# Utilizing LLM-based Multi-agent Framework for Pipelining Complex Data Discovery Tasks

Qianyu Yang
Arizona State University
Tempe, Arizona, USA
qyang129@asu.edu

Jin Wang
Arizona State University
Tempe, Arizona, USA
jinwang18@asu.edu

## ABSTRACT

Data discovery from data lakes is an essential task in many real world applications and has gained great traction from the database community. While many previous efforts have improved the performance of data discovery from different aspects, they failed to adequately support complex tasks that require multiple operations to finish. In this paper, we envision a Large Language Model (LLM) based multi-agent system that aims to pipeline complex data discovery tasks. We comprehensively discuss the challenges of building such an end-to-end data discovery framework and analyze the benefits of LLMs in addressing them. Then, we come up with a multi-agent system as the solution and introduce the potential design choices of different agents. Finally, we provide some initial experimental results based on LLM and raise several open questions to be resolved in realizing this vision.

**VLDB Workshop Reference Format:**
Qianyu Yang and Jin Wang. Utilizing LLM-based Multi-agent Framework for Pipelining Complex Data Discovery Tasks. VLDB 2026 Workshop: DASHSys: Systems for Data-centric Agents with Human-in-the-loop.

## 1 INTRODUCTION

The past few decades have witnessed a rapid growth in the number of open and shared datasets from governments, academic institutions, and enterprises. To explore and gain insights from such massive data collections which are also known as *data lake*, researchers from both industry and academia have made great efforts to build many *data discovery* systems. Data discovery plays a significant role in modern data science, as it could help users find and access specific data they need from massive data lakes for the required downstream tasks. It is essential in many real-world applications, such as machine learning, data cleaning and exploratory data analysis [1, 5].

Due to its importance, data discovery has become a popular topic in the database community for a long time. Many previous studies [3, 6, 9, 10, 17] have focused on improving the effectiveness, efficiency and usability of different data discovery tasks, such as table union search, joinable table search and domain discovery. While these efforts could improve the performance of some particular data discovery operation, there is still a gap in applying them to real world scenarios where multiple operations are needed to finish one complicated task. Some early efforts towards this goal have been made by the BLEND framework [4]. However, it primarily considered the operations in relational databases but cannot effectively support semantic ones, which are widely required in real application scenarios now. This could be illustrated in the following example in Figure 1:

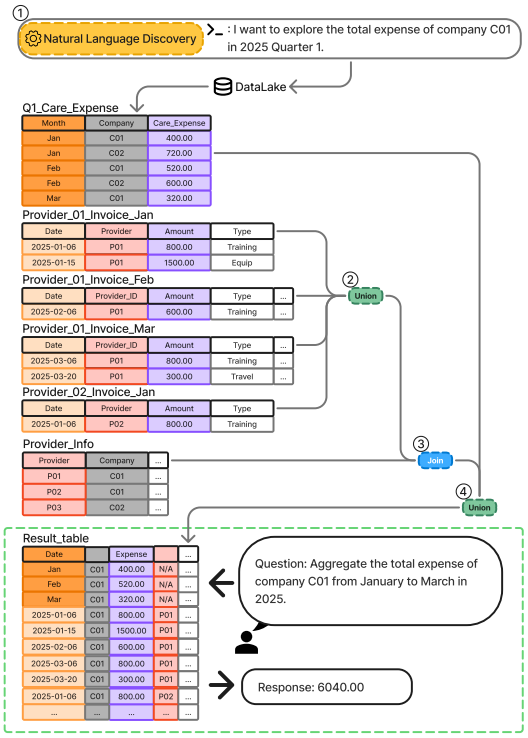


**Figure 1: Motivation Example of Complex Data Discovery.**

*Example 1.1.* Suppose a data scientist wants to conduct analysis on the enterprise data lakes to facilitate business decision making, the first step is to identify a table to start with. To this end, she could employ a data discovery operation that retrieves relevant tables according to the natural language input and then identify a table about Care Expense (Step 1). Then she would like to further find tables that are unionable or joinable with the table to enrich


Proceedings of the VLDB Endowment. ISSN 2150-8097.

the business insights (Step 2 to Step 4). As a result, she would obtain a table with sufficient information on the expenses of a company, which could be regarded as the output of the complex data discovery task. After that, she could make further data analysis on such a result table, e.g. ask the business question in natural language, and get the answer accordingly.

