Large Language Models as Pretrained Data Engineers: Techniques and Opportunities

Yin Lin, Bolin Ding, Jingren Zhou
Alibaba Group
{yin.lin, bolin.ding, jingren.zhou}@alibaba-inc.com

Abstract

The advent of large language models (LLMs) has yielded promising results across various fields. With their ability to understand, retrieve, synthesize information, and perform advanced reasoning, LLMs have shown significant potential for facilitating and automating data management. We propose that the extensive knowledge embedded within these models enables them to serve as pre-trained data engineers, enhancing both the effectiveness and scope of data tasks.

In this paper, we illustrate an architectural framework for integrating LLMs into data engineering workflows, aiming to enhance the applicability and usability of LLM-assisted data engineering solutions. We explore key techniques and opportunities across three critical stages: (a) data wrangling, to simplify and optimize data preparation and transformation; (b) analytical querying, to extend querying capabilities and interfaces in data systems; and (c) table augmentation, to enhance the original tables with additional data, improving the performance of data-centric tasks like machine learning.

1 Introduction

Large Language Models (LLMs) have recently demonstrated remarkable progress across a variety of tasks, including natural language processing, question answering, code generation, and information retrieval [3, 57, 8, 75, 60, 25]. These models, trained on extensive data corpora, encompass vast amounts of integrated knowledge and possess advanced reasoning capabilities [79, 80, 81]. Moreover, they have shown strong cross-task generality on a wide range of natural language tasks.

Building on these impressive capabilities, the rapid development of LLMs has motivated data researchers to reconsider traditional challenges in data management [54, 21, 72]. Data engineering, which includes the preparation, analytical querying, and enrichment of data, demands specialized domain expertise and is often time-consuming, lacking a one-size-fits-all solution due to the complexity of data-related tasks [65]. For example, considering a data imputation task, imputing the categories of items might require row-by-row interactions to understand contextual nuances, whereas imputing the Body Mass Index (BMI) can be completed by using standardized formulas that rely on height and weight. Similarly, to answer a user question such as "*which item is the most positively reviewed*" might require semantic parsing of each row, whereas answering "*what are the monthly sales for MacBook Pro*" may involve translating the question into an aggregate query to obtain the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

result. Pioneering studies [21, 85, 52, 8] have illustrated that well-crafted *prompts* can effectively guide LLMs to achieve state-of-the-art performance in specific data engineering tasks, such as data imputation, entity matching, and Text2SQL. However, integrating LLMs into complex and heterogeneous data engineering workflows remains a challenging endeavor.

First, designing a single prompt for each distinct data preparation task is suboptimal, as data wrangling generally necessitates a multi-step process involving diverse tasks. Employing domain-specific solutions customized for each data set and problem may not be uniformly effective. Secondly, while LLMs have empowered data querying with strong semantic reasoning capabilities and world knowledge [79, 80, 81], challenges arise in designing effective interfaces for integrating LLMs into data systems to facilitate efficient analysis. Recent efforts have focused on developing SQL-like [45, 4, 68, 86] and Pandas-like [58] programming interfaces to improve LLM usability across diverse systems. Additionally, to assist non-experts who may lack the ability to write code and queries, facilitating the translation of natural language questions into executable queries [82, 64, 77] is also critical for enhancing accessibility. Lastly, to enhance the performance of data-centric applications such as machine learning, augmenting the original tabular data can be highly beneficial. However, challenges remain to effectively utilize LLMs for this purpose. LLMs have the potential to enrich original datasets by retrieving relevant information [34, 20, 33] and performing feature engineering [46, 31] to generate useful features.

In this paper, we introduce an architectural framework for integrating the powerful capabilities of LLMs into data engineering, focusing on three key stages: *data wrangling*, *analytical querying*, and *table augmentation for machine learning*. Within each stage, we review current research and demonstrate how LLMs can facilitate and potentially expand the scope of data-related tasks.

In particular, we elaborate on three systems we have recently developed: UniDM [61], which proposes a unified framework for data wrangling; DAIL-SQL [22], a Text2SQL solution that systematically examines both in-context learning and supervised fine-tuning to optimize performance; and SMARTFEAT [46], which automates feature engineering for machine learning by identifying and applying appropriate transformations on the original table. Furthermore, we envision future directions to further extend LLM-assisted data engineering applications, including (1) developing automated, unified systems for efficient end-to-end data preparation, (2) enhancing querying capabilities for unstructured data and world knowledge in data systems through flexible querying interfaces, and (3) constructing automated solutions for machine learning data processing.

2 Integrating LLM Modules in Data Engineering

Data engineering often contains complex and labor-intensive tasks that traditionally require substantial engineering effort. Figure 1 outlines an architectural framework for integrating LLM modules into data engineering workflows, centered on three stages: 1) *data wrangling*, (2) *analytical querying*, and (3) *table augmentation for machine learning*.

Data Wrangling. Real-world data are often heterogeneous, incomplete, or contain errors, rendering them unsuitable for direct analytical or modeling tasks. Data wrangling typically encompasses tasks such as entity matching [19], error detection [1, 30], and data imputation [26], enabling the integration, cleaning, and transformation of raw data into structured formats suited to downstream applications. To facilitate complex data wrangling, the framework incorporates LLMs into *data wrangling operators*. In these operators, LLMs are prompted with the *task parameters*, such as the task name, task query, and relevant *contextual inputs* from the dataset. The LLMs are then applied to each row to predict the next word for completing tasks [54, 72, 61], or to the entire dataset to generate code [13] to process data more efficiently.

Analytical Querying. Data engineers and analysts often query datasets to extract meaningful insights. The framework extends the traditional definition of relational querying by incorporating LLM capabilities, which enable querying not only of structured data but also of unstructured textual content, documents, and world knowledge through various querying interfaces. Operators leveraging LLMs facilitate access to extensive corpora of

