iDataLake: An LLM-Powered Analytics System on Data Lakes

Jiayi Wang, Guoliang Li, Jianhua Feng
Department of Computer Science, Tsinghua University, Beijing, China
{liguoliang@tsinghua.edu.cn}

Abstract

Existing data lakes struggle to effectively analyze heterogeneous data – like unstructured, semi-structured, and structured data – because of difficulties in heterogeneous data linking, semantic understanding, and execution pipeline orchestration. In this paper we highlight challenges for supporting data analytics on data lakes, and present a system `iDataLake` to address these challenges, which takes natural language query as input, orchestrates a pipeline for the query and outputs the results for the query. We first define semantic operators to support heterogeneous data analytics on data lakes. Then we introduce our data embedding method for the alignment of multi-modal data and introduce how to efficiently discover those data relevant to the query from the data lake. Next, we introduce the pipeline orchestration method, which converts input natural language queries into executable pipelines built from predefined semantic operators. By executing the pipeline over the discovered data, the data analytics queries can be efficiently answered with high accuracy and low cost.

1 Introduction

Data lakes are increasingly becoming the storage paradigm for managing large volumes of heterogeneous data, including structured, semi-structured, and unstructured data. Structured data, such as relational tables, is well-suited to traditional analytics methods like SQL. However, semi-structured data (e.g., JSON, XML) and unstructured data (e.g., text documents) present significant challenges because of their inherent complexity (e.g., nested structure or or complete lack of structure) and the absence of predefined schemas. Despite these challenges, performing analytics on such diverse datasets offers tremendous value, enabling organizations to derive deeper insights and make data-driven decisions across various domains and data formats.

Example 6: Consider a query: "Identify the top-5 directors whose movies in the 1980s received the highest ratio of positive reviews." This query is highly relevant for film study. However, answering this query poses significant challenges due to the fragmented nature of the required data. For instance, structured information, such as production years, is typically stored in structured tables, while review content resides in unstructured text documents. Additionally, resolving the query necessitates multi-step reasoning, including data integration, sentiment analysis, and ranking, which further complicates the process.

While SQL is effective for querying structured data, it cannot directly handle semi-structured and unstructured data, which lack the well-defined schemas required for SQL. Furthermore, SQLs cannot express semantic predicates, which limits the ability to process data in a way that reflects its inherent meaning. Although some approaches often attempt to extract structured information from unstructured data, such as transforming textual content into tables with the help of machine learning models [1, 2],

this extraction process inevitably leads to data loss and may compromise the accuracy and depth of the analysis. Therefore, traditional methods fall short in handling the diverse nature of data lakes, even when unstructured data is converted into a structured format.

Recent advances in large language models (LLMs) have shown a great opportunity in addressing these challenges. LLMs possess advanced semantic understanding capabilities, enabling them to handle unstructured data (e.g., text) more effectively than traditional methods [3, 5–8]. Their ability to process and generate meaningful representations of natural language data makes them a powerful tool for performing analytics over semi-structured and unstructured datasets [9, 10]. However, integrating LLMs into data analytics workflows for data lakes remains an open research problem, as it faces the following challenges.

C1. Heterogeneous Data Modeling and Linking. Data analytical queries often involve diverse subsets of correlated data. Accurately identifying and linking the relevant data for an input query is crucial for both the accuracy and efficiency of the analytics process. Inaccurate data linking can lead to incorrect results and significant delays. Developing effective methods to semantically model and link heterogeneous data types effectively is essential for accurate and efficient data analytics.

C2. Semantic Data Processing. Semantic understanding is essential for processing semi-structured and unstructured data, as it allows the system to interpret the inherent meaning of the data rather than relying on rigid schemas and exact textual matching. Effectively leveraging the advanced semantic capabilities of LLMs to process and analyze these data types (e.g., semantic filtering, semantic grouping) is a critical challenge.

C3. Automatic Pipeline Orchestration. It is crucial to generate an execution plan for an analytical query on data lakes. However, orchestrating an efficient and accurate query execution pipeline in a multi-modal data lake is complex. Unlike traditional databases with deterministic query plans based on relational algebra and predefined data schemas, data lakes require flexible and adaptive approaches to handle diverse data types. Automating the generation and orchestration of such pipelines to ensure efficiency and accuracy remains a key challenge.

C4. Efficient Pipeline Execution. Executing pipelines in a multi-modal data lake involves balancing accuracy and efficiency. Enumerating all potential execution pipelines may yield high accuracy but incur significant computational costs. Conversely, executing a single pipeline can improve efficiency but may reduce accuracy, since semantic operator execution over heterogeneous data may fail or meet unforeseen errors. Designing an adaptive plan selection process that balances both accuracy and efficiency dynamically during execution is a challenging but essential task.

In response to these challenges, we propose `iDataLake`, a novel LLM-powered analytics system designed to handle data analytics queries over multi-modal data lakes. `iDataLake` addresses the aforementioned challenges with the following contributions:

(1) **Unified Embedding-Based Data Linking (for C1)**: We introduce a unified embedding approach to efficiently link heterogeneous data types within a multi-modal data lake. We embed different types of data into a shared semantic embedding space and align embeddings of relevant data through a contrastive learning method. This alignment ensures that semantically relevant data is accurately identified and retrieved even in heterogeneous formats, thus improving both the accuracy and efficiency of query responses.

