TAIJI: MCP-based Multi-Modal Data Analytics on Data Lakes

Chao Zhang, Shaolei Zhang, Quehuan Liu, Sibei Chen, Tong Li, Ju Fan*
Renmin University of China
{fanj@ruc.edu.cn}

Abstract

The variety of data in data lakes presents significant challenges for data analytics, as data scientists must simultaneously analyze multi-modal data, including structured, semi-structured, and unstructured data. While Large Language Models (LLMs) have demonstrated promising capabilities, they still remain inadequate for multi-modal data analytics in terms of accuracy, efficiency, and freshness. First, current natural language (NL) or SQL-like query languages may struggle to precisely and comprehensively capture users' analytical intent. Second, relying on a single unified LLM to process diverse data modalities often leads to substantial inference overhead. Third, data stored in data lakes may be incomplete or outdated, making it essential to integrate external open-domain knowledge to generate timely and relevant analytics results. In this paper, we envision a new multi-modal data analytics system to address the aforementioned limitations. Specifically, we propose a novel architecture built upon the Model Context Protocol (MCP), an emerging paradigm that enables LLMs to collaborate with knowledgeable agents. First, we define a semantic operator hierarchy tailored for querying multi-modal data in data lakes and develop an AI-agent–powered NL2Operator translator to bridge user intent and analytical execution. Next, we introduce an MCP-based execution framework, in which each MCP server hosts specialized foundation models optimized for specific data modalities. This design enhances both accuracy and efficiency, while supporting high scalability through modular deployment. Finally, we propose a updating mechanism by harnessing the deep research and machine unlearning techniques to refresh the data lakes and LLM knowledges, with the goal of balancing the data freshness and inference efficiency.

1 Introduction

Modern data-intensive applications continuously generate diverse types of data stored in data lakes [16], encompassing structured data (e.g., tables, graphs), semi-structured data (e.g., JSON, HTML), and unstructured data (e.g., text, images, video). This diversity introduces a critical challenge of *data variety* [15], underscoring the need for integrated analysis across *multi-modal datasets*, which often contain complementary information essential for extracting timely and valuable insights. For example, in healthcare scenarios, physicians simultaneously analyze heterogeneous patient data, such as X-ray images and textual diagnostic reports, to support accurate and timely diagnoses. In e-commerce, users seek products by jointly exploring visual content, textual descriptions, and structured metadata.

Recently, the rapid advancement of Large Language Models (LLMs) has opened new opportunities for analyzing multi-modal data at the semantic level through natural language interactions. For example, GPT-4 [2] demonstrates the ability to reason over text, images, and JSON data within a unified conversational interface. However, despite these advancements, current LLM-based approaches have yet to fully unlock the potential of multi-modal data analytics, primarily due to the following limitations:

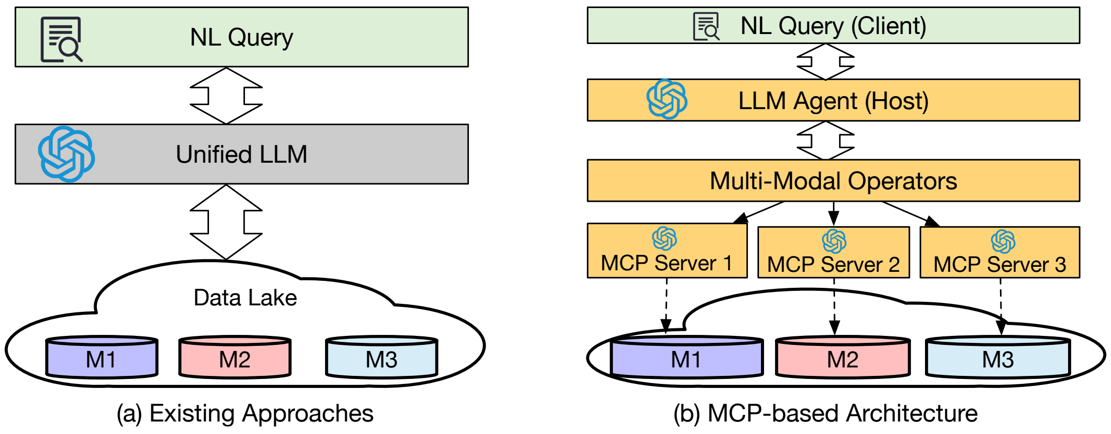


Figure 1: An Illustrate Example on MCP-based Multi-Modal Data Analytics.

Limitation 1: Limited Query Expressiveness. Existing approaches typically support querying only a subset of data modalities and suffer from limited expressiveness in capturing complex user intents. For example, ELEET [24] supports analytics over text and tabular data; Palimpzest [11] is restricted to text and image processing; and AOP [25] primarily targets document and text data. Underlying these systems are three main strategies for query translation: (1) mapping natural language (NL) directly to modality-specific operations [25], (2) translating NL to SQL using NL2SQL techniques [13, 6], or (3) requiring users to write declarative SQL-like query languages [11]. However, these methods remain inadequate for data lakes, where user queries are often ambiguous and span a wide range of data modalities.

Limitation 2: High Inference Overhead. Most existing approaches rely on a single, unified LLM to process heterogeneous data modalities, which leads to significant inference overhead. This issue is attributed to the computational cost of state-of-the-art LLMs, such as GPT-4, which contains over a trillion parameters. Moreover, queries that span multiple modalities introduce complexity in query planning and execution, further impacting efficiency. From a query processing perspective, users typically expect low-latency responses, making it imperative to reduce inference overhead.