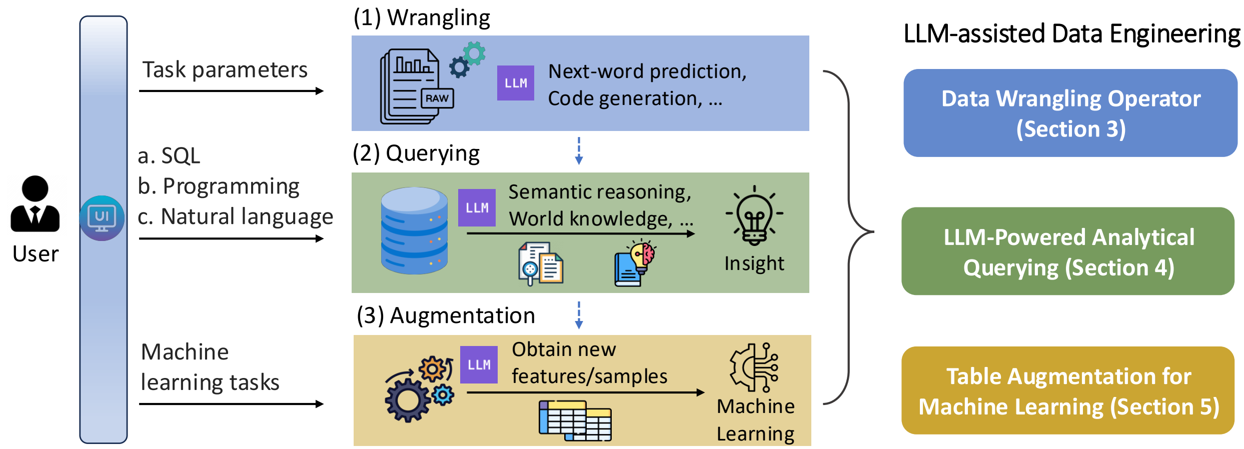

Figure 1: Integrating LLM-based modules into data engineering workflows: (1) facilitating data wrangling tasks, (2) supporting complex analytical querying through multiple interfaces, and (3) augmenting original tabular data for improved machine learning performance.

unstructured text [68, 86] and can even function as knowledge storage mechanisms [7]. The framework describes multiple interfaces for utilizing LLMs in analytical querying, including *SQL interfaces* [70, 15, 5, 45, 4, 68, 86], *programming interfaces* [4, 58], and *natural language interfaces* [48, 7], in which LLMs are commonly embedded as SQL extensions or as Pandas-like data science APIs. They can also translate natural language (NL) into logical operators [49, 82, 64, 77, 47, 62] and enable interactive, conversational NL-based interfaces [48, 7].

Table Augmentation for Machine Learning. Table augmentation can enrich original datasets with new features or samples, thereby improving performance in data-centric tasks such as machine learning. However, this process often relies heavily on domain expertise and demands substantial manual effort from data scientists. Given a machine learning task, the framework leverages LLMs to support two types of table augmentation [14]: generation-based augmentation [46, 31], which uses LLMs to iteratively generate semantically meaningful features for original datasets, and retrieval-based augmentation [34, 27], which retrieves additional external data.

In the following sections, we review the detailed techniques for integrating LLM modules in each stage. We then discuss unique opportunities to expand the scope and improve the usability of LLM-assisted data engineering solutions.

3 Data Wrangling with LLMs

Data wrangling involves a series of data preparation steps that transform datasets into formats suitable for analysis and modeling. To leverage LLMs in facilitating complex data preparation tasks, we first formalize these tasks using a generalized *data wrangling operator*, defined as follows:

Definition 3.1 (Data Wrangling Operator) *Let $\mathcal{D}$ be a dataset requiring data wrangling and let T denote the task parameters. An LLM-based data wrangling operator* opr_{LLM} *is defined as a function*

$$\mathsf{opr}_{LLM} : (\mathcal{D}, T) \rightarrow \mathcal{D}_o,$$

where LLM is the selected large language model. Given T and $\mathcal{D}$, the operator constructs prompts to instruct the LLM. The outputs generated by the LLM are applied to $\mathcal{D}$, resulting in a clean and structured dataset $\mathcal{D}_o$.

Data Preparation Tasks. Data wrangling is rarely a single-step process; it usually involves multiple individual preparation steps, each implemented by a data wrangling operator. These tasks can be categorized into three broad categories based on their purposes: *data integration* [17], *data cleaning* [63], and *data transformation* [29, 35]. Data integration involves discovering and combining data from various sources, which typically includes tasks such as schema matching [71], entity resolution [10, 78, 44], and join/union discovery [34, 20, 33]. Data cleaning aims to remove or replace inaccurate data values with more accurate ones. It typically includes tasks such as deduplication, error detection [1, 30], and data imputation [26]. Data transformation involves restructuring and filtering data, including tasks such as changing schema [35], and removing irregular rows or values [26]. Recent research has explored the use of LLMs to address one or more of these tasks, showing promising results and, in some cases, achieving state-of-the-art performance.

Prompts. LLMs have shown remarkable proficiency in text-intensive tasks. Researchers have also demonstrated that LLMs trained to predict the next word (e.g., "*Mary has a little*" – "*lamb*") can be adapted to data-related tasks by providing natural language descriptions of those tasks [54]. For example, to perform entity matching, one might ask an LLM "*Are iPhone 16 Pro and iPhone 16 Pro Max the same?*" and obtain the answer "*No.*". Moreover, models such as GPT-3 [9], GPT-4 [57], and LLaMA [75] also possess robust code-generation capabilities like human programmers.

Therefore, within data wrangling operators, the LLM is guided by *prompts* derived from task parameters, such as the task name, task query, and contextual inputs extracted from the data. These prompts can be structured in various forms to perform row-wise executions using the ability of LLMs in performing next-word prediction. For example, for a data imputation task, the prompt can be formatted such as a *question* (e.g., "`What is the timezone of Copenhagen?`"), a *cloze* (e.g., "`Copenhagen is in the ____ timezone.`"), or a *completion* (e.g., "`The timezone of Copenhagen is...`"). Furthermore, prompts can also be designed to synthesize code for domain-specific solutions [55, 52, 13], such as "write a Python regular expression to extract dates in the format 'YYYY-MM-DD'." The operator then incorporates a parser to extract outputs from LLMs and may apply additional transformations to the original datasets, thereby producing the output data.

3.1 Unifying LLM-based Data Wrangling

While the integration of LLMs into data wrangling has demonstrated exceptional performance in various data preparation tasks [54, 44, 72, 84], these approaches often rely on specifically tailored prompt designs for each task, which limits their usability as end-to-end platforms that encompass multiple data preparation steps.

In this subsection, we explore the generality of data preparation tasks based on Definition 3.1. We propose that disparate data preparation tasks could be unified, applying a general procedure to solve different tasks. To this end, we introduce a unified system, UniDM [61], which generalizes data preparation tasks into three main steps as shown in Figure 2. The core objective of UniDM is to unify diverse data tasks by developing a general mechanism that transforms task parameters T, and contextual inputs derive from $\mathcal{D}$ into a prompt understandable by LLMs to complete the task.

The first step, **context retrieval**, extracts the relevant context from the dataset $\mathcal{D}$ necessary for solving the task. Prior approaches often require users to manually specify the context [54, 6] or rely on attribute similarity to identify useful records [2, 51]. However, these methods can be less effective when dealing with a diverse range of complex data preparation tasks. In contrast, UniDM introduces an automated retrieval process that leverages LLMs through two specialized prompt templates. These templates guide LLMs to determine the relevant *attributes* and *rows* in $\mathcal{D}$ required for completing the task T. For example, consider a data imputation task shown in Figure 2, where the goal is to impute the timezone for Copenhagen. In this case, the first prompt identifies the "`country`" attribute as essential for inferring the timezone and the second prompt selects a subset of relevant records to guide this task (e.g., via few-shot prompting). One such record might be: "`city: Alicante, country: Spain, timezone: Central European Time`".

