UniDM: A Unified Framework for Data Manipulation with Large Language Models [Scalable Data Science]

Yichen Qian[1], Yongyi He[1,2], Rong Zhu[1], Jintao Huang[1,2], Zhijian Ma[1], Haibin Wang[1], Yaohua Wang[1], Xiuyu Sun[1], Defu Lian[2], Bolin Ding[1], Jingren Zhou[1]
[1]Alibaba Group, [2]University of Science and Technology of China
{yichen.qyc, yongyi.hyy, red.zr, huangjintao.hjt, zhijian.mzj, binke.whb, xiachen.wyh, xiuyu.sxy, bolin.ding, jingren.zhou}@alibaba-inc.com, liandefu@ustc.edu.cn

ABSTRACT

Data lakes with a massive collection of files provide a friendly interface for users to manage data. Designing effective data manipulation methods, such as data discovery, extraction, cleaning and integration, is a long standing problem in data lakes. Traditional methods, which rely on rules or machine learning models, require extensive human efforts on training data collection and tuning models. Recent methods apply Large Language Models (LLMs) as a universal knowledge base to automatically retrieve relevant information to resolve multiple data manipulation tasks. They exhibit bright benefits in terms of performance but still require customized designs to fit each specific task. This is very costly and can not catch up with the requirements of big data lake platforms.

In this paper, inspired by the cross-task generality of LLMs on NLP tasks, we pave the *first* step to design an *automatic* and *general* solution to tackle with data manipulation tasks. We propose `UniDM`, a unified framework which establishes a new paradigm to process data manipulation tasks using LLMs. `UniDM` formalizes a number of data manipulation tasks in a unified form and abstracts three main general steps to solve each task. For each step, we design effective prompts to retrieve relevant context information and guide LLMs to produce high quality results. By our comprehensive evaluation on a variety of benchmarks, our `UniDM` exhibits great generality and state-of-the-art performance on lots of data manipulation tasks, including but not limited to data imputation, data transformation, error detection and entity resolution. Meanwhile, it also indicates the potential to be applied on more complex data and tasks.

1 INTRODUCTION

1.1 Background and Motivation

Data lake is a general system to store vast amounts of data with heterogeneous schemas and structures. They provide an efficient interface that allows users to manage and manipulate data with various kinds of tools. Users could flexibly define different workflows to clean, integrate, interpret and analyze data according to their applications [41, 42]. This advantage facilitates users' customized demands on data processing, but also brings remarkable shortcomings. For any new application, the corresponding data processing workflow needs to be redesigned and tuned by experts from scratch. This is very costly and can not catch up with the new applications which may occur every day in big data lake platforms [25].

Literature works have devoted considerable research efforts to designing *automatic* and *general* methods that are applicable to different data manipulation tasks in data lakes. Traditional rule-based methods [13, 14, 17, 18, 28, 36, 48] require specialized model construction and rule tuning for each data task, which are not automatic enough. Recent works apply machine learning [5, 8, 9, 26, 30–32, 57], especially deep learning techniques [19, 22, 39, 61], to learn the adaptive solution for each task. However, their performance heavily relies on the quality of the trained models, which require a large amount of labeled training data and specific domain knowledge related to each task.

Large Language Models and Prompts. In recent time, Large Language Models (LLMs), such as BERT [20], GPT-3 [10], and LaMDA [50], have shown incredible performance on a broad set of downstream tasks [34, 62]. LLMs are typically deep neural networks with transformer architecture [53]. They are pre-trained on enormous corpora of text to learn universal world knowledge.

LLMs could be regarded as a very large knowledge base applicable to numerous tasks, particularly for tasks requiring to interpret the semantics of data. Instead of fine-tuning the model to fit each task, we can simply apply *prompts* to guide LLMs to solve each task. Specifically, a prompt is an intuitive interface written in natural text to interact with LLMs. It can take various forms (e.g., phrases or complex sentences) to guide or ask the LLMs to extract their knowledge to perform lots of different jobs, such as code generation, question answering, creative writing, etc. For example, a simple prompt, such as "Translate English to French: hello =>", could directly do the language translation.

The performance of the LLMs is very sensitive to the prompt [10]. To obtain high quality results, we often design prompts with context information to provide more instructions to LLMs. The context information could be a few input/output examples or other information relevant to the task. When combined with the task description, LLMs have more background to extract more accurate knowledge to answer the questions. For example, for the prompt "Fill in the value like Genre: Folk; Artist: Bob Dylan. Genre: Jazz; Artist: ?", the LLM would imitate the example to find a jazz artist, e.g., "Bill Evans", as a result.

LLMs for Tasks on Data Lakes. Some very recent works [11, 12, 33, 37, 38, 40, 51, 52, 54] observe the potential benefits of bringing LLMs into some data manipulation tasks. For example, we could ask LLMs to automatically judge whether a value is valid for an attribute using its intrinsic knowledge instead of designing numerous error detection rules for each domain. Prior works [27, 35, 44, 51] have verified the effectiveness of applying LLMs to answer questions on data tables. Later, LLMs have been applied on data pre-processing tasks [33], binary classification on tables [49], data cleaning and

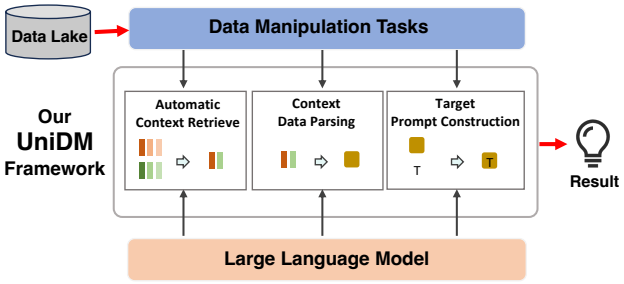


Figure 1: An overview of our UniDM framework.

integration tasks[38, 40]. These works provide enough evidence to exhibit that the LLM-based methods could attain very promising, sometimes state-of-the-art, performance on these tasks.

However, even avoiding human efforts on providing domain knowledge, current LLM-based methods are not generally applicable to data manipulation tasks. The procedures, along with the prompts, of these methods are all specifically designed for each task, which require users to manually extract customized context information to guide the LLMs. The benefits of the LLMs and the shortcomings of existing methods motivate us to ask the following question:

Could we find a unified solution with LLMs such that it is both general and automatic to different data manipulation tasks on data lakes requiring no manual efforts?

1.2 Challenges and Our Contributions

On NLP tasks, LLMs have shown remarkable cross-task generality. This is because the NLP field has accumulated decades of experience in designing a standard paradigm to unify and solve different NLP tasks [10, 45]. Whereas, for data manipulation tasks, the relevant experience is almost blank, which makes this problem to be extensively challenging. To resolve it, we need to answer the following two key questions:

1) *How to design a framework to elegantly unify different data manipulation tasks?* This framework should be general enough to subsume common and new tasks occurred in data lake applications and easy to bring LLMs into the solution.

2) *How to design a general solution under this unified framework?* This solution should contain abstract procedures that are adaptive to different tasks and at the same time, maximize the effectiveness of LLMs.

