applied sciences

MDPI

Article

Data Imputation Based on Retrieval-Augmented Generation

Xiaojun Shi, Jiacheng Wang, Gregorius Justin Chung, Derick Julian and Lianpeng Qiao *

School of Computer Science & Technolohy, Beijing Institute of Technology, Beijing 100081, China; xjshi0124@126.com (X.S.); wangjc@bit.edu.cn (J.W.); gregoriusjustin@bit.edu.cn (G.J.C.); derickjulian@bit.edu.cn (D.J.)
* Correspondence: qiaolp@bit.edu.cn

Abstract

Modern organizations collect increasing volumes of data to drive decision-making, often stored in centralized repositories such as data lakes, which consist of diverse structured and unstructured datasets. However, these repositories often suffer from issues such as incomplete, inconsistent, and low-quality data, which hinder data-driven insights. Existing methods for data imputation, including statistical techniques and machine learning approaches, often rely heavily on large amounts of labeled data and domain-specific knowledge, making them labor-intensive and limited in handling semantic heterogeneity across data formats. To address these challenges, this study proposes a novel retrieval-augmented generation (RAG) framework for data imputation that effectively combines the strengths of retrieval mechanisms and large language models (LLMs). This approach constructs semantic-based indexes for heterogeneous data, employs a recall and re-ranking strategy to enhance retrieval relevance, and proposes a method to reduce inference cost while ensuring imputation quality. Extensive experiments on real-world datasets demonstrate that the proposed framework significantly outperforms machine learning and deep learning approaches.

Keywords: retrieval-augmented generation; data imputation; data mining

check for updates

Academic Editor: Keun Ho Ryu

Received: 6 April 2025
Revised: 4 June 2025
Accepted: 11 June 2025
Published: 30 June 2025

Citation: Shi, X.; Wang, J.; Chung, G.J.; Julian, D.; Qiao, L. Data Imputation Based on Retrieval-Augmented Generation. *Appl. Sci.* **2025**, *15*, 7371. https://doi.org/10.3390/app15137371

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid advancement of Big Data technologies, leveraging data to drive economic growth [1,2], enhance government service levels [3], strengthen regulatory capabilities [4], and improve social governance [5] has become a prominent trend. Among the various types of data, relational data is widely used in practical applications, such as e-commerce recommendation systems [6,7], medical diagnosis [8–11], stock prediction [12], electricity consumption forecasting [13], and traffic prediction [14], among others. However, the promise of these applications of relational data is often hindered by incomplete or missing entries, which remain a critical bottleneck in realizing Big Data's full potential—a challenge that has persisted in database research for over three decades. Consequently, the task of data imputation, which aims to address missing or incomplete data values to establish a unified, high-quality data view, is of paramount importance. Nowadays, multi-source relational data may originate from sources such as the World Wide Web, various data lakes, and databases. These datasets frequently contain a significant number of missing values, exacerbating the problem of incomplete data. Therefore, applying effective imputation algorithms to enhance the quality of raw relational data is crucial for ensuring more accurate and reliable data analysis. However, existing data imputation methods struggle with missing, inconsistent, and low-quality inputs [15]; in this work, we address

this gap by proposing a retrieval-augmented generation (RAG) framework tailored for robust data imputation.

The importance and challenges of relational data imputation have been widely recognized by the international academic community. Over the past several decades, the field of relational data imputation has gone through several key development stages. In the early stages, research focused on utilizing statistical methods [16–20] to handle missing relational data. The earlier statistical imputation method involves using statistics (e.g., mean, median, or mode values) or the most similar among training data to replace missing components, such as mean imputation [17], hot deck imputation (HDI) [20], cold deck imputation (CDI) [19], and K nearest neighbor imputation (KNNI) [18]. Subsequently, to address the challenges posed by the heterogeneity of multisource data, researchers introduced techniques such as deep learning pretraining [21,22], concentrating on domain-level data imputation tasks. However, the unique structure of relational data makes its semantic information more difficult to capture and utilize. Specifically, different tables within the same databases or different columns within the same table may involve different domains [23,24], leading to significant differences in embedded semantics. Currently, deep learning techniques still exhibit two main shortcomings in relational data imputation: (1) they struggles to effectively capture and process complex semantics in cross-domain relational data [25], resulting in reduced quality of data imputation [26]; (2) model training requires large-scale data, leading to expensive manual labeling costs (for instance, on Amazon's crowdsourcing platform, it costs approximately USD 0.1 to impute a table with missing values) [27]. In summary, traditional data imputation strategies face the primary issues of low quality and high costs.

In recent years, large language models (LLMs) have undergone rapid advancements. Leveraging extensive parameter scales and complex model architectures, LLMs have acquired broad knowledge from vast Internet datasets [28], demonstrating strong capabilities in natural language understanding and reasoning. Moreover, compared to human labor, LLMs offer relatively low inference costs (for instance, the inference cost of GPT-4 is only USD 0.03 per thousand tokens). Additionally, pre-trained LLMs typically require only a small amount of extra labeled data or prompts to perform well across multiple tasks, significantly reducing the effort and cost of training models. Therefore, utilizing LLMs to tackle data imputation tasks—while simultaneously ensuring high-quality predictions and keeping monetary costs manageable—has become a vibrant area of research. Recent work has explored prompt-based strategies that guide LLMs to infer and fill missing values in tabular datasets [29]. However, to fully realize the dual goals of improving imputation accuracy and minimizing cost, the following key challenges remain to be addressed:

1. **How to equip LLMs with Cross-Domain Knowledge.** Although LLMs have been trained on vast corpora and have absorbed a wealth of domain knowledge, the complexity and semantic heterogeneity of relational data still pose challenges for understanding and reasoning across domains. Therefore, it is crucial to study how to better utilize LLMs to solve the problem of cross-domain knowledge understanding and reasoning, which will help improve the quality of data imputation using LLMs.

2. **How to Enhance the Performance of LLMs in Data Imputation Tasks by Optimizing Prompt Strategies.** Prompt design has a major impact on LLM performance in imputation tasks. In zero-shot settings, the model only sees a task description. This minimal context makes it hard to infer missing values [30,31]. Few-shot prompting adds a small set of labeled examples. These examples help the model learn patterns and improve accuracy. However, results can vary greatly depending on which examples are chosen [32,33]. Iterative prompting methods, such as chain-of-thought or tree-of-thought methods, break imputation into smaller reasoning steps. This gives the model richer, more structured

context [34,35]. While prompt engineering can enable LLMs to tackle data imputation tasks simply by constructing appropriate inputs, each prompt style still limits the total accessible information. Therefore, optimizing prompt strategies to enhance the performance of LLMs is critically important.

3. **How to Balance Imputation Quality and LLM Inference Cost** Prompt detail often drives LLM accuracy. Richer prompts or retrieval-augmented inputs supply more context and reduce errors, yet longer prompts directly raise inference costs. When imputing large datasets, even small cost increases multiply rapidly. Frequent calls to the LLM become expensive in terms of both time and money. Moreover, excessive context can slow down inference and complicate prompt design. Thus, a core challenge is finding the sweet spot, i.e., supplying enough information to ensure high-quality imputation while keeping token usage and latency within practical limits.

To address the aforementioned issues, we propose the following approach:

Semantic Indexing of Data Lakes for Understanding of Cross-Domain Knowledge.

Data lakes contain vast, diverse knowledge in structured tables and long, unstructured texts. To make retrieval both fast and precise, we first convert all data to a unified semantic representation. Structured records, documents, and tables are embedded into the same vector space. We then build indexes on these embeddings so that cross-domain content can be retrieved efficiently and consistently.

Recall and Re-Ranking for Prompt Construction

With semantic indexes in place, we perform a coarse-grained recall to pull all potentially relevant data for a given imputation task. However, the recall set usually contains irrelevant information, which may risk introducing noise and hallucinations when fed entirely into LLMs. To prevent this, we apply a fine-grained ranking step to select only the top k most contextually relevant snippets. These high-quality snippets form the prompt's knowledge context, optimizing LLM focus and reducing error rates.

Cost-Aware Quality Optimization

To control inference costs in large datasets, we leverage LLMs to generate and validate executable code. Our framework iteratively produces and tests scripts under a given budget, selecting those that maximize imputation accuracy. By executing these scripts locally—rather than issuing repeated API calls—we sharply reduce costs while maintaining high-quality imputations.

We summarize our main contributions as follows:

- Semantic Indexing Framework: We propose a unified and optimized semantic indexing framework for large-scale data lakes, significantly improving data retrieval efficiency and accuracy.
- Retrieval-Enhanced Inference: We designe a retrieval-enhanced method combining high recall scores and precise ranking, greatly enhancing the accuracy of data imputation inference.
- Reduce Costs while Ensuring Inference Quality: We propose a code generation framework based on LLMs for data imputation on large-scale datasets, significantly reducing costs while ensuring the quality of data imputation.

2. Related Work

2.1. Data Imputation

Missing data is a common issue in data analysis and machine learning tasks [36,37]. During data collection, various factors can lead to missing data, such as respondents refusing to answer questions [38], privacy concerns [39], or technical limitations preventing the recording of information [40,41]. Even a small amount of missing data can introduce significant biases and degrade the accuracy and quality of analysis [42,43]. Consequently,

research on missing data has received attention since the early stages of research in the field [44,45], and various data imputation methods have been proposed.

Based on the information sources utilized during data imputation, methods can be broadly categorized into internal imputation and external imputation [46]. Internal imputations rely on signals or redundancy within the dataset itself, inferring missing values solely from the existing data within the same dataset [47,48]. While effective for self-contained datasets, these methods often lack generalizability to other datasets. In contrast, external imputations utilize user-specified external datasets [49,50], which typically require carefully curated knowledge sources explicitly linked to the target data. However, these methods often depend on human intervention during the final stages of dataset imputation, limiting scalability and increasing costs.

Based on different implementation approaches, data imputation methods can be categorized into four types. The simplest approach is to discard samples containing missing values [51]. However, this method introduces bias, compromises the representativeness of the dataset, exacerbates its incompleteness, and ultimately causes the final results to deviate from reality. Subsequently, a range of techniques emerge, spanning from simple statistical methods [17–20] to traditional machine learning approaches [52–55] and further to modern deep learning algorithms [21,22,56–59].