Limitation 3: Knowledge and Data Staleness Data in the lake is often incomplete or becomes outdated over time, which undermines the freshness and reliability of analytical results. Moreover, the knowledge embedded in LLMs can also become stale as new data and emerging patterns (e.g., novel diseases or evolving user behavior) are not reflected in the original training data. For example, an LLM may fail to recognize symptoms of a newly discovered disease or interpret related medical images correctly. Existing approaches largely ignore this issue and operate under static assumptions. Therefore, it is essential to develop mechanisms for augmenting stale data and refreshing LLM knowledge to ensure up-to-date and contextually relevant analytics.

Model Context Protocol (MCP) [8] is a novel framework that standardizes the interaction among AI agents (e.g., knowledge-grounded LLMs), external tools (e.g., database engines, function calls), and data resources (e.g., structured and unstructured sources). By defining a unified interface, MCP enables seamless integration of real-time inputs, environmental variables, and domain-specific constraints, thereby supporting robust and scalable decision-making across diverse applications. In the context of multi-modal data analytics, MCP provides an effective abstraction layer over heterogeneous data lakes. Users can interact with AI agents through natural language, while the analytics workload is offloaded to dedicated MCP servers, each tailored to handle a specific data modality. Moreover, with the growing ecosystem of MCP servers, thousands of off-the-shelf components will be readily available for processing multi-modal data efficiently.

Building upon the capabilities of MCP, we propose a novel multi-modal data analytics system, named TAIJI, to address the aforementioned limitations. TAIJI introduces a new architecture that leverages MCP to offload modality-specific analytical tasks to dedicated MCP servers. As illustrated in Figure 1, TAIJI differs fundamentally from conventional approaches that rely on a single, unified LLM to process all modalities (e.g., M1, M2, M3 represent different data modalities). Instead, TAIJI adopts a client–host architecture: the client receives an NL query, and a host-side LLM agent interprets the user intent. This agent decomposes the query into a set of modality-specific operators and formulates a structured query plan. Each sub-plan is dispatched to an appropriate MCP server, where a tailored LLM, which is optimized for the corresponding modality, executes the analysis. This modular and distributed design yields several key advantages: (1) higher inference accuracy by leveraging specialized models, (2) improved scalability through parallelism across servers, and (3) significantly reduced inference overhead compared to monolithic LLM-based solutions.

In summary, this paper has the following key contributions:

(1) **MCP-Based Multi-Modal Analytical System.** We propose a novel multi-modal data analytics framework built upon the Model Context Protocol (MCP), which addresses the limitations of existing LLM-based systems. The core idea is to assign each MCP server a tailored LLM optimized for a specific data modality, embracing the principle that "one size does not fit all".

(2) **Semantic Operator Hierarchy.** We introduce a hierarchical set of semantic operators that spans structured, semi-structured, and unstructured data. This design enables TAIJI to support advanced tasks such as cross-modal joins, while maintaining compatibility with existing operators for relational data, documents, graphs, images, audio, and video, thus avoiding "reinventing the wheels".

(3) **Query-Driven Model Optimization and Data Discovery.** We develop a query-driven fine-tuning strategy to optimize the reasoning ability of LLMs on each MCP server. Moreover, we propose a unified embedding-based semantic representation combined with a hybrid indexing mechanism for multi-modal data discovery.

(4) **Dynamic Knowledge and Data Refreshing.** We propose a twofold updating mechanism to ensure freshness of both the data and the LLMs. First, we leverage deep research techniques to enrich the data lake with the latest documents and web content. Second, we introduce a lightweight editing mechanism (supporting insert, update, and delete operations) augmented by machine unlearning techniques to refresh LLM knowledge.

2 Preliminaries

2.1 Problem Definition

Multi-Modal Query Processing (MMQP). Given a natural language (NL) query Q over a collection of datasets $\mathbb{D}$, where each dataset $D_i \in \mathbb{D}$ is associated with a modality $M_i \in \mathbb{M}$, the goal is to compute a result set R that satisfies the query predicates, where each result tuple $r \in R$ may contain data items spanning multiple modalities in $\mathbb{M}$. To evaluate the performance of MMQP, we adopt three standard metrics: (1) recall, which measures the completeness of the retrieved results, (2) precision, which quantifies the ratio of the correct ones among the returned results. and (3) latency, which reflects the query execution efficiency.

In general, the MMQP problem has the following challenges:

Challenge 1: Cross-Modality Query Planning. Unlike traditional query planning in relational databases that focuses solely on relational operators, orchestrating operator pipelines across multiple data modalities is more complex. This complexity arises from the heterogeneous nature of operators and data formats. Therefore, MMQP requires an effective yet lightweight mechanism to generate reasonable execution plans.

Challenge 2: Efficient Inference on Large-Scale Data. Compared with using a general-purpose LLM with a massive number of parameters, employing a tailored, relatively lightweight LLM for modality-specific data can significantly reduce inference overhead. However, this approach still encounters scalability challenges when dealing with large volumes of data, particularly unstructured data such as text, images, or videos.

Challenge 3: Data and Knowlege Freshness. High-quality data in the data lake is often insufficient to precisely and comprehensively answer user queries. Therefore, it is crucial to continuously update both the data lake and the LLM knowledge within each MCP server. However, open-domain data sources are vast and challenging to explore, making integration into the data lake cumbersome and resource-intensive.

2.2 Related Work

In recent years, LLMs have made remarkable advances in natural language understanding, generation, and reasoning, opening up new opportunities for multi-modal data management. By expressing their query requirements in natural language, users can interact with systems that leverage LLMs to interpret query intent and dynamically orchestrate external databases, knowledge bases, or APIs to execute complex analytical tasks.