Our Contributions. In this paper, we pave the first step towards resolving this problem. We propose UniDM, a unified solution that is verified to attain state-of-the-art performance on a variety of data manipulation tasks on data lakes. Specifically, we make the following contributions:

1) **We propose a unified framework to describe data manipulation tasks.** We formalize a data manipulation task T as a function $F_T()$ to tackle with some records R and attributes S on a data table D in the data lake. We show that this framework subsumes a number of commonly occurred tasks on structured data and can be easily extended to new and complex tasks even on unstructured data. (in Section 2)

2) **We abstract the general procedures to solve different tasks using LLMs.** We observe that, the key to solving a data manipulation task is to find a proper prompt to inspire the LLMs to produce accurate results [54]. However, due to the complexity of our tasks, it is difficult and ineffective to directly ask the LLMs for final results by a singleton prompt [40]. As a result, we decompose a data manipulation task (that could be described by our unified framework) into several consistent steps such that each step is a simple, direct and easy job for LLMs.

As illustrated in Figure 1, our solution contains three main steps. The first step automatically extracts relevant context information from data table to serve as demonstrations or background knowledge to solve the task. The second step transforms the context information from tabular form to logic text so the LLMs can more easily capture its semantics. Finally, the third step applies prompt engineering to construct the target prompt to obtain the final results. In such a way, we attain generality across different tasks and effectiveness by LLMs. (in Section 3)

3) **We design effective (templates of) prompts for each main step in our solution.** For each main step, we abstract the knowledge that needs to be acquired from LLMs and design a general template of prompt to automatically extract such knowledge. In such a way, LLMs could do well in each step and improve the quality of the final results. (in Section 3)

4) **We conduct extensive experiments to evaluate the performance of our solution UniDM.** The evaluation results on lots of benchmarks exhibit that UniDM attains state-of-the-art results on a variety of data manipulation tasks, including data imputation, data transformation, error detection and entity resolution. Meanwhile, the effectiveness of each main step in UniDM is also verified by ablation study. (in Section 4)

5) **We point out a number of future research directions.** Our work takes the first step on applying LLMs to design general solutions for data manipulation tasks on data lakes. There still reserves much room for improvement upon our work, including the extension to more types of data (e.g., domain-specific data, unstructured or semi-structured data), the new paradigm of large models tailored to database tasks and new methods with better efficiency and more friendly to system deployment. (in Section 5)

Organizations. Section 2 formalizes our target data manipulation tasks into a unified framework. Section 3 presents the technical details of our general solution UniDM. Section 4 reports the evaluation results. Section 5 discusses the possible future directions. Finally, Section 6 concludes this paper.

2 PROBLEM DEFINITION

In this section, we propose a unified framework to formalize the data manipulation tasks we target to solve on data lakes. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ be a data lake. In this paper, we assume that each element $D_i \in \mathcal{D}$ is a relational data table containing a number of records (tuples). We denote the schema, as well as the set of attributes, of table D_i as S_i. Unlike with the relational database, the join relations are not specified for tables in the data lake $\mathcal{D}$. For any record r and attribute s, we denote the value of r on s as $r[s]$.

Let T represent a data manipulation (e.g., data cleaning or integration) task on $\mathcal{D}$. We assume that the task description and the parameters of the task (e.g., a question on the table) are all encoded

in T. We could formalize a number of different tasks in a unified manner as follows:

Input: a data lake $\mathcal{D}$, a subset of records $R \subseteq D_i$ extracted from a table $D_i \in \mathcal{D}$, a subset of attributes $S \subseteq S_i$ under the schema S_i and a target task T;

Output: we have a function F_T related to the task T that produces a value $Y = F_T(R, S, \mathcal{D})$.

For each different data manipulation task T, the function F_T is defined in different ways according to the applications. We list a number of example tasks that are commonly used in real-world applications and can be subsumed by our framework as follows:

- **Data Imputation**: S contains an attribute in S_i and R contains a singleton record in D_i having missing value on attribute S, $F_T(R, S, \mathcal{D})$ outputs the desired missing value of $R[S]$.
- **Data Transformation**: S contains an attribute in S_i and R contains a singleton record in D_i, $F_T(R, S, \mathcal{D})$ transforms the original value $R[S]$ to another new value $R'[S]$ by user specified rules.
- **Error Detection**: S contains an attribute in S_i and R contains a singleton record in D_i, $F_T(R, S, \mathcal{D})$ predicts whether the value $R[S]$ is normal or not.
- **Entity Resolution**: S contains a number of attributes describing the property of each record in D_i and $R = \{r_1, r_2\}$ contains two different records in D_i, $F_T(R, S, \mathcal{D})$ outputs whether the two records r_1 and r_2 refer to the same real-world entity or not.
- **Table Question Answering**: $S = S_i$ and $R = D_i$ contain all attributes and records in D_i, $F_T(R, S, \mathcal{D})$ outputs the result of a question, encoded as natural language text in T, on R and S. A detailed explanation is given in Appendix A in the full version [3].
- **Join Discovery**: $S = S_i \cup S_j$ contains all attributes in two tables and $R = D_i \cup D_j$ contains all records in two tables, $F_T(R, S, \mathcal{D})$ outputs all pair of attributes where $s \in S_i, s' \in S_j$ that could be joined with each other. A detailed example is given in Appendix B [3].

Notably, in our proposed framework, we just consider the very fundamental form for the data manipulation tasks in data lakes. In the following content, we apply the above data manipulation asks subsumed by our framework to showcase how to design a general solution using LLMs. Whereas, our framework could be easily extended with rich definitions to support new tasks on unstructured or semi-structured data. Due to space limits, we defer the details of the extensions into Appendix C [3].

3 OUR GENERAL SOLUTION

We present the details of our method on applying LLMs to solve the data manipulation tasks on data lakes. We analyze this problem and outline our solution in Section 3.1. Then, Section 3.2 to Section 3.4 elaborate on the details of each key technique in our method. Finally, the generality of our method is discussed in Section 3.5.

3.1 Overview

Problem Analysis. Recall that, we could consult the LLMs using a *prompt* to acquire the desired knowledge. The data manipulation tasks could also be solved using proper prompts. As shown in the following example, we apply two simple prompts to resolve the data imputation and transformation tasks. Here each prompt is a textual template in natural language containing: 1) the task description (marked in red) so the LLMs could understand what to be done, e.g., filling in the missing value or transforming the data; 2) the context information (marked in blue) serving as the demonstrations or examples to guide the LLMs to perform the task, e.g., information of other relevant attributes or examples of the transformation; and 3) the placeholder (marked in orange) to provide the input of the task.

Prompt A:
Data Imputation: city:Copenhagen,country:Denmark, timezone:?
Prompt B:
Data Transformation: 20210315 to Mar 15 2021, 20201103 to ?

It has been shown that the quality of the LLM results is very sensitive to the format of the prompt [10], i.e., the words, symbols and order of the text would all affect whether the LLMs could accurately capture the semantics in the prompt. Simple and straightforward prompts (such as the above two examples) would often have mediocre performance on data manipulation tasks. This is because LLMs are often trained on the corpus to reason about direct and simple events, e.g. "x is contained in y" or "y is the same as z", but may fail on complex multi-hop reasoning problems that they have no explicit evidence [16], e.g. "Is x contained in z?". The process of solving a data manipulation task is too complicated for LLMs. It requires LLMs to interpret the task description, to extract (and possibly transform) the context information and to fill in a proper value in the placeholder at the same time.

