UniDM: A Unified Framework for Data Manipulation with Large Language Models

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]

Abstract

Designing effective data manipulation methods 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) 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. We develop an automatic context retrieval to allow the LLMs to retrieve data from data lakes, potentially containing evidence and factual information. For each step, we design effective prompts to 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 a wide variety of data manipulation tasks.

1 Introduction

Data lake is a general system to store vast amounts of data with heterogeneous schemas and structures. It provides 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 (Nargesian et al., 2019; Ouellette et al., 2021). 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 (Hai et al., 2021).

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 (Dalvi et al., 2013; Singh et al., 2017; Chu et al., 2013; 2015; Dallachiesa et al., 2013; Mayfield et al., 2010; Jin et al., 2020) require specialized model construction and rule tuning for each data task, which are not automatic enough. Recent works apply machine learning (Bilenko & Mooney, 2003; Konda et al., 2016; Heidari et al., 2019; Biessmann et al., 2019; Li et al., 2021a;b; Wu et al., 2021; Alserafi et al., 2019), especially deep learning techniques (Mudgal et al., 2018; Ebraheem et al., 2018; Zhao & He, 2019; Deng et al., 2022), 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.

In recent time, Large Language Models (LLMs), such as BERT (Devlin et al., 2018), GPT-3 (Brown et al., 2020), and LaMDA (Thoppilan et al., 2022), have shown incredible performance on a broad set of downstream tasks (Zhou et al., 2023; Liang et al., 2022). LLMs are typically deep neural networks with transformer architecture (Vaswani et al., 2017). They are pre-trained on enormous corpora of text to learn universal world knowledge. 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 (Radford et al., 2019; Brown et al., 2020).

[*] Equal contribution [1] Alibaba Group [2] University of Science and Technology of China. Correspondence to: Rong Zhu <red.zr@alibaba-inc.com>, Defu Lian <liandefu@ustc.edu.cn>, Bolin Ding <bolin.ding@alibaba-inc.com>, Jingren Zhou <jingren.zhou@alibaba-inc.com>.

Proceedings of the 5^{th} MLSys Conference, Santa Clara, CA, USA, 2024. Copyright 2024 by the author(s).

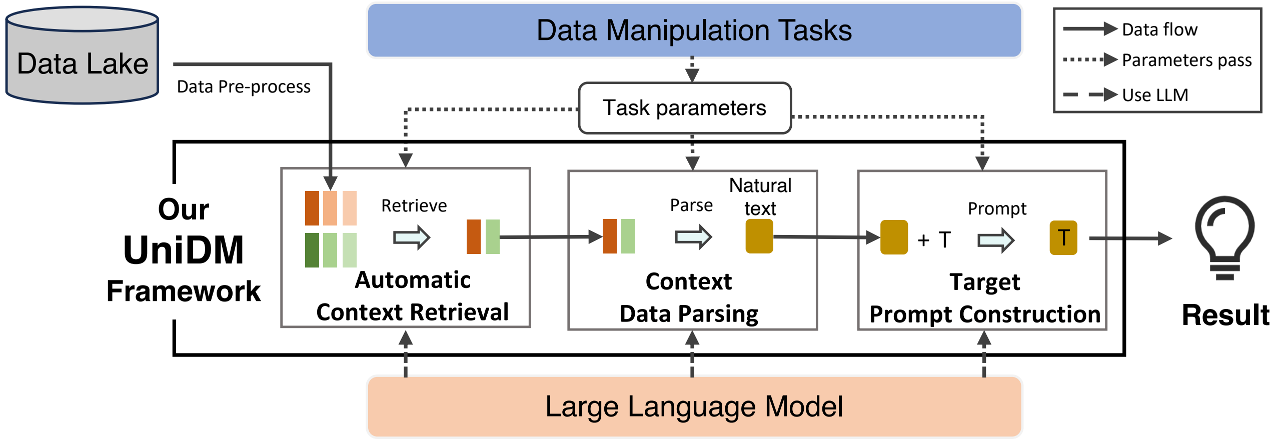


Figure 1. An overview of our UniDM framework.

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 3)

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 (Wang et al., 2022). 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 (Narayan et al., 2022). 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 4)

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 4)

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 5)

2 Background and Motivation

Large Language Models and Prompts. LLMs could be regarded as a foundation model 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 (Brown et al., 2020). 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 Data Tasks. Some very recent works (Mohammad et al., 2023; Brunner & Stockinger, 2020; Li et al., 2020; Mei et al., 2021; Chen et al., 2023; Narayan et al., 2022; Wang et al., 2022; Trummer, 2022a;b) 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 (Herzig et al., 2020; Liu et al., 2021; Peeters & Christian, 2021; Trummer, 2022a) have verified the effectiveness of applying LLMs to answer questions on data tables. Later, LLMs have been applied on data pre-processing tasks (Li et al., 2020), binary classification on tables (Stefan et al., 2022), data cleaning and integration tasks(Mohammad et al., 2023; Narayan et al., 2022). 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?

3 Problem Definition

In this section, we propose a unified framework to formalize the data manipulation tasks we target to solve on data lakes. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_l\}$ 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**: This task is to repair dirty data and impute the missing value within a record. S contains an attribute in S_i and R contains a singleton record in D_i having a missing value on attribute S, $F_T(R, S, \mathcal{D})$ outputs the desired missing value of $R[S]$.
- **Data Transformation**: This task is the process of converting data from one format to another required format within a record. 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**: This task is to detect attribute error within a record in a data cleaning system. 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**: This task is the process of predicting whether two records are referencing the same real-world thing. 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.

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 tasks 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.

4 Unified Framework for Data Manipulation

To generally support common data manipulation tasks, we propose the UniDM, a unified framework based on LLMs. We first analyze this problem and outline our solution in Section 4.1. Then, Section 4.2 to Section 4.4 elaborate on the details of each key technique in our method. Finally, the generality of our method is discussed in Section 4.5.

4.1 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 (Brown et al., 2020). 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 (Creswell et al., 2022), 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.