CAESURA [23] (also known as ELEET [24]) utilizes LLMs as multimodal query planners that translate NL inputs into executable multimodal query plans. Its key idea is leveraging LLMs to understand user intent and dynamically construct efficient query workflows tailored to diverse data modalities. Building upon this, Gorilla [18] enhances LLMs' reasoning and tool-use capabilities by incorporating retrieval-augmented generation (RAG), enabling real-time access to external knowledge bases and APIs during multimodal query processing. Toolformer [21] further advances LLM tool integration by fine-tuning models with limited examples, empowering them to autonomously invoke external APIs, such as database engines and computational tools, during generation. This overcomes LLMs' traditional limitation to static text generation, enabling dynamic interaction with external data sources for complex analytical tasks. ThalamusDB [9] introduces a hybrid architecture combining a dedicated query planner with deep learning inference. It allows SQL queries to include NL predicates, offloading parts of the execution to neural models for image and text processing, while leveraging approximate query processing (AQP) to balance efficiency and accuracy. However, it still relies on manual annotations, which may hinder scalability. PALIMPZEST [12, 11] extends declarative query languages to express AI workloads, generating optimized plans that jointly consider the cost of both AI inference and relational operations. AOP [25, 26] proposes defining semantic operators over documents, text, and structured tables, and employs LLMs to iteratively generate executable query plans for multi-modal content.

However, existing works have three main limitations: First, they support only a few data modalities and are hard to extend, i.e., adding new modalities or operators often requires heavy manual work. In contrast, TAIJI supports a wide range of multimodal operators with a scalable MCP-based architecture. Second, most systems use a single LLM to handle all data types, which leads to low efficiency and accuracy. TAIJI instead uses tailored LLMs for different modalities, improving performance. Third, they focus on static data and ignore updates. TAIJI addresses this by automatically enriching the data and refreshing model knowledge over time.

3 TAIJI's Overview

Figure 2 presents an overview of TAIJI, which consists of three main components: **MCP Client Manager**, **MCP Server Manager** and **MCP Augmentor**. In the following, we introduce them in details.

3.1 MCP Client Manager

3.1.1 NL2Operator

Understanding user intent is critical for accurate query answering. However, neither SQL-like declarative languages nor UDF-based procedural languages are well-suited for this task. This is due to two main challenges: First, NL queries are inherently ambiguous and difficult to interpret accurately. Second, translating NL queries into complex declarative or procedural languages increases the likelihood of inaccuracies or errors.

To address this, we propose NL2Operator, a method that defines a hierarchical set of semantic operators, each linked to specific MCP servers. An LLM-based agent maps user queries to these operators based on query intent. For example, semi-structured data processing is handled by an intermediate MCP server, which

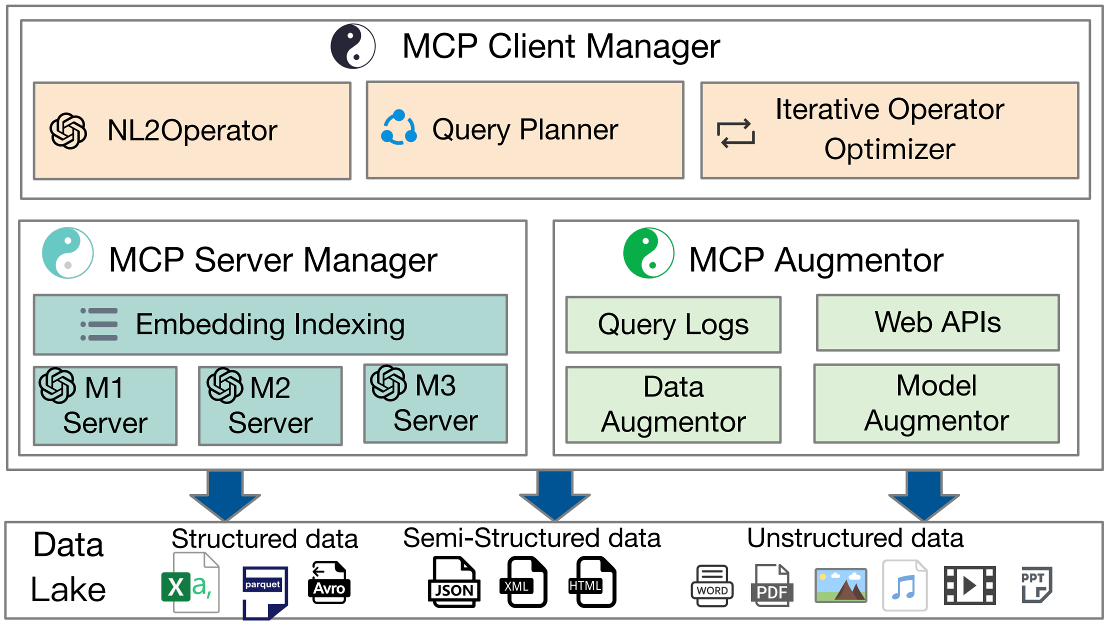


Figure 2: An Overview of TAIJI.

can either process the query directly or delegate it to specialized sub-servers (e.g., for JSON, HTML, XML). This hierarchical design offers three advantages: (1) Scalability and flexibility, by modularizing the processing across different servers; (2) High concurrency support, through distributed workload balancing; (3) Simplified translation, by mapping NL to high-level modality-specific operators, reducing the burden on the LLM.