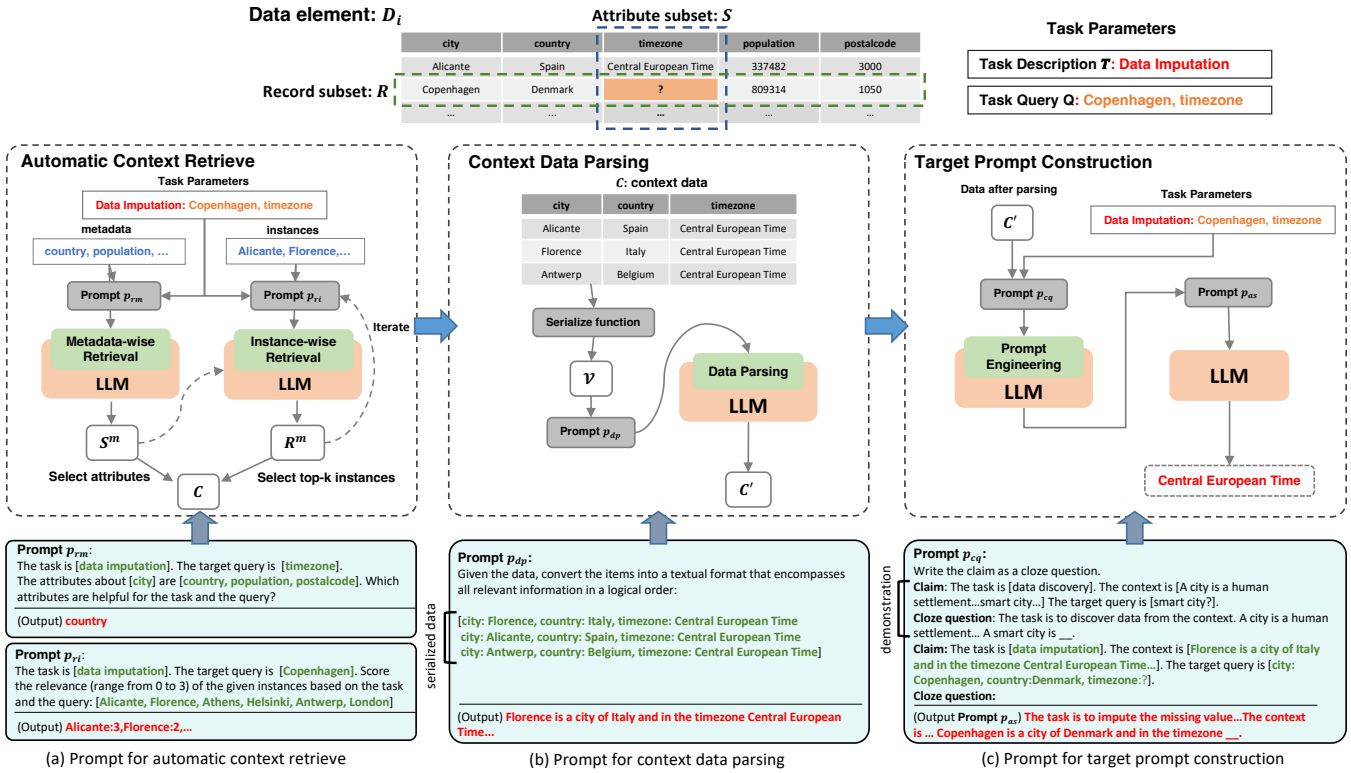
Some literature works [12, 54] have devoted the efforts on designing and tuning prompts for some specific data manipulation tasks. Whereas, this is very expensive and not extensible to numerous customized tasks occurring every day. What we actually need is a general solution that is applicable to produce effective prompts for each different data manipulation task.

Our Main Idea and Solution. To attain this goal, we decompose a data manipulation task (that could be described by our unified framework in Section 2) into several consistent steps. Each step is a simple, direct and easy task to reason about for LLMs using its evidence and logic. Based on this, for each step, we abstract the knowledge that needs to be acquired from LLMs and design a general template of prompt to extract such knowledge. In such a way, LLMs could do well in each step and we improve the quality of the final results. Meanwhile, we attain generality across different tasks.

The pipeline and architecture of our solution, namely `UniDM`, are illustrated in Figure 2. Generally, `UniDM` proceeds a data manipulation task in three main steps:

1) **Context Retrieval**: Given the task T, the records $R \subseteq D_i$ and the attributes $S \subseteq S_i$ on a data lake $\mathcal{D}$, at the very beginning, we need to extract the relevant contextual information C from $\mathcal{D}$ to resolve T. C may contain additional information from other records and attributes to guide the LLMs to capture the semantics for task T. As shown in [29], high quality context information plays a crucial role in improving the performance of LLMs. To this end, we design two templates of prompts to automatically retrieve context information from $\mathcal{D}$ using LLMs. The first prompt p_{rm} aims at

Yichen Qian[1], Yongyi He[1,2], Rong Zhu[1], Jintao Huang[1,2], Zhijian Ma[1], Haibin Wang[1], Yaohua Wang[1], Xiuyu Sun[1], Defu Lian[2], Bolin Ding[1], Jingren Zhou[1]



(a) Prompt for automatic context retrieve (b) Prompt for context data parsing (c) Prompt for target prompt construction

Figure 2: The pipeline of UniDM by taking data imputation task as the example.

retrieving meta-wise information, e.g., a number of attributes that may provide useful knowledge to task T on attributes in S. Based on its results, the second prompt p_{ri} identifies the most helpful records related to R to resolve the task in a more fine-grained manner. After that, we obtain the context information C in a tabular form.

2) **Context Parsing**: The raw context information in C in tabular form is often not friendly to be understood by the LLMs (as LLMs are mainly trained to interpret the natural language text). Therefore, the next step is to transform the original context C into another form that is more easily to be interpreted by the LLMs. Similar to previous works [40], we first apply a serialize function to transform C into a regular text $\mathcal{V}$ with pairs of attributes and values, e.g., "city:Florence, country:Italy". Then, we design a prompt template p_{dp} to further convert the text $\mathcal{V}$ into the natural text C' reflecting the logic relations among different attributes, e.g. "Florence is a city of Italy". C' is more smooth and close to the natural language, so the LLMs could find more relevant information in its corpus for downstream procedures.

3) **Target Prompt Construction**: Finally, we combine the serialized text C', the description of the task T and the task input R and S together to consult the LLMs to obtain the final result Y. We utilize prompt engineering to automatically elicit prompts for any data manipulation task. Specifically, all data manipulation tasks described by our unified framework in Section 2 could be equivalently transformed into a cloze question. Cloze question is friendly to LLMs as it is written in natural language with placeholders. To automatically generate a proper cloze question, we design a template of prompt p_{cq} that provides the LLMs a small set of demonstrations, where each one is a pair of a data manipulation task and its corresponding cloze question. These demonstrations include both task-specific and task-agnostic examples, where LLMs could learn to identify the most suitable template for any task to output a cloze question p_{as}. The target prompt p_{as} is fed into LLMs to obtain the final result Y.