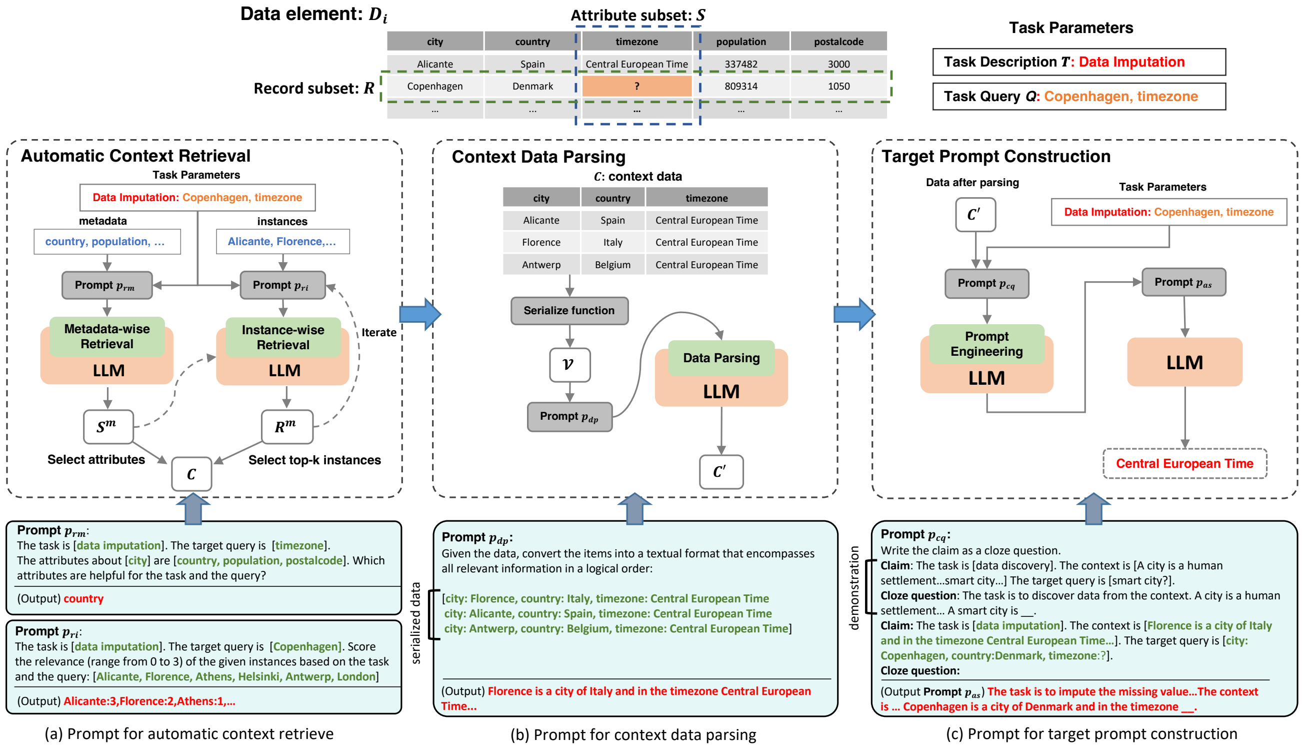
Some literature works (Chen et al., 2023; Wang et al., 2022) have devoted the efforts on designing and tuning prompts for some specific data manipulation tasks. Whereas, this is very expensive and not extensible to numerous customized tasks occurring every day. What we actually need is a general solution 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 in Section 3) into several consistent steps. The pipeline and architecture of UniDM are illustrated in Figure 2. Each step is a simple and direct 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. UniDM takes data from a data lake as input and performs a data manipulation task in an iterative and interactive fashion. 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 $\mathcal{C}$ from $\mathcal{D}$ to resolve T. $\mathcal{C}$ may contain additional information from other records and attributes to guide the LLMs to capture the semantics for task T. We design two templates of prompts to automatically retrieve context information from $\mathcal{D}$ using LLMs. The first prompt p_{rm} aims at 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 $\mathcal{C}$ in a tabular form.

2) **Context Parsing**: The raw context information in $\mathcal{C}$ 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 $\mathcal{C}$ into another form that is more easily to be interpreted by the LLMs. Similar to previous works (Narayan et al., 2022), we first apply a serialize function to transform $\mathcal{C}$ 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 $\mathcal{C}'$ reflecting the logic relations among different attributes, e.g. "Florence is a city of Italy". $\mathcal{C}'$ is smoother and closer 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 $\mathcal{C}'$, 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 3 could be equivalently transformed into a cloze question. Cloze question is friendly to LLMs as it is written in natural language with placeholders. To au-



city	country	timezone	population	postalcode
Alicante	Spain	Central European Time	337482	3000
Copenhagen	Denmark	?	809314	1050
...	...	...	...	...

city	country	timezone
Alicante	Spain	Central European Time
Florence	Italy	Central European Time
Antwerp	Belgium	Central European Time

Prompt p_{rm}:
The task is [data imputation]. The target query is [timezone].
The attributes about [city] are [country, population, postalcode]. Which attributes are helpful for the task and the query?

(Output) country

Prompt p_{ri}:
The task is [data imputation]. The target query is [Copenhagen]. Score the relevance (range from 0 to 3) of the given instances based on the task and the query: [Alicante, Florence, Athens, Helsinki, Antwerp, London]

(Output) Alicante:3,Florence:2,Athens:1,...

(a) Prompt for automatic context retrieve

Prompt p_{dp}:
Given the data, convert the items into a textual format that encompasses all relevant information in a logical order:

[city: Florence, country: Italy, timezone: Central European Time
city: Alicante, country: Spain, timezone: Central European Time
city: Antwerp, country: Belgium, timezone: Central European Time]

(Output) Florence is a city of Italy and in the timezone Central European Time...

(b) Prompt for context data parsing

Prompt p_{cq}:
Write the claim as a cloze question.
Claim: The task is [data discovery]. The context is [A city is a human settlement...smart city...] The target query is [smart city?].
Cloze question: The task is to discover data from the context. A city is a human settlement... A smart city is __.
Claim: The task is [data imputation]. The context is [Florence is a city of Italy and in the timezone Central European Time...]. The target query is [city: Copenhagen, country:Denmark, timezone:?].
Cloze question:

(Output **Prompt p_{as}**) The task is to impute the missing value...The context is ... Copenhagen is a city of Denmark and in the timezone __.

(c) Prompt for target prompt construction

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

tomatically 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.