3.1.2 Query Planner

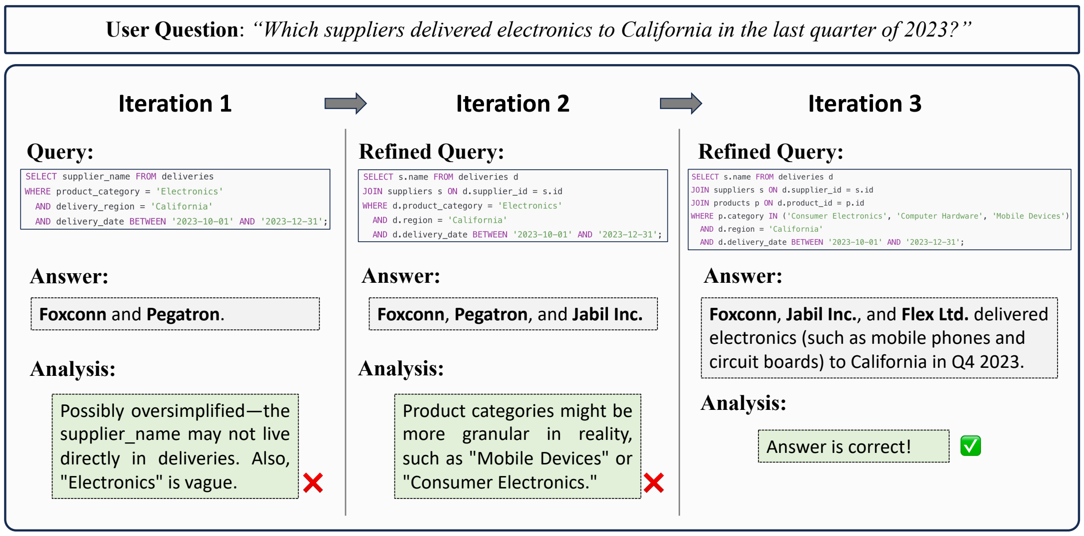
The goal of query planner is to find the execution plan with the lowest cost, thereby improving overall efficiency. Based on a hierarchy of semantic operators, we construct a directed acyclic graph (DAG) to represent the query workflow. We then implement a sampling-based cost estimation optimizer, which includes selectivity estimation, cost modeling, plan evaluation, and final plan selection.

The optimization process works as follows: First, an operator cost sampler collects runtime statistics, such as per-tuple processing time and selectivity, for each operator. Using this data, the optimizer builds a latency-based cost model to evaluate candidate plans. The plan with the lowest estimated execution time is selected for execution. This sampling-based approach supports accurate, data-driven cost prediction and adaptive plan selection. It offers two key advantages: (1) it maintains reliable cost estimates under different data distributions and runtime conditions, and (2) it produces more deterministic plans compared to LLM-based query planning.

3.1.3 Iterative Operator Optimizer

Each MCP server is responsible for processing data from a specific modality, autonomously interacting with its underlying data source, whether structured (e.g., SQL-based) or unstructured (e.g., document corpus), via an Iterative operator optimization [10, 22]. Unlike traditional pipelines that rely on a single retrieval pass, TAIJI introduces a bidirectional communication channel between the MCP server and the MCP client (or host). This feedback loop enables the server to request prompt refinements or clarifications from the upstream controller when retrieval results are insufficient, supporting a dynamic, feedback-driven, and iterative optimization process.

An example of the iterative architecture in TAIJI is illustrated in Figure 4. The process starts when the MCP


User Question: "Which suppliers delivered electronics to California in the last quarter of 2023?"
Iteration 1
Iteration 2
Iteration 3
Query:
Refined Query:
Refined Query:
Answer:
Foxconn and Pegatron.
Answer:
Foxconn, Pegatron, and Jabil Inc.
Answer:
Foxconn, Jabil Inc., and Flex Ltd. delivered electronics (such as mobile phones and circuit boards) to California in Q4 2023.
Analysis:
Possibly oversimplified—the supplier_name may not live directly in deliveries. Also, "Electronics" is vague.
Analysis:
Product categories might be more granular in reality, such as "Mobile Devices" or "Consumer Electronics."
Analysis:
Answer is correct!

```sql
SELECT supplier_name FROM deliveries
WHERE product_category = 'Electronics'
  AND delivery_region = 'California'
  AND delivery_date BETWEEN '2023-10-01' AND '2023-12-31';
```

```sql
SELECT s.name FROM deliveries d
JOIN suppliers s ON d.supplier_id = s.id
WHERE d.product_category = 'Electronics'
  AND d.region = 'California'
  AND d.delivery_date BETWEEN '2023-10-01' AND '2023-12-31';
```

```sql
SELECT s.name FROM deliveries d
JOIN suppliers s ON d.supplier_id = s.id
JOIN products p ON d.product_id = p.id
WHERE p.category IN ('Consumer Electronics', 'Computer Hardware', 'Mobile Devices')
  AND d.region = 'California'
  AND d.delivery_date BETWEEN '2023-10-01' AND '2023-12-31';
```

Figure 3: An illustrative example of iterative operator optimization in TAIJI.

server receives a sub-task from the central LLM agent. Guided by the task's semantics and structure, the server formulates an initial query and retrieves candidate results from its local database. These results are evaluated along multiple dimensions, including coverage, redundancy, ambiguity, and informativeness. If deficiencies are detected (e.g., sparse or misaligned results), the server engages a reasoning-based refinement loop to revise the query and improve alignment with the task's underlying intent. This iterative cycle continues until the result set meets a predefined confidence threshold. Moreover, we employ a curriculum learning strategy that gradually increases task complexity, from simple key-value lookups to multi-hop reasoning over relational and multimodal data, thus guiding the agent through progressively more abstract levels of query planning and evaluation.