In the following content, we describe the details of each component in our solution. For ease of presentation, we use the data imputation task as an example to introduce the technical details. We discuss how our solution generalizes to other data manipulation tasks in Section 3.5.

3.2 Automatic Context Retrieve

We require that the context retrieve component could identify useful information while filtering irrelevant data to facilitate the LLMs. Previous works [8, 40] ask users to specify the instances (records) and attributes relevant to the task or learn to identify useful records based on the similarity of attribute values [37, 38]. Unlike with them, we design a purely automatic strategy for extracting helpful attributes and records with the aid of the LLMs. On a high level, our strategy consists of two steps. The first step, i.e., meta-wise retrieval, finds relevant attributes from the holistic view. Then, the second step, i.e., instance-wise retrieval, extracts useful records in a more detailed manner.

Meta-wise Retrieval. In the first step, we provide a number of candidate attributes S' and ask LLMs to select valuable ones for our task T and target attribute S. LLMs could apply the inherent

knowledge to measure the relationships between S' and S and reserve only promising attributes. This step could filter irrelevant information in the data lake $\mathcal{D}$ in a coarse-grained manner. The results would contain helpful meta-wise information describing the high level domain knowledge of these attributes. Specifically, we construct the prompt p_{rm} using the following template:

Prompt p_{rm}:
The task is $[T]$. *The target query is* $[Q]$. *The candidate attributes are* $[s_1, s_2, ..., s_n]$. *Which attributes are helpful for the task and the query?*

Here T is the task description such as "data imputation". The query Q combines our inputs on the target record R and attribute S for task T. It has different forms in different tasks. In data imputation, we represent it as "the primary key of record R, the attribute S", e.g., "Copenhagen, timezone", to indicate that we want to fill the value of S for record R. The set of candidate attributes $S' = S_i - S = \{s_1, s_2, \ldots, s_n\}$ contains all remaining attributes in S_i of table D_i. For other tasks, the forms of the query Q and set S' are different. We reserve the details in Section 3.5. In this paper, we do not apply cross-table attributes in table $D_{j \neq i} \in \mathcal{D}$ as the experimental results on D_i (shown in Section 4) are competitive enough.

We denote S^m as the attributes returned by the LLMs. In our example in Figure 2, for the target attribute "timezone", we obtain the attribute "country" to fill in its value.

Instance-wise Retrieval. Next, we perform fine-grained filtering on all records in $R' = D_i - R = \{r_1, r_2, \ldots, r_m\}$ to identify relevant ones w.r.t. target records in R. This requires to examine the relevance between R' and R. The *relevance* can be interpreted in different ways for different tasks. For example, for data imputation, we hope to find records similar to target records R and attributes S to find the missing value. For the error detection task, we may want to obtain records reflecting the distribution of the domain value to identify whether the target value $R[S]$ is abnormal. We still drive LLMs by the following prompt p_{ri} to consult the knowledge base to automatically measure the relevance scores as:

Prompt p_{ri}:
The task is $[T]$. *The target query is* $[Q]$. *To score the relevance (range from 0 to 3) of given instances based on the task and the query:* $\{r_1[S^m], r_2[S^m], ..., r_m[S^m]\}$

Here for each $r_j \in R'$, we only reserve the attributes in S^m as the other ones are identified to be not helpful to our task. After examining all records in R' (or touching a time limit), we order all instances according to the relevance score and select the top-k instances R^m as the context C for the downstream procedures. This makes our context information more compact.

3.3 Context Data Parsing

The context information C, represented in a tabular form, is not friendly for the LLMs to interpret its underlying semantic, since the LLMs are often trained on large corpus of text. To resolve this problem, we consider how to transform C into a more effective format for LLMs. As all records R^m in C are all organized in a regular structure under the schema S^m, we could easily serialize C into a textual string. Specifically, let $\{(s, r[s]) | r \in R^m, s \in S^m\}$ denote the set of all pairs of each attribute s and its value in record r. The information of C is losslessly encoded. Our $serialize()$ function directly concatenates all pairs to produce a text $\mathcal{V}$.

Previous works [40] directly feed the text $\mathcal{V}$ into LLMs to serve as the contextual information for our task. Whereas, we further try to integrate the pairs in $\mathcal{V}$ into a logic text C' reflecting the relations among different attributes. For example, in Figure 2, the text "country:Italy, timezone: Central European Time" is converted into "The country Italy is in the Central European Time timezone". Obviously, the former one rarely occurs in any article except some tables while the latter one may frequently occur in some scientific articles in the training corpus. Therefore, providing C' rather than $\mathcal{V}$ to LLMs could improve its probability of hitting relevant texts in inference and produce more accurate results.

Notably, converting the text $\mathcal{V}$ to C' is an easy job for LLMs. The logic relations among different attributes are often common and fixed, e.g. "a city is in a country in a timezone", so the LLMs could directly capture such knowledge. In our solution, we apply the following data parsing template prompt p_{dp} to perform this job. The generated context representation C' is applied in the subsequent procedures.

Prompt p_{dp}: *Given the data, convert the items into a textual format that encompasses all relevant information in a logical order:* $[\mathcal{V}]$

3.4 Effective Target Prompt Construction

To apply LLMs for our task, the ultimate (as well as the most important) step is to find an effective prompt to organize the task description T, the context information C' in logic text and the query Q encoding the input records R and attributes S (defined in Section 3.2) together. Moreover, we hope that the method to find the prompt is generally applicable to different tasks to avoid exhaustive tuning efforts.

We observe that, all of our tasks described by the claims (containing T, C', R and S as stated above) could be equivalently summarized as a cloze question. For example, the data imputation task is to fill the missing value of $R[S]$ and the error detection task is to fill a normal or abnormal answer for the value $R[S]$. Cloze question is friendly to LLMs as it is written in natural language with placeholders. Therefore, our problem is how to automatically organize our claims for different tasks into a proper cloze question.

Certainly, it requires `UniDM` to capture the semantics of each element in our claims, e.g., which element should be placed in front of others, and organize them in a smooth natural text. In similar to context data parsing, this job is suitable to be done by the LLMs themselves. We apply the following prompt p_{cq} to do this transformation:

Prompt p_{cq}:
Write the claim as a cloze question.
***Claim**: xxx...*
***Cloze question**: xxx__ ....*
.....
***Claim**: The task is* $[T]$. *The context is* $[C']$. *The target query is* $[Q]$.
***Cloze question**:*

Motivated by the discrete prompt search methods [6, 24], we provide a number of claims and their corresponding cloze questions as demonstration examples in p_{cq}. The pairs of claim and cloze question include: 1) examples pertaining to our commonly applied tasks (such as data imputation and error detection) that are verified to produce accurate results; and 2) some task-agnostic transformation strategies which are verified to be generally applicable to different tasks. LLMs could learn from these examples to identify the most suitable template to transform our claims on a task to a cloze question p_{as}. In such a way, we attain both high effectiveness (on common tasks) and cross-task generality (on new and unseen tasks) in generating prompts. We only need to maintain the demonstration examples according to the applications in periodical while avoiding specialized prompt design for each upcoming task.

