (2020).

[43] Esko Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica* 14, 3 (1995), 249–260.

[44] Jianwei Wang, Kai Wang, Ying Zhang, Wenjie Zhang, Xiwei Xu, and Xuemin Lin. 2025. On llm-enhanced mixed-type data imputation with high-order message passing. *arXiv preprint arXiv:2501.02191* (2025).

[45] Richard Wu, Aoqian Zhang, Ihab Ilyas, and Theodoros Rekatsinas. 2020. Attention-based learning for missing data imputation in holoclean. *Proceedings of Machine Learning and Systems* 2 (2020), 307–325.

[46] Sen Wu, Xiaodong Feng, Yushan Han, and Qiang Wang. 2012. Missing categorical data imputation approach based on similarity. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2827–2832.

[47] Yangyang Wu, Chen Yang, Mengying Zhu, Xiaoye Miao, Wei Ni, Meng Xi, Xinkui Zhao, and Jianwei Yin. 2025. A Zero-Training Error Correction System with Large Language Models. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 2949–2962.

[48] Xiaobo Yan, Weiqing Xiong, Liang Hu, Feng Wang, and Kuo Zhao. 2015. Missing value imputation based on gaussian mixture model for the internet of things. *Mathematical Problems in Engineering* 2015, 1 (2015), 548605.

[49] Chenyu Yang, Yuyu Luo, Chuanxuan Cui, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Data Imputation with Limited Data Redundancy Using Data Lakes. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3354–3367.

[50] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*. PMLR, 5689–5698.

[51] Chengqi Zhang, Xiaofeng Zhu, Jilian Zhang, Yongsong Qin, and Shichao Zhang. 2007. GBKII: An imputation method for missing values. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 1080–1087.

[52] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024. Jellyfish: Instruction-tuning local large language models for data preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 8754–8782.

[53] Shichao Zhang, Jilian Zhang, Xiaofeng Zhu, Yongsong Qin, and Chengqi Zhang. 2008. Missing value imputation based on data clustering. In *Transactions on computational science I*. Springer, 128–138.

A DETAILED PERFORMANCE ANALYSIS

This appendix provides the detailed time complexity analysis of the main components of LDI, including supporting lemmas and step-by-step derivations.

A.1 Supporting Lemmas

We first present two lemmas that are used in the analysis of both the attribute selection and tuple selection phases.

Lemma 1 (Frequent Substring Detection). *Given a collection of n strings, each of length ℓ, all substrings that appear in at least q fraction of the strings can be found in time $O(n \cdot \ell + r)$ and space $O(n \cdot \ell)$, where r is the number of substrings reported.*

Proof. We build a Generalized Suffix Tree (GST) for the n input strings, each of length ℓ, after appending a unique terminal symbol to each string. This tree can be constructed in $O(n \cdot \ell)$ time and space using Ukkonen's algorithm [43]. Each leaf is labeled with its corresponding string ID, and a post-order traversal is used to annotate each internal node with the set of string IDs in its subtree. For each node, if the number of distinct string IDs is at least $\lceil qn \rceil$, we report the substring represented by that node. Reporting all such substrings takes $O(r)$ time, where r is the number of substrings found. Therefore, the total time and space complexity is $O(n \cdot \ell + r)$ and $O(n \cdot \ell)$, respectively. □

Lemma 2 (Uniqueness Across Groups). *Given a collection of s strings, each of length ℓ, partitioned into m groups, the time to determine whether each string appears uniquely within a single group is $O(s \cdot \ell)$.*

Proof. To determine whether each string appears in only one group, we use a hashmap that associates each string with the set of group identifiers in which it appears. We iterate over all s strings and, for each one, insert it into the hashmap and record its group identifier. Since each string has length ℓ, inserting or comparing a string in the hashmap takes $O(\ell)$ time. Processing all s strings therefore takes $O(s \cdot \ell)$ time. After building the hashmap, we scan its entries to check whether any string is associated with more than one group. As there are at most s unique strings, this step takes $O(s)$ time. Hence, the overall time complexity is $O(s \cdot \ell)$. □

A.2 Attribute Selection

The attribute selection phase depends on the number of candidate attributes a_c, the number of groups m induced by the target attribute, the number of sampled tuples per group n, the average string length ℓ, and the number of extracted substrings r per group.

Step 1: Group Sampling. Tuples are grouped by their values in the target attribute and a subset of m groups is sampled, with n tuples selected per group. This requires a single pass over the dataset and takes $O(|\mathcal{D}|)$ time.

Step 2: Pattern Detection. For each candidate attribute and each group, we detect substrings that appear in at least a q fraction of the sampled tuples using a generalized suffix tree. By Lemma 1, this step takes $O(n \cdot \ell + r)$ time per group. Across all m groups and a_c candidate attributes, the total cost is

$$O(a_c \cdot m \cdot (n \cdot \ell + r)).$$

Step 3: Uniqueness Checking. All substrings extracted from different groups are checked for uniqueness across groups. Letting $s = m \cdot r$ be the total number of substrings per attribute, Lemma 2 implies a cost of $O(s \cdot \ell)$ per attribute. Across all candidate attributes, this results in a total time of

$$O(a_c \cdot m \cdot r \cdot \ell).$$

Step 4: Optional Redundancy Filtering. Optionally, substrings that are fully contained within longer ones are removed within each group to improve interpretability. This requires pairwise comparisons among the r substrings in a group, yielding a cost of $O(r^2 \cdot \ell)$ per group. Across all groups and candidate attributes, the total cost is

$$O(a_c \cdot m \cdot r^2 \cdot \ell).$$

Overall Complexity. Combining all steps, the total time complexity of the attribute selection phase is:

$$O(|\mathcal{D}| + a_c \cdot m \cdot \ell \cdot (n + r))$$

when redundancy filtering is disabled, and

$$O\Big(|\mathcal{D}| + a_c \cdot m \cdot \ell \cdot (n + r^2)\Big)$$

when redundancy filtering is enabled.