3.2 MCP Server Manager

3.2.1 Embedding Indexing

For high-dimensional vector data, traditional vector indexing methods [3] typically construct a graph structure by linking neighboring data points based on nearest-neighbor distance. However, when metadata-based filters are applied, many of these connections become invalid, resulting in disconnected subgraphs. This leads to search failures, as the index may no longer be able to reach the true nearest neighbors. To address this, TAIJI introduces a filter-aware vector indexing approach designed to maintain search effectiveness even under filtering constraints. The key idea is to selectively augment the vector graph by adding edges that connect only those data points that comply with the filtering conditions. This ensures that the search process remains efficient while achieving high recall. Figure 4 illustrates the core concept. The key components of our method include:

(1) Condition-aware graph augmentation: Dynamically reinforce valid connections among filter-compliant nodes to ensure index traversability even after metadata-based filtering is applied.

(2) Hybrid search robustness: Maintain a balance between filtering precision and vector search accuracy by preserving essential traversal paths within the index.

(3) Recall preservation: Ensure that the augmented index structure delivers competitive nearest-neighbor retrieval performance, comparable to that of unfiltered searches.

This approach extends traditional vector search to support constrained queries where both semantic similarity (via embeddings) and structured criteria (via metadata) must be jointly satisfied.

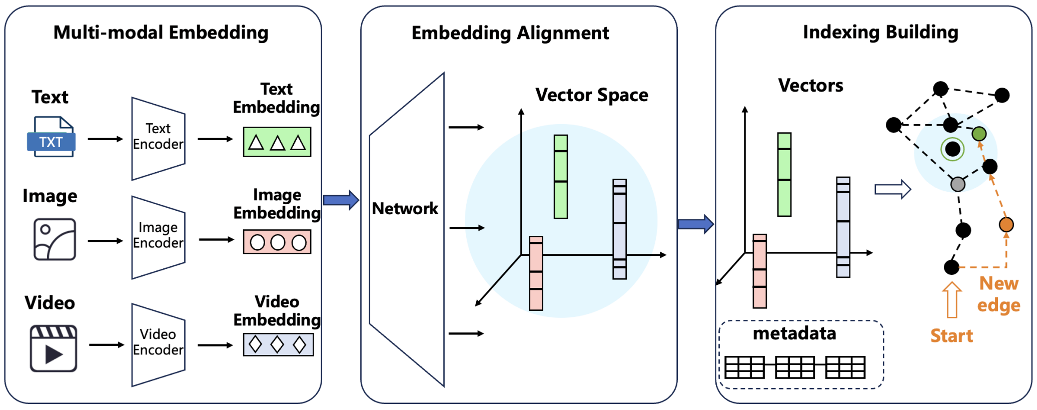


Figure 4: Embedding of Multi-Modal Data.

3.2.2 Multi-Modal MCP Servers

Multi-Modal MCP Servers deal with heterogeneous data ecosystems by unifying structured (e.g., relational databases), semi-structured (e.g., JSON logs, XML), and unstructured data (e.g., text, images, sensor streams) under a single adaptive framework. Leveraging LLMs and multi-modal AI architectures, these servers dynamically interpret, classify, and contextualize diverse data formats through techniques such as natural language processing (NLP) for unstructured text, computer vision for visual data, and graph-based reasoning for structured relationships. By embedding semantic understanding into the MCP layer, the system autonomously generates metadata, enforces cross-modal data governance policies, and enables federated queries that bridge tabular sales records, semi-structured IoT telemetry, and unstructured social media feeds. For instance, an LLM-powered MCP Server could correlate customer sentiment (extracted from unstructured reviews) with structured transactional data to optimize supply chain decisions, while simultaneously parsing semi-structured maintenance logs to predict equipment failures.

3.3 MCP Augmentor

3.3.1 Data Augmentor

After executing queries on modality-specific databases, each MCP server autonomously initiates an update procedure involving both data augmentation and model capability enhancement, as shown in Figure 5. In the data augmentation phase, TAIJI adopts a query-driven retrieval-then-synthesis strategy to keep the data lake semantically enriched and aligned with the latest developments. Upon completion of a user query, the MCP server triggers targeted information harvesting routines. These leverage both web crawlers and structured API-based agents configured to access dynamic and authoritative sources such as arXiv papers, GitHub repositories, Stack Overflow, and enterprise technical documentation.

The collected documents undergo a multi-stage processing pipeline designed to transform unstructured content into validated, semantically indexed knowledge.

(1) Structural Reconstruction: The pipeline begins with document structure parsing, utilizing tools such as ScienceParse [7] and GROBID [14] to extract key structured elements, including titles, abstracts, section headers, equations, tables, figures, and code snippets. These tools employ a combination of rule-based heuristics and machine learning techniques to reconstruct the hierarchical organization of scientific and technical documents from raw formats such as PDFs and HTML.

(2) Redundancy Elimination: In the redundancy elimination stage, a hybrid strategy combines MinHash-based fingerprinting for fast approximate set similarity detection with dense embedding-based retrieval. Specifically, documents are first fingerprinted using MinHash signatures generated over token-level n-grams. In paral-