The second step, **context data parsing**, transforms the retrieved context from a tabular format into a textual representation, enabling LLMs to better capture the semantics. Unlike conventional methods that *serialize* input rows into a simple text string [54, 72], UniDM employs a prompt template to instruct LLMs to create a

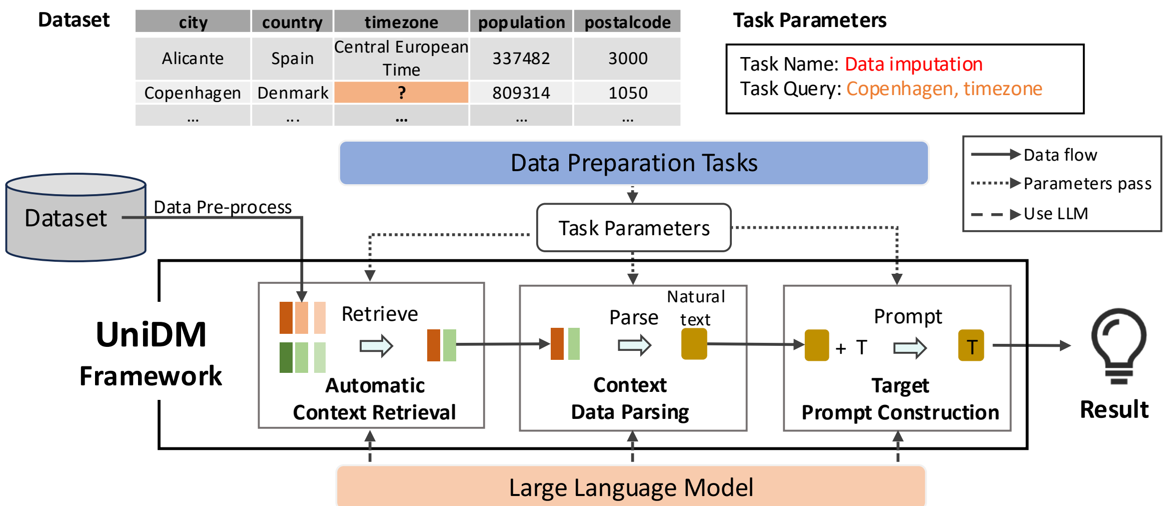

Figure 2: An overview of the UniDM framework.

semantically rich text representation of the context. For example, the tabular record "`city: Alicante, country: Spain, timezone: Central European Time`" is transformed into: "`Alicante is a city in Spain and is in the Central European timezone.`".

The final step is **target prompt construction**, where UniDM synthesizes the task parameters T, including the task name, task query, and the contextual inputs of dataset $\mathcal{D}$ obtained from the previous two steps, into a final prompt for the LLMs to generate results. The construction of this final prompt is also relied on LLMs. For the data imputation task, the final prompt could take the form of a cloze-style prompt: "`This task involves imputing the missing value`...; `The context is: Alicante is a city in Spain and is in the Central European timezone`...; `Copenhagen is a city in Denmark and is in the ____ timezone.`" This prompt is then used to generate the final result for the data preparation task.

By abstracting these steps, UniDM unifies disparate data preparation tasks, offering a systematic approach to leverage LLMs effectively and generalizing across multiple task types. In a similar vein, another recent framework, CHORUS [38], presents a unified approach for synthesizing data discovery and exploration tasks. Unlike UniDM's three-step procedure, CHORUS decomposes prompts into six fixed components and employs specialized templates and context retrieval methods to automatically construct the final prompt. It has shown highly effective performance for data discovery tasks such as table-class detection, column-type annotation, and join-column prediction.

3.2 LLM-as-a-Compiler for Data Wrangling

While UniDM offers a unified framework for automating data wrangling effectively, a limitation lies in its reliance on *row-wise execution*, which can become inefficient and costly for large datasets due to the need for LLM invocations on each record. To overcome this limitation, more recent work, SEED [13], proposes an improved approach by introducing an optimizer that automatically selects from four LLM-assisted modules: *CodeGen*, which generates code; *CacheReuse*, which reuses previous LLM query results; *ModelGen*, which distills an LLM into a smaller machine learning model; and *LLM*, which directly generates answers. By combining these modules, SEED automatically produces a hybrid data wrangling solution that achieves performance comparable to row-wise execution while dramatically reducing the number of LLM calls.

4 Data Analytics with LLMs

Traditional relational queries enable users to perform scalable and accurate analyses on structured data through formal querying languages. However, much of the information that data users want to query [50] resides not only in structured datasets, but also in various types of unstructured sources such as text, documents, and even in the vast world knowledge encoded within LLMs. Moreover, traditional systems often assume users have the technical expertise to write complex queries, which can be a barrier for non-expert users. For example, a restaurant owner may want to analyze customer reviews (e.g., *identifying the most positively reviewed dishes in Japanese restaurants in New York City*) but may lack the skills to write complex queries or code.

LLMs have demonstrated the ability to understand, extract, and answer questions using unstructured data and world knowledge through methods such as retrieval-augmented generation (RAG) [40]. Recent research [7] has demonstrated the promising potential of integrating LLM capabilities into data systems to combine strong semantic reasoning and LLM knowledge with the efficient computational query execution capabilities of traditional relational data systems.

In this section, we provide an expanded definition of analytical querying that extends the relational querying definition $Q : (\mathcal{D}_s, \mathsf{q}) \rightarrow \mathcal{R}$, where q represents a query in the formal query language, and $\mathcal{D}_s$ is the structured dataset, and $\mathcal{R}$ is the query result. The LLM-powered analytical querying definition includes support for unstructured data, integrates knowledge from LLMs, and accommodates diverse interfaces, as follows:

Definition 4.1 (Analytical Querying) *Let $\mathcal{D}_s$ denote the structured dataset, $\mathcal{D}_u$ denote the unstructured text data that may be leveraged for question answering, and $\mathcal{D}_{LLM}$ denote the world knowledge encoded within a large language model LLM. Given a user question q^+, an LLM-powered analytical querying is expressed as*

$$Q_{LLM} : (\mathcal{D}_s, \mathcal{D}_u, \mathcal{D}_{LLM}, \mathsf{q}^+) \rightarrow \mathcal{I},$$

where the output $\mathcal{I}$ represents the insights derived from the query result.

In Definition 4.1, the LLM-powered analytical querying not only enables analysis of structured data $\mathcal{D}_s$ but also enables the extraction and processing of information from unstructured text data $\mathcal{D}_u$ to answer the question. It also facilitates querying access and exploits the LLM knowledge through $\mathcal{D}_{LLM}$. In addition, the querying interface is not restricted to formal query languages; instead, the user question q^+ supports various formats according to the data system and user requirements, including SQL queries [45, 4, 68, 86], data science code snippets [4, 58], and natural language inputs [82, 64, 77, 47, 62, 48, 7]. It may also incorporate a conversational agent to translate query results into more easily interpretable insights in natural language [7, 48].