Early statistical methods, such as mean imputation [17], hot deck imputation (HDI) [20], cold deck imputation (CDI) [19], and K-nearest neighbors imputation (KNNI) [18], are computationally simple and widely used. However, they typically assume that data are MAR and are ineffective at handling MNAR scenarios. Additionally, they struggle to manage complex missing data patterns, which may lead to biased imputations and reduced analytical accuracy. Traditional machine learning approaches, such as XGBoost Imputation (XGBI) [60], MissForest Imputation (MissFI) [54], Multiple Imputation by Chained Equations (MICE) [61], Imputation via Individual Models (IIM) [55], Soft Imputation (SI) [62], Matrix Factorization Imputation (MFI) [63], Principal Component Analysis Imputation (PCAI) [64], Multilayer Perceptron Imputation (MLPI) [53], and Round-Robin Sinkhorn Imputation (RRSI) [52], have gained prominence due to their ability to effectively capture non-linear patterns. Furthermore, some methods, like MICE, offer the advantage of incorporating uncertainty through multiple imputations. Despite their advantages, these methods depend heavily on labeled data and require careful tuning of hyperparameters and assumptions about data structure. Modern deep learning algorithms leverage the power of generative models for imputation tasks. These methods either enhance the prior or posterior distributions of explicit generative models, such as deep autoencoders (AEs) [65,66], or adopt alternative training objectives through implicit generative models like Generative Adversarial Networks (GANs) [67,68]. While these methods excel in capturing complex patterns and are highly scalable for large-scale, high-dimensional datasets due to their hierarchical representation learning capabilities, they face several significant challenges: (1) Sensitivity to missing data patterns: The effectiveness of these models can be influenced by the distribution and extent of missing values. (2) High-quality training data requirements: These models require large, diverse datasets to achieve optimal performance. (3) Computational complexity: Both GANs and AEs demand substantial computational resources.

With their rapid development, the use of LLMs for data imputation has garnered significant attention from researchers [29,69–71]. Prompt engineering enables the application of LLMs to specific downstream tasks, improving their accuracy without modifying model parameters [72]. This method leverages the contextual learning capabilities of LLMs by constructing inputs that incorporate additional prompts, such as task instructions or labeled examples, into the task descriptions.

Studies have shown that prompt quality critically affects LLM performance in data imputation [73]. In zero-shot learning—where only the task description is provided—no labeled examples are needed, but results often lag due to missing task-specific context. In contrast, few-shot learning supplies a small set of labeled examples, enabling LLMs to better capture imputation patterns and generally outperform zero-shot methods [74,75]. However, its effectiveness varies widely with the choice of examples, as different example combinations can lead to large swings in accuracy.

Subsequent research has explored training methods to enable LLMs to better understand the structure and knowledge of tabular data, thereby improving their performance in data imputation [73]. However, as the knowledge in these models is stored within their parameters, even with enhanced training, there is no guarantee that the imputed values will achieve consistently high levels of accuracy and reliability [69]. A promising approach to address this limitation is the adoption of retrieval-augmented methods [76]. Among these, RATA [77] leverages a retrieval-augmented transformer model for table enhancement tasks, including data imputation. However, RATA primarily focuses on table-level retrieval, such as considering the number of rows and columns, without leveraging specific external knowledge to estimate missing values more accurately within the tabular data.

While training methods and retrieval-augmented approaches have shown potential in improving LLMs' capabilities for data imputation, these methods have certain limitations [78,79]. The reliance on model parameters or table-level retrieval alone often falls short when addressing tasks that require deeper contextual understanding or integration of external knowledge [80].

Building upon advancements in training methods and retrieval-augmented techniques, some studies have taken an alternative approach by leveraging prompt engineering techniques. By decomposing complex tasks into smaller steps and utilizing multi-turn dialogues [81,82], these methods enable LLMs to process information in a structured, chain-like, or tree-like manner [34]. While prompt engineering enables LLMs to address data imputation tasks with relatively simple input construction, it inherently limits the amount of task-related information the model can access. Consequently, for highly specialized or complex tasks, prompt engineering alone may fail to deliver satisfactory results.

Previous studies have shown that applying LLMs to data imputation tasks is a flexible and efficient approach. However, the limitations of LLMs in complex and domain-specific tasks have prompted the exploration of complementary strategies to fully unleash their potential.

2.2. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) [83] is an innovative framework that combines retrieval mechanisms with generative language models to improve the relevance and precision of generated output [84,85]. This approach involves extracting relevant information from external knowledge sources in response to a query and using a generative model, ranging from smaller-scale models to large language models (LLMs), to produce responses contextualized by the retrieved data [86].

With the emergence of LLMs like ChatGPT, generative models have garnered widespread attention for their powerful natural language understanding and reasoning capabilities. However, LLMs often face issues such as generating responses that are inconsistent with user queries or producing hallucinated information. To address these challenges, RAG was proposed. By integrating contextual retrieval, RAG effectively leverages the strengths of LLMs, enhancing the reliability and factual accuracy of their responses [87–89]. This integration enables RAG to address complex tasks requiring domain-specific knowledge and precise factuality, such as open-domain question answering and fact checking.

Beyond text-based tasks, RAG has shown promise in table-related applications, including TableQA and table augmentation [90,91]. These tasks rely heavily on the retrieval of table-specific knowledge to generate meaningful outputs.

The typical RAG workflow consists of four key steps: (1) Creating a knowledge base: External knowledge sources (e.g., documents, or APIs) are encoded into vector representations using language models and stored in vector databases to form a searchable knowledge base. (2) Semantic retrieval: Queries are transformed into vector embeddings, which are then matched against the knowledge base to retrieve the most relevant results using techniques such as cosine similarity or nearest-neighbor search. (3) Prompt augmentation: Retrieved data is incorporated into the input prompt, providing contextual knowledge to the generative model and enabling more accurate, task-specific outputs. (4) Updating the knowledge base: The knowledge base is periodically refreshed, either in real time or via batch updates, to ensure the retrieved data remains relevant and up-to-date.

Despite its versatility, current RAG implementations face significant limitations that hinder their scalability and effectiveness when dealing with unstructured data [92–94]. Most RAG systems are optimized for structured data, such as well-organized tables and metadata, which limits their ability to leverage the rich, unstructured information commonly found in modern data repositories (e.g., text documents and images) [95]. Some systems attempt to unify retrieval by converting structured data into unstructured formats (e.g., plain text), but this often eliminates the inherent structural properties of the data, making it difficult to utilize inter-table relationships or support holistic discovery. Other approaches map structured and unstructured data into a shared embedding space (e.g., subject–predicate–object triples) [96] while introducing high computational overhead and scalability challenges, as well as potentially losing important structural features essential for accurate retrieval.

These gaps are particularly evident in tasks such as data imputation [97], which involves retrieving and completing missing or incomplete data within large, heterogeneous datasets. Data imputation requires reasoning over both structured and unstructured data sources simultaneously, as it relies on the leveraging of diverse types of information to fill in gaps effectively. Structured data—such as tables, relational databases, or spreadsheets—provides well-defined schemata, relationships, and numerical or categorical values. These features are crucial for identifying patterns, trends, and correlations within the dataset. However, structured data alone often lacks the contextual richness necessary for accurate imputation, especially in cases of sparsity or ambiguity. Unstructured data, such as text documents, images, or logs, can provide complementary context [92]. For instance, textual descriptions in reports or emails might offer insights into specific missing entries in a table, while images or metadata can help deduce or verify missing attributes. Thus, effective data imputation requires integration and reasoning across both structured and unstructured sources. This entails aligning, retrieving, and synthesizing information from disparate formats—a task that often exceeds the capabilities of traditional RAG frameworks. The inability to fully address this complexity highlights a significant challenge in optimizing RAG systems for data imputation. Recent studies [98] have demonstrated the effectiveness of applying RAG-based techniques to noisy relational data imputation, further motivating our approach.

3. Methods

3.1. Framework

The retrieval-augmented generation (RAG) framework proposed in this study integrates advanced retrieval mechanisms with the generative capabilities of LLMs to facilitate precise

and contextually relevant data imputation by leveraging external heterogeneous datasets. As shown in Figure 1, the framework is structured around three primary components:

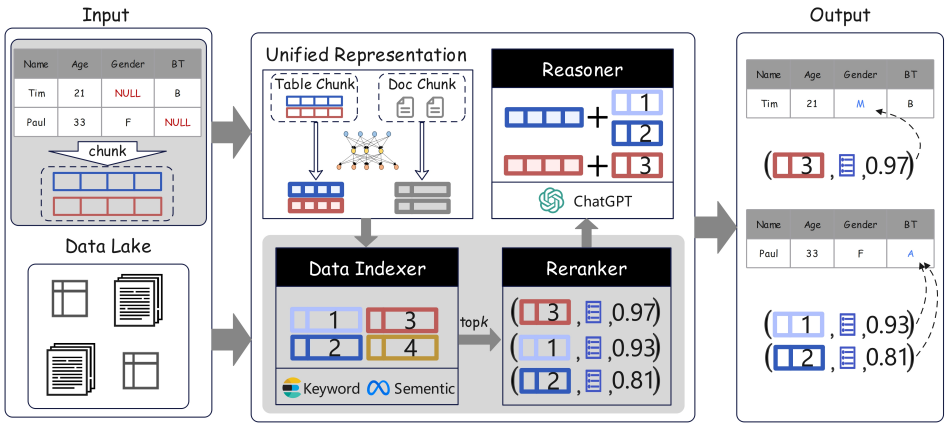

Figure 1. Overall framework.

Semantic Data Lake Indexing: This component focuses on the characterization and indexing of heterogeneous data stored in data lakes. By constructing semantically based indexes, the framework ensures efficient and accurate retrieval of relevant data, even amidst diverse data formats and sources.

Recall and Re-ranking Retrieval Mechanism: This component implements a two-stage retrieval process to maximize the relevance and precision of retrieved data. The recall stage retrieves a broad pool of potentially relevant data, and the subsequent re-ranking stage refines these results by prioritizing the most pertinent data. This approach minimizes noise and ensures that only the most relevant and high-quality data is fed into the LLMs for the downstream data imputation task.

Cost-Constrained Reasoning Quality Optimization: This component addresses the trade-off between data imputation quality and computational cost by framing the task as a code generation problem. Leveraging the LLMs' powerful natural language understanding, data semantic analysis, and code generation capabilities, the framework generates task-specific code for data imputation. This approach reduces the frequency of LLM calls, thereby lowering the cost of data imputation. Task decomposition is employed to break down the complex data imputation process into smaller, manageable sub-tasks, each optimized individually. Iterative code generation and validation further enhance the system's ability, combining multiple code snippets to handle complex scenarios while maintaining cost efficiency.

The subsequent sections delve into each of these components in greater detail, elucidating the methodologies and technologies.

3.2. Semantic Data Lake Indexing Technology

The presence of massive heterogeneous data in data lakes poses significant challenges for data retrieval and integration. To enable efficient and accurate retrieval-augmented methods, we propose a unified indexing approach for heterogeneous data based on feature extraction and joint representation learning, as illustrated in Figure 2.