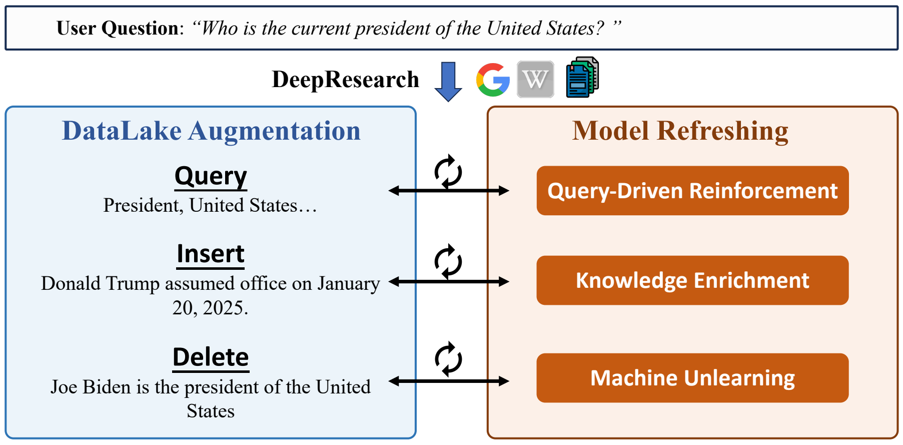

User Question: "Who is the current president of the United States? "
DeepResearch
DataLake Augmentation
Query
President, United States…
Insert
Donald Trump assumed office on January 20, 2025.
Delete
Joe Biden is the president of the United States
Model Refreshing
Query-Driven Reinforcement
Knowledge Enrichment
Machine Unlearning

Figure 5: Data Augmentation and Model Refreshing in TAIJI.

lel, dense vector embeddings are computed using models like Sentence-BERT [20], and clustered using HNSW-based approximate nearest neighbor search (via libraries like FAISS) to detect semantically similar passages. Duplicate or near-duplicate entries are then filtered out based on a configurable similarity threshold.

(3) Entity Extraction: Next, domain-adapted named entity recognition (NER) is applied using fine-tuned transformer-based models trained on labeled datasets from scientific and software domains. To resolve synonyms and disambiguate entities across sources, we apply context-aware entity linking using string similarity, co-occurrence statistics, and external knowledge bases (e.g., DBpedia, PapersWithCode APIs). To identify novel concepts and facts, we compute concept-level hashes using SimHash, which captures the semantic footprint of passages. These hashes are used to cluster conceptually similar content and measure deviation from existing entries in the data lake. We also apply graph-based semantic clustering over the extracted entities and their relationships to reveal emergent topics not previously indexed.

(4) Data Augmentation: In the final augmentation and indexing stage, each candidate entry is validated across multiple independent sources to ensure factual consistency and reliability. Only entries that are corroborated—i.e., referenced in at least two unrelated sources—are retained. The resulting validated knowledge units are indexed along multiple axes: modality (e.g., text, code, image), source credibility, and temporal metadata (e.g., crawl time, publication date). This indexing supports temporal knowledge reasoning, such as prioritizing more recent findings or applying decay-weighted relevance during downstream model retrieval.

3.3.2 Model Refreshing

In the Model Refreshing phase, to align model knowledge with the evolving data lake, we introduce a modality-specific parameter update mechanism tailored to three data operations: querying, insertion, and deletion. Thanks to the built-in subscription function of MCP server, the MCP client can monitor the changes of data resources via subcription, then we can conduct the model refreshing.

(1) Query-Driven Reinforcement: To enhance model performance based on frequently asked or emergent user intents identified from query logs, we first perform an in-depth analysis to extract the most common and critical intent patterns. These patterns are then leveraged to generate task-specific training samples. Furthermore, to ensure that these synthesized samples effectively contribute to the model's learning process, we apply importance sampling during the fine-tuning phase. Importance sampling assigns a higher weight to these specific training examples, based on their frequency and relevance in real-world use cases. This mechanism helps the model prioritize the learning of high-value user intents while preventing overfitting to less common or irrelevant patterns. By combining these methods, we significantly improve the MCP server's ability to understand and respond to emergent user needs in real-time applications.

(2) Insert via Knowledge Enrichment: When new knowledge is ingested into the data lake, it undergoes a structured transformation where relevant text spans or structured tuples are converted into instruction-style training samples. These samples are then systematically appended to the fine-tuning dataset. The conversion process involves identifying key knowledge components and framing them as prompts and expected responses, ensuring that the newly introduced information is aligned with the model's reasoning mechanisms. By incorporating these structured samples, the model is exposed to explicit tasks that require reasoning over the updated knowledge, enabling it to more effectively process and apply this new information during inference. This method directly enhances the model's capacity for knowledge integration, thereby improving its performance on tasks that necessitate reasoning over both the pre-existing and newly added knowledge. Furthermore, this fine-tuning approach mitigates the risk of knowledge forgetting, as the model learns to consistently reference the most recent data alongside the foundational knowledge it was initially trained on.

(3) Deletion via Machine Unlearning: To address the removal of obsolete or sensitive knowledge, we employ advanced techniques such as gradient ascent unlearning or influence function-based data deletion to effectively erase the target information without compromising the integrity of the remaining model knowledge [27]. Specifically, we first compute data influence scores, which quantify the contribution of individual training samples to the model's predictions. These scores are utilized to identify the most influential data points that need to be removed. Subsequently, we employ a fine-tuning strategy where negatively-weighted gradients are applied to the model, directing it to unlearn the targeted knowledge. This process is executed in a manner that is mindful of the need to preserve the model's retained knowledge, ensuring that the unlearning of specific information does not lead to catastrophic forgetting of other critical data or degrade the overall performance [5]. The integration of these techniques enables a controlled and efficient forgetting mechanism that minimizes interference with the model's pre-existing knowledge, making it an ideal approach for handling sensitive or outdated information.

This unified update mechanism ensures that each MCP server maintains alignment between its internal LLM and its evolving modality-specific data lake, enabling accurate, efficient, and up-to-date multi-modal response. In particular, it facilitates synchronized parameter tuning and schema-aware knowledge injection across modalities such as vision, audio, and text, thereby preventing semantic drift and enhancing the consistency of cross-modal representations. This design also supports incremental updates without requiring full model retraining, significantly reducing computational overhead while preserving reasoning fidelity.