A.3 Tuple Selection

To impute a missing value in a tuple t_i, LDI selects k similar and diverse complete tuples. Let c be the number of complete tuples (i.e., tuples with known values in A_T) and a_s the number of selected attributes.

For each selected attribute, we build a generalized suffix tree over the attribute values of t_i and all c complete tuples. By Lemma 1, constructing the GST takes $O(c \cdot \ell)$ time per attribute. After the tree is built, the longest common substring between t_i and any complete tuple can be computed in $O(\ell)$ time.

Thus, computing similarity scores between t_i and all c complete tuples across a_s attributes takes $O(a_s \cdot c \cdot \ell)$ time. The c tuples are then sorted by similarity in $O(c \log c)$ time, and a final scan ensures diversity among the top-k selected tuples. Therefore, the total time complexity of tuple selection for one incomplete tuple is

$$O(a_s \cdot c \cdot \ell + c \log c).$$

B IMPLEMENTATION AND EXPERIMENT DETAILS

B.1 Datasets and Evaluation Setup

We evaluate LDI on four real-world datasets from three domains: Buy, Restaurant, Zomato, and Phone. The incomplete attributes are categorical, with vocabulary sizes ranging from 49 to 385. Buy and Restaurant have fewer attributes but are standard benchmarks [9, 28, 30, 39], while Zomato and Phone have more attributes, enabling a comprehensive evaluation of attribute selection and scalability. Experiments for large datasets were limited to 1,000 rows to manage LLM calls.

Experiments were conducted on Ubuntu 20.04 with an AMD Ryzen 9 3900X (12-core) and 64 GB RAM, using Python 3.9.20. Group sampling was configured with $m = 10$ groups and $n = 10$ samples per group, with the maximum available used for smaller

Table 10: Zero-shot vs. LDI (k=10)

Dataset	Zero-shot	LDI
Buy	0.909 (0.951)	0.974 (0.991)
Restaurant	0.794 (0.858)	0.832 (0.889)
Zomato	0.146 (0.657)	0.974 (0.983)
Phone	0.945 (0.961)	0.970 (0.980)

datasets. Inner threshold q and outer threshold p were set to 0.9 for Buy and Phone, and 0.8 for Zomato and Restaurant, allowing minor noise while maintaining strong evidence for dependencies [5]. We report results with $k = 3$ and $k = 10$ in-context examples, matching the default settings of FMW for fair comparison, and baseline hyperparameters follow the ranges reported in the original papers. LLM-based imputation primarily used GPT-4o-mini (version 2024-07-18), with Llama 3.2 3B included to evaluate performance under constrained resources. GPT-4o-mini provides higher accuracy, while Llama 3.2 3B shows LDI's portability and generalizability.

B.2 LLM Prompt Template

Each input is serialized as key–value pairs and organized into three parts: a task description, example tuples, and a query tuple with a missing target value.

```
[Context]
  A brief description of the imputation task.
[Examples]
  Example 1:
  Attr. A: Val. A1, Attr. B: Val. B1, ...
  Target Attr.: Target Val. 1
  Example 2:
  Attr. A: Val. A2, Attr. B: Val. B2, ...
  Target Attr.: Target Val. 2
  ...
[Query]
  Attr. A: Val. Ax, Attr. B: Val. Bx, ...
  Target Attr.: ?
```

B.3 Zero-shot Baseline

To better understand the contribution of LDI beyond the inherent knowledge of LLMs, we evaluate a zero-shot baseline in which the model is queried without any retrieved examples (i.e., $k = 0$) or attribute selection. In this setting, the LLM relies solely on its pretrained knowledge to infer missing values.

Table 10 compares the exact match accuracy (and ROUGE-1 score in parentheses) of this baseline with LDI using $k = 10$ examples. While zero-shot prompting achieves competitive performance on some datasets, its behavior is inconsistent and often lacks precision. In particular, without access to relevant examples, the model tends to generate plausible but less specific predictions, as it cannot ground its output in dataset-specific patterns or variations. This effect is consistent with the observations in Section 6.6.2: providing relevant and similar examples enables the model to produce more precise and fully correct predictions, rather than approximate or generic ones. By selecting examples that reflect the underlying data distribution, LDI reinforces correct specificity and reduces ambiguity in the generated outputs. This difference is especially pronounced in the Zomato dataset, where zero-shot performance is substantially lower. In such cases, accurate imputation depends on subtle, dataset-specific signals that are not captured by general knowledge alone. By contrast, LDI retrieves informative examples that expose these patterns, enabling the model to make accurate and contextually grounded predictions. Overall, these results show that the gains of LDI stem not only from leveraging LLM capabilities, but from guiding the model with relevant, localized examples that improve precision and reliability.

[5] Section 6.7 reports a sensitivity analysis showing that LDI remains robust across a wide range of p and q values.