(2) **Semantic Operators for Data Analytics (for C2)**: We present a set of semantic operators tailored for data analytics over multi-modal data lakes. These operators are designed to perform statistical and semantic analysis across structured, semi-structured, and unstructured data, enabling `iDataLake` to handle a wide range of complex queries that traditional systems struggle with.

(3) **Pipeline Orchestration Algorithm (for C3)**: We propose an iterative two-stage algorithm for automatic pipeline orchestration. It iteratively selects an appropriate operator to reduce the query and gradually form the execution pipeline. In each step, it filters out irrelevant operators using a

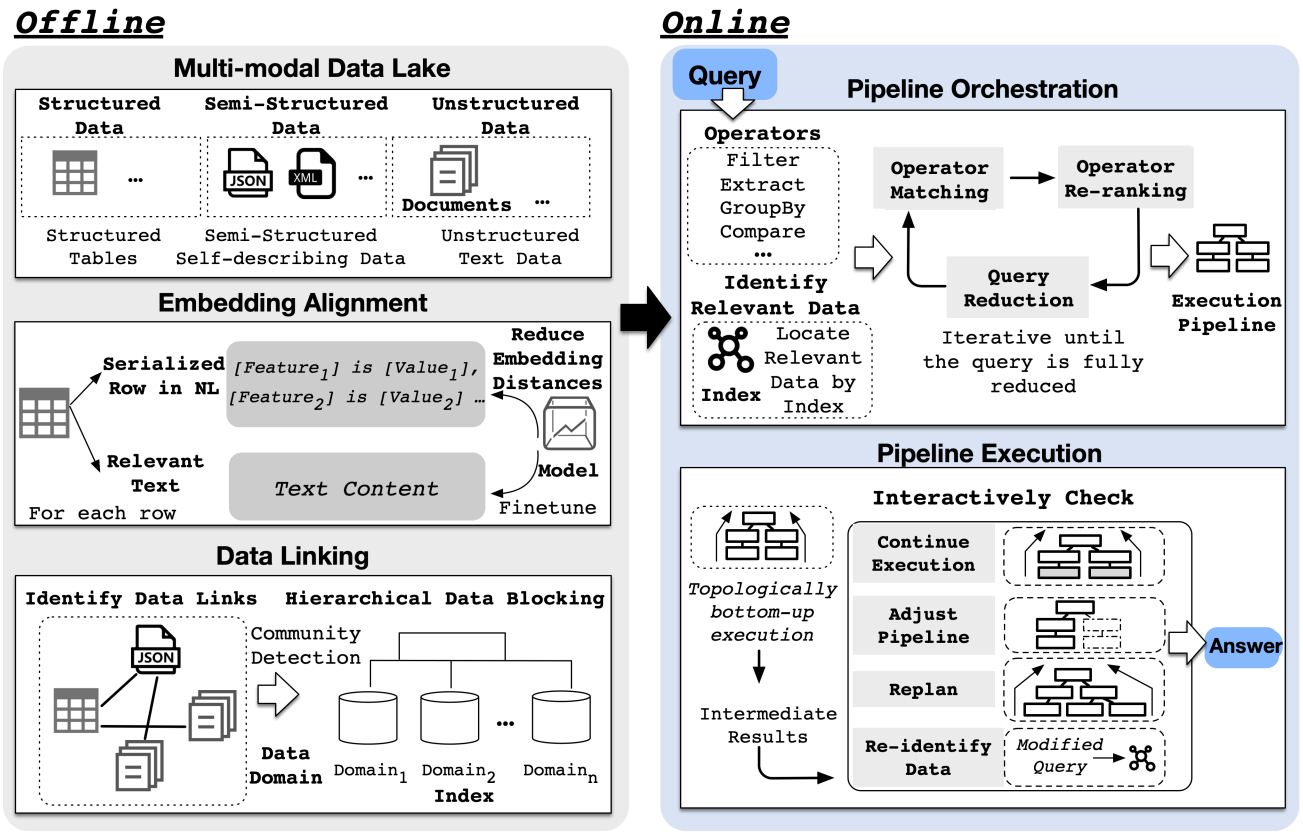

Figure 1: Framework of iDataLake.

low-cost approach and focuses on selecting from the remaining operators and organizing the relevant operators based on the query and available data. This ensures efficient and accurate query processing in a heterogeneous data environment.

(4) **Dynamic Pipeline Adjustment (for C4)**: Our system incorporates dynamic pipeline adjustment during query execution, allowing it to adapt to intermediate results. This flexibility enables iDataLake to optimize performance by skipping unnecessary computations and adjusting to unforeseen intermediate results, ensuring both efficiency and robustness.

In general, iDataLake represents a significant step forward in enabling high-accuracy, practical data analytics on data lakes. Unlike previous approaches that rely on lossy data extraction or are limited by SQL's rigid schema, iDataLake employs the semantic understanding capability of LLMs effectively to provide a more holistic and efficient solution.

2 LLM-Powered Analytics System on Data Lakes

We first introduce the system architecture of iDataLake (Figure 1), and then present the components in iDataLake.