Representation and Chunking of Heterogeneous Data. In the feature extraction stage, unstructured data—such as text—can be directly represented in a structured format using bag-of-words techniques without relying on external knowledge. Although this process unifies the representation format of features, the resulting vector spaces for different data types remain distinct, making it difficult to construct effective indexes. Therefore, it is necessary to map both structured and unstructured data from their respective representation spaces into a shared vector space during the joint representation learning stage. To

achieve this, we employ contrastive learning to construct the joint representation model, ensuring that semantically related heterogeneous data are embedded closer together in the new vector space.

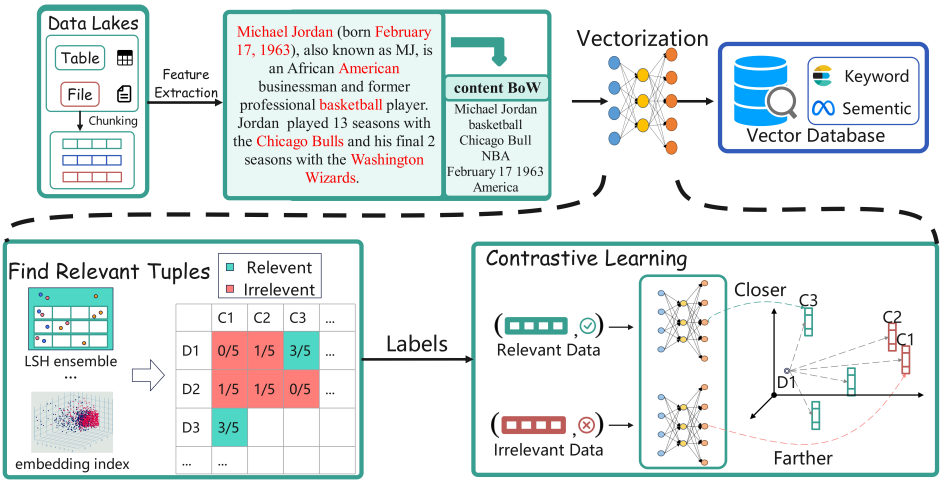

Figure 2. Semantic data lake indexing technology.

Unified Representation Based on Contrastive Learning. Contrastive learning plays a pivotal role in this process. Its core objective is to achieve semantic alignment in the representation space by minimizing the distance between semantically similar sample pairs while maximizing the distance between dissimilar ones during training. Specifically, given a sample pair (x_i, x_j), the goal of contrastive learning is to optimize the following loss function:

$$L = -\log\left(\frac{\exp(\text{sim}(x, x^+))}{\exp(\text{sim}(x, x^+)) + \sum_{x^-} \exp(\text{sim}(x, x^-))}\right) \quad (1)$$

where x represents the current training data, x^+ is a positive example, x^- is a negative example, and $\text{sim}(x, x^+)$ and $\text{sim}(x, x^-)$ are similarity functions between the pairs. This formulation aims to optimize the training process by minimizing the similarity between positive examples while maximizing the similarity between negative examples. Additionally, since contrastive learning lacks labeled training samples, innovative solutions are required, such as keyword-based indexing and semantic vector nearest neighbor graph indexing, to infer correlations and construct high-quality labeled training sets.

To more accurately distinguish semantically related and unrelated data points during the training phase, the framework adopts a Siamese network architecture consisting of dual encoders with shared weights. This design enables the model to more effectively capture semantic differences between relevant and irrelevant training samples.

Construction of Semantically Based Indexes. Effective indexing is paramount in terms of enabling rapid and accurate data retrieval. Once the heterogeneous data has been uniformly characterized and embedded into the unified vector space, semantically based indexes are constructed using vector databases. These indexes facilitate efficient similarity searches, allowing the framework to retrieve semantically relevant data points swiftly. Techniques such as Locality-Sensitive Hashing (LSH) and nearest neighbor graph indexing are integrated to optimize the indexing process, ensuring scalability and performance, even with large-scale datasets.

Efficiency in Large-Scale Similarity Searches. In the context of data lakes containing millions of data points, performing large-scale similarity search efficiently requires unified representation of all data. After standardizing the representations, the trained neural

network is used to embed the data into a shared vector space, upon which an index is built using a vector database to enable efficient subsequent retrieval.

3.3. Recall and Re-Ranking Retrieval-Augmented Technology

Accurate and efficient retrieval of relevant data is critical for enhancing the performance of LLMs in data imputation tasks. The recall and re-ranking retrieval-augmented technology component implements a two-stage retrieval strategy designed to maximize both the relevance and precision of retrieved data , as depicted in Figure 3.

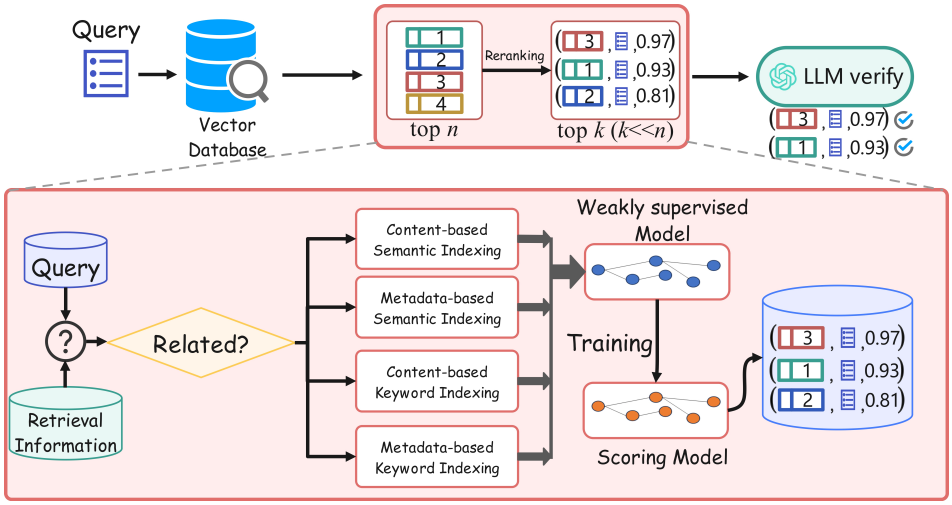

Figure 3. Recall and re-ranking retrieval-augmented technology.

Recall Stage. The initial stage focuses on retrieving a broad set of task-relevant data from the semantically based indexes. Utilizing both keyword-based and semantic search techniques, this stage efficiently scans the vector database to return the top n retrieved results that are potentially relevant to the specific data imputation task at hand. This stage prioritizes recall efficiency, ensuring that a comprehensive pool of relevant data is collected without being bogged down by computational overhead.

Re-ranking Stage. Following the recall stage, a fine-grained re-ranking approach is employed for the specified task to reduce the noise contained in the results and lower the information load that the model needs to process, selecting the k most relevant (where k « n) data points for further validation by the LLMs. To address this, a dedicated scoring model must be constructed to rank the recall results. During the offline training of the scoring model, a subset of data is retrieved from the data lake using various retrieval strategies. The results from different indexes are then aggregated, and a weakly supervised model is employed to annotate the pairwise relevance among these data points, labeling them as either relevant or irrelevant. In the retrieval process, considering that there may be various correlations between different modalities of data, retrieval should be conducted from three perspectives: keyword matching, semantic similarity, and additional information matching. The additional information includes metadata of the tables or documents where the retrieved information is located, such as table names and column names. Next, based on these labeled training data (composed of queries and retrieved information), the scoring model is trained. Subsequently, when a query arrives, the recall results can be quickly scored, and the top k results with the highest scores are fed into the LLMs for final reasoning validation to determine the results of data imputation. For example, in the task of filling in missing values, given a tuple with missing values, n related pieces of information can first be retrieved from the data lake. After re-ranking these n pieces of information using the scoring model, k pieces are selected and sent to the LLMs for final verification.

When a new query is issued, the recall stage retrieves the top n most relevant data points from the vector database. These data points are then passed through the scoring model, which assigns relevance scores based on the learned patterns from the training phase. The top k data points with the highest relevance scores are selected and forwarded to the LLMs for final reasoning and validation. This refined selection process ensures that the LLMs operate on the most pertinent and high-quality data, thereby enhancing the accuracy and reliability of the imputed results.

3.4. Inference Optimization Methods Under Cost Constraints

In practical applications, organizations often face the challenge of how to ensure the quality of data imputation while controlling costs. The application of large language models (LLMs) in data imputation has partially reduced the overall cost; however, large-scale datasets still require multiple invocations of LLMs, resulting in substantial cost overhead. To address this issue, we propose the leveraging of the code generation capabilities of LLMs to reduce imputation costs.

The Code Generation Solution. This study optimizes the task of data imputation by directly using LLMs to generate code for the imputation process. The quality of the generated code is validated through a validation subset of the dataset to ensure the effectiveness of this approach in completing the data imputation task. In this method, LLMs not only leverage retrieval augmentation to better understand task requirements but also utilize the code generation capability of LLMs to reduce costs. This approach strikes a balance between the quality and cost of the data imputation task.

Certain data imputation tasks, particularly those involving large-scale data or full dataset analysis, are best suited for code generation or tool invocation methods. In this context, code generation involves automatically generating code for a specific task using LLMs, followed by iterative optimization and validation. This process can be broken down into the following steps:

- Task Decomposition: An LLM (acting as an advisor) first decomposes the task into smaller, manageable components. Each component is then translated into a specific code generation objective.
- Code Generation and Verification: A code generator produces the corresponding code snippets, which are iteratively refined using automated test cases and debugging tools.
- Error Handling and Refinement: A verifier identifies logical or semantic errors in the generated code, feeding this information back to the advisor for further refinement until all errors are resolved.

This iterative process minimizes human involvement while ensuring the high quality of the generated code, making it suitable for various complex data imputation tasks, such as extracting structured data from unstructured formats or implementing complex combinatorial logic.

Challenges in Code Generation. Despite its promise, LLM-driven code generation faces several challenges: (1) Quality and robustness: LLMs may struggle to produce high-quality code for complex or large-scale tasks, particularly when the logic involves intricate combinations of rules or constraints. (2) Prompt engineering: Crafting effective prompts to guide LLMs in generating accurate code requires domain expertise and significant time investment. (3) Validation complexity: Verifying the correctness of generated code, especially for large and diverse datasets, is non-trivial. Logical errors, inefficiencies, and edge cases may escape detection.

To address these challenges, this study proposes two techniques:

- Validated code generation: This method integrates code generation and debugging into a unified process. By leveraging the reasoning capabilities of LLMs, the sys-

tem iteratively refines code snippets until all test cases are satisfied. This approach minimizes user input and enhances code quality.