4 Preliminary Experiments

In this section, we evaluate the performance of a prototype of TAIJI, demonstrating the efficiency and effectiveness of the proposed MCP-based architecture on handling multi-modal data.

4.1 Experimental Setup

Dataset. We use the Craigslist furniture dataset [9], which contains both relational and image data, to evaluate execution accuracy and latency of TAIJI. The dataset comprises 3000 listings from the "furniture" category in website craigslist.org, detailing items such as sofas, tables, and chairs. It consists of two tables: *furniture* and *image*. The *furniture* table includes information of each entity; with each furniture has one or more images related to it, the *image* table contains the path of images corresponding to each furniture.

We design three queries to verify that TAIJI can demonstrate advantages concerning varied selectivity and task complexity. For simplicity, the query plan is fixed by performing a filter on the furniture table and then conducting matches on images with specified predicates. The intermediate result size refers to the size of set filtered by relational predicates, and image predicate refers to the image classification task. Table 1 gives the summary of the workload.

Table 1: Summary of the Workload

Query ID	Intermediate Size	Image Predicate	NL Query
1	33	images with two chairs	Find a set of two chairs
2	126	images with a black chair	Find a black leather chair
3	347	images with table and chair	Find a set of wood table and chair

MCP Client Setting and Server Setting. As introduced before, we implement a disaggregated MCP-based data analytical system. Specifically, we leverage the ChatGPT API [1] in the MCP client, which is built upon the GPT-4.1 architecture [17]. Additionally, We use the Qwen2.5-VL-7B [4] as an MCP server to analyze complex images. We also deploy another MCP server [19] to manage the structured data and semi-structured data based on a PostgreSQL database.

Baseline. We compare our method to a pure GPT-4.1 model, which handles the query translation and data analysis altogether. On the contrary, our approach splits the task to three sub-tasks with MCP: query translation, table filtering, and image analysis. Note that as GPT-4.1 is not good at processing tabular data, we utilize PostgreSQL to help it handle the table filtering for a fair comparison.

Evaluation Metric. For the evaluation of TAIJI, we employ three metrics: recall, accuracy and latency, which are respectively employed to evaluate correctness and efficiency. We compare the result set retrieved by client with the ground truth, to compute recall and precision. We also evaluate latency of performing the queries in an end-to-end fasion.

4.2 Experimental Results

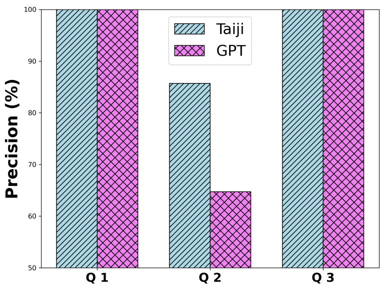


Figure 6: Precision Evaluation.

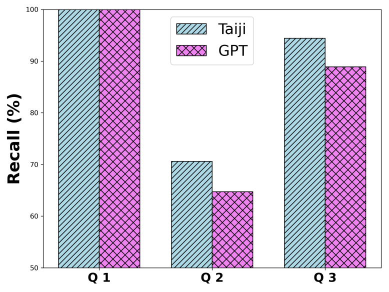


Figure 7: Recall Evaluation.

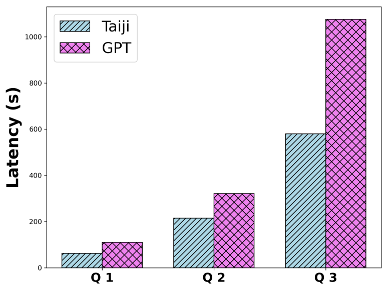


Figure 8: Latency Evaluation.

As shown in Figure 6 and Figure 7, TAIJI also outperforms GPT in accuracy (for Q2) and recall (on Q2 and Q3). In specific, TAIJI achieves an accuracy of 85% on Q2 while GPT has an accuracy of 65%.; TAIJI has a recall of 71% and 94% on Q2 and Q3, respectively while GPT has a s a recall of 65% and 89%. Take Q2 as an example where TAIJI is better than GPT both in precision and recall, we found that it is hard to find the black chair from the candidate images, the reason is that GPT can hardly figure out black chair and black sofa.

Figure 8 depicts the latency between TAIJI and GPT-4.1. It is clearly visible that our approach is largely faster than GPT. As the intermediate results increase (i.e., Q1-Q3), the performance gap increases. On average, TAIJI improves the latency by 43%. Take Q3 as an example where the intermediate size has reached up to 347, GPT is too heavy to offer high efficiency with its trillion parameters.

Implications of MCP-based Architecture over pure LLM. Through the preliminary experiments, we have the following insights. First, compared with pure LLM, MCP-based architecture can judiciously select and harness most proper LLMs to process multi-modal data. For instance, GPT may be excel at text processing and query translation, but it turns out that Qwen is better than GPT on image processing. Second, leveraging a tailored LLM (i.e., Qwen) with a much smaller number of parameters can not only improve the accuracy, but also it can result in a much lower inference overhead. This implies that, MCP-based architecture is very promising to combine many small LLMs for a wide range of data modality. Third, since MCP-based architecture can offload a sub-task to a local MCP server, it further reduces the burden of the centric LLM. Such an observation motivates us to explore more optimizations in the future, such as asynchronous execution, task scheduling, etc.