2.1 iDataLake Architecture and Workflow

As illustrated in Figure 1, iDataLake is designed to process data lakes containing structured (e.g., relational tables), semi-structured (e.g., JSON, XML), and unstructured data (e.g., text documents).

Table 1: Summary of Semantic Operators

Operator	Category	Description
OrderBy	Transformation	Sort data according to specified criteria.
Compare	Transformation	Compare two input values and return the one that meets the criteria.
Refine	Transformation	Adjust text for clarity or precision, aiding subsequent processing.
Translate	Transformation	Convert text between languages using LLM or external tools.
Transform	Transformation	Convert data from one format to another, e.g., table to text.
Validate	Transformation	Verify the accuracy of generated information through external sources.
Scan	Retrieve	Load data files and enumerate their elements.
Filter	Retrieve	Remove irrelevant data based on specified criteria.
Augment	Retrieve	Fetch relevant data from external sources to enhance LLM responses.
Extract	Extraction	Extract relevant information from documents/tuples, similar to projection.
Conceptualize	Extraction	Identify key concepts to simplify complex queries.
Generate	Generation	Produce coherent text based on input, often for final responses.
Explain	Generation	Provide explanations or justifications for decisions.
GroupBy	Partition	Organize data into groups for computing summary statistics.
Cluster	Partition	Cluster similar data together.
Classify	Partition	Categorize or label entities using LLM or external ML models.
Link	Link	Identify and link related data, such as tables and documents in the data lake.
SetOP	Aggregation	Perform set operations, e.g., Union, Intersection, Complement.
Integrate	Aggregation	Combine information from multiple sources into a cohesive response.
Aggregate	Aggregation	Compute aggregation results, such as sum or average, from data.
Summarize	Aggregation	Condense text into shorter summaries for improved context consumption.

During offline pre-processing, `iDataLake` constructs an index tailored to the characteristics of the data, capturing inherent correlations across various data modalities. To enable efficient data linking and retrieval, `iDataLake` employs a unified embedding approach. Data of different types is transformed into a shared semantic embedding space, aligning heterogeneous data for seamless integration. Then using community detection algorithms, `iDataLake` discovers hierarchical clustering relationships within the data, partitioning the data into domain-specific clusters. This design ensures that relevant data can be located efficiently for any given query. Moreover, a vector index is built on top of embeddings to support efficient embedding retrieval. To address the out of distributions (OOD) issues among different data types, we also build OOD vector index to facilitate cross-modal retrieval.

For online analytics, `iDataLake` introduces a suite of semantic operators specifically designed for multi-modal data processing. When a query is received, `iDataLake` first utilizes the data linking index to identify relevant data within the data lake. The pipeline orchestration component then constructs an execution plan by iteratively selecting and applying appropriate semantic operators. The execution plan is executed by the query execution engine, which dynamically adjusts the pipeline based on intermediate results to maintain robustness and efficiency. Once the entire pipeline is executed, `iDataLake` generates the final result and returns it to the user. By following logically reasonable pipelines constructed over query-relevant data, `iDataLake` ensures high accuracy in its results.

2.2 Semantic Operators

Traditional relational operators are insufficient for unstructured data analytics, as they lack semantic processing capabilities and require data to adhere to strict schemas. To address this limitation, we introduce a set of semantic operators that serve as the fundamental building blocks for performing analytics over multi-modal data lakes. These operators are designed to handle diverse data types, bridge the gap between heterogeneous data sources, and enable joint query execution based on semantic

understanding rather than relying on predefined schemas.

2.2.1 Logical Operators

The logical operators in `iDataLake` are categorized into seven types, each designed to address a specific aspect of the analytics process. Table 1 summarizes these operators.

Data Transformation Operators. These operators transform data to generate intermediate representations or perform computations. Both input and output are typically lists of data. Examples include:

OrderBy: Sort input data based on specified criteria and return the ordered results.

Transform: Convert data from one format to another, such as transforming a table into a text paragraph describing its content.

Data Retrieval Operators. These operators retrieve relevant data from external sources. The input is typically a query or criteria, and the output is typically a list of data. Examples include:

Filter: Remove irrelevant data from specified data based on user-specific criteria and output the result.

Augment: Fetch additional data relevant to the query from external sources to enhance LLM responses.

Data Extraction Operators. These operators extract relevant information from certain data sources. The input is generally a list of data and the output is also a list. Examples include:

Extract: Extract specific information from documents or tuples, similar to the projection operator in relational algebra.

Conceptualize: Identify key concepts from input query, e.g., converting a description to its corresponding term, to simplify complex queries and improve understanding.

Data Generation Operators. These operators generate target content based on given requirements and input information. The input is generally text paragraphs, and the output is the target text result. Examples include:

Generate: Produce coherent text based on input, often used for final responses.

Explain: Provide explanations or justifications for decisions. The explanations of different reasoning paths are generally integrated to generate the final answer.

Data Partition Operators. These operators partition data into meaningful subsets for further analysis. The input is generally a list of data, and the output is a list of data lists. Examples include:

GroupBy: Organize data into groups for computing summary statistics or comparing information across groups.

Cluster: Cluster data together based on similarity metrics using clustering algorithms or LLM identification.