4.2 Automatic Context Retrieval

To capture data knowledge from data lakes in a more interpretable and modular way, we augment LLMs with an automatic context retrieval component. We require that the context retrieval component could identify useful information while filtering irrelevant data to facilitate the LLMs. Previous works (Biessmann et al., 2019; Narayan et al., 2022) 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 (Mei et al., 2021; Mohammad et al., 2023). Unlike with them, we design a purely automatic strategy for extracting helpful attributes and records with the aid of the LLMs. We first apply meta-wise retrieval to find relevant attributes from the holistic view. Then we apply instance-wise retrieval to extract 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 4.5. 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 5) are competitive enough.

We denote S^t as the task-relevant attributes returned by the LLMs. In our example in Figure 2 (left), when given the task description "data imputation" and the target query (attribute) "timezone", the LLMs select the attribute "country" towards inferring about the missing value.

Instance-wise Retrieval. Next, we perform fine-grained filtering on records to identify relevant ones w.r.t. target records in R. We first shrink the data in $D_i - R = \{r_1, r_2, \ldots, r_m\}$ to provide a set of candidate records R' by random-sampling. After that, we use the LLMs 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 to consult the semantic knowledge to measure the relevance scores of the records for the target task by a prompt p_{ri} using the following template:

> **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:* $\{r_1[S^t], r_2[S^t], ..., r_m[S^t]\}$

Here for each $r_j \in R'$, we only reserve the task-relevant attributes in S^t 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^t as the task-relevant context $\mathcal{C}$ for the downstream procedures.

4.3 Context Data Parsing

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

Previous works (Narayan et al., 2022) 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 $\mathcal{C}'$ reflecting the relations among different attributes. For example, in Figure 2 (middle), 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 $\mathcal{C}'$ 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 $\mathcal{C}'$ 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 $\mathcal{C}'$ 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:* $[\mathcal{V}]$

4.4 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 $\mathcal{C}'$ in logic text and the query Q encoding the input records R and attributes S (defined in Section 4.2) 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. Specifically, cloze question asks the model to fill in the remaining text ("Australia and Switzerland won __ gold medals in total."). Cloze question is friendly to LLMs as it is written in natural language with placeholders. 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 $R[S]$. 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*: The task is data imputation. The context is...*
Cloze question*: ... China's population is __.*
Claim*: The task is data transformation. The context is...*
Cloze question*: ... The roman numeral III can be transformed to normal number __.*
......
Claim*: The task is* $[T]$. *The context is* $[\mathcal{C}']$. *The target query is* $[Q]$.
Cloze question*:*

Motivated by the discrete prompt search methods (Gao et al., 2021; Arora et al., 2022), 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 5 indicate that the prompts generated by our method are very effective on a variety of data manipulation tasks.

4.5 Generalization to More Tasks

Our UniDM framework could be easily generalized to other tasks listed in Section 3 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 4.2). On the other hand, our method can flexibly combine various modules for different tasks. 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 other tasks that could be subsumed by our framework defined in Section 3, we could also adjust the parameters and module combination according to the semantics of the task.

5 Experiment

In this section, we conduct extensive experiments on different data manipulation tasks to evaluate the generalization and quality of our UniDM. We also perform an in-depth analysis of UniDM on more data types and task forms. And then, we provide several model variants to show the effectiveness of the proposed method.

5.1 Experimental Setup

Implement Details. We implement our UniDM using the GPT-3-175B parameter model (Brown et al., 2020)(text-davinci-003) in the OpenAI API (OpenAI, 2021) as the LLM without fine-tuning. In addition, we give fine-tuning results for open-source LLMs GPT-J-6B (Ben & Aran, 2021) and LLaMA2-7B (LLaMA2, 2023). In our method, in the default setting, we apply the automatic context retrieval method proposed in Section 4.2. In detail, we extract one attribute from the candidate set in the metadata-wise retrieval (see prompt p_{rm} in Section 4.2) and top-3 records from 50 records randomly sampled in the dataset in the instance-wise retrieval (see prompt p_{ri} in Section 4.2).

Evaluation Tasks and Datasets. We evaluate UniDM on a number of different data manipulation tasks including data imputation, data transformation, error detection and entity resolution. We evaluate the performance of our method on different benchmark datasets. For data imputation, we choose two challenging benchmark datasets, namely Restaurants and Buy, from (Mei et al., 2021). For Restaurants dataset, the target attribute to be impute is "city". For Buy dataset, the target attribute to be impute is "manufacturer". We manually mask the values in the target attributes. Ground truth information is available for the missing values. For data transformation, we follow the TDE benchmark in (Yeye et al., 2018) and choose two datasets, namely StackOverflow and Bing-QueryLogs. This benchmark covers diverse types of transformation tasks (e.g., ip, address, phone, etc). For error detection task, we choose the benchmark Hospital and Adult datasets widely used in data cleaning papers (Rekatsinas et al., 2017; Heidari et al., 2019). Errors amount to 5% of the total data. Ground truth information is available for all cells. For entity resolution, we follow the standard Magellan benchmark in (Konda et al., 2016) and choose four datasets across different domains. Each dataset consists of candidate pairs from two structured tables of entity records of the same schema. The ground truth labels (positive or negative) are available for the entity pairs.

Baseline Approaches. We compare UniDM with a variety of state-of-the-art (SOTA) methods on data manipulation tasks. The **FM** (Narayan et al., 2022) method is shown to attain SOTA performance on multiple data manipulation tasks with simple prompt learning on LLMs. We reproduce FM following the original paper and its open-source code. 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 tasks. We also reproduce a random-sample version of FM, where the context information of records is randomly selected from the table.

For data imputation task, we select several methods following different technical routines, including a statistics-based method **HoloClean** (Rekatsinas et al., 2017; Wu et al., 2020), a clustering-based method **CMI** (Shichao et al., 2008) and a deep learning based method **IMP** (Mei et al., 2021). For data transformation task, we select a search-based method **TDE** (Yeye et al., 2018). For error detection task, we select two machine learning based methods **HoloClean** (Rekatsinas et al., 2017) and **HoloDetect** (Heidari et al., 2019). For entity resolution task, we use a deep learning method **Ditto** (Li et al., 2020).