We can see that in the above scenario, it requires four data discovery operations, i.e. one Natural Language based table discovery (NLD) [17], two table union searches and one joinable table discovery [3], to finish the task. And it is essential to use semantic union/join operators in this process. Recently, Transformer-based Large Language Models (LLM) have been widely utilized to address table-centric tasks [7, 11, 13–15] due to their power in understanding human instructions as well as generating structured outputs. In this paper, we envision an LLM-based system as the solution to complex data discovery tasks by leveraging their superior abilities of planning and semantic understanding. The system adopts a multi-agent architecture [8] with four agents to provide an agentic data discovery workflow from user input understanding, pipeline construction, performance optimization to plan execution. We make several discussions about the potential design choices of the overall system as well as each agent to support complex data discovery tasks. Finally, we report some initial results of LLM-based data discovery operations and raise some open problems to be resolved in the next step.

## 2 SYSTEM OVERVIEW

To support end-to-end data discovery tasks with multiple steps, we propose a multi-agent framework that could automatically generate the data discovery plan based on the instructions from users. An overview of the system architecture is shown in Figure 2.

We will introduce the basic workflow of the system by referring to the application scenario illustrated in Figure 1 again: Firstly, users could send the requests via a predefined user interface. Following the example of systems that provide the NL query interface, the user could express the intention with a paragraph like *"I am a data scientist and would like to make analysis over the financial expenses of companies."*. The user input is then sent to the **Plan Agent**, which is responsible for parsing the user input and identifying the required functionalities. For example, the system could identify a logical plan with four data discovery operations ordered as NLD, Union, Join and Union, as shown in Figure 1. Next, the logical plan would be sent to the **Selection Agent** and transformed into a physical plan. The selection could happen in two aspects: on the one hand, there might be multiple tools to realize one data discovery operation. For example, there are multiple ways, from rule-based heuristics to Language Model-based methods, to implement the Union operation [5], indicating different kinds of trade-offs between effectiveness and efficiency; On the other hand, we need data selection over the datasets to be processed. Specifically, some language-model-based methods [6, 18] need to reduce the input size to satisfy the window length limitation or save the cost, and thus only a subset of the original table will be kept. For example, when deciding whether two tables are joinable with LLM, we need to include only a subset of each table rather than putting two full tables into the prompt to save the LLM cost. Finally, the **Execution Agent** receives the physical plan and returns the result table. For example, the result table in Figure 1 is obtained from the related tables of *Care Expense*, *Provider Invoice* and *Provider Info*, respectively. And the **Evaluation Agent** could answer questions from user input or benchmarking tasks on the result table.

Next, we would like to introduce the functionalities and research challenges in each agent of the multi-agent system:

*Plan Agent.* This agent aims to understand user intent, identify the essential steps of the pipeline and decide the required operations accordingly. To this end, it is essential to support various kinds of user interfaces, such as GUI that provides form-based input, Natural Language (NL) and code snippets in high level programming language. A recent trend [12, 19] is to support NL query interfaces where users could specify their requirements in a paragraph of natural language. Thus, an essential research challenge lies in how to turn user input in the format of NL description into structured logical plans, especially with the help of LLMs.

*Selection Agent.* The functionality of this agent is to transform the logical plan into a physical plan. As introduced in the above example, there are two kinds of selection to be realized: tool and data. For tool selection, the agent should maintain a list of potential tools to realize each logical operation, and make a selection based on requirements specified by users. To this end, the agent should be able to estimate the benefits of each tool and make decisions accordingly. LLM-based solutions for routing and planning might be helpful here. For data selection, there should be a set of potential operations that could filter out irrelevant parts of the table as well as conduct essential cleaning and normalization over the targeted table, especially when the selected tools are based on LLMs. Our previous work on table understanding tasks [18] could serve as a good foundation to reach this goal. An alternative solution is to let the LLM generate Python code to select the relevant data.