- Evolutionary code integration: This technique generates multiple code snippets with complementary strengths, combining them iteratively to handle complex tasks. By optimizing the integration of these snippets, the framework ensures robustness and scalability.

Next, we elaborate on these two techniques in more detail. To generate robust and high-quality code while minimizing human involvement, this method leverages the observation that LLMs perform better when additional reasoning processes are included in the prompt. For example, compared to a simple prompt like "fill in the missing sales data", a prompt that includes an additional reasoning process—"Approach: First, analyze the pattern of the missing data, then use the sales trends of similar periods to impute, and finally verify the consistency of the imputed data with the overall sales trend"—typically generates higher-quality code, as this reasoning provides the LLMs with valuable guidance on how to solve the problem.

Validated Code Generation. In this study, rather than relying on users to manually write down their suggestions, we introduce a separate LLM dialogue to automatically generate recommendations. As shown in Figure 4, the validated code generation method comprises three key components: an LLM advisor, LLM code generator, and verifier. The LLM advisor communicates with the LLM code generator and verifier, providing suggestions for generating code snippets based on the task. The verifier automatically checks the semantic and logical correctness of the generated code. The code and error messages are then relayed back to the LLM advisor based on the verification results, repeating this process until the code passes all test cases or times out. Through such iterative processes, the LLMs can produce more robust and resilient code snippets.

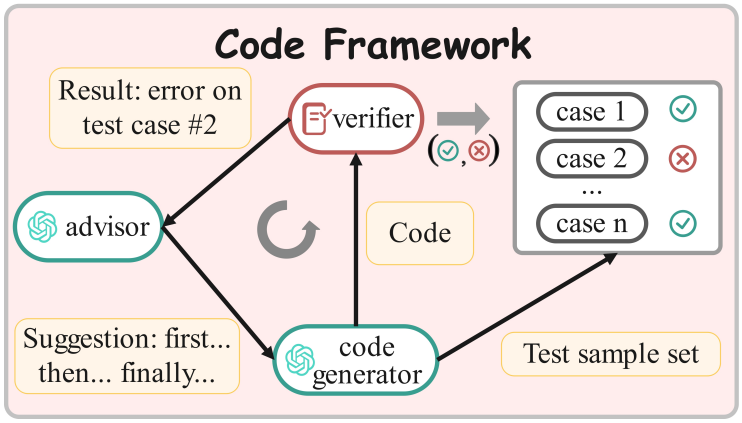

Figure 4. Code framework.

Evolutionary Code Integration. For tasks requiring intricate logic and multiple combinatorial rules, evolutionary code integration generates a diverse set of complementary code snippets. As shown in Figure 5, these snippets undergo iterative optimization, where less effective code is discarded and successful code snippets are combined to handle complex scenarios. This evolutionary approach ensures that the final codebase is both robust and capable of handling the multifaceted requirements of data imputation tasks.

Since theoretically verifying the correctness of LLM-generated code remains an open issue, this study explores empirical validation methods to comprehensively verify code and minimize potential errors. Two methods are introduced: logical correctness assessment and example generation. Both aim to minimize user involvement and the required testing samples. Note that the code verification process proposed in this study is flexible and can

support any validation method, as long as it returns the verification result in the form of <result, error information>.

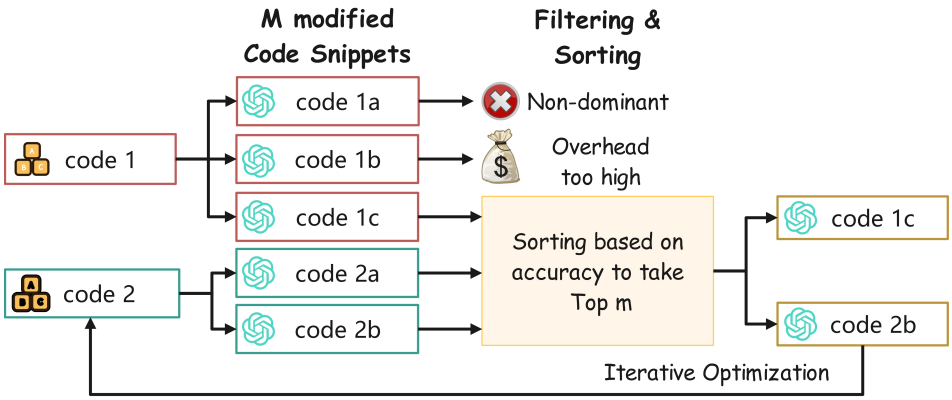

Figure 5. Code process.

Logical Correctness Assessment: Given that LLMs can understand natural language requirements and code logic, they can be directly asked to assess the logical correctness of the generated code and its alignment with the task requirements. Specifically, similar to the verifier in the code generation process, the LLM is provided with complete profile information: the code to be verified, the program runtime environment, etc. The LLM is explicitly asked to provide judgments and reasoning processes, which are then used as evidence. However, since this type of validation heavily relies on the performance of the LLM, it is typically less reliable. Therefore, this study allows the LLM advisor to discard suspicious verification results using the SKIP CASE method. Logical correctness assessment is the most fundamental and general solution for using LLMs in code validation.

Example (test case) Generation: Human programmers typically rely on test cases to check their code. When test cases are available, the generated code is placed in a sandbox and tested against all examples. A result of result=1 is returned only when the generated code passes all test cases, requiring no evidence. Failure on any test case, including compilation errors, format errors, timeouts, or incorrect outputs, is marked as result=0, with errors returned as evidence. In many cases, users do not provide sufficient test examples. Therefore, it is necessary to use LLMs to automatically generate examples as test cases. Since LLMs can assess the logical correctness of code and identify error conditions, this method requires the LLMs to deliberately generate adversarial test cases to evaluate the robustness of the generated code. Thus, this method can evaluate code snippets requiring very few user-provided examples.

4. Experimental Verification

We divide the experiments into four parts. The first part presents the overall evaluation of the data imputation task, while the remaining three parts focus on the independent analysis of each module. These contents are presented in the following four subsections.

4.1. Evaluation of the Overall Experimental Effect of Data Imputation

In this chapter, we systematically evaluate the effectiveness of the complete retrieval-augmented imputation method, with a focus on analyzing the impact of different retrieval strategies and language model combinations on imputation performance.

4.1.1. Dataset Preparation

To make our experiments more compelling, we collected and preprocess datasets from various domains, as described below.

Dataset Collection: Our goal is to showcase datasets with specific features: (1) they should originate from real-world scenarios, and (2) they should cover a variety of data domains. Guided by these criteria and existing work on missing value imputation, we collected the three datasets from real-world data sources:

- WikiTuples [99]: This dataset is composed of 207,912 tables from various fields within the data lake, such as sports, politics, and arts. It contains 10,003 tuples, with missing values present in 807 tables. These missing values span multiple attributes, including Party, Director, Team, Album, etc. On average, each tuple with missing values corresponds to 4.38 target tuples, reflecting the complexity of locating target tuples among 2,674,164 tuples in the data lake. Although WikiTuples and the retriever's training data come from the same source, the latter did not undergo the same rigorous annotation process. Furthermore, only 41,000 tables were used during the pre-training phase, which is significantly smaller than the size of the data lake.
- Education [100]: This dataset includes information about elementary and high schools in Chicago, containing 654 tuples, with missing values in the Address, Zip Code, and Phone Number columns. It comprises 11,132 tuples, with each tuple missing values corresponding to four target tuples.
- Cricket Players [101]: The Cricket Players dataset also originates from RetClean and focuses on the sports domain. It contains 213 incomplete tuples, with the data lake containing 94,164 tuples. On average, each tuple with missing values corresponds to 1.38 target tuples. The missing information pertains to players' nationality and bowling style attributes.

Candidate Tuple Construction: For tuples with missing values (i.e., incomplete tuples), we first create a candidate set by selecting tuples that may assist in imputing the missing values. Given the large number of tuples in the data lake, we utilize explicit information, such as matching the same subject ID or linking to the same entity in the original data. This helps us establish effective filtering rules to identify potential candidates for incomplete tuples.
Expert Annotation: The final step is managing the target tuples for each incomplete tuple. For each tuple with missing values, we obtain the corresponding candidate set from the previous step. We present each tuple, along with its candidate tuples, to experts, who evaluate whether any of the candidates can fill at least one missing value in the tuple, thereby being identified as target tuples. This manual process is very time-consuming and requires specific domain knowledge, making attempts to automate it quite challenging. To reduce annotation costs, we primarily focus on cases where the candidate set contains 10 or fewer tuples.

4.1.2. Baseline Model

Reasoner Baseline: We use the following baselines for end-to-end data imputation:

- GPT-3.5/GPT-4.0: We directly use GPT-3.5 and GPT-4.0 as the reasoners for data imputation without the retrieval module. Additionally, to guide the model in recognizing the format of missing values, we add an extra complete tuple to the table that requires imputation.
- BM25-based GPT-3.5/GPT-4.0: To study the impact of different retrieval methods in our retrieval-enhanced imputation framework, we use BM25 [102] as the retriever module. This setup allows us to evaluate the influence of various retrieval strategies on the accuracy of data imputation.

4.1.3. Evaluation Metrics

Data Imputation Metric: We apply Exact Match (EM) accuracy to assess our method in the end-to-end data imputation task. A generated value for a missing cell is considered

correct if it matches any entry in the normalized acceptable answers list. This normalization step involves converting to lowercase, removing punctuation, and eliminating duplicate spaces. For each missing value, we collect all semantically equivalent cell values to form the corresponding answer list.

4.1.4. Experimental Results and Analysis

For the settings involving retrieval methods, the top five retrieved results are fed into the LLMs, along with the incomplete tuple that needs imputation. Table 1 shows the experimental results.

Table 1. Evaluation of data imputation performance. The best results are bolded. We ran each experiment three times and report the average ± standard deviation.

Reasoner	Retrieval Modules	WikiTuples	Education	Cricket Players
GPT-3.5	w/o	0.715 ± 0.008	0.017 ± 0.002	0.896 ± 0.005
	w/tuples (BM25)	0.577 ± 0.007	0.892 ± 0.004	0.889 ± 0.006
	w/tuples (ours)	0.886 ± 0.005	0.976 ± 0.003	0.964 ± 0.004
GPT-4.0	w/o	0.752 ± 0.009	0.597 ± 0.006	0.863 ± 0.005
	w/tuples (BM25)	0.800 ± 0.006	0.925 ± 0.004	0.909 ± 0.004
	w/tuples (ours)	**0.902 ± 0.004**	**0.979 ± 0.002**	**0.972 ± 0.003**

The results indicate that our method significantly improves the accuracy of data imputation compared to using the LLMs alone across all datasets. GPT-3.5 shows an average improvement of 41%, while GPT-4.0 shows an average improvement of 33.8%. Directly applying the LLMs resulted in suboptimal outcomes because the knowledge stored in the LLMs' parameters is not sufficiently accurate, introducing a degree of uncertainty. Thus, even for datasets in the Wikipedia domain included in the LLMs' training data, the imputation accuracy is low. In more specific and proprietary fields, such as business and education, both GPT-3.5 and GPT-4 struggle to estimate missing values accurately. The accuracy of GPT-3.5 is similarly very low, while GPT-4's accuracy is slightly higher but still limited. In contrast, our method demonstrates exceptional performance due to the effective integration of the LLMs' advanced reasoning capabilities with the rich knowledge from accurately retrieved tuples from the data lake.