Evaluation Metrics. Following previous works, we employ widely-used metrics, *accuracy*, *precision*, *recall* and *F1-score* to evaluate the effectiveness of these methods. For data imputation and data transformation, we use accuracy to denote the fraction of correct repairs over the total number of repairs performed for cells in the labeled data. For error detection and entity resolution, we use F1-score based on precision and recall.

Evaluation Goals. The experimental results mainly answer the following questions:

- What is the performance of our UniDM solution on different data manipulation tasks? (in Section 5.2)
- What is the contribution of each component in our UniDM solution? (in Section 5.3)

5.2 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:

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.

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	Adult
HoloClean	51.4	54.5
HoloDetect	94.4	99.1
FM	97.1	99.1
UniDM	99.8	99.7

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 accuracy. The reasons are analyzed in the above experiment. This also verifies the advantages of LLM-based methods over other approaches.

Error Detection. We report the F1-score obtained by UniDM and competing approaches. UniDM also achieves a similar behavior on the error detection task. As shown in Table 3, UniDM outperforms the baseline methods Holo-

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

Clean, HoloDetect and FM by up to 2.7% in terms of the F1-score in the Hospital dataset. For the Adult dataset, UniDM achieves a high F1-score of 99.7%, as it uses the information on data source. It proves that our method is useful in interpreting the domain knowledge to detect errors.

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 (Narayan et al., 2022). To avoid this, Ditto utilizes domain data to fine-tune the model and FM manually selects instances to learn the domain knowledge.

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
LLaMA2-7B	NA	40.6
LLaMA2-7B (fine-tune)	NA	89.4
GPT-3-175B	87.0	88.2

For fairness, we also conduct a lightweight fine-tuning on our UniDM as LLMs' model capacity scaling, specifically targeting the entity resolution task. We conduct the lightweight fine-tuning experiments based on the HuggingFace(Wolf et al., 2020) library. In this setting, we freeze most of the pre-trained parameters and augment the model with a small trainable head (Ding et al., 2021). For the fine-tuning process, we use the training split of the Walmart-Amazon dataset, which consists of 6144 tuples. 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 also reproduced this experiment on FM under the same setting.

Table 6. UniDM results on data imputation with LLMs variants.

Model	Data Imputation Acc (%)	
	Restaurant	Buy
GPT-3-175B	93.0	98.5
GPT-4-Turbo	96.5	98.5
Claude2	89.5	96.9
LLaMA2-7B	86.0	95.4
LLaMA2-70B	88.4	96.9
Qwen-7B	86.0	93.8

Table 7. Token consumption (per-query) comparison with FM.

Method	Token Consumption	
	Restaurant	Buy
FM	174	246
UniDM (w/o retrieval)	325	384
UniDM	6860	7323

We scale LLMs parameter size from 175B down to 6B/7B. As shown in Table 5, the fine-tuned 6B/7B LLM exhibits comparable performance to the much larger 175B model, 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.

To showcase the performance of our UniDM across different base models, we further evaluate UniDM over five LLM variants: GPT-4-Turbo (OpenAI, 2021), Claude2 (about 100B) (Claude2, 2023), LLaMA2 (7B and 70B) (LLaMA2, 2023), and Qwen-7B (Qwen, 2023) on a data imputation task. We observe consistent high performance with UniDM across these different base LLMs, demonstrating its adaptability and robustness. As shown in Table 6, even on 7B models, our UniDM maintains impressive performance, and it can achieve better results on larger models at a higher cost.

In Table 7, we compare the LLMs' tokens between the FM and our UniDM. It is evident that our method incurs greater token consumption than the FM. However, our UniDM automates the processes of context retrieval and target prompt construction, which significantly reduces the human labor.

5.3 Impact of Model Components

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 8,

Table 8. Ablation study of UniDM 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

Table 9. Ablation study of UniDM on data imputation task (Buy dataset).

Instance-wise Retrieval	Meta-wise Retrieval	Target Prompt Construction	Context Data Parsing	Acc (%)
				90.8
✓				92.3
	✓			90.8
✓	✓			92.3
✓	✓	✓		96.9
✓	✓	✓	✓	98.5

Table 10. Ablation study of UniDM on data transformation task.

Target Prompt Construction	Context Data Parsing	Data Transformation Acc (%)	
		Stack Overflow	Bing QueryLogs
		63.3	52.0
	✓	65.3	52.0
✓		65.3	54.0
✓	✓	67.4	56.0

Table 9 and Table 10. 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, for the data imputation task, by simply using instance-wise and meta-wise retrieval, the accuracy of UniDM could improve 2.3% and 8.1% on Restaurant dataset, respectively. Additionally, on the Buy dataset, using context retrieval also improves the accuracy by 1.5%. 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. For the data transformation task, it does not require to extract context data and this step is not included in the ablation study.

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, for the data imputation task, data parsing also helps to improve the result accuracy, with a 1.1% increase on the Restaurant dataset and a 1.6% increase on the Buy dataset. Similarly, for the data transformation task, using data parsing improves the accuracy by 2% on the StackOverflow 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. For the data imputation task, we find that our constructed target prompt using cloze questions improves accuracy by 1.2% on the Restaurant dataset and by 4.6% on the Buy dataset. Furthermore, for the data transformation task, the accuracy could also improve 2% on both the StackOverflow and Bing-QueryLogs datasets by using this method. 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.

5.4 Discussion

The principal objective of our experimental study is to demonstrate the effectiveness of LLMs in enhancing data lakes and minimizing human effort. The challenge of dealing with massive amounts of data is a common issue faced by data lake systems. Data relationships take on various forms. For instance, in some cases, we may have shared values or keys; in others, the data may be complementary and thus have no value-overlap at all. To address this challenge, our method utilizes LLMs to understand data relationships, integrate heterogeneous data sources, and automatically identify the relational data for data tasks. Our findings suggest that data retrieval can boost performance by selecting relational data and filtering noisy data. Another grand challenge for data lake systems is supporting various on-demand queries. Our UniDM offers a combination of data modules and task modules in a flexible way. From a data module perspective, automatic data retrieval is used to extract useful information, while data parsing is used for data interpretation. From a task module perspective, prompt engineering demonstrates remarkable cross-task capabilities across various data tasks. Overall, our UniDM makes data in data lakes actionable and enriches data lakes in a flexible manner.