To support this extended analytical querying, recent research efforts have focused on enhancing or developing next-generation data systems [49] that leverage the capabilities of LLMs. In the following subsections, we will explore the specific mechanisms through which these systems achieve the goal:

- **Building declarative querying interfaces.** These interfaces enable users to ask questions based on their needs rather than focusing on the technical execution details [49]. They can express their queries in a more intuitive and user-friendly manner, which the systems then translate into the necessary operations.

- **Translating natural language questions into queries.** These systems may leverage semantic parsers driven by LLMs to convert user questions expressed in natural language into executable queries [48], which bridges the gap between user intent and system execution.

- **Expending the set of logical operators and enabling optimizations.** These systems offer an enriched set of logical operators that augment traditional relational operations [58]. By incorporating LLM-powered semantic data processing and information extraction capabilities, these operators work with traditional relational operators to facilitate optimization and effective query planning.

4.1 Interfaces for Analytical Querying

In real-world analytical querying applications, there are various design considerations when integrating LLMs into data analytics systems. State-of-the-art research primarily focuses on three types of user interfaces: *SQL interface*, *programming interface* and *natural language interface*.

SQL Interface. To incorporate user questions into executable operations within database management systems (DBMS), modern DBMS vendors have explored integrating LLMs as extensions of SQL user-defined functions (UDFs). For example, systems like BigQuery [70], Databricks [15], and Redshift [5] uses AI modules to perform information extraction based on the prompted user question. However, these LLM UDFs typically support only row-by-row execution, making them inefficient and prohibitively expensive for large datasets.

To address the challenge of querying both structured and unstructured data, several systems have proposed transforming unstructured data and LLM knowledge into a structured format compatible with SQL. For example, ZenDB [45] automatically extracts semantic hierarcal structure from text documents allowing users to impose a schema on their documents and query the document with SQL interface. Evaporate [4] generates structured views of data from input documents by employing efficient entity extraction techniques on semi-structured data. GALIOS [68] enables users to query large language models via an SQL interface, executing some parts of the query plan with prompt-based interactions to retrieve data from the LLM. Similarly, HQDL [86] extends SQL beyond the data at hand, using LLMs to answer questions that require additional context.

Programming Interface. Another approach to integrating LLMs into analytical querying is through programmatic interfaces provided by data science platforms, such as packages in Python, which enable developers to construct flexible data analysis pipelines. For example, Palimpzest [4] allows users to declaratively query text and images using an API, similar to querying tables in a relational database. The LOTUS system [58] provides a Pandas-like interface enriched with *semantic operators*, such as semantic filtering, ranking, and aggregation. This enables developers to make use of both traditional relational functionalities and advanced LLM-driven semantic reasoning capabilities in data analytics.

Natural Language Interface. Recent advancements have demonstrated that LLMs can translate natural language questions into executable relational queries with high accuracy [82, 64, 77, 47, 62]. Research has also explored developing interactive or conversational agents that provide end-to-end query support for non-expert users. For example, SUQL [48] builds a conversational interface to answer questions over semi-structured data by executing queries derived from user utterances that correspond to specific query intentions. Similarly, TAG [7] proposes a general-purpose query model that translates natural language inputs into queries and generates answers in natural language based on the query results. Although natural language interfaces greatly improve usability for non-expert users, they inherently carry ambiguities [41]. Therefore, we believe that data systems should not rely solely on natural language as the querying interface. An ideal design would map natural language inputs to expressive SQL-like or programming languages, ensuring accurate and efficient execution within the data system. Systems such as SUQL and TAG exemplify this approach by first interpreting and generating SQL queries from natural language questions, then executing these queries and presenting the results in natural language.

4.2 Text2SQL Semantic Parsers

A critical challenge in bridging user intentions to executable database queries lies in accurately translating natural language questions into executable queries. While Text2SQL has long been a focus in both natural language processing and database research, LLMs have recently emerged as a transformative paradigm for this task [77, 47, 62].

In this subsection, we present DAIL-SQL [22] which provides an integrated tool for improving LLM-based Text2SQL solutions. It systematically explores two key directions: *prompt engineering for in-context learning* on closed-source LLMs and *supervised fine-tuning (SFT)* on open-source LLMs.

Prompt Engineering. Effective prompt design is crucial to promote accurate SQL generation from LLMs [9]. DAIL-SQL examines both question representation and example selection to optimize prompt design.

- *Question representation:* This involves representing user questions and database information in the most informative format (e.g., natural text, code-like schemas). We extensively evaluate five widely adopted approaches [73, 12, 56, 53, 47]. We find that certain representations yield higher execution accuracy than others. In particular, using the OpenAI demonstration prompts [56] or the code representation [53] that includes comprehensive schema details tends to perform better. DAIL-SQL adopts the code representation and finds that including supplementary details, such as foreign key information and implication rules like the "*with no explanation*" implication, also helps improve execution accuracy.

- *Example selection and organization:* The in-context learning ability of LLMs [18] is also helpful in improving Text2SQL performance with carefully selected examples. DAIL-SQL demonstrates that few-shot prompting is most effective when examples are similar to the user's query. To identify effective examples, DAIL-SQL masks domain-specific terms in both the natural language questions and the pre-generated SQL for the user query and candidate examples. It then ranks the candidates based on question and query similarity. Experimental results indicate that selecting examples based on both question and SQL similarity consistently outperforms random selection. Furthermore, while including full example details may yield optimal performance, we find that for powerful LLMs like GPT-4, retaining only the question-SQL mapping is an efficient and effective approach.

Supervised Fine-Tuning (SFT). Beyond prompting, DAIL-SQL explores fine-tuning open-source LLMs (e.g., LLaMA variants [75, 73, 74, 87, 67, 76]) to improve Text2SQL performance. Empirical results show that supervised fine-tuning is essential to achieve competitive accuracy. With carefully curated training data (e.g., from historical Text2SQL workloads), DAIL-SQL shows that fine-tuned open-source models exhibit strong potential for Text2SQL tasks. Unlike in-context learning with closed-source LLMs, fine-tuned open-source LLMs do not learn from contextual examples, e.g., queries in the current data analytical pipeline, but we can always applying in-context learning on fine-tuned LLMs to incorporate the learned experience and characteristics of the workload in an online data analytical pipeline.