Furthermore, the results emphasize the importance of effective retrieval methods within the retrieval-augmented imputation framework. BM25 exhibits poorer results than our method on most datasets, except for the business dataset, where there is a large lexical overlap between the incomplete tuples and the target tuples, allowing BM25 to perform well. Therefore, the effectiveness of the LLMs combined with BM25 is almost always inferior to that of our method. It is also evident that using suboptimal retrievers like BM25 adversely affects the data imputation accuracy of less advanced models like GPT-3.5. This is primarily because inaccurately retrieved tuples can mislead the LLMs, especially when they lacks strong reasoning capabilities, impairing predictive performance.

Additionally, in most cases, GPT-4 shows better performance than GPT-3.5 across all datasets, except for the Cricket Players dataset without retrieved tuples. This anomaly arises because both models lack sufficient domain-specific knowledge to accurately estimate missing values in these cases. In such instances, GPT-3.5 tends to guess answers based on the content of the complete tuples. However, without additional contextual information, GPT-4 often does not provide an answer, resulting in empty responses. Nevertheless, GPT-4's advanced reasoning capabilities generally lead to better performance, particularly in identifying target tuples and inferring missing values. Notably, when implementing

retrieval-augmented imputation, the performance gap between GPT-3.5 and GPT-4 narrows. This suggests that retrieval-augmented methods can effectively mitigate the limitations of less advanced language models in data imputation.

Our method significantly outperforms baseline LLMs and those paired with weaker retrieval methods, improving data imputation accuracy across various domains and compensating for the shortcomings of less advanced language models.

For the top retrieved tuples fed into the LLMs, the success rate of retrieval tends to increase with k, meaning that a greater number of retrieved tuples is more likely to contain the target tuples needed for the input missing values. Intuitively, providing more retrieved tuples to the LLMs appears beneficial. However, increasing the length of input data introduces complexity in the LLMs' reasoning process. To further explore the impact of the number of retrieved tuples fed into the LLMs on data imputation, we conducted comparative experiments. Specifically, we used all test tuples from the ShowMovie and Cricket Players datasets and sampled 500 tuples with missing values from the other three datasets. We then provided different quantities of tuples retrieved by the re-ranker to GPT-3.5 to obtain the corresponding data imputation results. To reduce costs, we did not use GPT-4 in this phase of the experiment.

From the three subplots in Figure 6, it is evident that the accuracy of data imputation does not significantly increase an the increase in the number of retrieved tuples fed into the LLMs. Instead, a decrease in accuracy is observed across all datasets. This result indicates a trade-off; the increase in the number of tuples required for the LLMs to process may negatively affect the accuracy of data imputation. Thus, the importance of an efficient retrieval module that can achieve high success rates with the minimal possible k is emphasized.

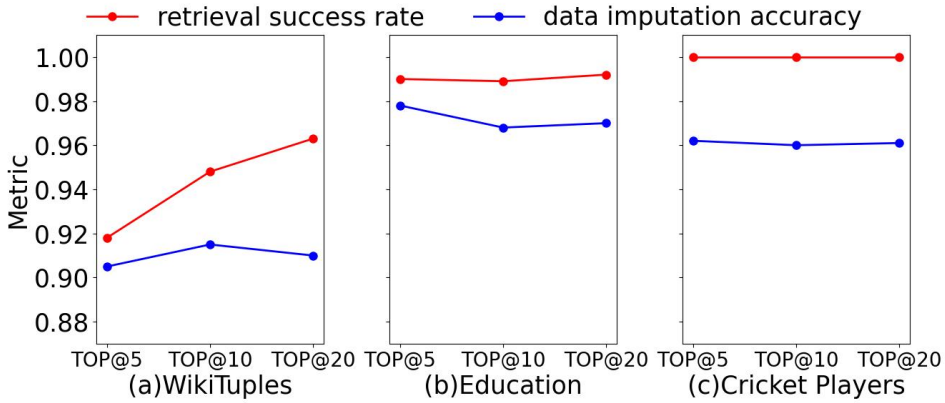

Figure 6. Retrieval success rate and data imputation accuracy.

While increasing the number of retrieved tuples generally improves the success rate of retrieval, it does not guarantee better imputation accuracy due to the complexity it introduces into the LLMs' reasoning process. This finding highlights the critical need for an effective retrieval module that can maximize success rates while minimizing the number of retrieved tuples.

4.2. Evaluation of Retriever Effectiveness

In this section, we systematically evaluate the performance of the retriever module, with a focus on comparing different retrieval baselines and analyzing how training data construction strategies affect retrieval effectiveness.

4.2.1. Dataset Preparation

We used the same datasets as in the evaluation of the overall experimental effect of data imputation.

4.2.2. Baseline Model

Retriever Baseline: We compare our retriever with several baselines. The first two methods have zero-shot capabilities and are, thus, applied directly. For the third method, we retrain it using the same training data for a fair comparison. The baselines are introduced as follows:

- BM25: BM25 is the most commonly used sparse retrieval method and demonstrates very robust performance.
- Contriever: Contriever is an unsupervised retriever that performs well in few-shot and zero-shot settings for paragraph retrieval.
- BERT [103] with Language Modeling (LM) Task: Previous work on table data representation has adopted pre-training tasks centered around language modeling and tabular data structures. For example, TabReformer and TURL utilize masked language modeling tasks, with the latter adding masked entity recovery. To demonstrate that contrastive learning methods are crucial for the success of tuple embeddings in retrieval, we trained a tuple encoder using the BERT-base model. This model shares the same basic structure as our encoder but was trained using a language modeling task. Specifically, for a given tuple (t), we randomly mask one cell, then input t into the encoder, asking it to predict the value of the masked cell. Additionally, we apply two enhancement operators—shuffle and delete—to tuple t to enrich our training data. Due to the structural nature of tables and their inherent permutation invariance, these operators are commonly used for tabular data.

4.2.3. Evaluation Metrics

Retriever Metric: In the retrieval phase, our goal is to retrieve as many target tuples as possible from the top retrieval results. Therefore, we use *Recall@K*, defined as the ratio of the number of successfully retrieved target tuples to the total number of target tuples present in the top results.

4.2.4. Experimental Results and Analysis

The experimental results, as shown in Table 2, demonstrate the effectiveness of the retriever, which achieves the highest recall and success rates across all datasets, except for Education. Additionally, it is noteworthy that the retriever is trained on 40,000 tables from Wikipedia and is directly applied to the three datasets mentioned above without any additional training, showcasing its robustness and generalization ability. Furthermore, the results highlight that contrastive learning is key to retrieving effective tuple encoders. The retriever and BERT share the same underlying model structure and are trained on the same table corpus. The primary distinction lies in the training tasks. However, BERT with the language modeling task lacks generalization ability and performed the worst across all datasets, indicating its unsuitability for our scenario. This contrast underscores the unique advantages and effectiveness of training the retriever using contrastive learning.

Table 2. Evaluation of retriever performance (average ± standard deviation). The best results are bolded. Each experiment was repeated three times.

Retriever	WikiTuples		Education		Cricket Players	
	S@1	S@5	S@1	S@5	S@1	S@5
Initial Retrieval	0.327 ± 0.006	0.279 ± 0.005	0.743 ± 0.004	0.901 ± 0.003	0.902 ± 0.002	0.043 ± 0.001
Contriever	0.485 ± 0.007	0.456 ± 0.006	0.758 ± 0.005	0.825 ± 0.004	0.074 ± 0.002	0.043 ± 0.001
BERT w/LM Task	0.275 ± 0.005	0.201 ± 0.004	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Re-ranker (ours)	**0.951 ± 0.003**	**0.791 ± 0.003**	**0.992 ± 0.002**	**0.923 ± 0.002**	**1.000 ± 0.000**	**0.989 ± 0.001**

In this study of the retriever, leveraging contrastive learning, outperformed baselines in recall and success rates across all datasets, proving its superior effectiveness and generalization ability. The construction of the pre-training dataset is crucial for the performance of the retriever. As previously mentioned, we provide an in-depth explanation of the construction methods. To assess the validity of our training data and identify key factors in dataset construction, we detail the anchor tuples, positive tuples, and negative tuples.

(1) Impact of Anchor Tuple Construction: The training data has two settings for anchor tuples: with or without missing values. Here, the construction process considers only one of these settings. We find that combining complete anchor tuples with anchor tuples that have missing values can enhance the performance of the retriever. This improvement may be due to the inclusion of missing values in the anchors, aligning with the intention of data imputation, while complete anchors help the retriever learn to identify tuples with the same subject entity as the positive examples, thereby ensuring higher recall.

(2) Impact of Positive Tuple Construction: Unlike traditional methods that only consider enhanced anchors as positive tuples, we regard tuples that satisfy certain conditions from other tables as positive tuples. To demonstrate the effectiveness of these added positive tuples, we construct a new dataset by removing these positive tuples and observe a significant drop in the retriever's performance across all datasets. This is because the target tuples in real-world scenarios may come from entirely different tables than the anchors, highlighting the importance of diverse positive samples.

(3) Impact of Negative Tuple Construction: We consider other tuples from the same table that the anchor is related to in the negative context. Since the retriever is applied to data imputation, for an anchor tuple (t) with a missing $t[j]$, it is natural to treat the augmented anchor with the j-th attribute removed as negative, i.e., a tuple very similar to the anchor but lacking information to assist in imputing the missing value. This type of negative sample is added to the training data. However, we observe that it leads to a significant decline in retrieval results, primarily because accurately capturing and distinguishing cell-level semantics when encoding each tuple as embeddings is challenging. Consequently, incorporating these negative samples adversely affected performance. Moreover, this change resulted in a substantial drop in performance compared to other discussed modifications.

Based on the above research and discussion, we find that the data construction method for the retriever is highly effective and that the combination of anchor tuples, positive tuples, and negative tuples is crucial. Notably, the impact of the methods used to construct positive and negative samples on performance is more significant than changes in anchor tuple construction.