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.

6.1 future works

UniDM can benefit from explicitly retrieving data, potentially containing evidence and factual information, before the processing of data manipulation. However, the model may be unreliable when dealing with domain-specific knowledge that is required for commercial use cases. In such scenarios, LLMs can generate information that may be problematic in practical applications where factual accuracy is crucial. Another limitation is explainability. In most data applications (e.g., root cause analysis), a good explanation that includes all the rules applied to reach a conclusion can be valuable to the user. While LLMs have exhibited remarkable success in various data tasks, their ability to reason and explain is often viewed as a limitation. We summarize some future research directions in terms of different perspectives of data, model, algorithm and efficiency as follows.

Integration with Domain Knowledge From the results in Section 5.2, 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. Currently, the widely adopted method is to fine-tune the LLMs with domain specific data. Whereas, there still remain challenges for fine-tuning, such as how to extract high quality data from data lakes as a corpus to tune the LLMs. Besides, it is very interesting to explore new integration methods except fine-tuning LLMs.

Designing Large Models for DB Tasks 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. Some previous works have attempted to leverage BERT (e.g., TaPas (Herzig et al., 2020), TaPEx (Liu et al., 2021), Tableformer (Yang et al., 2022), TURL(Deng et al., 2022)) 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 applied in our method bring benefits but also entail an increase in computational resource. In the future work, it is rather important to consider how to improve the efficiency while retaining the effectiveness of LLM-based methods. One possible way is to design more efficient retrieval methods to extract 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.

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. 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.

ACKNOWLEDGEMENTS

The work was supported by grants from the National Natural Science Foundation of China (No. 92367110) and the program of Alibaba Innovative Research.

References

Alserafi, A., Abelló, A., Romero, O., and Calders, T. 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*, pp. 35–49. Springer, 2019.

Arora, S., Narayan, A., Chen, M. F., Orr, L. J., Guha, N., Bhatia, K., Chami, I., Sala, F., and Ré, C. Ask me anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441*, 2022.

Ben, W. and Aran, K. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021. URL https://github.com/kingoflolz/mesh-transformer-jax.

Biessmann, F., Rukat, T., Schmidt, P., Naidu, P., Schelter, S., Taptunov, A., Lange, D., and Salinas, D. Datawig: Missing value imputation for tables. *Journal of Machine Learning Research*, 20(175):1–6, 2019. URL http://jmlr.org/papers/v20/18-753.html.

Bilenko, M. and Mooney, R. J. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 39–48, 2003.

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. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Brunner, U. and Stockinger, K. 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*. Open-Proceedings, 2020.

Chen, Z., Gu, Z., Cao, L., Fan, J., Madden, S., and Tang, N. Symphony: Towards natural language query answering over multi-modal data lakes. *The Conference on Innovative Data Systems Research*, 2023.

Chu, X., Ilyas, I. F., and Papotti, P. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 458–469. IEEE, 2013.

Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., and Ye, Y. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1247–1261, 2015.

Claude2. Claude2, 2023. URL https://claude.ai/.

Cong, T., Gale, J., Frantz, J., Jagadish, H., and Demiralp, Ç. Warpgate: A semantic join discovery system for cloud data warehouse. *arXiv preprint arXiv:2212.14155*, 2022.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A., Ilyas, I. F., Ouzzani, M., and Tang, N. Nadeef: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 541–552, 2013.

Dalvi, N., Rastogi, V., Dasgupta, A., Das Sarma, A., and Sarlós, T. Optimal hashing schemes for entity matching. In *Proceedings of the 22nd international conference on world wide web*, pp. 295–306, 2013.

Deng, X., Sun, H., Lees, A., Wu, Y., and Yu, C. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ding, N., Hu, S., Zhao, W., Chen, Y., Liu, Z., Zheng, H.-T., and Sun, M. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*, 2021.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., and Tang, N. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

Flores Herrera, J. d. J., Nadal Francesch, S., and Romero Moral, Ó. 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*, pp. 433–438. OpenProceedings, 2021.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*, 2021.

Hai, R., Quix, C., and Jarke, M. Data lake concept and systems: a survey. *arXiv preprint arXiv:2106.09592*, 2021.

Hao, Q., Cai, R., Pang, Y., and Zhang, L. From one tree to a forest: a unified solution for structured web data extraction. In *Proceeding of the 34th International ACM SIGIR*

Conference on Research and Development in Information Retrieval (SIGIR 2011). Association for Computing Machinery, Inc., July 2011.

Heidari, A., McGrath, J., Ilyas, I. F., and Rekatsinas, T. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*, pp. 829–846, 2019.

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., and Eisenschlos, J. M. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*, 2020.

Jin, Z., He, Y., and Chauduri, S. Auto-transform: learning-to-transform by patterns. *Proceedings of the VLDB Endowment*, 13(12):2368–2381, 2020.

Konda, P., Das, S., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., et al. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment*, 9(13):1581–1584, 2016.

Li, G., Zhou, X., Sun, J., Yu, X., Han, Y., Jin, L., Li, W., Wang, T., and Li, S. opengauss: An autonomous database system. *Proceedings of the VLDB Endowment*, 14(12): 3028–3042, 2021a.

Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., and Zhang, C. Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 13–24. IEEE, 2021b.

Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60, 2020.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Liu, Q., Chen, B., Guo, J., Ziyadi, M., Lin, Z., Chen, W., and guang Lou, J. Tapex: Table pre-training via learning a neural sql executor, 2021.

LLaMA2. Llama2, 2023. URL https://ai.meta.com/llama/.

Mayfield, C., Neville, J., and Prabhakar, S. Eracer: a database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 75–86, 2010.

Mei, Y., Song, S., Fang, C., Yang, H., Fang, J., and Long, J. Capturing semantics for imputation with pre-trained language models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 61–72. IEEE, 2021.

Mohammad, S. A., Zan, A. N., Mohamed, E., Mourad, O., and Nan, T. Retclean: Retrieval-based data cleaning using foundation models and data lakes. *arXiv preprint arXiv:2303.16909*, 2023.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 19–34, 2018.

Narayan, A., Chami, I., Orr, L., and Ré, C. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911*, 2022.

Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., and Arocena, P. C. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12): 1986–1989, 2019.