To evaluate the performance of Text2SQL techniques, a variety of benchmarks have been proposed, such as Spider [83] and Bird [42], along with metrics like execution accuracy and query efficiency. DAIL-SQL demonstrates that both prompt engineering and supervised fine-tuning can significantly enhance Text2SQL's execution accuracy, providing valuable insights for data systems aiming to translate natural language questions into precise and reliable SQL queries. More recently, XiYan-SQL [23], a state-of-the-art Text2SQL system, further improved the benchmark performance by integrating schema linking techniques to retrieve only relevant columns for a given natural language question, designing a more efficient schema representation, and implementing an ensemble strategy for LLM in candidate generation and selection.

However, Text2SQL research has primarily focused on relational queries over structured data, which covers only a fraction of the questions posed by real-world users [7]. The incorporation of LLMs into data systems brings significant opportunities to leverage world knowledge and semantic reasoning, thereby extending relational queries through a broader set of logical operators, as we will describe next.

4.3 Logical Operators

Executing analytical queries relies on a set of logical operators and their corresponding physical execution to complete the tasks. Integrating LLMs into data analytics does not mean overwriting traditional relational operators. SQL and data science operators remain essential for supporting scalable and efficient queries over structured data. The objective of developing LLM-powered *logical operators* is to augment these existing operators with AI-based transformations.

Recent research has primarily focused on extending SQL operators because they can be seamlessly integrated with database management systems and have the potential to leverage existing query engines and optimizing compilers. For example, SUQL [48] augments SQL with free-text primitives (`SUMMARY` and `ANSWER`) powered by LLMs. These primitives enable the system to provide row-by-row summaries and answer user questions on unstructured text data, while automatic optimizations are considered to reduce the number of LLM calls to improve efficiency. HQDL [86] treats an LLM as a virtual table that can be queried for information unavailable in structured data, and TAG [7] integrates LLM-based retrieval capabilities into table querying. In contrast, LOTUS [58] expands Pandas' operator set with semantic operators (e.g., `sem_map`, `sem_join`, and `sem_filter`), facilitating both logical query plan optimization and operator execution optimization. We see significant potential for developing additional operators. A key design criterion is to ensure these operators are expressive and comprehensive while enabling execution optimizations. For instance, strategies such as model selection or prompt engineering can optimize LLM performance, while using code generation instead of row-wise execution can improve efficiency.

Another crucial aspect is planning the query execution to provide answers to user questions. For instance, when a user poses a question in natural language, it's vital to generate a query execution plan that may involve extracting unstructured information or utilizing LLM knowledge. Traditional Text2SQL approaches, as discussed, are insufficient for generating queries and utilizing the optimizers in DBMS for the expanded set of operators. Utilizing the reasoning capabilities of LLMs to decompose questions into executable operations and enable optimization of query execution plans is important. Current methods often require users to specify updated schemas [45] or explicitly define the query plan [58, 4], and incorporate rule-based optimization techniques like operator reordering, predicate pushdown, and lazy evaluation to improve performance.

5 Table Augmentation for Machine Learning with LLMs

Machine learning (ML) is a pivotal data-centric application that plays a critical role in numerous decision-making processes [16, 66, 39]. It derives sophisticated patterns from historical data to produce predictive outcomes. However, real-world data are often unsuitable for direct ML training due to limited data entries and potentially insufficient features [14]. Acquiring more high-quality tabular data with well-suited features is essential for improving ML performance. However, this process often relies heavily on domain expertise, demanding considerable human effort [59] and remaining a persistent challenge in both database and data science research [11].

Numerous approaches have been proposed to automate the ML data processing pipeline [88], yet they often depend on machine learning or deep learning recommendations, which require substantial data collection of training, or employ rule-based systems, thereby limiting their applicability. Recent advancements in LLMs present significant opportunities to enhance tabular data augmentation process. Existing work broadly falls into two categories: *generation-based* approaches, which generate new data based on the original table, and *retrieval-based* approaches, which identify and extract relevant data from external sources. Building on the general definition of table augmentation tasks [14], we formally define the LLM-assisted table augmentation task for ML as follows:

Definition 5.1 (Table Augmentation for Machine Learning) *Let $\mathcal{T}_{\mathcal{A}}$ be the original table with feature set $\mathcal{A}$, and $\mathcal{P}$ be the data pool available for augmentation. Consider a downstream machine learning model $f_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ represents the model parameters. The objective of the table augmentation task is to transform $\mathcal{T}_{\mathcal{A}}$ into an augmented table $\mathcal{T}'_{\mathcal{A}_{new}}$ to enhance the performance of $f_{\boldsymbol{\theta}}$. Formally, the table augmentation task is defined as an augmentation function assisted with a large language model LLM:*

$$\mathsf{aug}_{LLM} : (\mathcal{T}_{\mathcal{A}}, \mathcal{P}, f_{\boldsymbol{\theta}}) \rightarrow \mathcal{T}'_{\mathcal{A}_{new}},\ s.t.,\ \mathbb{E}(f_{\boldsymbol{\theta}}^{\mathcal{T}'_{\mathcal{A}_{new}}}) < \mathbb{E}(f_{\boldsymbol{\theta}}^{\mathcal{T}_{\mathcal{A}}}),$$

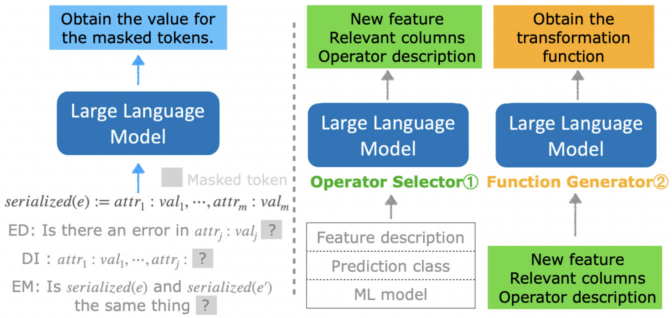

Figure 3: SMARTFEAT overview: advancing from row-level to feature-level generation.

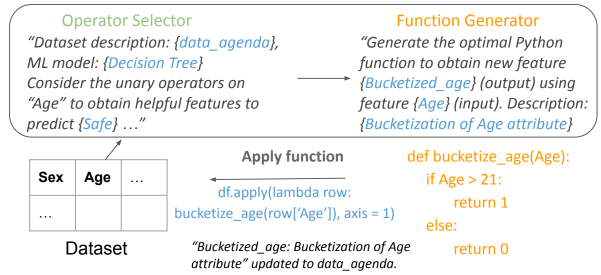

Figure 4: Example: constructing *Bucketized Age*.

where $\mathbb{E}(f_{\boldsymbol{\theta}}^{\mathcal{T}_{\mathcal{A}}})$ *and* $\mathbb{E}(f_{\boldsymbol{\theta}}^{\mathcal{T}'_{\mathcal{A}_{new}}})$ *represent the empirical errors of the machine learning model trained on* $\mathcal{T}_{\mathcal{A}}$ *and* $\mathcal{T}'_{\mathcal{A}_{new}}$*, respectively.*

5.1 Generation-based Table Augmentation

An effective approach to improving tabular datasets is *feature engineering*, which involves augmenting existing features to create more relevant inputs for machine learning models. This process often leads to substantial performance enhancements [88].