4.3. Evaluation of Re-Ranker Effectiveness

In this section, we systematically evaluate the performance of the re-ranker module, with a focus on comparing various baseline methods—including both traditional models and LLM-based approaches—and analyzing their success rates and influencing factors across multiple datasets.

4.3.1. Dataset Preparation

We used the same datasets as in the evaluation of the overall Experimental effect of data imputation.

4.3.2. Baseline Model

Re-ranker Baseline:We compare our re-ranker with several baseline methods, selecting representative re-ranker models for evaluation. The experiments were conducted using the same datasets and evaluation metrics. Through these comparisons, we gain a more

comprehensive understanding of the performance of our re-ranker and its position relative to the state-of-the-art methods. The baselines are introduced as follows:

- monoBERT [104]: This is a model designed for query-based paragraph re-ranking, utilizing the BERT model. This process involves concatenating the query with the paragraph text, then using the [CLS] vector to compute the re-ranking score. To compare monoBERT with our re-ranker, we use the provided parameters to initialize it and fine-tune it on our training data.
- RoBERTa-LCE: This introduces a new loss called Local Contrastive Estimation (LCE) to train a more robust re-ranker. We fine-tune the RoBERTa-base model using the LCE loss on our training data for a fair comparison.
- GPT-3.5 with List: Recent studies have highlighted the exceptional performance of LLMs in relevance re-ranking tasks. Thus, we use GPT-3.5 as a re-ranker employing the list method. Specifically, GPT-3.5 is fed a set of retrieved tuples and an incomplete tuple, instructing the model to generate a re-ranked list based on relevance to the incomplete tuple. To manage input limitations, we adopt a sliding window strategy with a window size of 30 and a step size of 14.

4.3.3. Evaluation Metrics

Re-ranker Metric: Since many tuples with missing values can only be filled by one target tuple, following previous research, we report *Success@K*, which is the percentage of incomplete tuples in the top retrieved tuples that contain at least one target tuple to evaluate the performance of the re-ranker.

4.3.4. Experimental Results and Analysis

To evaluate the performance of the re-ranker, this study compares it against four baseline methods, two of which innovatively use LLMs as re-rankers. Due to cost constraints, for the methods utilizing LLMs, only a subset of tuples is sampled from the test set of each dataset. Specifically, 180 tuples are selected from the Cricket Players dataset, while 200 tuples are sampled from the remaining datasets. The results for the complete datasets are reported in Table 3, while the results for the sampled subsets are shown in Table 4.

Table 3. Evaluation of re-ranker performance (average ± standard deviation). The best results are bolded. Each experiment was repeated three times.

Re-Ranker	WikiTuples		Education		Cricket Players	
	S@1	S@5	S@1	S@5	S@1	S@5
Initial Retrieval	0.531 ± 0.006	0.803 ± 0.004	0.577 ± 0.005	0.923 ± 0.003	0.830 ± 0.003	0.989 ± 0.001
monoBERT	0.501 ± 0.007	0.799 ± 0.005	0.973 ± 0.003	0.984 ± 0.002	0.622 ± 0.004	0.931 ± 0.002
RoBERTa-LCE	0.615 ± 0.006	0.837 ± 0.004	0.970 ± 0.002	0.984 ± 0.002	0.902 ± 0.003	0.974 ± 0.002
Re-ranker (ours)	**0.713 ± 0.005**	**0.927 ± 0.003**	**0.976 ± 0.002**	**0.986 ± 0.001**	**0.941 ± 0.002**	**1.000 ± 0.000**

First, the results in Table 3 indicate that our re-ranker achieves the highest *Success@5* across various datasets compared to other re-rankers that require fine-tuning. Additionally, on all datasets, the re-ranker shows an average improvement of 24.3% in *Success@1* and 6.2% in *Success@5* compared to the initial results from Initial Retrieval. Only six tuples with missing values were used for training, and we observe that all re-rankers need fine-tuning to outperform the retriever on this dataset.

Table 4. Evaluation of performance on re-ranker subsets (average ± standard deviation). The best results are bolded. Each experiment was repeated three times.

Re-ranker	WikiTuples		Education		Cricket Players	
	S@1	S@5	S@1	S@5	S@1	S@5
Initial Retrieval	0.231 ± 0.005	0.733 ± 0.004	0.577 ± 0.004	0.923 ± 0.003	0.830 ± 0.003	0.989 ± 0.001
GPT-3.5/Pairwise	0.470 ± 0.006	0.756 ± 0.004	0.260 ± 0.005	0.960 ± 0.002	0.932 ± 0.002	0.989 ± 0.001
GPT-3.5/Listwise	0.210 ± 0.007	0.676 ± 0.005	0.255 ± 0.006	0.825 ± 0.003	0.750 ± 0.003	0.841 ± 0.002
Re-ranker (ours)	**0.630 ± 0.005**	**0.915 ± 0.003**	**0.976 ± 0.002**	**0.986 ± 0.001**	**0.941 ± 0.002**	**1.000 ± 0.000**

Second, Table 4 demonstrates that the re-ranker has a significant advantage over existing re-ranking methods that utilize LLMs across various datasets. The GPT-3.5 model using the Listwise method is clearly the most effective, as its *Success@5* falls below the re-ranker's initial retrieval results across all datasets. Moreover, its *Success@1* also lags behind the initial retrieval results on three datasets. This poor performance is primarily attributed to two factors: first, GPT-3.5's limited ability to understand tabular data and, second, the inherent challenge GPT-3.5 faces in reordering a list of tuples based on the relevance of incomplete tuples, especially when dealing with large lists. In contrast, when GPT-3.5 is combined with a pairwise method, the model selects one of two candidate tuples, resulting in significant improvements. This is because making a choice between two options is simpler than ranking an entire list. However, this method performs poorly compared to the re-ranker on most datasets. The reason is that if GPT-3.5 incorrectly ranks a target tuple low in the pairwise comparison, that tuple has almost no chance to improve its subsequent ranking.

When sufficient training data is available, the re-ranker can outperform traditional re-ranking methods and those utilizing LLMs, demonstrating excellent performance across various datasets, particularly in enhancing success rates.

4.4. Evaluation of Code Generator Effectiveness

As a typical data cleaning task, data imputation fills in missing or corrupted data with substitute values. In this section, we systematically evaluate the effectiveness of the code generation module in data imputation tasks, with a focus on its performance on real-world datasets and its comparative advantages over statistical methods, supervised learning approaches, and LLM-based baselines.

4.4.1. Dataset Preparation

Data Configuration Plan. Data imputation typically benefits from rule-based methods. Therefore, its data management plan supports code and data access modules. Additionally, we briefly discuss the performance of small models on this task to confirm the superiority of the code and model in this area.

Datasets. We use the Buy dataset and the Restaurant dataset for evaluation. The Buy dataset has three attributes: product name, product description, and manufacturer. In this dataset, the manufacturer is masked and considered the missing attribute to estimate. The Restaurant dataset has five attributes: restaurant name, address, city, phone number, and food type. In this dataset, the city is masked and regarded as the missing attribute to estimate. These two datasets are drawn from real-world domains. The *Buy* dataset comprises product listings collected from an e-commerce platform, while the *Restaurant* dataset contains restaurant records sourced from a public directory. Both have one key attribute masked (manufacturer or city) to create a realistic imputation task.

4.4.2. Baseline Model

The only available baseline for few-shot imputation on the Buy dataset is FMs, which is a pure end-to-end LLM solution with 10-shot examples. We also compare our method with statistical cleaning methods like HoloClean and supervised methods like IMP, which use large training datasets.

4.4.3. Evaluation Metrics

In this study, we also report imputation accuracy, where data imputation is considered correct if the imputed values match the true values exactly.

4.4.4. Experimental Results and Analysis

Table 5 displays the results of our experiments. Our method achieves an imputation accuracy of 96.1% on the Buy dataset using only three examples, significantly outperforming the HoloClean general data cleaning method. Additionally, its performance is comparable to the supervised SOTA solution IMP trained on thousands of labeled data. Compared to FMs, which use LLMs on every row of the table, SEED reduces the number of LLM calls by over 50% on the Buy dataset while only sacrificing about 2% to 5% accuracy. We also experimented with a plan using model modules instead of code modules, and the results indicate that using code modules is, indeed, more suitable for this task. Meanwhile, our method effectively reduces the number of calls to LLMs, especially when handling similar tasks within the same batch, leveraging code reuse mechanisms to efficiently complete multiple tasks.

Table 5. Test of effects on data generator (average ± standard deviation). The best results are bolded. Each experiment was repeated three times.

Method	Buy		Restaurant	
	acc	LLMs Ratio	acc	LLMs Ratio
HoloClean	0.162 ± 0.003	N/A	0.331 ± 0.004	N/A
IMP	0.965 ± 0.004	N/A	0.772 ± 0.005	N/A
FMs	0.963 ± 0.003	**1.000** ± 0.000	0.875 ± 0.004	**1.000** ± 0.000
Our method (LLMs only)	**0.976 ± 0.002**	**1.000** ± 0.000	**0.890 ± 0.003**	**1.000** ± 0.000
Our method	0.961 ± 0.004	0.439 ± 0.012	0.821 ± 0.005	0.000 ± 0.000

5. Conclusions

Data lakes store massive amounts of structured, unstructured, and heterogeneous data, encompassing a wealth of knowledge across different domains. To support efficient and precise retrieval enhancement, it is necessary to represent and index this heterogeneous data. Unstructured data often has a long length and contains a lot of content unrelated to the query; thus, in this study, it was processed in chunks to facilitate subsequent retrieval. Additionally, since heterogeneous data exists in different spaces, this study unified its representation, enabling effective support for unified and precise retrieval. Finally, to achieve efficient retrieval, this study constructed a corresponding index based on the representations.

After establishing the index for the data lake, it is necessary to adaptively recall texts or tables containing relevant knowledge from the data lake based on the specific data imputation tasks. However, this recall is relatively coarse-grained, suitable for filtering out a large portion of content related to the task. If all this content is sent to the LLMs, along with the task, much of the noise can negatively impact the model's generation results, potentially leading to hallucinations. Therefore, in this study, the recalled results underwent

more refined ranking, similar to a search engine, selecting content that is more closely related to the task to feed into the LLMs, thereby achieving more precise reasoning results.

In many cases, the cost of completing data imputation tasks is constrained. If the imputation for each data entry were entirely dependent on the reasoning capabilities of LLMs, it would incur substantial costs for large-scale data. Therefore, an urgent problem to solve is how to reduce costs while ensuring the quality of data imputation. We can leverage the strong natural language understanding and code generation capabilities of LLMs by having them generate code to complete data imputation tasks, which would significantly reduce the cost of performing such tasks.

