An Agentic Approach to Metadata Reasoning

Anonymous ACL submission

Abstract

As LLM-driven autonomous agents evolve to perform complex, multi-step tasks that require integrating multiple datasets, the problem of discovering relevant data sources becomes a key bottleneck. In addition to the challenge stemming from sheer number of available data sources, data-source selection is difficult because the semantics of data are extremely nuanced and involve considering many aspects of the data. We introduce METADATA REASONER, an agentic approach to metadata reasoning, which is designed to identify a small set of data sources that are both sufficient and minimal for a given analytical task. METADATA REASONER uses a table-search engine to retrieve a set of candidate tables for the task, but then autonomously consults other aspects of the available metadata to decide whether the candidates fit the requirements of the task. We describe a set of empirical studies to demonstrate the effectiveness of METADATA REASONER. We evaluated METADATA REASONER on the real-world KramaBench datasets for data discovery, and show that it achieves an average F1-score of 83.16%, outperforming state-of-the-art baselines by a substantial margin of over 32 percentage points. We also test specific properties of METADATA REASONER on a synthetic dataset based on the BIRD data lake. There we show that METADATA REASONER exhibits robust resistance to the presence of redundant and low-quality tables that may be in the data lake, maintaining a 86.5% F1-score and a 100% success rate in avoiding low-quality data.

1 Introduction

The rapid advancement of LLM-driven autonomous agents has fundamentally transformed data-centric workflows (Fu et al., 2025; Zhu et al., 2025), enabling unprecedented levels of automation. These agents hold the promise to autonomously execute high-level tasks, such as orchestrating complex data engineering pipelines (Jin et al., 2025) or answering business analytics questions. However, a primary bottleneck in developing these agents lies in the effective discovery and selection of data sources that are relevant to a given task (Wei et al., 2025; Kosiuk et al., 2026). This challenge is especially acute when these agents are meant to operate over massive-scale repositories present in modern data lakes.

In addition to the sheer number of available data sources, data selection faces two additional challenges. The first challenge is that deciding whether a data source is relevant to a task is extremely nuanced. The schema and textual description of a table are obviously crucial, but there are a multitude of factors that need to be considered, including the quality and recency of the data, whether it is governed, how it was created (including its code and data lineage), the purpose it was created for and other hidden assumptions that may have gone into its creation. While present day data catalogs can accommodate many of these aspects, they are built for humans. They rely on rich interfaces and implicit context that a human analyst can effortlessly interpret, yet this context remains opaque to an LLM agent.

The second challenge is that executing a task typically requires a combination of data sources rather than a single one. In doing so, it is necessary to guarantee that the collection of data sources together covers all the entities and attributes needed for the task, and that the data sources can be combined (e.g., via joins or unions) to yield the correct result.

One might assume that the expanding context windows of modern LLMs would solve this by allowing agents to ingest vast amounts of metadata and reason over it directly (Chung et al., 2025; Comanici et al., 2025; Yu et al., 2025). However, performance often degrades with context length due to attention dilution across extended contexts, that causes the model to overlook crucial schema constraints (Liu et al., 2024a; Shi et al., 2023; Li et al., 2024). Furthermore, the scale of enterprise data lakes makes reasoning over the entire metadata repository computationally prohibitive.

To manage the sheer scale of available data sources and their associated computational costs, the de facto approach to data discovery today utilizes ranking-based retrieval systems (Brickley et al., 2019; Balaka et al., 2025). The systems may integrate multiple retrieval signals: a lexical full-text search (such as BM25 (Robertson and Zaragoza, 2009)) to handle exact string matches, a semantic vector search (Malkov and Yashunin, 2018) that encodes the descriptions of data sources into an embedding space to capture semantic meaning, and a re-ranking mechanism to refine final ranking. How-

ever, because these methods are inherently designed to score and retrieve datasets individually, they remain inadequate for retrieving combinations or integrations of multiple data sources needed for a task. Consider, queries requiring attributes $\{A_1, A_2\}$ from distinct tables, a ranking-based search may return sources highly relevant to A_1 while omitting any sources relevant to A_2. Even if the search does find tables covering both attributes, these tables may not be connected, either because they contain different sets of entities or because we need an intermediate table to serve as a relational bridge. Furthermore, when a join path exists, determining whether it is semantically valid – for instance, distinguishing between a *customer* table joined to *orders* versus *returns* – requires reasoning about the intent of the relationship, not just the presence of a shared key. Ranking models optimize for topic relevance but are not designed to capture other aspects of metadata reasoning.