OpenAI. Openai api, 2021. URL https://openai.com/api/.

Ouellette, P., Sciortino, A., Nargesian, F., Bashardoost, B. G., Zhu, E., Pu, K. Q., and Miller, R. J. Ronin: data lake exploration. *Proceedings of the VLDB Endowment*, 14(12), 2021.

Pasupat, P. and Liang, P. 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)*, pp. 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142.

Peeters, R. and Christian, B. Dual-objective fine-tuning of bert for entity matching. *Proceedings of the VLDB Endowment*, 14:1913–1921, 2021.

Qwen. Qwen, 2023. URL https://tongyi.aliyun.com/qianwen/.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment*, 10:1190–1201, 2017.

Shichao, Z., Jilian, Z., Xiaofeng, Z., Yongsong, Q., and Chengqi, Z. Missing value imputation based on data clustering. *Transactions on computational science*, pp. 128–138, 2008.

Singh, R., Meduri, V. V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J.-A., Solar-Lezama, A., and Tang, N. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment*, 11(2):189–202, 2017.

Stefan, H., Alejandro, B., Hunter, L., Monica, A., Xiaoyi, J., and David, S. Tabllm: Few-shot classification of tabular data with large language models. *arXiv preprint arXiv:2210.10723*, 2022.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

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

Trummer, i. Db-bert: a database tuning tool that "reads the manual". *SIGMOD*, 2022b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, P., Zeng, X., Chen, L., Ye, F., Mao, Y., Zhu, J., and Gao, Y. Promptem: prompt-tuning for low-resource generalized entity matching. *arXiv preprint arXiv:2207.04802*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Wu, R., Zhang, A., Ilyas, I., and Rekatsinas, T. Attention-based learning for missing data imputation in holoclean. *Proceedings of Machine Learning and Systems*, 2:307–325, 2020.

Wu, Z., Yu, P., Yang, P., Zhu, R., Han, Y., Li, Y., Lian, D., Zeng, K., and Zhou, J. A unified transferable model for ml-enhanced dbms. *arXiv preprint arXiv:2105.02418*, 2021.

Yang, J., Gupta, A., Upadhyay, S., He, L., Goel, R., and Paul, S. Tableformer: Robust transformer modeling for table-text encoding. *arXiv preprint arXiv:2203.00274*, 2022.

Yeye, H., Xu, C., Kris, G., Yudian, Z., Vivek, N., and Surajit, C. Transform-data-by-example (tde) an extensible search engine for data transformations. *Proceedings of the VLDB Endowment*, 11(10):1165–1177, 2018.

Zhao, C. and He, Y. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*, pp. 2413–2424, 2019.

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

Appendices

A Prompt Template

We provide a running example to explain how our method automatically generates the desired cloze question of target task based on the in-context learning of LLMs. The prompt example is as follows:

(Input to LLMs):

Write the claim as a cloze question.
Claim:
The task is data imputation which produces the missing data with some value to retain most of the data. The context is Wenham, Marysville, and Westmont are cities in the United States, identified by the ISO3 code USA. The target is city:New Cassel, iso3:USA, country:?
Cloze question:
Wenham, Marysville, and Westmont are cities in the United States, identified by the ISO3 code USA. New Cassel is the name of a city whose ISO3 country code is USA. New Kassel belongs to the country _.
Claim:
The task is data transformation which is the process of converting data from one format to another required format within a record. The context is data before transformation: 20000101 data after transformation: 2000-01-01. The target is 19990415:?
Cloze question:
20000101 can be transformed to 2000-01-01, and 19990415 can be transformed to _.
Claim:
The task is error detection which detect attribute error within a record in a data cleaning system. The context is the address of 2505 u s highway 431 north is not an error, the county name of mxrshxll is an error. The target is whether there is an error in city:sheffxeld.
Cloze question:
The address ”2505 U.S. Highway 431 North” has no error, whereas the county name ”mxrshxll” contains an error. It is required to identify if there is an error in the city name ”sheffxeld”. Is there an error in the city name? Yes or No. _
Claim:
The task is entity resolution which is the process of predicting whether two records are referencing the same real-world thing. The context is A is the Punch! Home Design Architectural Series 4000 v10, manufactured by Punch! Software, is priced at $199.99. B is The Punch Software 41100 Punch! Home Design Architectural Series 18, manufactured by Punch Software, is priced at $18.99. The target is are A and B the same?
Cloze question:

(Output of LLMs):
Punch! Home Design Architectural Series 4000 v10, manufactured by Punch! Software, is priced at $199.99, whereas Punch Software 41100 Punch! Home Design Architectural Series 18, also manufactured by Punch Software, is priced at $18.99. Are these two products the same? Yes or No. _

B Case Study

For case study, we present the final results of FM (random setting), FM (manual setting), and our retrieval-based UniDM. For FM method (random setting), we have the final prompt and the output of LLMs as follows:

(Input to LLMs):
name: anthonys. addr: 3109 piedmont rd. just south of peachtree rd.. phone: 404/262-7379. type: american. What is the city? atlanta
name: rose pistola. addr: 532 columbus ave.. phone: 415/399-0499. type: italian. What is the city? san francisco
name: american place. addr: 2 park ave. at 32nd st.. phone: 212/684-2122. type: american. What is the city? new york
name: ruth's chris steak house (los angeles). addr: 224 s. beverly dr.. phone: 310-859-8744. type: steakhouses. What is the city?

(Output of LLMs): los angeles
Ground Truth: Beverly Hills

For FM method (manual setting), we have the final prompt and the output of LLMs as follows:

(Input to LLMs):
name: oceana. addr: 55 e. 54th st.. phone: 212/759-5941. type: seafood. What is the city? new york
name: oceana. addr: 55 e. 54th st.. phone: 212-759-5941. type: seafood. What is the city? new york city
name: ruth's chris steak house (los angeles). addr: 224 s. beverly dr.. phone: 310-859-8744. type: steakhouses. What is the city?