This study focuses on how to establish effective indexing for multisource heterogeneous data in data lakes, how to refine the retrieved data, and how to achieve low-cost data imputation tasks while ensuring high quality. Since the data imputation tasks used in this study share strong similarities with other data imputation tasks, the proposed framework can be easily adapted to other tasks.

Currently, with the increasing challenges of big data in terms of volume and variety, issues such as data incompleteness and low data quality have emerged. The data cleaning method based on RAG proposed in this study can, to some extent, overcome key bottlenecks in big data analysis applications. This, in turn, promotes better economic development, enhances government services and regulatory capabilities, and improves social governance.

Limitation: We note that the performance of our RAG-based framework relies on the quality of retrieved contexts; retrieval errors or noisy passages may degrade imputation results.

Future Work: In future work, we will explore retrieval-error mitigation strategies—such as adaptive ranking refinement and chain-of-thought prompting—to further enhance the robustness of our approach.

Author Contributions: Conceptualization, X.S.; Methodology, X.S.; Software, X.S.; Validation, J.W.; Formal analysis, G.J.C.; Investigation, G.J.C.; Resources, L.Q.; Data curation, D.J.; Writing—original draft, J.W.; Writing—review & editing, J.W.; Visualization, D.J.; Project administration, L.Q.; Funding acquisition, L.Q. All authors have read and agreed to the published version of the manuscript.

Funding: Xiaojun SHI is supported by the National Key R&D Program of China (Grant No. 2024YFE0209000), the NSFC (Grant No. U23B2019).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Blazquez, D.; Domenech, J. Big Data sources and methods for social and economic analyses. *Technol. Forecast. Soc. Chang.* **2018**, *130*, 99–113. [CrossRef]
2. Can, U.; Alatas, B. Big social network data and sustainable economic development. *Sustainability* **2017**, *9*, 2027. [CrossRef]
3. Arora, A.; Vats, P.; Tomer, N.; Kaur, R.; Saini, A.K.; Shekhawat, S.S.; Roopak, M. Data-Driven Decision Support Systems in E-Governance: Leveraging AI for Policymaking. In *Proceedings of the International Conference on Artificial Intelligence on Textile and Apparel*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 229–243.
4. Linkov, I.; Trump, B.D.; Poinsatte-Jones, K.; Florin, M.V. Governance strategies for a sustainable digital world. *Sustainability* **2018**, *10*, 440. [CrossRef]
5. van Veenstra, A.F.; Kotterink, B. Data-driven policy making: The policy lab approach. In Proceedings of the Electronic Participation: 9th IFIP WG 8.5 International Conference, ePart 2017, St. Petersburg, Russia, 4–7 September 2017; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2017; pp. 100–111.
6. Rendle, S. Factorization machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 995–1000.
7. He, X.; Chua, T.S. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 355–364.

8. Jiang, P.; Xiao, C.; Cross, A.; Sun, J. Graphcare: Enhancing healthcare predictions with personalized knowledge graphs. *arXiv* **2023**, arXiv:2305.12788.
9. Wang, L.; Liu, Q.; Zhang, M.; Hu, Y.; Wu, S.; Wang, L. Stage-aware hierarchical attentive relational network for diagnosis prediction. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 1773–1784. [CrossRef]
10. Ye, M.; Cui, S.; Wang, Y.; Luo, J.; Xiao, C.; Ma, F. Medpath: Augmenting health risk prediction via medical knowledge paths. In Proceedings of the Web Conference, Ljubljana, Slovenia, 19–23 April 2021; pp. 1397–1409.
11. Yang, K.; Xu, Y.; Zou, P.; Ding, H.; Zhao, J.; Wang, Y.; Xie, B. KerPrint: Local-global knowledge graph enhanced diagnosis prediction for retrospective and prospective interpretations. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 5357–5365.
12. Cui, C.; Wang, W.; Zhang, M.; Chen, G.; Luo, Z.; Ooi, B.C. Alphaevolve: A learning framework to discover novel alphas in quantitative investment. In Proceedings of the 2021 International Conference on Management of Data, Xi'an, China, 20–25 June 2021; pp. 2208–2216.
13. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 11106–11115.
14. Ren, Y.; Chen, Y.; Liu, S.; Wang, B.; Yu, H.; Cui, Z. TPLLM: A traffic prediction framework based on pretrained large language models. *arXiv* **2024**, arXiv:2403.02221.
15. Adhikari, D.; Jiang, W.; Zhan, J.; He, Z.; Rawat, D.B.; Aickelin, U.; Khorshidi, H.A. A comprehensive survey on imputation of missing data in internet of things. *ACM Comput. Surv.* **2022**, *55*, 1–38. [CrossRef]
16. Little, R.J.; Rubin, D.B. *Statistical Analysis with Missing Data*; John Wiley & Sons: Hoboken, NJ, USA, 2019; Volume 793;
17. Farhangfar, A.; Kurgan, L.A.; Pedrycz, W. A novel framework for imputation of missing values in databases. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Humans* **2007**, *37*, 692–709. [CrossRef]
18. Altman, N.S. An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.* **1992**, *46*, 175–185. [CrossRef]
19. Jerez, J.M.; Molina, I.; García-Laencina, P.J.; Alba, E.; Ribelles, N.; Martín, M.; Franco, L. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artif. Intell. Med.* **2010**, *50*, 105–115. [CrossRef]
20. Twala, B.; Cartwright, M.; Shepperd, M. Comparison of various methods for handling incomplete data in software engineering databases. In Proceedings of the IEEE 2005 International Symposium on Empirical Software Engineering, Noosa Heads, Australia, 17–18 November 2005; p. 10.
21. Gondara, L.; Wang, K. Multiple imputation using deep denoising autoencoders. *arXiv* **2017**, arXiv:1705.02737.
22. Mattei, P.A.; Frellsen, J. MIWAE: Deep generative modelling and imputation of incomplete data sets. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 4413–4423.
23. Zheng, Y. Methodologies for cross-domain data fusion: An overview. *IEEE Trans. Big Data* **2015**, *1*, 16–34. [CrossRef]
24. Lv, F.; Liang, T.; Chen, X.; Lin, G. Cross-domain semantic segmentation via domain-invariant interactive relation transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 4334–4343.
25. Manjunath, G.; Murty, M.N.; Sitaram, D. Combining heterogeneous classifiers for relational databases. *Pattern Recognit.* **2013**, *46*, 317–324. [CrossRef]
26. Sayyadian, M.; LeKhac, H.; Doan, A.; Gravano, L. Efficient keyword search across heterogeneous relational databases. In Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, 15–20 April 2006; pp. 346–355.
27. Drutsa, A.; Fedorova, V.; Ustalov, D.; Megorskaya, O.; Zerminova, E.; Baidakova, D. Crowdsourcing practice for efficient data labeling: Aggregation, incremental relabeling, and pricing. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, Portland, OR, USA, 14–19 June 2020; pp. 2623–2627.
28. Zhang, C.; Zhong, H.; Zhang, K.; Chai, C.; Wang, R.; Zhuang, X.; Bai, T.; Qiu, J.; Cao, L.; Fan, J.; et al. Harnessing Diversity for Important Data Selection in Pretraining Large Language Models. *arXiv*, **2024**, arXiv:2409.16986.
29. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
30. Narayan, A.; Chami, I.; Orr, L.; Arora, S.; Ré, C. Can foundation models wrangle your data? *arXiv* **2022**, arXiv:2205.09911. [CrossRef]
31. Wei, J.; Bosma, M.; Zhao, V.Y.; Guu, K.; Yu, A.W.; Lester, B.; Du, N.; Dai, A.M.; Le, Q.V. Finetuned language models are zero-shot learners. *arXiv* **2021**, arXiv:2109.01652.
32. Xu, D.D.; Mukherjee, S.; Liu, X.; Dey, D.; Wang, W.; Zhang, X.; Awadallah, A.; Gao, J. Few-shot task-agnostic neural architecture search for distilling large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 28644–28656.
33. Tian, P.; Li, W.; Gao, Y. Consistent meta-regularization for better meta-knowledge in few-shot learning. *IEEE Trans. Neural Networks Learn. Syst.* **2021**, *33*, 7277–7288. [CrossRef]

34. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
35. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 11809–11822.
36. Qahtan, A.A.; Elmagarmid, A.; Castro Fernandez, R.; Ouzzani, M.; Tang, N. FAHES: A robust disguised missing values detector. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2100–2109.
37. Soliman, M.A.; Ilyas, I.F.; Ben-David, S. Supporting ranking queries on uncertain and incomplete data. *VLDB J.* **2010**, *19*, 477–501. [CrossRef]
38. Berti-Équille, L.; Harmouch, H.; Naumann, F.; Novelli, N.; Thirumuruganathan, S. Discovery of genuine functional dependencies from relational data with missing values. *Proc. VLDB Endow.* **2018**, *11*, 880–892. [CrossRef]
39. Tian, Y.; Zhang, K.; Li, J.; Lin, X.; Yang, B. LSTM-based traffic flow prediction with missing data. *Neurocomputing* **2018**, *318*, 297–305. [CrossRef]
40. Song, H.; Szafir, D.A. Where's my data? evaluating visualizations with missing data. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 914–924. [CrossRef]
41. Wei, Z.; Link, S. Embedded functional dependencies and data-completeness tailored database design. *Proc. VLDB Endow.* **2019**, *12*, 1458–1470. [CrossRef]
42. Režnáková, M.; Tencer, L.; Plamondon, R.; Cheriet, M. Forgetting of unused classes in missing data environment using automatically generated data: Application to on-line handwritten gesture command recognition. *Pattern Recognit.* **2017**, *72*, 355–367. [CrossRef]
43. Zhao, B.; Wu, B.; Wu, T.; Wang, Y. Zero-shot learning posed as a missing data problem. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 2616–2622.
44. Li, Z.; Qin, L.; Cheng, H.; Zhang, X.; Zhou, X. TRIP: An interactive retrieving-inferring data imputation approach. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2550–2563. [CrossRef]
45. Song, S.; Sun, Y.; Zhang, A.; Chen, L.; Wang, J. Enriching data imputation under similarity rule constraints. *IEEE Trans. Knowl. Data Eng.* **2018**, *32*, 275–287. [CrossRef]
46. Lakshminarayan, K.; Harp, S.A.; Samad, T. Imputation of missing data in industrial databases. *Appl. Intell.* **1999**, *11*, 259–275. [CrossRef]
47. Wahl, S.; Boulesteix, A.L.; Zierer, A.; Thorand, B.; van de Wiel, M.A. Assessment of predictive performance in incomplete data by combining internal validation and multiple imputation. *BMC Med Res. Methodol.* **2016**, *16*, 144. [CrossRef]
48. Edwards, J.K.; Cole, S.R.; Troester, M.A.; Richardson, D.B. Accounting for misclassified outcomes in binary regression models using multiple imputation with internal validation data. *Am. J. Epidemiol.* **2013**, *177*, 904–912. [CrossRef] [PubMed]
49. Thiesmeier, R.; Bottai, M.; Orsini, N. Imputing missing values with external data. *arXiv* **2024**, arXiv:2410.02982.
50. Edwards, J.K.; Cole, S.R.; Fox, M.P. Flexibly accounting for exposure misclassification with external validation data. *Am. J. Epidemiol.* **2020**, *189*, 850–860. [CrossRef]
51. Salgado, C.M.; Azevedo, C.; Proença, H.; Vieira, S.M. Missing data. In *Secondary Analysis of Electronic Health Records*; Springer: Berlin/Heidelberg, Germany, 2016 ; pp. 143–162.
52. Muzellec, B.; Josse, J.; Boyer, C.; Cuturi, M. Missing data imputation using optimal transport. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 7130–7140.
53. García-Laencina, P.J.; Sancho-Gómez, J.L.; Figueiras-Vidal, A.R. Pattern classification with missing data: A review. *Neural Comput. Appl.* **2010**, *19*, 263–282. [CrossRef]
54. Stekhoven, D.J.; Bühlmann, P. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* **2012**, *28*, 112–118. [CrossRef]
55. Zhang, A.; Song, S.; Sun, Y.; Wang, J. Learning individual models for imputation. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; pp. 160–171.
56. McCoy, J.T.; Kroon, S.; Auret, L. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine* **2018**, *51*, 141–146. [CrossRef]
57. Nazabal, A.; Olmos, P.M.; Ghahramani, Z.; Valera, I. Handling incomplete heterogeneous data using vaes. *Pattern Recognit.* **2020**, *107*, 107501. [CrossRef]
58. Spinelli, I.; Scardapane, S.; Uncini, A. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Netw.* **2020**, *129*, 249–260. [CrossRef]
59. Yoon, J.; Jordon, J.; Schaar, M. Gain: Missing data imputation using generative adversarial nets. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5689–5698.
60. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.