This paper introduces the METADATA REASONER, an agentic approach to metadata reasoning, which is designed to identify a small set of data sources that are both sufficient and minimal for a given analytical task. In addition to an analytical task (e.g., a query), METADATA REASONER takes as input a comprehensive metadata catalog that stores all known properties of the available data sources – capturing not just the schemas and text descriptions of tables, but also their creation history, data quality metrics, usage patterns, and broader business definitions such as glossaries or ontologies.

Rather than bringing all the available metadata into the LLM's context window, METADATA REASONER executes a series of steps that involve retrieving candidate tables, reasoning about the metadata attached to them, and consulting specialized tools that provide additional metadata. For example, the specialized tools may determine whether two tables can be meaningfully joined, whether a particular value is present in a given table, or consult a business glossary to better understand the intent of the task. METADATA REASONER autonomously decides which metadata to consult at any given point, depending on how it is progressing towards its goal.

We evaluate the proposed METADATA REASONER in two environments: KramaBench (Lai et al., 2025), a real-world benchmark representing "messy" data lake across six diverse domains, and a synthetically scaled version of the BIRD dataset (Li et al., 2023). We selected KramaBench because it challenges systems with high-cardinality search spaces (approximately 1,500 candidate tables in the *Astronomy* domain) and requires multi-source reasoning. To systematically stress-test the system's resilience against the complexities of modern enterprise data environments, we expanded the BIRD dataset by injecting horizontal partitions, redundant duplicates, and structural noise. We compare our framework against state-of-the-art vector search engines and an LLM-powered hybrid retrieval baseline, Pneuma (Balaka et al., 2025).

Our empirical results demonstrate that the METADATA REASONER achieves superior table selection accuracy, yielding an average F1-score of 83.16% on KramaBench, substantially outperforming both the standard vector search baseline (50.77%) and Pneuma (45.12%). The performance boost is most substantial in the large-scale Astronomy domain, where our method achieves an F1-score of 72.31%, more than doubling the performance of both vector search (32.80%) and Pneuma (27.70%). Furthermore, in the highly noisy synthetic BIRD data lake containing more than 100 similar-looking tables, the METADATA REASONER maintains a robust F1-score of 92.4%, compared to just 37.2% for the vector search baseline. The system achieves perfect discrimination against noise-injected tables, whereas the baseline retrieves invalid or low-quality tables 34.2% of the time. Crucially, this high-precision data selection directly translates to downstream success: in an end-to-end Text-to-SQL evaluation, using the tables recommended by the METADATA REASONER improved SQL execution accuracy to 64.15%, up from the 54.72% achieved by top-10 vector retrieval.

2 Related Works

Dataset search and Data Integration. Dataset search aims to fulfill information needs across vast collections like data lakes (Chapman et al., 2020; Hulsebos et al., 2024; Leventidis et al., 2024; Fan et al., 2023a; Fernandez et al., 2018). While early keyword-based methods often struggle with semantic ambiguity (Paton et al., 2023; Brickley et al., 2019), later approaches identify joinable and unionable tables (Koutras et al., 2021; Nargesian et al., 2018) utilizing approximate containment search (Bogatu et al., 2020; Zhu et al., 2019), knowledge graphs (Koutras et al., 2025) or language models (Fan et al., 2023b). Recently, systems like Pneuma integrated LLMs for hybrid search (Balaka et al., 2025). Concurrently, classical data integration systems (Doan et al., 2012) rely on formal descriptions of data sources and query translation is performed by a specialized algorithm (e.g., answering queries using views (Halevy, 2001)). Formal descriptions cannot capture nuances that arise in real data lakes and which are best described in natural language. The METADATA REASONER, on the other hand, accepts nuanced descriptions of data sources and reasons about them appropriately.

Retrieval within agentic systems. Recent advancements in agentic information retrieval (Singh et al., 2025) and reasoning-intensive retrieval (Zhang et al., 2024; Su et al., 2024) utilize LLMs to navigate dynamic information states via multi-step reasoning, query expansion, specialized dense models, and advanced reranking techniques (Long et al., 2025; Shao et al., 2025; Das et al., 2025). However, these systems typically process unstructured text. Meanwhile, METADATA REASONER operates on structured and semi-structured data, such as metadata catalogs and database schemas, and selects a combination of data sources rather than ranking individual items

Improving LLMs for tool usage. As LLMs have

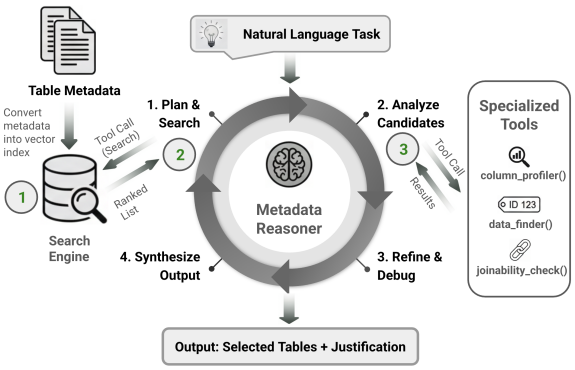

Figure 1: The architecture of the METADATA REASONER. Given an analytic task and a metadata catalog, the agent autonomously executes a series of planning, search, and reasoning steps. It leverages both attached metadata from retrieved candidate tables and on-the-fly metadata accessed via specialized tools to output a sufficient and minimal set of tables, along with a natural-language justification.