Data Linking Operators. These operators identify and link related data across different sources. The input is typically a set of data elements, and the output is a linked structure. An example is:

Link: Identify and link related data, such as tables and documents in a data lake, supporting different granularities (e.g., table level, tuple level, paragraph level, and document level).

Data Aggregation and Integration Operators. These operators integrate information from multiple sources or compute aggregate insights. The input is typically a list of data, and the output is a single value or cohesive response. Examples include:

Aggregate: Compute statistical aggregation results, such as sum or average, from data.

Summarize: Condense text into shorter summaries to reduce context consumption for LLMs.

2.2.2 Physical Operators

Each logical operator has multiple physical implementations tailored to different data types, corresponding to different physical operators. Based on the involvement of LLMs in the analytics process, physical

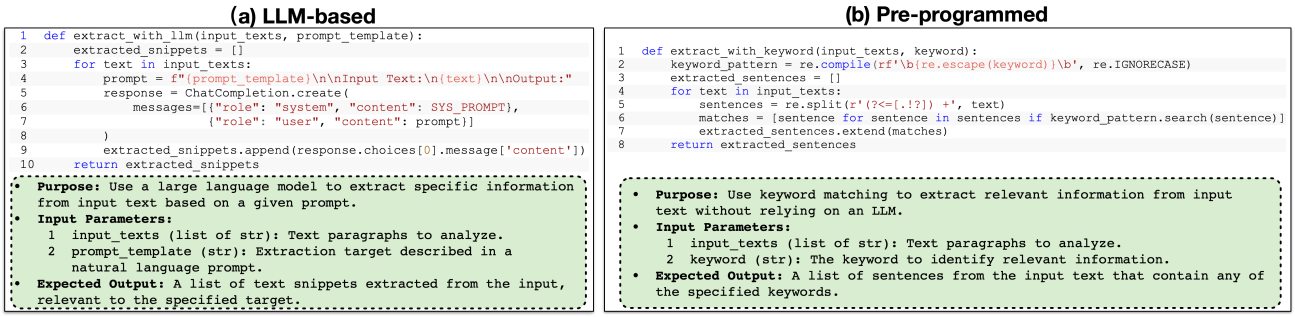

Figure 2: Example physical operators for the Extract operator.

operators are classified into two categories:

LLM-Based Physical Operators: These implementations leverage LLMs for tasks requiring semantic understanding by formulating appropriate prompts. For example, as shown in Figure 2(a), the *Extract* operator can utilize an LLM to identify and extract text snippets by providing prompts that specify the extraction target, such as *information related to sports*.

Pre-Programmed Physical Operators: These implementations rely on pre-defined logic to process data, similar to the physical operators in relational databases. For example, as shown in the *Extract* operator can extract information using keyword matching or regular expressions to extract information that follows certain patterns.

In order to select the appropriate physical operator for each logical operator in the plan, `iDataLake` maintains detailed text descriptions for each physical operator, including its purpose, input parameters, and expected outputs. If such descriptions are unavailable, `iDataLake` invokes the LLM to generate them with few-shot examples of existing operators automatically. With this information, the overall pipeline can be constructed accurately by LLMs based on actual needs.

To support user-defined functions (UDFs), we invoke LLMs to generate codes for UDFs.

2.2.3 Adding Other Operators

In practical applications, existing operators may not satisfy specific analytical requirements. `iDataLake` supports the seamless addition of new operators to address such needs. Only the following information is required to add a physical operator: (1) The programmed implementation of the physical operator for execution. (2) A detailed text description of the operator, including its purpose, input parameters, and outputs.

If these descriptions are incomplete, `iDataLake` can call the LLM to generate them. By defining these elements, `iDataLake` ensures that the newly added operators integrate seamlessly into the analytics framework.

2.3 Embedding and Linking

As introduced in Section 1, a key challenge for data analytics over data lakes is to efficiently identify the data that needs processing, as processing the complete data lake, extensively large in volume, is impractical. The effectiveness of identifying relevant data is critical since all subsequent operations are performed on this subset. Therefore, the identification process must achieve both high recall and high precision. High recall with low precision retains excessive irrelevant data, leading to inefficiencies. Conversely, high precision with low recall excludes relevant data, rendering correct results unattainable due to incomplete information.

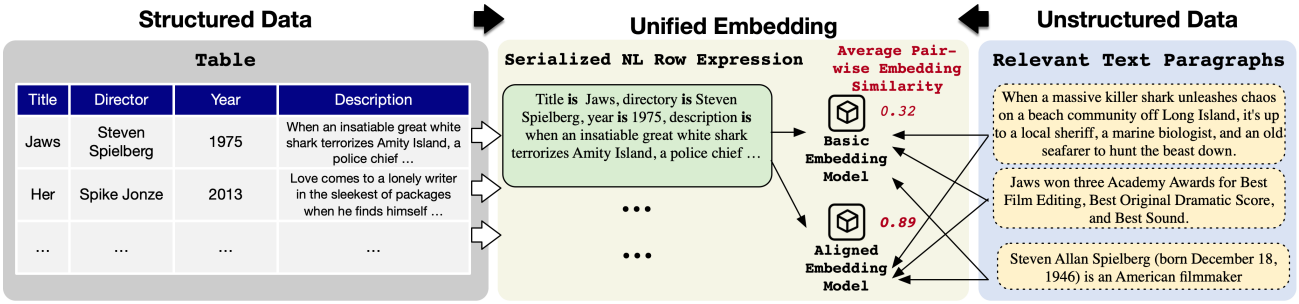

Figure 3: Example of embedding alignment between tables and text.