61. Royston, P.; White, I.R. Multiple imputation by chained equations (MICE): Implementation in Stata. *J. Stat. Softw.* **2011**, *45*, 1–20. [CrossRef]
62. Mazumder, R.; Hastie, T.; Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.* **2010**, *11*, 2287–2322.
63. Lee, D.; Seung, H.S. Algorithms for non-negative matrix factorization. *Adv. Neural Inf. Process. Syst.* **2000**, *13* .
64. Josse, J.; Pagès, J.; Husson, F. Multiple imputation in principal component analysis. *Adv. Data Anal. Classif.* **2011**, *5*, 231–246. [CrossRef]
65. Doersch, C. Tutorial on variational autoencoders. *arXiv* **2016**, arXiv:1606.05908.
66. Germain, M.; Gregor, K.; Murray, I.; Larochelle, H. Made: Masked autoencoder for distribution estimation. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 881–889.
67. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 214–223.
68. Chen, J.; Wu, Y.; Jia, C.; Zheng, H.; Huang, G. Customizable text generation via conditional text generative adversarial network. *Neurocomputing* **2020**, *416*, 125–135. [CrossRef]
69. Mei, Y.; Song, S.; Fang, C.; Yang, H.; Fang, J.; Long, J. Capturing semantics for imputation with pre-trained language models. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 61–72.
70. Chen, Y.; Wang, X.; Xu, G. Gatgpt: A pre-trained large language model with graph attention network for spatiotemporal imputation. *arXiv* **2023**, arXiv:2311.14332.
71. Ding, Z.; Tian, J.; Wang, Z.; Zhao, J.; Li, S. Data imputation using large language model to accelerate recommendation system. *arXiv* **2024**, arXiv:2407.10078.
72. Radford, A. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://www.mikecaptain.com/resources/pdf/GPT-1.pdf (accessed on 10 June 2025).
73. Ruder, S.; Peters, M.E.; Swayamdipta, S.; Wolf, T. Transfer learning in natural language processing. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, Minneapolis, MN, USA, 2–7 June 2019 ; pp. 15–18.
74. Luo, G.; Han, Y.T.; Mou, L.; Firdaus, M. Prompt-based editing for text style transfer. *arXiv* **2023**, arXiv:2301.11997.
75. Izacard, G.; Lewis, P.; Lomeli, M.; Hosseini, L.; Petroni, F.; Schick, T.; Dwivedi-Yu, J.; Joulin, A.; Riedel, S.; Grave, E. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.* **2023**, *24*, 1–43.
76. Wang, Z.; Sun, J. Transtab: Learning transferable tabular transformers across tables. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 2902–2915.
77. Yang, C.; Luo, Y.; Cui, C.; Fan, J.; Chai, C.; Tang, N. Retrieval Augmented Imputation Using Data Lake Tables. Available online: https://openreview.net/forum?id=EyW92b6DyY (accessed on 10 June 2025).
78. Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; Kasneci, G. Deep neural networks and tabular data: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 7499–7519. [CrossRef]
79. Carballo, K.V.; Na, L.; Ma, Y.; Boussioux, L.; Zeng, C.; Soenksen, L.R.; Bertsimas, D. TabText: A Flexible and Contextual Approach to Tabular Data Representation. *arXiv* **2022**, arXiv:2206.10381.
80. Ren, W.; Zhao, T.; Huang, Y.; Honavar, V. Deep Learning within Tabular Data: Foundations, Challenges, Advances and Future Directions. *arXiv* **2025**, arXiv:2501.03540.
81. Yi, Z.; Ouyang, J.; Liu, Y.; Liao, T.; Xu, Z.; Shen, Y. A Survey on Recent Advances in LLM-Based Multi-turn Dialogue Systems. *arXiv* **2024**, arXiv:2402.18013.
82. Bai, G.; Liu, J.; Bu, X.; He, Y.; Liu, J.; Zhou, Z.; Lin, Z.; Su, W.; Ge, T.; Zheng, B.; et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv* **2024**, arXiv:2402.14762.
83. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv* **2023**, arXiv:2312.10997.
84. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
85. Wu, Y.; Zhao, Y.; Hu, B.; Minervini, P.; Stenetorp, P.; Riedel, S. An efficient memory-augmented transformer for knowledge-intensive nlp tasks. *arXiv* **2022**, arXiv:2210.16773.
86. Jiang, Z.; Xu, F.F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; Neubig, G. Active retrieval augmented generation. *arXiv* **2023**, arXiv:2305.06983.
87. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M. Retrieval augmented language model pre-training. In Proceedings of the International Conference on Machine Learning. Virtual Event, 13–18 July 2020; pp. 3929–3938.
88. Zhuang, X.; Peng, J.; Ma, R.; Wang, Y.; Bai, T.; Wei, X.; Qiu, J.; Zhang, C.; Qian, Y.; He, C. Meta-rater: A Multi-dimensional Data Selection Method for Pre-training Language Models. *arXiv* **2025**, arXiv:2310.09263.

89. Ram, O.; Levine, Y.; Dalmedigos, I.; Muhlgay, D.; Shashua, A.; Leyton-Brown, K.; Shoham, Y. In-context retrieval-augmented language models. *Trans. Assoc. Comput. Linguist.* **2023**, *11*, 1316–1331. [CrossRef]
90. Li, P.; He, Y.; Yashar, D.; Cui, W.; Ge, S.; Zhang, H.; Fainman, D.R.; Zhang, D.; Chaudhuri, S. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv* **2023**, arXiv:2310.09263. [CrossRef]
91. Li, P.; He, Y.; Yashar, D.; Cui, W.; Ge, S.; Zhang, H.; Rifinski Fainman, D.; Zhang, D.; Chaudhuri, S. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. *Proc. ACM Manag. Data* **2024**, *2*, 1–28. [CrossRef]
92. Zhang, D.; Yin, C.; Zeng, J.; Yuan, X.; Zhang, P. Combining structured and unstructured data for predictive models: A deep learning approach. *BMC Med. Inform. Decis. Mak.* **2020**, *20*, 280. [CrossRef]
93. Alessandro, M.D.; Calabrés, E.; Elkano, M. A Modular End-to-End Multimodal Learning Method for Structured and Unstructured Data. *arXiv* **2024**, arXiv:2403.04866.
94. Bai, T.; Yang, L.; Wong, Z.H.; Peng, J.; Zhuang, X.; Zhang, C.; Wu, L.; Qiu, J.; Zhang, W.; Yuan, B.; et al. Multi-Agent Collaborative Data Selection for Efficient LLM Pretraining. *arXiv* **2024**, arXiv:2410.08102.
95. Ebrahimi, S.; Arik, S.O.; Dong, Y.; Pfister, T. Lanistr: Multimodal learning from structured and unstructured data. *arXiv* **2023**, arXiv:2305.16556.
96. Yang, S.; Zhang, R.; Erfani, S.M.; Lau, J.H. UniMF: A Unified Framework to Incorporate Multimodal Knowledge Bases intoEnd-to-End Task-Oriented Dialogue Systems. In Proceedings of the IJCAI, Montreal, QC, Canada, 19–27 August 2021; pp. 3978–3984.
97. Santos, M.S.; Abreu, P.H.; Fernández, A.; Luengo, J.; Santos, J. The impact of heterogeneous distance functions on missing data imputation and classification performance. *Eng. Appl. Artif. Intell.* **2022**, *111*, 104791. [CrossRef]
98. Guțu, B.M.; Popescu, N. Exploring Data Analysis Methods in Generative Models: From Fine-Tuning to RAG Implementation. *Computers* **2024**, *13*, 327. [CrossRef]
99. Naeem, Z.A.; Ahmad, M.S.; Eltabakh, M.; Ouzzani, M.; Tang, N. RetClean: Retrieval-Based Data Cleaning Using Foundation Models and Data Lakes. *arXiv* **2024**, arXiv:2303.16909.
100. Deng, X.; Sun, H.; Lees, A.; Wu, Y.; Yu, C. TURL: Table Understanding through Representation Learning. *arXiv* **2020**, arXiv:2006.14806. [CrossRef]
101. City of Chicago. Chicago Data Portal. 2023. Available online: https://data.cityofchicago.org/Public-Safety/Crimes-2023/xguy-4ndq/about_data (accessed on 30 June 2023).
102. Robertson, S.; Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends® Inf. Retr.* **2009**, *3*, 333–389. [CrossRef]
103. Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
104. Nogueira, R.; Yang, W.; Cho, K.; Lin, J. Multi-stage document ranking with BERT. *arXiv* **2019**, arXiv:1910.14424.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.