Finally, we feed the prompt p_{as} into LLMs to yield the final answer of our task. The experimental results in Section 4 indicate that the prompts generated by our method are very effective on a variety of data manipulation tasks.

3.5 Generalization to More Tasks

Our UniDM framework could be easily generalized to other tasks listed in Section 2 by minor adaptions to the form of the query Q, which encodes the target records R and attributes S, and the set S' of candidate relevant attributes in prompt p_{rm} (defined in Section 3.2). The details are listed as follows.

For the data transformation task, we directly set $Q = R[S]$ to give the attribute value to be transformed. For error detection, we represent it as "S: $R[S]$?" to indicate whether $R[S]$ is a valid value for S. For entity resolution where $R = \{r_1, r_2\}$, we set Q to be "Entity A is r_1, Entity B is r_2" to identify whether r_1 and r_2 refer to the same entity.

For table question answering, we directly set the query Q to be the question contained in the task description, e.g., "how many gold medals did Australia and Switzerland total?". For entity resolution and table question answering, as we are concerned with finding useful information among all attributes, in the prompt p_{rm} for retrieval meta-wise information, we set $S' = S$. We present an example in Appendix A [3] to show that UniDM could not only process instance-level tasks such as data imputation and error detection, but also good at retrieving table-level information in table question answering task.

For join discovery task, we set Q to be the table name of D_i and D_j and $S' = S_i \cup S_j$. We obtain the contextual information $C = C_i \cup C_j$ where C_i and C_j are extracted from table D_i and D_j, respectively. We present an example in Appendix B [3] to show how to apply UniDM to solve this task.

For other tasks that could be subsumed by our framework defined in Section 2, we could also adjust the parameters in all prompts according to the semantics of the task. We present an example on information retrieval task in Appendix C [3].

4 EXPERIMENT

We conduct extensive experiments to evaluate the performance of our UniDM, whose implementation is publicly available in [4]. The experimental settings are introduced in Section 4.1. The experimental results mainly answer two crucial questions: 1) the performance of our UniDM solution on different data manipulation tasks (in Section 4.2); and 2) the importance of each component in our UniDM solution (in Section 4.3).

4.1 Experimental Details

Baselines and Evaluation Tasks. We implement our UniDM using the GPT-3-175B parameter model [10](text-davinci-003) in the OpenAI API [2] as the LLM without fine-tuning. In addition, we give fine-tuning results for an open-source LLM GPT-J-6B [7]. In our method, in the default setting, we apply the automatic context retrieval method proposed in Section 3.2. In detail, we extract one attribute from the candidate set in the metadata-wise retrieval (see prompt p_{rm} in Section 3.2) and top-3 records from 50 records randomly sampled in the dataset in the instance-wise retrieval (see prompt p_{ri} in Section 3.2). We evaluate UniDM on a number of different data manipulation tasks including data imputation, data transformation, error detection and entity resolution. Further results on more complex tasks (e.g., join discovery and information extraction) are provided in the Appendix B and C [3].

Besides, we compare UniDM with a variety of state-of-the-art (SOTA) methods on data manipulation tasks. The **FM** [40] method is shown to attain SOTA performance on multiple data manipulation tasks with simple prompt learning on LLMs. We implement FM following the original paper and its open-source code [1]. In its default setting, the context information and the target prompt are manually selected by guiding rules and it only applies serialization in context data parsing. We evaluate FM on data imputation, data transformation, error detection and entity resolution task.

For data imputation task, we select several methods following different technical routines, including a statistics-based method **HoloClean** [46, 56], a clustering-based method **CMI** [47] and a deep learning based method **IMP** [37]. For data transformation task, we select a search-based method **TDE** [59]. For error detection task, we select two machine learning based methods **HoloClean** [46] and **HoloDetect** [26]. For entity resolution task, we use a deep learning method **Ditto** [33].

Following previous works, we employ widely-used metrics, *accuracy* and *F1-score* to evaluate the effectiveness of these methods. For data imputation and data transformation, we use *accuracy*. For error detection and entity resolution, we use *F1-score*.

Datasets. We evaluate the performance of baselines on different benchmark datasets. For data imputation, we choose two challenging benchmark datasets, namely Restaurants and Buy, from [37]. For data transformation, we follow the TDE benchmark in [59]. For error detection task, we choose the benchmark Hospital dataset widely used in data cleaning papers [26, 46]. For entity resolution, we follow the standard Magellan benchmark in [30].

4.2 Performance Evaluation

Data Imputation. As shown in Table 1, we conduct experiments to compare UniDM with other methods. Here we also compare the performance of UniDM and FM with another setting, where the context information of records is randomly selected from the table. We find that:

Table 1: Accuracy on data imputation task with SOTA.

Method	Data Imputation Accuracy (%)	
	Restaurant	Buy
HoloClean	33.1	16.2
CMI	56.0	65.3
IMP	77.2	96.5
FM (random)	81.4	86.2
FM (manual)	88.4	98.5
UniDM (random)	87.2	92.3
UniDM	93.0	98.5

Table 2: Accuracy on data transformation task with SOTA.

Method	Data Transformation Accuracy (%)	
	StackOverflow	Bing-QueryLogs
TDE	63.0	32.0
FM	65.3	54.0
UniDM	67.4	56.0

Table 3: F1 score on error detection task with SOTA.

Method	Error Detection F1 Score (%)
	Hospital
HoloClean	51.4
HoloDetect	94.4
FM	97.1
UniDM	99.8

1) Overall, UniDM attains significantly higher accuracy than the SOTA results. Although FM applies costly manual selection of context information, the accuracy of UniDM is still 4.6% higher than FM on Restaurant dataset and comparable on Buy dataset. This verifies the effectiveness of our UniDM solution, especially the automatic retrieval of the context information.

2) For the random-setting with the same context information selected randomly from the table, UniDM still outperforms FM by 5.8% on the Restaurant dataset and 6.1% on the Buy dataset. This is because UniDM applies logic transformation of the context information, rather than only simple serialization employed in FM. Meanwhile, UniDM searches for the most effective target prompt by utilizing the knowledge in the LLM, rather than simple construction by users. This verifies the success on our design choices of the context data parsing and target prompt construction.

Data Transformation. UniDM also achieves the most promising results on the data transformation task. As shown in Table 2, UniDM outperforms the search-based method TDE and the LLM-based FM. In comparison to FM (the current SOTA results), UniDM yields nearly 2% gain on the two datasets in terms of the F1 score. The reasons are analyzed in the above experiment. This also verifies the advantages of LLM-based methods over other approaches.

Error Detection. UniDM also achieves a similar behavior on the error detection task. As shown in Table 3, UniDM outperforms the baseline methods HoloClean, HoloDetect and FM by up to 2.7% in terms of the F1 score. It proves that our method is useful in interpreting the domain knowledge to detect errors.

Table 4: F1 score on entity resolution task with SOTA.