We introduce SMARTFEAT [46], a system that leverages LLMs to automate the feature engineering pipeline. Compared with traditional rule-based automated feature engineering tools, SMARTFEAT leverages the reasoning capabilities of LLMs to search for meaningful and interpretable features. Furthermore, instead of generating new feature values through row-by-row LLM executions, which could be computationally expensive, SMARTFEAT identifies promising features upfront and then synthesizes the transformation code to compute these features efficiently at scale. SMARTFEAT operates through an iterative search process, progressively refining and expanding the feature set. As shown in Figure 3, the system consists of two main components:

- *Operator selector* ①: This component takes as input (a) dataset feature descriptions, (b) the prediction task (e.g., classification), and (c) the downstream machine learning model. It applies operator-guided feature generation using various operator prompt templates, such as unary, binary, group-by–aggregate, and extractor operators. The operator selector interacts with the LLM to determine suitable operators to apply and outputs the *name of the new feature*, the *relevant columns* for computing the new feature, and a *descriptive explanation* of the feature.

- *Function generator* ②: Based on the outputs of the operator selector, this component generates an executable transformation function. The function is applied to the original dataset to compute the values of the new feature, and the augmented dataset and feature descriptions are updated for further iterations. Including detailed feature descriptions during this process is highly beneficial in guiding function generation. When no suitable function can be derived (e.g., extracting the capital city for each country), SMARTFEAT resorts to row-level LLM generation to obtain feature values, leveraging the common knowledge and reasoning abilities of LLMs.

In each iteration, the operator selector first chooses a semantically meaningful operator, and then the function generator obtains the transformation to compute feature values. For example, as shown in Figure 4, given the ML prediction task, model selection, and feature description, the operator selector chooses a unary operator to bucketize the `Age` column. It generates a new feature name (`Bucketized_age`), a description (e.g.,

Table 1: Comparison of average AUC values (↑) for different ML models: SMARTFEAT vs. baseline methods.

Methods	Diabetes	Heart	Bank	Adult	Housing	West Nile Virus	Tennis
Initial AUC	82.20	67.38	91.46	76.81	86.72	78.96	77.93
SMARTFEAT	**86.76 (+4.3%)**	**72.15 (+7.0%)**	91.47 (≈)	**87.00 (+13.3%)**	**92.19 (+6.3%)**	**82.12 (+4.0%)**	87.39 (+9.5%)
CAAFE	-	69.67 (+3.4%)	**91.73 (+0.3%)**	83.10 (+8.2%)	92.15 (+6.3%)	80.11 (+1.8%)	**88.50 (+13.6%)**
Featuretools	82.24 (≈)	66.78 (-0.9%)	91.04 (-0.5%)	73.85 (-3.9%)	79.47 (-8.1%)	73.12 (-7.4%)	81.29 (+4.3%)
AutoFeat	75.24 (-8.4%)	64.92 (-3.7%)	-	-	77.63 (-10.5%)	70.90 (-10.2%)	71.73 (-8.0%)

"`Bucketization of Age attribute`"), and identifies the relevant column(s). The function generator then translates this information into executable code, applies the transformation to the dataset, and adds the resulting feature to the feature set.

The feature generation process begins by exploring potential unary operators on the original features. Using a prompt template, the operator selector iterates over each original feature, prompting LLMs to propose potentially beneficial unary operations (e.g., bucketization or scaling). Once an operator is selected, the function generator retrieves the corresponding transformation function and applies it to the dataset. Building on the original and unary features, SMARTFEAT prompts LLMs to suggest binary and group-by-aggregate operators that may further enhance the data set. Finally, the process considers extractors that can operate on arbitrary columns to generate additional features.

In addition to SMARTFEAT, another feature engineering tool, CAAFE [31], also uses LLMs to produce Python code for feature engineering. Unlike SMARTFEAT, which utilizes a pre-defined operator-guided search for new features, CAAFE employs chain-of-thought instructions [79] to guide a series of intermediate steps to generate new features.

Lastly, we compare the performance of SMARTFEAT and CAAFE against traditional automated feature engineering tools based on *expansion-selection* methods [37]: Featuretools [36], which exhaustively generates features using predefined operators and applies feature selection, and AutoFEAT [32], which constructs a large set of nonlinear features followed by a search algorithm to select an effective subset. Using these tools, we processed seven datasets from *Kaggle*, performed classification, and evaluated the Area Under the ROC Curve (AUC) as the primary performance metric across four ML classification models: *linear regression*, *GaussianNB*, *random forest*, and *extra tree*. The average AUC of the four models is presented in Table 1.

The results show that the LLM-based approaches significantly outperform traditional methods. SMARTFEAT enhances the original AUC score by up to 13.3% on the *Adult* dataset, while CAAFE improves by 8.2%. We observe that CAAFE recommends a smaller set of features, and the operator-guided search in SMARTFEAT generates a more comprehensive feature set. The new features generated by Featuretools and AutoFEAT are agnostic to the dataset context and prediction task, and thus exhibit comparatively lower performance. However, this evaluation is limited to publicly available datasets, which LLMs might have encountered during training, potentially leading to overly optimistic results. Evaluating the performance of LLM-assisted feature engineering on private datasets remains a topic for future exploration.

5.2 Retrieval-based Table Augmentation

When generating additional information directly from the original table is insufficient, an alternative approach to data augmentation involves performing dataset searches and utilizing external text to enrich the data [14].

LLMs have demonstrated promising potential in identifying joinable [34] and unionable [20, 33] structured data, facilitating data discovery of related tables to broaden available information. Additionally, unstructured sources such as Wikipedia can be harnessed by performing entity linking [69] and extracting relevant content. For instance, FeSTE [27] combines web search with a fine-tuned BERT model to supplement data with Wikipedia-derived information, demonstrating the potential of LLMs for retrieval-based augmentation.

6 Concluding Remarks

In this paper, we explored three essential stages of the data engineering workflow: data wrangling, analytical querying, and table augmentation for machine learning, which often require significant effort from data engineers. Traditional automated approaches, which rely on rule-based algorithms or machine learning models, can struggle with complex scenarios or demand significant human involvement for training dataset collection. Recently, large language models (LLMs) have shown growing promise in automating these tasks due to their extensive training data corpora and cross-task generality. LLMs hold the potential to function as pretrained data engineers, facilitating the automation of data engineering workflows, optimizing performance, and expanding the scope of data tasks.