*Execution Agent.* This agent aims to execute the generated physical plan and return the result table. The major research challenge in designing the execution agent is to ensure the scalability and reduce the latency of the system. Due to the huge size of the data lakes, it could be very expensive to run multiple semantic Union and Join operations in the plan. Thus, we need to develop additional techniques such as indexing and query optimization for data lakes to provide system-level support, which also remains to be an open research challenge in the field of data lakes [1].

*Evaluation Agent.* This agent aims to connect the output of the Execution Agent, i.e. results of the complex data discovery task, to downstream tasks. To seamlessly integrate our system with other data science toolkit, we propose the Evaluation Agent to ease the evaluation of downstream tasks over such results. With the help of the Evaluation Agent, users could apply their benchmarking questions to the output of data discovery tasks and obtain the experimental results. If they are not satisfied with the initial results, they can provide some feedback to our system so as to obtain the desired data discovery outcomes. We also plan to support human-in-the-loop operations in our system so that the Plan, Selection and Execution agents could benefit from the user feedback. In this way, the results could be optimized gradually based on the collaboration of users and built-in strategies provided by our system.

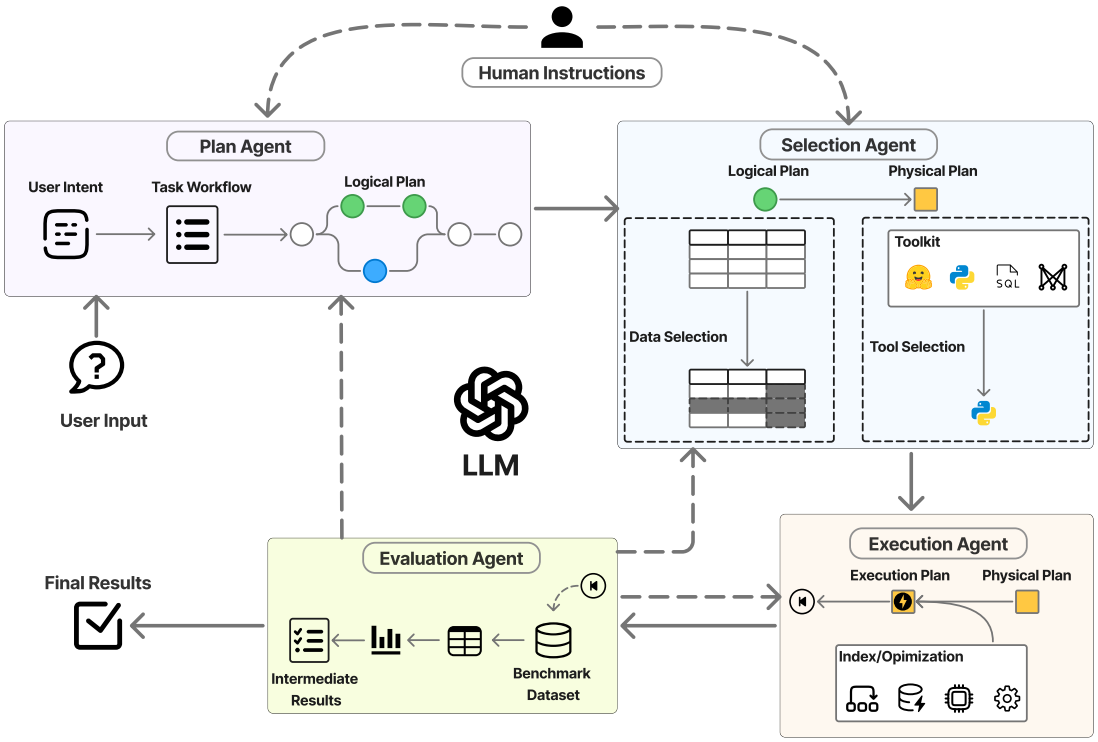


**Figure 2: System Architecture**

## 3 EVALUATION

In this section, we report some initial results about LLM-based data discovery operations. Specifically, we focus on the tasks of Table Union Search and Joinable Table Discovery. Note that these two tasks are inherited Information Retrieval style, where the input query and output results are both tables. Due to the huge size of data lakes, it is difficult to do the whole retrieval process via prompts over LLM. Thus, we believe that a reasonable way to apply LLM for these two tasks is to ask LLM to judge whether two tables are unionable or joinable, while the candidate tables will be retrieved by existing solutions. Specifically, given a query table, we first retrieve $K$ candidate tables with Starmie [6] and DeepJoin [3] for the unionable and joinable tables, respectively. Then, for each candidate table, we use a prompt to let the LLM verify whether it is unionable/joinable with the query table.