Method	Entity Resolution F1 Score (%)			
	Beer	Amazon-Google	iTunes-Amazon	Walmart-Amazon
Magellan	78.8	49.1	91.2	71.9
Ditto	94.4	75.6	97.1	86.8
FM(random)	92.3	60.7	96.3	73.8
FM(manual)	100	63.5	98.2	87.0
UniDM	96.3	64.3	96.3	88.2

Table 5: Fine-tuning experiments: F1 score of UniDM on entity resolution task (Walmart-Amazon dataset).

LLM	F1 Score (%)	
	FM	UniDM
GPT-J-6B	17.6	17.8
GPT-J-6B (fine-tune)	84.2	86.6
GPT-3-175B	87.0	88.2

Entity Resolution. As shown in Table 4, UniDM is also effective on the entity resolution task. In comparison to FM with random context information, UniDM always attains higher (or at least comparable) accuracy. In comparison to Magellan and Ditto which fine-tunes the model with large amounts of task-specific labeled data, UniDM still achieves comparable or better results in most cases. Sometimes, the accuracy of UniDM is lower than Ditto and FM with manually selected context. This is because these datasets contain very specific domain words that do not commonly occur in the corpus. As a result, the LLMs have little knowledge on their semantics and may make errors in inference. A similar phenomenon is also observed in [40]. To avoid this, Ditto utilizes domain data to fine-tune the model and FM manually selects instances to learn the domain knowledge.

For fairness, we also conduct a lightweight fine-tuning on the LLMs and apply our UniDM on the fine-tuned model. We conduct the lightweight fine-tuning experiments based on the HuggingFace[55] library. In this setting, we freeze most of the pre-trained parameters and augment the model with a small trainable head [21]. During fine-tuning, we use an AdamW optimizer and a cosine annealing learning rate scheduler with the linearly warm-up step of 100, initial learning rate of 4e-5 and final learning rate of 1e-5. Our model is trained over 8 V100 GPUs for 30 epochs with a batch size of 16.

We scale LLMs parameter size from 175B to 6B. Table 5 shows that the fine-tuned 6B LLM is comparable to the 175B LLM, suggesting UniDM has the potential to scale to even smaller models with proper fine-tuning. Meanwhile, on the fine-tuned small model, UniDM performs better than FM. This indicates that our UniDM could also attain higher accuracy by fine-tuning.

4.3 Ablation Study

For ablation study, we analyze the effectiveness of each component in UniDM. Specifically, we disable one or more components in UniDM and compare the performance of UniDM with its variants. The results are shown in Table 6. Our findings are described as follows:

Context Retrieval. When disable context retrieval component, UniDM randomly samples the same number of attributes and/or records from the table as context information. At this time, the accuracy of UniDM significantly decreases. We observe that, by simply

Table 6: Accuracy of UniDM with different components on data imputation task (Restaurant dataset).

Instance-wise Retrieval	Meta-wise Retrieval	Target Prompt Construction	Context Data Parsing	Acc (%)
				82.6
✓				84.9
	✓			90.7
✓	✓			90.7
✓	✓	✓		91.9
✓	✓	✓	✓	93.0

using instance-wise and meta-wise retrieval, the accuracy of UniDM could improve 2.3% and 8.1% on Restaurant dataset, respectively. This is because our context retrieval leverages LLMs to capture relevant attributes and/or instances with semantic relationships, which provides richer background knowledge for the target prompt.

Context Data Parsing. Without context data parsing, we only apply the serialization function to convent the tabular context information into a string. We observe that data parsing also helps to improve the result accuracy, i.e., 1.1% on Restaurant dataset. This is because our data parsing bridges the gap between structured tabular data and natural language representation to make the context information more friendly to be interpreted by LLMs.

Target Prompt Construction. We also compare UniDM with the simple prompt that directly combines the task description, the context information and the task inputs to obtain the final result. We find that our constructed target prompt using cloze questions improve the accuracy by 1.2%. This verifies the effectiveness of our target prompt construction method. It learns from examples to identify the most suitable prompt, rather than the simple combination without semantic connections.

5 FUTURE RESEARCH DIRECTIONS

Although we have made some progress on applying LLMs to data manipulation tasks on data lakes, there still reserves much room for improvement upon our work. In this section, we summarize some future research directions in terms of different perspectives of data, model and algorithm as follows:

Integration with Domain Knowledge. From the evaluation results in Section 4.2, we observe that UniDM can perform well on universal data but may fall on domain specific data. However, data lakes often contain data from highly specialized domains, e.g., financial, biological and academic data. Therefore, it is worthwhile to investigate how to incorporate domain-specific knowledge into the knowledge base of LLMs. Currently, the widely adopted method is to fine-tune the LLMs with domain specific data. Whereas, there still remain some challenges for fine-tuning. First is how to extract high quality data from data lakes as a corpus to tune the LLMs. The second is how to design fine-tuning methods for LLMs which are not open-source (such as GPT-3 [10]). Besides, it is very interesting to explore new integration methods except fine-tuning LLMs.

Extension to More Data Types. Our solution on data manipulation tasks mainly considers the structured data in relation tables in this paper. We present an example in Appendix B in the full version [3] on applying our method to retrieve information on xml data. However, data lake contains various data including semi-structured (e.g., xml), unstructured (e.g., graph) data and even multi-modal data (e.g., tables, texts and images). It is very helpful to extend our methods to process these types of data.

Designing Large Models for DB Tasks. Currently, the LLMs are mainly trained on a corpus of texts to resolve NLP tasks. Although we could design serialization functions and prompts to apply LLMs on tabular data, it is essentially a process of cutting the foot to fit the shoe. A better way is to design and train large models on tabular (and other types of) data from scratch to capture semantics for database tasks. This is a very challenging but promising direction. Some previous works have attempted to leverage BERT (e.g., TaBERT [60], TaPas [27], TaPEx [35], Tableformer [58], TURL[19]) to understand (semi-)structured data. However, all of the concepts, model structures, training methods and the whole paradigm of large models need to be re-designed to fit database tasks.

Efficiency Consideration. LLMs apply in our methods bring benefits in terms of the result accuracy but also entail a significant increase in computational resource and memory cost. Our UniDM requires more GPU resources to do inference on LLMs than conventional methods. Therefore, in the future work, it is rather important to consider how to improve the efficiency while retaining the effectiveness of LLM-based methods. We suggest two possible ways. One is to design more efficient retrieval methods to extract very relevant information from data to minimize the computation overhead. Another way is to adapt to select the LLMs with minimal computation cost to fulfill each task. As we show in Table 5, a fine-tuned LLM in a smaller size is possible to match the performance of a universal LLM in a much larger size.

Cooperation between LLM-based and Traditional Methods. In spite of our LLM-based solutions exhibiting superiority in terms of result effectiveness, they can not totally replace traditional methods. Until now, LLM is still a black-box which is difficult to interpret, debug and analyze. These are all risky factors for database systems which require rock-solid stability. Traditional methods relying on rules and logic tuned by human experience over decades have their unique advantages. They are more friendly to system deployment. Therefore, LLM-based and traditional methods are not conflicting but rather complementary to each other. It would be very practical to combine their advantages together to control the deployment risk while still attaining high result effectiveness.