To address this, we propose a bi-encoder-based method that transforms diverse data types into a unified embedding space, where the similarity between embedding vectors serves as a metric to identify relevant data across modalities. However, training separate embedding models for different data types inevitably leads to misaligned embeddings. For example, embeddings for text and table data, although of the same dimension, are inherently unaligned. Achieving semantic alignment between these embeddings is a recognized challenge [11]. To the best of our knowledge, we are the first to semantically align embeddings for table data and text data.

Existing methods extend text embedding models to structured table data by serializing table rows into natural language (NL) sentences using formatting rules such as *[Feature1] is [Value1], [Feature2] is [Value2], ...* or *[Feature1]: [Value1], [Feature2]: [Value2], ...*, where *[Feature]* denotes the column name and *[Value]* denotes the cell value. Figure 3 illustrates this approach. However, directly computing embeddings for table rows from their serialized NL expressions is inaccurate, as the serialized format differs significantly from standard NL expressions, leading to embedding similarities that fail to reliably identify relevant data.

Embedding Alignment Between Different Types of Data. To overcome this limitation, we introduce a fine-tuning process that aligns text embeddings with embeddings derived from serialized table data. We collect a corpus of tables paired with ground-truth relevant text paragraphs from diverse sources. During fine-tuning, we employ a contrastive learning framework: table rows paired with their ground-truth text paragraphs are treated as positive examples, while table rows paired with unrelated text paragraphs serve as negative examples. Using these training pairs, we compute the Multiple Negatives Ranking Loss [12], which minimizes embedding distances for positive pairs and maximizes embedding distances for negative pairs.

The fine-tuning process proceeds iteratively, adjusting the embedding space to bridge the semantic gap between table and text data. Upon convergence, embeddings achieve meaningful alignment, enabling similarity measures in the unified space to accurately identify relevant data across data types. This alignment ensures both high recall and high precision in retrieval tasks, significantly enhancing system performance.

Data Linking. The vast size of data lakes results in an enormous number of data embeddings, posing significant challenges for efficiently locating relevant data. To address this, we observe that relevant data often forms natural clusters. Specifically, subsets of data relevant to one query are likely to be relevant or irrelevant to other queries together, as they often represent related entities. Leveraging this insight, we propose a preprocessing approach that identifies and organizes these clustering relationships within the data lake. This process divides the data into distinct blocks, where related data locate in the same block and the relevant blocks are linked.

To achieve this, we employ the Louvain community detection algorithm [13], which identifies hierarchical clustering relationships in the data. The algorithm starts by constructing a graph where

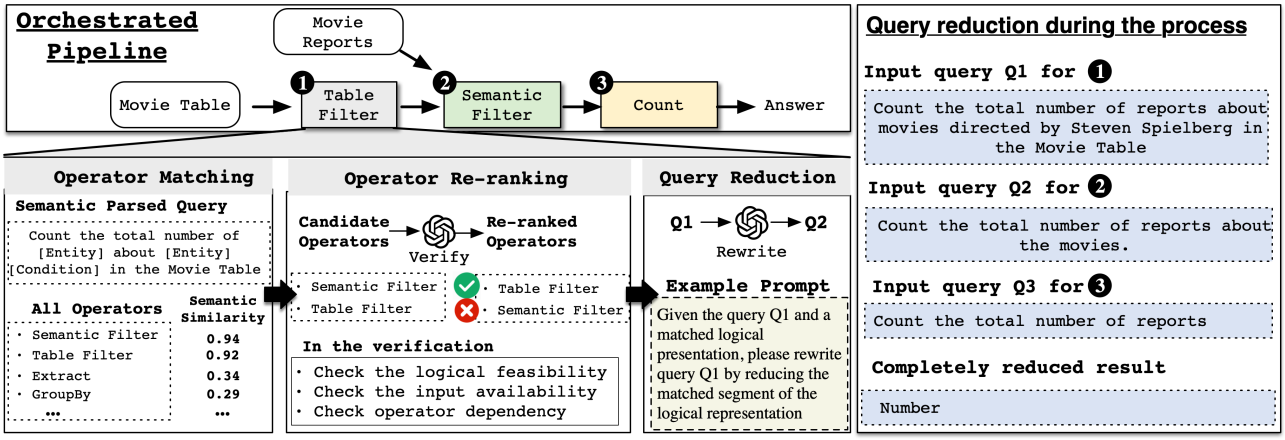

Figure 4: An example pipeline orchestration process, detailing the selection of the Table Filter operator.

nodes represent data items and edges represent embedding similarities. It iteratively optimizes modularity to detect communities, i.e., subsets of data points with high pair-wise similarities, producing a hierarchy of clusters. These clusters are then used to organize the data into blocks, with each block representing semantically similar data. When a query is issued, we first identify the relevant blocks and then locate specific data within each block. This hierarchical clustering approach reduces the data needing consideration, enhancing both retrieval efficiency and query accuracy.