To save the cost of using LLM and improve the quality of the result, we employ some simple data selection techniques on tables proposed in our previous studies [6, 18] when constructing the prompts. Since tables in the data lakes are much larger than those in the table understanding tasks [18], we just keep up to 1% of rows in each query/candidate table for tables with more than 10,000 rows in the prompt. We use a row-wise order to sequentialize each table into the prompt and keep the header information. And the LLM is asked to output a binary decision based on the input question and tables while no demonstration example is provided. For Table Union Search, we evaluate on the SANTOS small dataset [10]; For Joinable Table Discovery, we select 40 query tables from the OpenData task in LakeBench [2] due to its huge size. The evaluation metric is Accuracy@K, which indicates whether the LLM could make correct prediction in the top-K retrieved tables. In this experiment, we vary the value of $K$ from 2 to 30. And we report the results of prompt over OpenAI APIs for GPT-5.4.

**Table 1: Results of Table Union Search**

| Value of K | 2 | 5 | 10 | 20 | 30 |
|---|---|---|---|---|---|
| Accuracy@K | 1.000 | 0.995 | 0.990 | 0.9775 | 0.9775 |
| # Input Tokens | 220,492 | 551,311 | 1,108,167 | 2,204,546 | 3,170,709 |
| # Output Tokens | 1,274 | 3,194 | 6,394 | 12,794 | 19,194 |

The results of Table Union Search are reported in Table 1. We observe that for unionable search, the results of accuracy are very promising as LLM is good at capturing the semantics from each table and decide the unionability accordingly. Such initial results illustrate that LLM could effectively verify whether two tables are unionable without seeing the full table. On the other hand, the LLM cost is somewhat high: we need more than 3 million input tokens when $K = 30$ as there are 1,200 pairs of tables to be verified.

Next, we look at the results of Joinable Table Discovery in Table 2. Compared with the unionable task, the accuracy is relatively lower.

**Table 2: Results of Joinable Table Discovery**

| Value of K | 2 | 5 | 10 | 20 | 30 |
|---|---|---|---|---|---|
| Accuracy@K | 0.763 | 0.760 | 0.793 | 0.821 | 0.833 |
| # Input Tokens | 386,566 | 978,593 | 2,011,032 | 4,129,612 | 6,246,717 |
| # Output Tokens | 6,398 | 15,755 | 30,129 | 56,931 | 83,570 |

The reason is that the joinability between two tables is decided by column-wise comparison. The LLM might not be able to recognize such information since the prompt just asked about table-level joinability. The results could be potentially improved by designing a better prompt: For instance, the tables could be sequentialized in a column-wise manner instead of row-wise one; There could be more instructions in the prompt template to remind the LLM to look at the joinability between pairs of columns; And some few-shot examples about joinability between columns could be provided.

From the above initial results, we could gain the following insights: Firstly, although the LLM-based methods for verifying the unionability and joinability are highly effective, the token cost could be huge due to the large size of data lake tables, even with aggressive data selection techniques. Thus, there should be multiple kinds of tools to implement the same operation in the system so as to make different types of trade-offs. Secondly, the system should be wisely guided about when to launch LLM-based methods. And it is essential to design advanced techniques in the Selection Agent and there could be filtering strategies that apply methods/models from cheap to expensive ones gradually. Thirdly, the system should provide an interface to let users provide their intention of goals about the trade-offs. For example, users could express the needs that they would like a plan that prioritizes saving the token cost. Then the Plan and Selection agents could use a cheaper LLM model or employ a more aggressive data selection strategy when constructing the prompt.

## 4 CONCLUSION AND FUTURE WORK

In this paper, we provided a blueprint for the multi-agent framework for pipelining complex data discovery tasks. We identified four essential agents in the agentic data discovery workflow: Plan, Selection, Execution and Evaluation. With the help of such agents, the system could automatically understand the user intention, generate an optimized plan and return the final answer by leveraging the power of LLMs. We reported some initial results of LLM-based data discovery operations and illustrated the importance of effective planning and data selection in the process.