(Output of LLMs): los angeles
Ground Truth: Beverly Hills

For our retrieval-based UniDM, we have the final prompt and the output of LLMs as follows:

(Input to LLMs):
The name of the place is Belvedere. The address is 9882 Little Santa Monica Blvd. The city is Beverly Hills.
The name of the grill is Jack Sprat's Grill and its address is 10668 W. Pico Blvd. in the city of Los Angeles.
The name of the establishment is Border Grill, located on 4th Street in Los Angeles.
Ruth's Chris Steak House (Los Angeles) is located at 224 S. Beverly Dr. Ruth's Chris Steak House (Los Angeles) is located in the city of _.

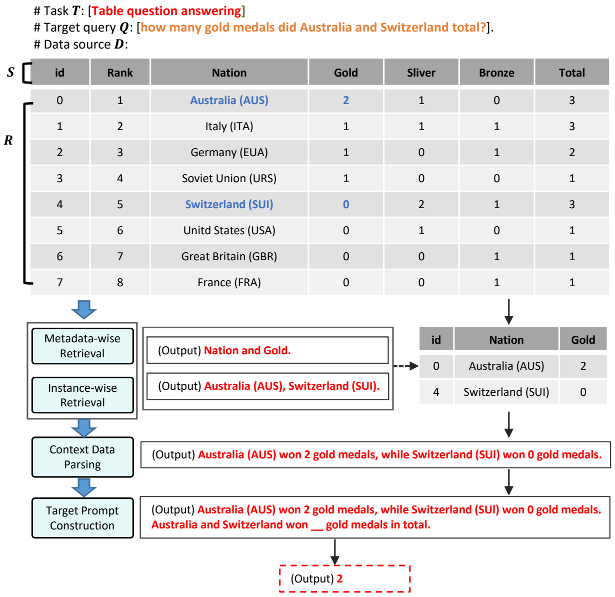
(Output of LLMs): Beverly Hills
Ground Truth: Beverly Hills

C 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 (Pasupat & Liang, 2015). 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 $\mathcal{C}$ 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 $\mathcal{C}$ is serialized and then parsed into a natural text representation $\mathcal{C}'$. 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.



# Task T: [Table question answering]
# Target query Q: [how many gold medals did Australia and Switzerland total?].
# Data source D:
S
R
Metadata-wise Retrieval
Instance-wise Retrieval
Context Data Parsing
Target Prompt Construction

id	Rank	Nation	Gold	Sliver	Bronze	Total
0	1	Australia (AUS)	2	1	0	3
1	2	Italy (ITA)	1	1	1	3
2	3	Germany (EUA)	1	0	1	2
3	4	Soviet Union (URS)	1	0	0	1
4	5	Switzerland (SUI)	0	2	1	3
5	6	Unitd States (USA)	0	1	0	1
6	7	Great Britain (GBR)	0	0	1	1
7	8	France (FRA)	0	0	1	1

(Output) Nation and Gold.

(Output) Australia (AUS), Switzerland (SUI).

id	Nation	Gold
0	Australia (AUS)	2
4	Switzerland (SUI)	0

(Output) Australia (AUS) won 2 gold medals, while Switzerland (SUI) won 0 gold medals.

(Output) Australia (AUS) won 2 gold medals, while Switzerland (SUI) won 0 gold medals. Australia and Switzerland won __ gold medals in total.

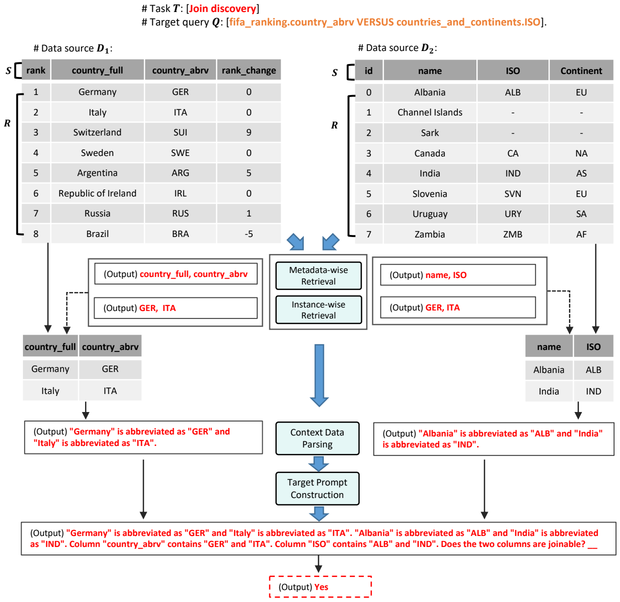
(Output) 2

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

D 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



# Task T: [Join discovery]
# Target query Q: [fifa_ranking.country_abrv VERSUS countries_and_continents.ISO].
# Data source D_1:
# Data source D_2:
S
R
Metadata-wise Retrieval
Instance-wise Retrieval
Context Data Parsing
Target Prompt Construction

rank	country_full	country_abrv	rank_change
1	Germany	GER	0
2	Italy	ITA	0
3	Switzerland	SUI	9
4	Sweden	SWE	0
5	Argentina	ARG	5
6	Republic of Ireland	IRL	0
7	Russia	RUS	1
8	Brazil	BRA	-5

id	name	ISO	Continent
0	Albania	ALB	EU
1	Channel Islands	-	-
2	Sark	-	-
3	Canada	CA	NA
4	India	IND	AS
5	Slovenia	SVN	EU
6	Uruguay	URY	SA
7	Zambia	ZMB	AF

(Output) country_full, country_abrv

(Output) GER, ITA

(Output) name, ISO

(Output) GER, ITA

country_full	country_abrv
Germany	GER
Italy	ITA

name	ISO
Albania	ALB
India	IND

(Output) "Germany" is abbreviated as "GER" and "Italy" is abbreviated as "ITA".

(Output) "Albania" is abbreviated as "ALB" and "India" is abbreviated as "IND".