2.4 Pipeline Orchestration

Orchestrating an accurate pipeline for query execution over multi-modal data lakes presents unique challenges due to the heterogeneity of data types and the absence of unified data schemas. Unlike traditional query execution in relational databases, which operates over structured data with predefined schemas and deterministic query plans, multi-modal data lakes require a flexible and adaptive pipeline orchestration mechanism capable of handling diverse data modalities and complex analytics tasks. In this section, we introduce the pipeline orchestration method employed in `iDataLake`, which iteratively selects and applies appropriate operators to reduce the query until it is fully solved. This method combines the semantic understanding capability of LLMs with a cost-efficient, two-stage operator matching strategy that minimizes LLM invocations to reduce cost and improve efficiency.

Overview of the Orchestration Process. The orchestration process in `iDataLake` follows an iterative workflow aimed at progressively reducing the query. Each iteration involves three key steps: (1) identifying suitable physical operators that can solve part of the query (coarse-grained matching); (2) validating the applicability and availability of the matched operators (fine-grained re-ranking), and (3) applying a feasible operator to simplify the query, solving part of the query logically. This process repeats until the query is fully resolved, which is verified by the LLM. We next introduce each step in detail.

Operator Matching. Efficiently matching a query with suitable physical operators is critical for effective pipeline orchestration. A naive approach of directly prompting an LLM with descriptions of all operators is infeasible due to constraints such as limited LLM context length and inaccurate selection among numerous options [4]. To address these issues, `iDataLake` employs a two-stage operator matching method:

Coarse-Grained Matching. This stage quickly eliminates irrelevant operators using low-cost checks. The key idea is that operators are likely to be applicable only if their use cases align with the query. However,

the specific values in the NL expressions are not helpful for the matching and can even mislead the matching. Therefore, to facilitate matching, both the query and operator use cases are transformed into *"logical representations"* by replacing concrete values with semantic placeholders such as [Entity] and [Condition]. The transformation is performed by instructing LLMs to conduct semantic parsing with few-shot examples. After computing these logical representations, the relevance between query and operators can be measured by the similarity of their semantic embedding vectors. For instance, the query *"Select the documents that are related to swimming"* corresponds to the logical representation of *"Select the documents that [Condition]"*, which has high semantic embedding similarity with the logical representation of a use case for the *Filter* operator *"Select the documents that satisfy [Condition]"* and thereby the matching can be conducted correctly.

Logical representations of operator use cases and their semantic embeddings can be precomputed. The logical representation of the query can be computed online. These embeddings enable efficient similarity-based comparisons, filtering out irrelevant operators accurately without involving the LLM.

After coarse-grained filtering, the remaining operators are further evaluated using the LLM. In this stage, the LLM verifies the applicability of each operator through prompts and few-shot examples. For applicable operators, the LLM determines the required inputs and checks their availability in the current context. This step relies an intermediate variable list maintained throughout the orchestration process, ensuring accurate and context-aware operator selection.

Fine-Grained Re-ranking. After coarse-grained filtering, the remaining operators are further evaluated using the LLM. In this stage, the LLM is instructed to verify the applicability of each operator through prompts and few-shot examples. For applicable operators, the LLM determines the required inputs and checks their availability in the current context. Specifically, an intermediate variable list is maintained throughout the whole orchestration process. The list is initialized with the whole data lake. During the orchestration, intermediate results are generated after the application of each operator. For each intermediate data, the LLM is instructed to generate a short text description of it. The generated text descriptions will then be added to the prompt to guide the LLM to select the appropriate inputs. Meanwhile, based on the dependency relationship between the inputs and outputs of the operators in the pipeline, the dependency between operators can also be determined. For each operator, only the dependency relationships with the operators whose outputs are prerequisites are necessary. In this way, a directed acyclic graph, i.e., a DAG-format pipeline can be constructed that can obtain much higher efficiency than sequential pipelines.

Query Reduction. Once an applicable operator is selected, it is applied to simplify the query. To ensure accurate query reduction, the LLM is instructed to conduct this reduction. This involves resolving part of the query logically. A concrete example of this process is illustrated in Figure 4.

Iterative Query Decomposition. The process iterates in a DFS manner until the query is fully reduced to a solved form. Each iteration involves selecting an appropriate operator, validating its applicability, and applying it to reduce the query. The resulting plan is represented as a directed acyclic graph (DAG), where nodes correspond to the selected physical operators, and edges represent dependencies between them. Notably, this orchestration method can generate multiple candidate pipelines by exploring multiple paths in the DFS process.

By combining coarse-grained filtering with embedding-based similarity and fine-grained LLM-driven operator selection, `iDataLake` achieves both high efficiency and accuracy in operator matching. This iterative reduction process enables `iDataLake` to systematically simplify complex queries over multi-modal data lakes while leveraging LLMs only when inevitable. When the query is completely solved by this iterative process, `iDataLake` constructs an efficient plan to solve complex queries effectively.

2.5 Pipeline Execution

Unlike traditional databases, where execution plans are deterministic to guarantee correct results, `iDataLake` needs to be able to adapt dynamically to handle the inherent unpredictability of both unstructured data and complex queries. This section describes how `iDataLake` executes interactively, adjusting its pipeline in response to intermediate results and unforeseen conditions. As illustrated in Figure 1, `iDataLake` can decide to continue execution, adjust the pipeline, replan, or re-identify relevant data based on the state of intermediate results.