6 CONCLUSION

In this paper, we design UniDM, a unified framework to solve data manipulation tasks on data lakes. UniDM summarizes a number of data manipulation tasks into a unified form and designs general steps to solve these tasks using LLMs with proper prompts. Experimental results demonstrate that UniDM exhibits superior performance when compared to traditional and learned methods on a variety of data manipulation tasks. In the future work, we show that there still reserves enough room for improvement in terms of data, model and algorithm. We hope the strategies proposed in UniDM could contribute and inspire more advances on exploring LLMs with database systems.

REFERENCES

[1] 2023. FM Implementation. https://github.com/HazyResearch/fm_data_tasks.

[2] 2023. OpenAI. https://openai.com/api.

[3] 2023. UniDM: A Unified Framework for Data Manipulation with Large Language Models [Full Version]. https://github.com/mx54039q/UniDM/blob/main/fullpaper.pdf.

[4] 2023. UniDM Implementation. https://github.com/mx54039q/UniDM.

[5] Ayman Alserafi, Alberto Abelló, Oscar Romero, and Toon Calders. 2019. Keeping the data lake in form: DS-kNN datasets categorization using proximity mining. In *Model and Data Engineering: 9th International Conference, MEDI 2019, Toulouse, France, October 28–31, 2019, Proceedings 9*. Springer, 35–49.

[6] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel J Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. Ask Me Anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441* (2022).

[7] Wang Ben and Komatsuzaki Aran. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax

[8] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6. http://jmlr.org/papers/v20/18-753.html

[9] Mikhail Bilenko and Raymond J Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 39–48.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[11] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *23rd International Conference on Extending Database Technology, Copenhagen, 30 March-2 April 2020*. OpenProceedings.

[12] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Sam Madden, and Nan Tang. 2023. Symphony: Towards Natural Language Query Answering over Multi-modal Data Lakes. *The Conference on Innovative Data Systems Research* (2023).

[13] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 458–469.

[14] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1247–1261.

[15] Tianji Cong, James Gale, Jason Frantz, HV Jagadish, and Çağatay Demiralp. 2022. WarpGate: A Semantic Join Discovery System for Cloud Data Warehouse. *arXiv preprint arXiv:2212.14155* (2022).

[16] Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712* (2022).

[17] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 541–552.

[18] Nilesh Dalvi, Vibhor Rastogi, Anirban Dasgupta, Anish Das Sarma, and Tamás Sarlós. 2013. Optimal hashing schemes for entity matching. In *Proceedings of the 22nd international conference on world wide web*. 295–306.

[19] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[21] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. OpenPrompt: An Open-source Framework for Prompt-learning. *arXiv preprint arXiv:2111.01998* (2021).

[22] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.

[23] Javier de Jesús Flores Herrera, Sergi Nadal Francesch, and Óscar Romero Moral. 2021. Towards scalable data discovery. In *Advances in Database Technology: EDBT 2021, 24th International Conference on Extending Database Technology: Nicosia, Cyprus, March 23-26, 2021: proceedings*. OpenProceedings, 433–438.

[24] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Association for Computational Linguistics (ACL)*.

[25] Rihan Hai, Christoph Quix, and Matthias Jarke. 2021. Data lake concept and systems: a survey. *arXiv preprint arXiv:2106.09592* (2021).

[26] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*. 829–846.

[27] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349* (2020).

[28] Zhongjun Jin, Yeye He, and Surajit Chauduri. 2020. Auto-transform: learning-to-transform by patterns. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2368–2381.

[29] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[30] Pradap Konda, Sanjib Das, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, et al. 2016. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1581–1584.

[31] Guoliang Li, Xuanhe Zhou, Ji Sun, Xiang Yu, Yue Han, Lianyuan Jin, Wenbo Li, Tianqing Wang, and Shifu Li. 2021. opengauss: An autonomous database system. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3028–3042.

[32] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A study for evaluating the impact of data cleaning on ml classification tasks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 13–24.

[33] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.

[34] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[35] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian guang Lou. 2021. TAPEX: Table Pre-training via Learning a Neural SQL Executor. arXiv:2107.07653 [cs.CL]

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

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

[38] Shahmeer Ahmad Mohammad, Ahmad Naeem Zan, Eltabakh Mohamed, Ouzzani Mourad, and Tang Nan. 2023. RetClean: Retrieval-Based Data Cleaning Using Foundation Models and Data Lakes. *arXiv preprint arXiv:2303.16909* (2023).

[39] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[40] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *arXiv preprint arXiv:2205.09911* (2022).

[41] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment* 12, 12 (2019), 1986–1989.

[42] Paul Ouellette, Aidan Sciortino, Fatemeh Nargesian, Bahar Ghadiri Bashardoost, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2021. RONIN: data lake exploration. *Proceedings of the VLDB Endowment* 14, 12 (2021).

[43] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1470–1480. https://doi.org/10.3115/v1/P15-1142

[44] Ralph Peeters and Bizer Christian. 2021. Dual-objective fine-tuning of BERT for entity matching. *Proceedings of the VLDB Endowment* 14 (2021), 1913–1921.

[45] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[46] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment* 10 (2017), 1190–1201.

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

[48] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment* 11, 2 (2017), 189–202.

[49] Hegselmann Stefan, Buendia Alejandro, Lang Hunter, Agrawal Monica, Jiang Xiaoyi, and Sontag David. 2022. TabLLM: Few-shot Classification of Tabular Data with Large Language Models. *arXiv preprint arXiv:2210.10723* (2022).

[50] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).

[51] Immanuel Trummer. 2022. CodexDB: Synthesizing code for query processing from natural language instructions using GPT-3 Codex. *PVLDB* 15, 11 (2022), 2921 – 2928.

[52] Immanuel Trummer. 2022. DB-BERT: a database tuning tool that "reads the manual". In *SIGMOD*.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[54] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. PromptEM: prompt-tuning for low-resource generalized entity matching. *arXiv preprint arXiv:2207.04802* (2022).

[55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 38–45.

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

[57] Ziniu Wu, Pei Yu, Peilun Yang, Rong Zhu, Yuxing Han, Yaliang Li, Defu Lian, Kai Zeng, and Jingren Zhou. 2021. A unified transferable model for ml-enhanced dbms. *arXiv preprint arXiv:2105.02418* (2021).

[58] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding. *arXiv preprint arXiv:2203.00274* (2022).

[59] He Yeye, Chu Xu, Ganjam Kris, Zheng Yudian, Narasayya Vivek, and Chaudhuri Surajit. 2018. Transform-data-by-example (TDE) an extensible search engine for data transformations. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1165–1177.

[60] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314* (2020).

[61] Chen Zhao and Yeye He. 2019. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*. 2413–2424.

[62] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419* (2023).

Appendices

A EXPLAINING THE TABLE QUESTION ANSWER TASK

To show the generality of our UniDM solution, we apply that it could be applied to the more complex table question answer (TableQA) task. This is a task to ask a question to retrieve answers from a data table. Figure 3 gives an illustrative example on WikiTableQuestions dataset [43]. Here we have a question Q: "how many gold medals did Australia and Switzerland total?" and the answer on the number of gold medals could be obtained from the table by finding the relevant information.