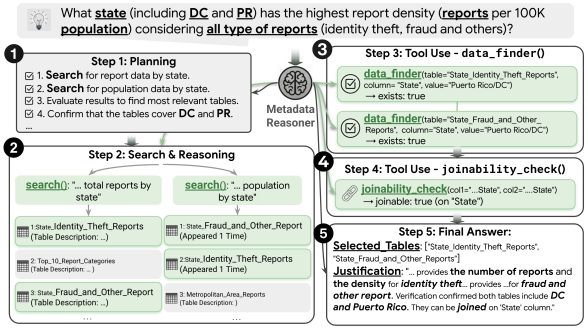

Figure 2: Illustrative Example of the METADATA REASONER Workflow. Given an analytical task, the METADATA REASONER first decomposes the query into multi-faceted search plans (e.g., reports vs. population) to retrieve candidate datasets. It then uses specialized tools to confirm entity existence (e.g., "Puerto Rico/DC") and to validate relational paths. The final output provides a justified selection of tables, ensuring they are both sufficient for the task and can be joined.

evolved to interact with external tools (Schick et al., 2023), a major bottleneck has been context window saturation caused by providing exhaustive API documentation. Recent studies address this by abstracting verbose documentation into concise instructions (Yuan et al., 2025) or employing structured thought-action-observation loops (Zhuang et al., 2024; Liu et al., 2024b). Inspired by these advances, METADATA REASONER translates high-level tasks into coordinated retrieval steps across diverse metadata catalogs.

3 The Architecture of the METADATA REASONER

We first formally define the goal of the METADATA REASONER and then describe its architecture and underlying design rationale.

3.1 Problem formulation

The METADATA REASONER accepts two **inputs**:

A data task Q: a typical example of a task is a question that can be answered by an SQL query, or a machine learning prediction or cause analysis task that requires Python code.

A metadata catalog $\mathcal{T}$: The catalog encapsulates all known properties of the available data sources within the data lake. While we use the term *tables* for simplicity throughout this paper, these resources can also encompass collections of structured and semi-structured datasets, such as those found in Web tables and open data portals. The catalog captures both structural and semantic information, including schemas and textual descriptions of tables and their attributes. It also includes information surrounding the tables, such as their lineage, quality, and usage patterns. The catalog also contains semantic information that is not attached to a specific table. For example, it may contain a glossary that explains business terms in the domain of the enterprise, an ontology that describes classes of entities and relationships between them, or a semantic model of the data.

The METADATA REASONER produces two **outputs**: The first is **a subset of tables** $T^* \subseteq \mathcal{T}$. T^* must satisfy the following conditions: (1) **sufficiency** (or coverage): T^* should contain tables that are required to solve the input task (e.g., tables for answering a query). Specifically, this means that the tables in T^* should cover all the entities and attributes relevant to Q, while ensuring that the tables can be joined correctly when needed, and (2) **minimality**: T^* should be minimal, in the sense that no smaller strict subset of T^* is sufficient for Q. This ensures that every individual table included in T^* is absolutely **necessary** to solve the task.

The METADATA REASONER also outputs **a natural-language justification** J, which explicitly provides evidence supporting the sufficiency and optimality of T^*. This contextual grounding ensures that subsequent agents in the pipeline understand exactly why and how the retrieved tables should be used.

Example 3.1. Figure 2 illustrates the METADATA REASONER workflow. The agent begins by analyzing the given query to understand which entities (e.g., *DC* and *PR*) and attributes (e.g., *total reports* and *population*) are necessary and what other constraints (e.g., *all type of reports*) to look out for. Based on this analysis, the agent creates an initial plan for its reasoning. In the example, the first two steps of the plan include involving the search engine to retrieve candidate datasets for *state reports* and *population*. It then utilizes `data_finder()` to confirm the presence of *DC* and *PR* and `joinability_check()` to validate that disparate report categories can be unified via shared `State` columns. If the candidates are insufficient or lack a valid join path, the METADATA REASONER recursively

reformulates its search strategy to expand the set of candidate tables. Upon satisfying all extracted constraints, the agent terminates and outputs a set of selected tables and a natural language justification for their relevance. In this instance, the reasoner clarifies how the `State_Identity_Theft_Reports` and `State_Fraud_and_Other_Reports` tables satisfy the "all types" requirement, ensuring both sufficiency (coverage of all attributes) and minimality (no smaller subset satisfies the constraints).