Determining Operator Inputs. Similar to the orchestration phase (Section 2.4), identifying the correct inputs for each operator is critical during execution. In the orchestration phase, the relationships between operators' inputs and outputs are established. During execution, `iDataLake` determines operator inputs according to these identified relationships. Additionally, after each intermediate execution step, the LLM generates a brief textual description of the results. If an input determined during orchestration is unavailable or unsuitable during execution, `iDataLake` dynamically re-evaluates the input selection by repeating the orchestration process. This re-evaluation uses the generated descriptions of intermediate results as part of the prompt to guide the LLM in selecting appropriate inputs for the operator.

Parallel Topological Execution. To maximize execution efficiency, `iDataLake` follows a bottom-up, parallelized execution strategy based on the pipeline's topological order. Operators are executed in parallel once their prerequisites are met, continuing until all operators have been executed and the final result is produced. After each operator executes, `iDataLake` verifies whether its intermediate result aligns with expectations. Discrepancies can occur due to issues in the initially discovered relevant data or the incorrectness of the orchestrated pipeline. Therefore, if discrepancies are found, `iDataLake` first re-examines the relevant data identification steps, querying the LLM to rewrite the original query in an alternative form and using the rewritten query to re-identify relevant data. If the re-identified data closely matches the initial results, the system triggers a dynamic adjustment to the pipeline, as described next.

Pipeline Adjustment During Execution. When an operator fails to produce the expected result, `iDataLake` dynamically adjusts the execution plan without restarting the entire planning and execution process. Early stopping conditions, identified by the LLMs, can prune unnecessary operations, enhancing the execution efficiency.

Incremental Execution. As discussed in Section 2.4, multiple candidate plans are generated during the planning phase. Although only one plan is executed, many of these plans share a common sequence of initial operators due to their generation by a depth-first search (DFS) strategy. When an operator fails, `iDataLake` selects the plan with the longest matching sequence of initial operators and resumes execution from the last successful operator. This prevents the need to regenerate the entire plan. However, if there is minimal overlap between plans or no matching plans, `iDataLake` initiates replanning.

Replanning for Adjustment. If no plan shares a sufficient initial sequence, `iDataLake` replans from the point of failure. This is faster than starting from scratch (replanning for the initial query), as the current query has already been partially simplified by the earlier execution. If no suitable pipeline can be found, `iDataLake` backtracks to the last successful operator, repeating this process until it reaches the root (i.e., backtracks to the initial query).

Pruning Based on Insights from Data Analysis. Analyzing unstructured data can uncover relationships such as equivalencies or exclusivities between entities or conditions that were not initially identified during the planning phase. These insights enable the optimization of query execution by eliminating redundant operations. For instance, if two conditions are found to be equivalent, applying both filters becomes unnecessary, and one can be skipped to improve efficiency. Conversely, if conditions are mutually exclusive, the query can be simplified or terminated early, avoiding unnecessary data processing. Consider the query: *List the mathematicians who have won the Fields Medal at the age of 50 or older.* By

analyzing the Fields Medal's eligibility criteria, it becomes evident that the conditions *"won the Fields Medal"* and *"at the age of 50 or older"* are mutually exclusive, as the award is restricted to individuals under 40. This insight allows the query execution to be halted immediately, saving computational resources by avoiding processing large datasets for a result that is guaranteed to be empty.

3 Challenges and Opportunities

Scalability and Efficiency. A major challenge in implementing LLM-powered analytics on a large data lake is the inherent high cost and low computation efficiency of LLMs. Multi-modal data lakes consist of structured, semi-structured and unstructured data, which require distinct processing pipelines. Efficiently querying such vast datasets with LLMs necessitates reducing invocation costs without sacrificing performance. Scaling LLMs to handle such vast quantities of heterogeneous data, demands careful optimization of both the model's computation and the overall query pipeline architecture. For example, ordering the operators in the pipeline based on cardinality estimation [14, 15] is crucial for efficiency. Research into more efficient LLM architectures, such as fine-tuning models for domain-specific tasks or incorporating hybrid models with different sizes for tasks of different complexity [17], could mitigate the scalability issue. Additionally, caching intermediate results in a distributed system or identifying and indexing critical data components, such as coresets [18, 19] could improve the system's efficiency in handling large datasets.

Interpretability and Transparency. LLMs are often criticized for their "black box" nature, making it difficult to understand how they generate results or why they make particular decisions. This lack of transparency poses challenges for users who need to trust and interpret the outcomes of their data analytics queries. In the context of multi-modal data lakes, where complex relationships between structured and unstructured data must be identified and analyzed, ensuring that the system's reasoning is interpretable becomes even more critical. Approaches to improving the interpretability of LLMs, such as attention mechanisms and counterfactual reasoning, can enhance the transparency of the system. By providing insights into the specific data features or relationships that influence the model's decision-making, these methods could increase user confidence in the results. Moreover, incorporating explainability into the pipeline design could facilitate the adoption of LLM-powered analytics in sensitive or regulated industries.