For data wrangling, we formally defined an LLM-based data wrangling operator, which uses task parameters and dataset context to construct prompts that guide LLMs in performing data preparation tasks. We introduced a unified system, UniDM, for LLM-based data wrangling, which automates prompt construction to handle different tasks in the multi-step wrangling process. However, a limitation of UniDM as we discussed is its reliance on row-level execution, which can be inefficient and costly for large datasets. To this end, we encourage ongoing efforts to explore approaches to utilize LLMs more efficiently for data wrangling. Looking ahead, we envision end-to-end data preparation platforms powered by LLMs, where users provide raw data, and necessary operators and optimizations are determined automatically to minimize costs and improve the wrangling quality.

For analytical querying, we are able to extend traditional relational queries by leveraging LLMs' capabilities to query unstructured data and LLM's knowledge via various interfaces (e.g., natural languages). To better interpret users' questions, we presented a Text2SQL framework, DAIL-SQL, which incorporates techniques including prompt engineering and supervised fine-tuning to improve the execution accuracy of translated SQL queries. We also discussed the importance of developing LLM-powered logical operators to expand the analytical capabilities of databases. We anticipate that future data systems will integrate these operators to harness LLMs' knowledge and reasoning ability. These systems should also support query planning and optimization, and offer both natural-language interfaces and declarative programming interfaces for analytical queries.

To support data processing for machine learning, we demonstrated that tabular data augmentation through generation- or retrieval-based approaches can boost downstream prediction performance. We introduced an automated feature engineering tool, SMARTFEAT, which employs operator-guided search to enable LLMs to generate meaningful and interpretable new features. We also discussed retrieval-based approaches using LLMs for searching and leveraging external text to enrich datasets. In the future, we can utilize LLMs to autonomously plan and execute table augmentation pipelines for generating more meaningful features and accessing additional data, while this may require careful pre- and post-processing to ensure efficiency and reliability.

There are several additional topics we touched upon that are worth further discussion. For example, in this paper, to optimize the performance of LLMs, we discussed approaches based on prompt engineering and supervised fine-tuning, which are also used to explore different avenues: Table-GPT [43] proposes fine-tuning LLMs with real tables to improve their ability to process two-dimensional data structures, while Jellyfish [84] explores instruction tuning to enhance LLMs as universal data processing solutions that better adhere to human instructions. In addition, reinforcement fine-tuning [28] has shown potential, particularly for reasoning-intensive models such as OpenAI's o1-mini, enabling stronger generalization and reducing reliance on large-scale labeled data. More recently, Deepseek-R1 [24], an open-source LLM based on reinforcement learning, has demonstrated strong reasoning capabilities without relying on supervised data. Collectively, these advancements in post-training techniques show significant potential to boost the performance of LLM-assisted data analytical tasks and open up new possibilities for addressing complex data engineering challenges. Moreover, current evaluations are predominantly based on public data sets, which can lead to overly optimistic results if LLMs have seen these datasets during training. Evaluating LLMs on private data lakes and unseen tasks is essential for assessing their generality and robustness.

References

[1] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.*, 9(12):993–1004, 2016.

[2] Mohammad Shahmeer Ahmad, Zan Ahmad Naeem, Mohamed Y. Eltabakh, Mourad Ouzzani, and Nan Tang. Retclean: Retrieval-based data cleaning using foundation models and data lakes. *CoRR*, abs/2303.16909, 2023.

[3] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. Palm 2 technical report. *CoRR*, abs/2305.10403, 2023.

[4] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proc. VLDB Endow.*, 17(2):92–105, 2023.

[5] Blessing Bamiduro and Anusha Challa. Large language models for sentiment analysis with amazon redshift ml (preview). `https://aws.amazon.com/blogs/big-data/large-language-models/-for-sentiment-analysis-with-amazon-redshift-ml-preview/`.

[6] Felix Bießmann, Tammo Rukat, Philipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. Datawig: Missing value imputation for tables. *J. Mach. Learn. Res.*, 20:175:1–175:6, 2019.

[7] Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E. Gonzalez, Carlos Guestrin, and Matei Zaharia. Text2sql is not enough: Unifying AI and databases with TAG. *CoRR*, abs/2408.14717, 2024.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[10] Ursin Brunner and Kurt Stockinger. Entity matching with transformer architectures - A step forward in data integration. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*, pages 463–473. OpenProceedings.org, 2020.

[11] Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. Data management for machine learning: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(5):4646–4667, 2023.

[12] Shuaichen Chang and Eric Fosler-Lussier. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *CoRR*, abs/2305.11853, 2023.

[13] Zui Chen, Lei Cao, Sam Madden, Ju Fan, Nan Tang, Zihui Gu, Zeyuan Shang, Chunwei Liu, Michael J. Cafarella, and Tim Kraska. SEED: simple, efficient, and effective data management via large language models. *CoRR*, abs/2310.00749, 2023.

[14] Lingxi Cui, Huan Li, Ke Chen, Lidan Shou, and Gang Chen. Tabular data augmentation for machine learning: Progress and prospects of embracing generative AI. *CoRR*, abs/2407.21523, 2024.

[15] Databricks. Ai functions on databricks. `https://docs.databricks.com/en/index.html`.

[16] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Trans. Neural Networks Learn. Syst.*, 28(3):653–664, 2017.

[17] AnHai Doan and Alon Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Mag.*, 26(1):83–94, 2005.

[18] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey for in-context learning. *CoRR*, abs/2301.00234, 2023.

[19] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[20] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *Proc. VLDB Endow.*, 16(7):1726–1739, 2023.

[21] Raul Castro Fernandez, Aaron J. Elmore, Michael J. Franklin, Sanjay Krishnan, and Chenhao Tan. How large language models will disrupt data management. *Proc. VLDB Endow.*, 16(11):3302–3309, 2023.

[22] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145, 2024.

[23] Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, Jinyang Gao, Liyu Mou, and Yu Li. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. *CoRR*, abs/2411.08599, 2024.

[24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[25] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[26] Mazhar Hameed and Felix Naumann. Data preparation: A survey of commercial tools. *SIGMOD Rec.*, 49(3):18–29, 2020.

[27] Asaf Harari and Gilad Katz. Few-shot tabular data enrichment using fine-tuned transformer architectures. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1577–1591. Association for Computational Linguistics, 2022.

[28] Hesam Sheikh Hassani. What is openai's reinforcement fine-tuning? `https://www.datacamp.com/blog/reinforcement-fine-tuning`.

[29] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek R. Narasayya, and Surajit Chaudhuri. Transform-data-by-example (TDE): an extensible search engine for data transformations. *Proc. VLDB Endow.*, 11(10):1165–1177, 2018.

[30] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 829–846. ACM, 2019.

[31] Noah Hollmann, Samuel Müller, and Frank Hutter. Llms for semi-automated data science: Introducing CAAFE for context-aware automated feature engineering. *CoRR*, abs/2305.03403, 2023.

[32] Franziska Horn, Robert Pack, and Michael Rieger. The autofeat python library for automated feature engineering and selection. In *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*, volume 1167 of *Communications in Computer and Information Science*, pages 111–120. Springer, 2019.