For the TableQA task, we directly set the task query Q to be the question contained in the task description. The set R of records and set S of attributes are set to contain all records and attributes in the table D_i, respectively. When applying UniDM to solve TableQA, in the first context information retrieval, we set the set of candidate attributes $S' = S$. By applying prompts p_{rm} and p_{ri}, UniDM first automatically retrieves a content snapshot C from the data table D_i. This snapshot contains a selection of columns ('Nation' and 'Gold' in our example) and rows ('Australia (AUS)' and 'Switzerland (SUI)' in our example) that summarize the information most relevant to the task and the query ('how many gold medals did Australia and Switzerland total?').

In the second step, the context snapshot C is serialized and then parsed into a natural text representation C'. In our example, we now know that "Australia (AUS) won 2 goal medals, while Switzerland (SUI) won 0 goal medals". To facilitate open-ended cloze question generation centered around the target query and contextual information drawn from the data, the prompt engineering module is employed. Ultimately, the resulting cloze question is fed into the LLM to yield an answer. UniDM could correctly output "2" as the answer. This exhibit that UniDM is good at not only processing instance-level tasks such as data imputation and error correction, but also can be applied to retrieve table-level information.

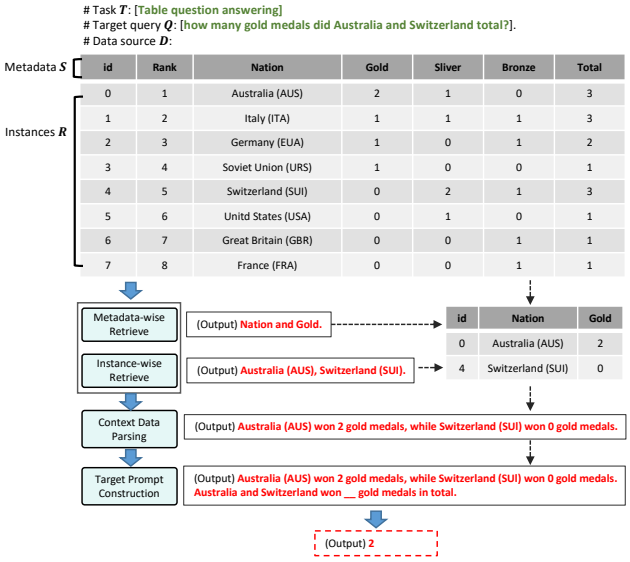


Figure 3: Explanation example for table question answering by the UniDM.

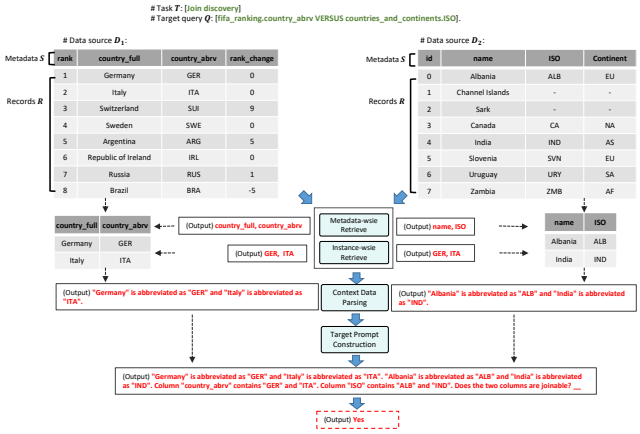


Figure 4: Explanation example for join discovery by the UniDM.

B JOIN DISCOVERY TASK

In data lakes, the join relations across tables are not specified. Join discovery task is a major challenge in data analysis. It aims at finding semantically joinable columns across different tables. This task could be subsumed and solved by our UniDM framework.

Figure 4 gives an illustrative example of join discovery task. The query Q is set to be the textual name of the two tables, e.g., "fifa_ranking.country_abrv " and "countries_and_continents.ISO". The set R of records and the set S of attributes are set to contain all records and attributes in the two tables D_1 and D_2, respectively. In the first context information retrieval, UniDM extracts joinable attributes and records C_1 and C_2 between the two tables. The context snapshots C_1 and C_2 are serialized and then parsed into natural text representations C_1' and C_2' separately. By using the prompt engineering module, UniDM constructs a cloze question with the target query and contextual information from the two tables. Ultimately, the resulting cloze question is fed into the LLM to yield an answer "Yes" that indicates the two tables are joinable.

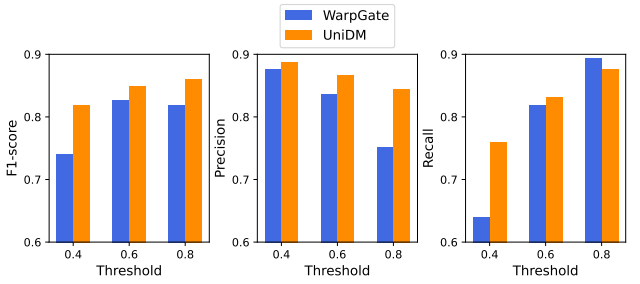


Figure 5: F1-score, precision and recall on join discovery task.

Figure 6: Explanation example for information extraction by the UniDM.

Table 7: Text F1-score on information extraction task.

Method	Information Extraction NBA player
Evaporate-code	40.6
Evaporate-code+	84.6
UniDM	70.1

For experiment on join discovery task, we use NextiaJD [23] that composes four splits according to their file size. The dataset labels the join quality of attribute pairs based on a measure that considers both containment and cardinality proportion with empirically determined thresholds. In experiments, we use a subset with 4404 pairs (2239 positive and 2164 negative) of attributes whose quality is labeled as Good and High. For the baseline, we select an embedding-based solution WarpGate [15], the SOTA method. As shown in Figure 5, we report precision, recall and F1-score under various threshold values. We find that UniDM consistently obtains higher F1-score compared with WarpGate [15] under various threshold values. This exhibits that UniDM's potential to manipulate data cross sources.

C INFORMATION EXTRACTION TASK

In order to demonstrate the performance of UniDM in handling more complex data manipulation tasks, we conduct experiments on an information extraction task. For the information extraction task, we aim to construct a structured view (i.e., tabular) of a set of semi-structured documents (e.g., HTML).

We also subsume the information extraction task by our framework. D_i is a semi-structured document and $R = D_i$. S represents a set of pre-defined attributes to be extracted from D_i. $F_T(R, S, \mathcal{D})$ outputs a set of attribute values from the document D_i. We set the query parameter $Q = S$ to give the target attributes to be extracted.

Figure 6 gives an illustrative example of information extraction task. The query Q is set to be the target attribute S, e.g., "player". Note that we temporarily removed the context retrieval module because the attributes and the instance is pre-defined by users. We slightly modified the context parsing prompt by adding the query Q. It aims to extract and transform the context information ("Kevin Durant is an American professional basketball player...") according to the query Q. We then construct the target cloze question and yield the final result (i.e., "Kevin Durant").

For experiment on information extraction task, we use the NBA players dataset that contains 100 randomly selected player pages and 19 attribute annotations. For the baseline, we select the LLM-based method Evaporate-code and Evaporate-code+. As shown in Table 7, UniDM outperforms the baseline methods Evaporate-code in terms of the F1 score. Evaporate-code+ achieves better results due to its utilization of ensemble methods. This exhibits that UniDM's potential to manipulate semi-structured data.