(Output) "Germany" is abbreviated as "GER" and "Italy" is abbreviated as "ITA". "Albania" is abbreviated as "ALB" and "India" is abbreviated as "IND". Column "country_abrv" contains "GER" and "ITA". Column "ISO" contains "ALB" and "IND". Does the two columns are joinable? __

(Output) Yes

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

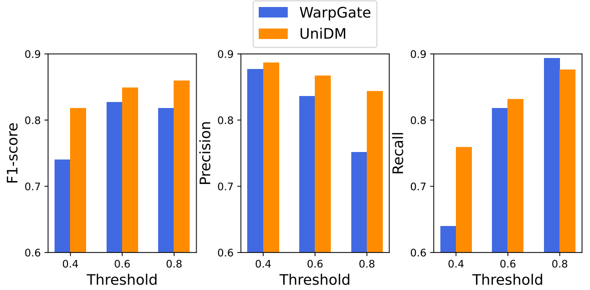


Figure 5. F1-score, precision and recall on join discovery task.

and records $\mathcal{C}_1$ and $\mathcal{C}_2$ between the two tables. The context snapshots $\mathcal{C}_1$ and $\mathcal{C}_2$ are serialized and then parsed into natural text representations $\mathcal{C}'_1$ and $\mathcal{C}'_2$ 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.

For experiment on join discovery task, we use NextiaJD (Flores Herrera et al., 2021) 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 (Cong et al., 2022), 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 (Cong et al., 2022) under various threshold values. This exhibits that UniDM's potential to manipulate data cross sources.

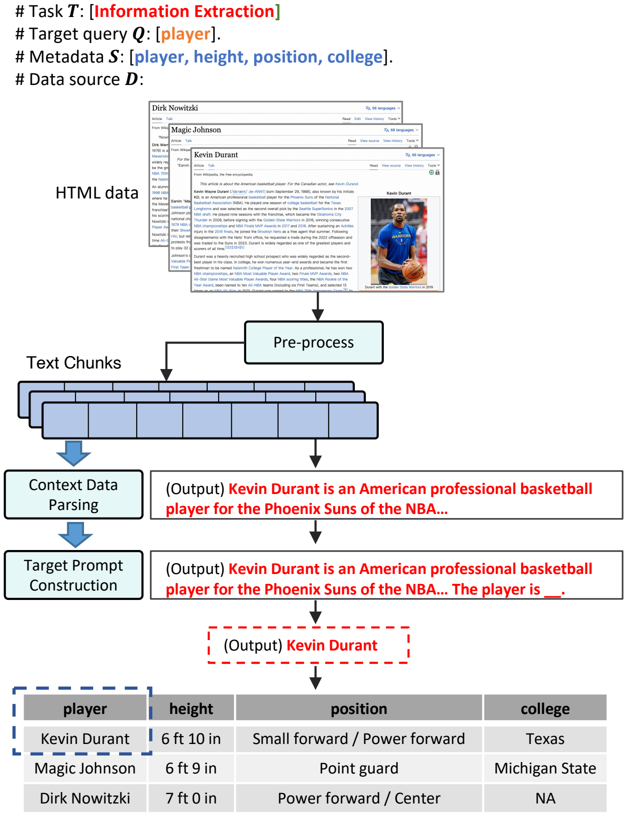
E Information Extraction Task

Table 11. Text F1-score on information extraction task.

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

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).

Figure 6 gives an illustrative example of information extraction task. 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. 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").



# Task *T*: [Information Extraction]
# Target query *Q*: [player].
# Metadata *S*: [player, height, position, college].
# Data source *D*:
HTML data
Dirk Nowitzki
Magic Johnson
Kevin Durant
Pre-process
Text Chunks
Context Data Parsing
Target Prompt Construction
(Output) Kevin Durant is an American professional basketball player for the Phoenix Suns of the NBA...
(Output) Kevin Durant is an American professional basketball player for the Phoenix Suns of the NBA... The player is __.
(Output) Kevin Durant

player	height	position	college
Kevin Durant	6 ft 10 in	Small forward / Power forward	Texas
Magic Johnson	6 ft 9 in	Point guard	Michigan State
Dirk Nowitzki	7 ft 0 in	Power forward / Center	NA

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

For information extraction, we use SWDE benchmark in (Hao et al., 2011) and choose NBAplayer dataset. This dataset includes complex HTML from the NBA Wikipedia pages for NBA players. Following the previous method, we consult closed information extraction setting, where a pre-defined schema is provided and UniDM is used to populate the table. Ground truth value is available for the target attributes. As shown in Table 11, 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.

F Algorithm

Algorithm 1 represents the main function for data manipulation tasks, executed by UniDM, in pseudo-code. The input integrates a data lake for a target data manipulation task, with user-provided parameters. These parameters include a subset of schema, a subset of records, a task description, and a target query. First, we automatically retrieve context information according to the task and the input query. This module uses LLMs to select valuable attributes for the task and the target attribute, and then perform a fine-grained filtering on all records to identify relevant ones w.r.t. target records. Next, the context information $\mathcal{C}$, represented in a tabular form, is transformed into a more effective format C' for LLMs. The target prompt construction is to find an effective prompt to organize the task description T, the context information C' and the query Q. Finally, this prompt is fed into LLMs to yield the final answer of our task.

Algorithm 1 Unified Framework for Data Manipulation (UniDM)

```
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 $S_i$, a task description $T$, a target
    query $Q$.
if Context Retrieval then
    // Metadata-wise retrieve
    $p_{rm} \leftarrow$ prompt_$p_{rm}(T, Q, S)$
    $S^m \leftarrow$ LLM$(p_{rm})$
    // Instance-wise retrieve
    $p_{ri} \leftarrow$ prompt_$p_{ri}(T, Q, R)$
    $\{score_i\}_{i=1}^m \leftarrow$ LLM$(p_{ri})$
    $R^m \leftarrow$ top-k$(\{score_i\})$
    // Select records based on the retrieved results
    $\mathcal{C} \leftarrow R^m[S^m]$
else
    $\mathcal{C} \leftarrow D_i$
end if
serialize$(C) = \{s : r[s]\}, (\forall r[s] \in C)$
$\mathcal{V} \leftarrow$ serialize$(C)$
if Context Data Parsing then
    // Parse the data into a natural text representation
    $p_{dp} \leftarrow$ prompt_$p_{dp}(\mathcal{V})$
    $\mathcal{C}' \leftarrow$ LLM$(p_{dp})$
else
    $\mathcal{C}' \leftarrow \mathcal{V}$
end if
// Recursively uses the LLM to reformat the data task.
$p_{cq} \leftarrow$ prompt_$p_{cq}(T, Q, C')$
$p_{tg} \leftarrow$ LLM$(p_{cq})$
$Y \leftarrow$ LLM$(p_{tg})$
Return $Y$
```