[33] Xuming Hu, Shen Wang, Xiao Qin, Chuan Lei, Zhengyuan Shen, Christos Faloutsos, Asterios Katsifodimos, George Karypis, Lijie Wen, and Philip S. Yu. Automatic table union search with tabular representation learning. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3786–3800. Association for Computational Linguistics, 2023.

[34] Zezhou Huang, Jiaxiang Liu, Haonan Wang, and Eugene Wu. The fast and the private: Task-based dataset search. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org, 2024.

[35] Zhongjun Jin, Michael R. Anderson, Michael J. Cafarella, and H. V. Jagadish. Foofah: Transforming data by example. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 683–698. ACM, 2017.

[36] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pages 1–10. IEEE, 2015.

[37] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. Explorekit: Automatic feature generation and selection. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 979–984. IEEE Computer Society, 2016.

[38] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. CHORUS: foundation models for unified data discovery and exploration. *Proc. VLDB Endow.*, 17(8):2104–2114, 2024.

[39] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.

[40] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[41] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. The dawn of natural language to SQL: are we fully ready? [experiment, analysis & benchmark]. *Proc. VLDB Endow.*, 17(11):3318–3331, 2024.

[42] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[43] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table fine-tuned GPT for diverse table tasks. *Proc. ACM Manag. Data*, 2(3):176, 2024.

[44] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, 2020.

[45] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. Towards accurate and efficient document analytics with large language models. *CoRR*, abs/2405.04674, 2024.

[46] Yin Lin, Bolin Ding, H. V. Jagadish, and Jingren Zhou. SMARTFEAT: efficient feature construction through feature-level foundation model interactions. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org, 2024.

[47] Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *CoRR*, abs/2303.13547, 2023.

[48] Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina J. Semnani, Chen Jie Yu, and Monica Lam. SUQL: conversational search over structured and unstructured data with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4535–4555. Association for Computational Linguistics, 2024.

[49] Samuel Madden, Michael J. Cafarella, Michael J. Franklin, and Tim Kraska. Databases unbound: Querying all of the world's bytes with AI. *Proc. VLDB Endow.*, 17(12):4546–4554, 2024.

[50] Steve McDowell. Komprise unleashes fresh insights about your unstructured data. `forbes.com/sites/stevemcdowell/2023/03/09/komprise-unleashes-fresh-insights-about-your-unstructured-data/`.

[51] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. Capturing semantics for imputation with pre-trained language models. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 61–72. IEEE, 2021.

[52] Xupeng Miao, Zhihao Jia, and Bin Cui. Demystifying data management for large language models. In Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan, editors, *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, pages 547–555. ACM, 2024.

[53] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *CoRR*, abs/2305.12586, 2023.

[54] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. Can foundation models wrangle your data? *Proc. VLDB Endow.*, 16(4):738–746, 2022.

[55] Nhan Nguyen and Sarah Nadi. An empirical evaluation of github copilot's code suggestions. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pages 1–5. ACM, 2022.

[56] OpenAI. Sql translate. `https://platform.openai.com/examples/default-sql-translate?spm=teamfile.libs.0.0.affb3f1dDq3ukY`.

[57] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[58] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. Semantic operators: A declarative model for rich, ai-based analytics over text data. *CoRR*, abs/2407.11418, 2024.

[59] Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. Dataset discovery and exploration: A survey. *ACM Comput. Surv.*, 56(4):102:1–102:37, 2024.

[60] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics, 2019.

[61] Yichen Qian, Yongyi He, Rong Zhu, Jintao Huang, Zhijian Ma, Haibin Wang, Yaohua Wang, Xiuyu Sun, Defu Lian, Bolin Ding, and Jingren Zhou. Unidm: A unified framework for data manipulation with large language models. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024.

[62] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models. *CoRR*, abs/2204.00498, 2022.

[63] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, 2017.

[64] Tonghui Ren, Yuankai Fan, Zhenying He, Ren Huang, Jiaqi Dai, Can Huang, Yinan Jing, Kai Zhang, Yifan Yang, and X. Sean Wang. PURPLE: making a large language model a better SQL writer. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*, pages 15–28. IEEE, 2024.

[65] El Kindi Rezig, Lei Cao, Michael Stonebraker, Giovanni Simonini, Wenbo Tao, Samuel Madden, Mourad Ouzzani, Nan Tang, and Ahmed K. Elmagarmid. Data civilizer 2.0: A holistic framework for data preparation and analytics. *Proc. VLDB Endow.*, 12(12):1954–1957, 2019.

[66] Rashida Richardson, Jason M Schultz, and Kate Crawford. Dirty data, bad predictions: How civil rights violations impact police data, predictive policing systems, and justice. *NYUL Rev. Online*, 94:15, 2019.

[67] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023.

[68] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. Querying large language models with SQL. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 365–372. OpenProceedings.org, 2024.

[69] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460, 2015.

[70] Abirami Sukumaran. Llm with vertex ai only using sql queries in bigquery. `https://cloud.google.com/blog/products/ai-machine-learning/llm-with-vertex-ai-only-using-sql-queries-in-bigquery`.

[71] Edhy Sutanta, Retantyo Wardoyo, Khabib Mustofa, and Edi Winarko. Survey: Models and prototypes of schema matching. *International Journal of Electrical and Computer Engineering (IJECE)*, 6(3):1011–1022, 2016.

[72] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Samuel Madden, and Mourad Ouzzani. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proc. VLDB Endow.*, 14(8):1254–1261, 2021.

[73] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[74] The Vicuna Team. Vicuna: An open-source chatbot impressing gpt-4 with 90 `https://lmsys.org/blog/2023-03-30-vicuna/`.

[75] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[76] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[77] Immanuel Trummer. From BERT to GPT-3 codex: Harnessing the potential of very large language models for data management. *Proc. VLDB Endow.*, 15(12):3770–3773, 2022.

[78] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. Promptem: Prompt-tuning for low-resource generalized entity matching. *Proc. VLDB Endow.*, 16(2):369–378, 2022.

[79] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[80] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[81] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[82] Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1962–1979. Association for Computational Linguistics, 2019.

[83] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics, 2018.

[84] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Jellyfish: A large language model for data preprocessing. *CoRR*, abs/2312.01678, 2023.

[85] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Large language models as data preprocessors. In *Proceedings of Workshops at the 50th International Conference on Very Large Data Bases, VLDB 2024, Guangzhou, China, August 26-30, 2024*. VLDB.org, 2024.

[86] Fuheng Zhao, Divyakant Agrawal, and Amr El Abbadi. Hybrid querying over relational databases and large language models. *CoRR*, abs/2408.00884, 2024.

[87] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[88] Marc-André Zöller and Marco F. Huber. Benchmark and survey of automated machine learning frameworks. *J. Artif. Intell. Res.*, 70:409–472, 2021.