5 Challenges and Opportunity

MCP-based Data Security. Despite the salient features discussed in the previous sections, a major challenge of MCP-based data analytic is how to preserve the data security while performing the LLM-empowered data analytics. This is because MCP servers are developed by different vendors and contributors all over the world, and analyzing the plain text/documents in the data lake poses high risks of data leakage. For instance, the MCP SDK or build-in LLM agents may upload the data to the web server. Therefore, it calls for new mechanisms to enable privacy-preserved data transmission and analytics. On the one hand, the data transmission tunnel should be secured among MCP host/clients, MCP servers, and resources. On the other hand, it shoud avoid accessing the plain text, other alternatives should be explored, such as query processing over ciphertext directly. However, it is challenging to balance the trade-off between query efficiency and data privacy.
Hybrid Deployment Optimization. In addition to the local deployment of MCP servers, it is also possible to deploy remote MCP servers, resulting in a hybrid deployment. Hence, it calls for new methods that can harness the capability of hyrid deployment. However, it is challenging to effectively determine the strategy of data placement and to migrate the data among the multiple MCP servers. Moreover, it is also hard to efficiently discover data of high quality due to the hugh search space and high latency.
Cost-Aware Model Mangement. As more and more MCP servers are involved, it has become a challenge to manage the multiple LLMs and their evolvement. The reason is two-fold. First, existing approaches [24, 9, 25] only care about the query performance, while neglecting the dollar cost of LLMs' invocation as they charge for tokens. Second, model knowledge can also be obsolete as discussed before, and performing the fine-tuning also incur significant cost, thus it is challenging to balance the knowledge freshness and training cost. As a result, it is required to judiciously select and fine-tune the LLM models with cost-aware optimization.
Multi-Modal Database Benchmark. Since there lacks a unified and standard multi-modal database benchmark, existing approaches basically evaluate the performance on different datasets and workloads. Hence, there is a pressing need to have a new multi-modal database benchmark that covers various data modality and workloads. However, it is challenging to collect or generate multi-modal data and workloads with both fidelity, utility, and generality, even for one modality, there could be numerous variants.

6 Conclusion

In this paper, we present TAIJI, a novel multi-modal data analytical system. Particularly, we design a new architecture based on Model Context Protocol (MCP) that offloads the multi-modal operators to specific MCP server. We propose a series of components including NL2Operator query translation, multi-modal query planning, iterative RAG, embedding indexing, data augmentation, model refreshing. To the best of our knowledge, this is the first MCP-based architecture for multi-modal analytical system. We believe MCP architecture opens the door for the researches over multi-modal data management. In the future, we will finish implementing the whole system and conduct more experiments.

References

[1] Introduction of openai text generation apis. `https://platform.openai.com/docs/guides/text-generation`.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] I. Azizi, K. Echihabi, and T. Palpanas. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proc. ACM Manag. Data*, 3(1):43:1–43:31, 2025.

[4] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025.

[5] R. Chourasia and N. Shah. Forget unlearning: Towards true data-deletion in machine learning. volume 202, pages 6028–6073. PMLR, 2023.

[6] J. Fan, Z. Gu, S. Zhang, Y. Zhang, Z. Chen, L. Cao, G. Li, S. Madden, X. Du, and N. Tang. Combining small language models and large language models for zero-shot nl2sql. *Proceedings of the VLDB Endowment*, 17(11):2750–2763, 2024.

[7] A. I. for AI. Science parse: An open-source library for extracting structured data from scientific pdfs, 2018. `https://github.com/allenai/science-parse`.

[8] X. Hou, Y. Zhao, S. Wang, and H. Wang. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*, 2025.

[9] S. Jo and I. Trummer. Thalamusdb: Approximate query processing on multi-modal data. *Proceedings of the ACM on Management of Data*, 2(3):1–26, 2024.

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[11] C. Liu, M. Russo, M. Cafarella, L. Cao, P. B. Chen, Z. Chen, M. Franklin, T. Kraska, S. Madden, R. Shahout, et al. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*, 2025.

[12] C. Liu, M. Russo, M. Cafarella, L. Cao, P. B. Chen, Z. Chen, M. Franklin, T. Kraska, S. Madden, and G. Vitagliano. A declarative system for optimizing ai workloads. *arXiv preprint arXiv:2405.14696*, 2024.

[13] X. Liu, S. Shen, B. Li, P. Ma, R. Jiang, Y. Zhang, J. Fan, G. Li, N. Tang, and Y. Luo. A survey of nl2sql with large language models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109*, 2024.

[14] P. Lopez. Grobid: Generation of bibliographic data, 2020. `https://github.com/kermitt2/grobid`.

[15] J. Lu and I. Holubová. Multi-model databases: a new journey to handle the variety of data. *ACM Computing Surveys (CSUR)*, 52(3):1–38, 2019.

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

[17] OpenAI and e. a. Achiam. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[18] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

[19] M. C. Protocol. Model context protocol servers, 2025. `https://github.com/modelcontextprotocol/servers`.

[20] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992, 2019.

[21] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[22] K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. Szlam, and J. Weston. Language models that seek for knowledge: Modular search and generation for dialogue and prompt completion. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 373–393, 2022.

[23] M. Urban and C. Binnig. Caesura: Language models as multi-modal query planners. *arXiv preprint arXiv:2308.03424*, 2023.

[24] M. Urban and C. Binnig. Eleet: Efficient learned query execution over text and tables. *Proc. VLDB Endow*, 17:13, 2024.

[25] J. Wang and G. Li. Aop: Automated and interactive llm pipeline orchestration for answering complex queries. CIDR, 2025.

[26] J. Wang, G. Li, and J. Feng. idatalake: An llm-powered analytics system on data lakes. *Data Engineering*, page 57.

[27] Y. Zhang, Z. Hu, Y. Bai, J. Wu, Q. Wang, and F. Feng. Recommendation unlearning via influence function. *arXiv preprint arXiv:2307.02147*, 2023.