There are many promising directions to realize the vision in this paper. Firstly, it requires essential efforts to understand user input and generate the initial pipeline accordingly. The success of LLM in task planning [16] paves a way to reach this goal. In addition, it is worth exploring effective techniques to make selections from tools to realize each step in the pipeline as well as keep useful and compact information in the targeted table data. Some initial efforts in the application of LLM-based table understanding [18] could be a good starting point towards this goal. Finally, it is important to propose efficient indexing and optimization techniques to improve the scalability of the system.

## REFERENCES


[1] Ziawasch Abedjan, Mahdi Esmailoghli, and Sainyam Galhorta. 2025. Data Discovery in Data Lakes: Operations, Indexes, Systems. *Proc. VLDB Endow.* 18, 12 (2025), 5455–5459.

[2] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. 2024. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proc. VLDB Endow.* 17, 8 (2024), 1925–1938.

[3] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-trained Language Models. *Proc. VLDB Endow.* 16, 10 (2023), 2458–2470.

[4] Mahdi Esmailoghli, Christoph Schnell, Renée J. Miller, and Ziawasch Abedjan. 2025. BLEND: A Unified Data Discovery System. In *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 737–750.

[5] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*. ACM, 69–75.

[6] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739.

[7] Juliana Freire, Grace Fan, Benjamin Feuer, Christos Koutras, Yurong Liu, Eduardo Peña, Aécio S. R. Santos, Cláudio T. Silva, and Eden Wu. 2025. Large Language Models for Data Discovery and Integration: Challenges and Opportunities. *IEEE Data Eng. Bull.* 49, 1 (2025), 3–31.

[8] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

[9] Yihao Hu, Jin Wang, and Sajjadur Rahman. 2025. LakeVisage: Towards Scalable, Flexible and Interactive Visualization Recommendation for Data Discovery over Data Lakes. *Proc. VLDB Endow.* 18, 10 (2025), 3545–3558.

[10] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proc. ACM Manag. Data* 1, 1 (2023), 9:1–9:25.

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

[12] Shu Liu, Soujanya Ponnapalli, Shreya Shankar, Sepanta Zeighami, Alan Zhu, Shubham Agarwal, Ruiqi Chen, Samion Suwito, Shuo Yuan, Ion Stoica, Matei Zaharia, Alvin Cheung, Natacha Crooks, Joseph Gonzalez, and Aditya G. Parameswaran. 2026. Supporting Our AI Overlords: Redesigning Data Systems to be Agent-First. In *16th Conference on Innovative Data Systems Research, CIDR 2026, Chaminade, CA, USA, January 18-21, 2026*.

[13] Zhengjie Miao and Jin Wang. 2023. Watchog: A Light-weight Contrastive Learning based Framework for Column Annotation. *Proc. ACM Manag. Data* 1, 4 (2023), 272:1–272:24.

[14] Koyena Pal, David Bau, and Renée J. Miller. 2025. Model Lakes. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*. 985–995.

[15] Chen Shen, Jin Wang, Sajjadur Rahman, and Eser Kandogan. 2024. Demonstration of a Multi-agent Framework for Text to SQL Applications with Large Language Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*. ACM, 5280–5283.

[16] Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. 2023. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2986–2997.

[17] Qiming Wang and Raul Castro Fernandez. 2023. Solo: Data Discovery Using Natural Language Questions Via A Self-Supervised Approach. *Proc. ACM Manag. Data* 1, 4 (2023), 262:1–262:27.

[18] Guorui Xiao, Dong He, Jin Wang, and Magdalena Balazinska. 2025. CENTS: A Flexible and Cost-Effective Framework for LLM-Based Table Understanding. *Proc. VLDB Endow.* 18, 11 (2025), 4574–4587.

[19] Guorui Xiao, Enhao Zhang, Nicole Sullivan, Will Hansen, and Magdalena Balazinska. 2026. KathDB: Explainable Multimodal Database Management System with Human-AI Collaboration. In *16th Conference on Innovative Data Systems Research, CIDR 2026, Chaminade, CA, USA, January 18-21, 2026*.