Evaluation and Benchmarking. The evaluation of LLM-driven analytics over multi-modal data lakes remains an open challenge, particularly in comparing the effectiveness of different methodologies. Metrics for traditional data analytics, such as precision, recall, or F1 score, may not adequately capture the nuances of integrating structured and unstructured data sources. Furthermore, due to the large-scale and dynamic nature of the data lake, it is difficult to develop comprehensive benchmarks that can consistently evaluate the model's performance across different types of queries and use cases. Developing new benchmarking methodologies tailored for multi-modal data lake analytics would be an essential step forward. These benchmarks could focus on both the accuracy of the results and the system's ability to integrate and analyze heterogeneous data sources. Additionally, creating standardized test datasets for data lake environments could encourage further research and provide a basis for consistently comparing different models and systems.

Model Adaptation to Domain-Specific Needs. LLMs are typically trained on a wide array of general-domain data, which may not always capture the specific nuances of particular industries or use cases, such as legal, healthcare, or finance. As a result, LLMs may not fully comprehend domain-specific terms, contexts, or relationships without further domain-specific fine-tuning. For example, the data linking may become inaccurate for personal domain data, even after the alignment finetuning. Domain adaptation of LLMs is a promising avenue for improving system performance in specialized applications.

By leveraging transfer learning or few-shot learning techniques, LLMs can be fine-tuned to perform better on industry-specific tasks. Furthermore, the system can be augmented with domain-specific knowledge bases or ontologies, which can enhance the model's understanding of complex, industry-relevant concepts.

4 Conclusion

In this paper, we present `iDataLake`, a novel LLM-powered analytics system designed to address the challenges of data analytics in multi-modal data lakes. By integrating large language models with advanced semantic operators, embedding-based data linking, and dynamic pipeline orchestration, `iDataLake` provides a unified framework for querying and analyzing diverse data types, including structured, semi-structured, and unstructured data. `iDataLake` introduces several key innovations including semantic operators tailored to the unique requirements of multi-modal analytics, unified embedding-based data linking for aligning heterogeneous data types in a common semantic space, dynamic pipeline adjustment to adapt to evolving query execution requirements and interactive and incremental plan execution to ensure robust and efficient query handling.

Acknowledgment. This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (61925205, 62232009, 62102215), Shenzhen Project (CJGJZD20230724093403007), Zhongguancun Lab, Huawei, and Beijing National Research Center for Information Science and Technology (BNRist).

References

[1] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, Christopher Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes *VLDB*, 2024.

[2] Xueqing Wu, Jiacheng Zhang, Hang Li Text-to-table: A new way of information extraction *ACL*, 2022.

[3] Matthias Urban, Carsten Binnig. Efficient Learned Query Execution over Text and Tables [Technical Report] *Corr*, 2024.

[4] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, Minlie Huang. Large Language Models Are Not Robust Multiple Choice Selectors *ICLR*, 2024.

[5] Samuel Madden, Michael J. Cafarella, Michael J. Franklin, Tim Kraska. Databases Unbound: Querying All of the World's Bytes with AI *VLDB*, 2024.

[6] Liana Patel, Siddharth Jha, Parth Asawa, Melissa Pan, Carlos Guestrin, Matei Zaharia. LOTUS: Enabling Semantic Queries with LLMs Over Tables of Unstructured and Structured Data *CoRR*, 2024.

[7] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeigham, Aditya G. Parameswaran, Eugene Wu. Towards Accurate and Efficient Document Analytics with Large Language Models *CoRR*, 2024.

[8] Jiayi Wang, Guoliang Li. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries *CIDR*, 2025.

[9] Matthias Urban, Carsten Binnig. CAESURA: Language Models as Multi-Modal Query Planners *CIDR*, 2024.

[10] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Samuel Madden, Nan Tang. Symphony: Towards natural language query answering over multi-modal data lakes *CIDR*, 2023.

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever Learning Transferable Visual Models From Natural Language Supervision *ICML*, 2021.

[12] Nils Reimers and Iryna Gurevych. *EMNLP-IJCNLP*, 2019.

[13] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[14] Jiayi Wang, Chengliang Chai, Jiabin Liu, Guoliang Li FACE: a normalizing flow based cardinality estimator *Proc. VLDB Endow.*, 2022.

[15] Jiayi Wang, Chengliang Chai, Jiabin Liu, Guoliang Li Cardinality estimation using normalizing flow *VLDB J.*, 2024.

[16] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, Alfons Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning *CIDR*, 2019.

[17] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Gerardo Vitagliano. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing *CIDR*, 2025.

[18] Jiayi Wang, Chengliang Chai, Nan Tang, Jiabin Liu, Guoliang Li. Coresets over multiple tables for feature-rich and data-efficient machine learning *Proc. VLDB Endow.*, 2023.

[19] Chengliang Chai, Jiayi Wang, Nan Tang, Ye Yuan, Jiabin Liu, Yuhao Deng, Guoren Wang. Efficient Coreset Selection with Cluster-based Methods *KDD*, 2023.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data *AISTATS*, 2017.

[21] Cynthia Dwork, Frank McSherry, Kobbi Nissim, Adam D. Smith. Calibrating Noise to Sensitivity in Private Data Analysis *TCC*, 2006.