3.2 An agentic approach to metadata reasoning

The preceding example illustrates that the agent leverages metadata across three distinct operational stages. First, during retrieval, table metadata is encoded as vectors describing the objects available to the search engine. Second, during evaluation, the agent reasons over the metadata explicitly attached to the retrieved candidate tables. Third, specialized tools provide the agent with on-demand access to highly specific metadata signals that would be impractical to provide ahead of time.

Recognizing that different types of metadata are optimal for different stages of the reasoning pipeline is the core principle driving the METADATA REASONER's design. The primary challenge is context saturation: it is computationally impractical and detrimental to reasoning performance if we supply the agent with all available table metadata simultaneously.

To overcome this, we architect the METADATA REASONER around an orchestration agent that autonomously fetches the exact fragments of metadata required at any given step. In the following sections, we detail the reasoning pipeline, highlighting the specific metadata utilized at each phase.

3.3 Query decomposition and planning

Upon receiving a query Q, the agent is instructed to decompose it into a set of search constraints, including named entities, measures, temporal scopes, and granularity requirements. This enables constructing a plan for searching the relevant tables. The agent then executes this plan by alternating between invoking the search tool, reasoning about the retrieved tables, and consulting the specialized tools for specific information.

3.4 The semantic search tool

The agent retrieves candidate tables from the catalog through a semantic-search tool. The tool has access to a set of vectors, each representing the embedding of metadata about a table. The search is based on approximate nearest neighbor with the embedding of the query.

The challenge that arises here is choosing which metadata to use in creating the embeddings of the table descriptions. A common failure mode in large-scale data lakes is semantic homogenization – when tables share high-level metadata, their embeddings become indistinguishable, degrading retrieval precision, leading to poor search results. For example, this may happen if many tables have similar schemas and nuanced differences between them occupy only a small part of their description.

3.4.1 Discrimination-Oriented Metadata Construction

To have more targeted search, we designed the embeddings to emphasize the differences between tables. Specifically, in deciding which metadata to embed, we prioritize descriptions of unique aspects, such as specific temporal or geographical scopes of a table. In addition, we ensure that technical identifiers and cryptic column names are mapped to standardized domain terminology and synonymous business logic. This helps align natural-language queries with the indexed vector embeddings of the metadata. We construct the embeddings with a two-stage LLM-driven process.

In the first stage, we group tables based on semantic relatedness. In general, grouping can be done based on the table descriptions. In some cases, such as for KramaBench, the tables are organized into a directory structure that provide such a grouping. We task the LLM with identifying shared and unique variables across a set of tables. The model then generates a prompt template designed to describe individual tables by highlighting their distinguishing features relative to the group. In the second stage, we apply the prompt template to each of the tables in the group to create the differentiated table description. We provide an example of the generated description in Appendix A.1.

3.4.2 Handling duplicate search results

When the agent performs multiple calls to the semantic-search tool, some tables may be returned for multiple search queries. If we included all the metadata for these tables again, this would pollute the agent context, leading the agent overlook distinct tables surfaced later in the retrieval results.

To address this challenge, we implemented a state-aware deduplication mechanism within the search tool. The mechanism maintains a session-level dictionary, $\mathcal{S}$, which tracks the identifiers (table ID) and retrieval occurrences of all previously surfaced tables. For each retrieved table ID, if $ID \notin \mathcal{S}$, then the table's metadata is added to the context window. However, if $ID \in \mathcal{S}$, the metadata is suppressed and replaced by a concise recurrence indicator (e.g., "Table ID: xxx (Appeared N times)"), and the occurrence counter in $\mathcal{S}$ is incremented. These are then concatenated into a single return string. This strategy prevents the agent from becoming stuck in high-recall redundancy, forcing it to attend to previously overlooked data sources. In scenarios where a search cycle yields only duplicates, the system returns a termination signal, prompting the agent to autonomously revise its search strategy.

3.5 Reasoning about retrieved tables

Once the search engine retrieves a set of candidate tables, the agent proceeds to reason about their efficacy

for the task Q. At this point, we face a choice of which metadata about a table T to feed directly into the agent when T is retrieved, versus which metadata to make available only on demand. We refer to these as *attached metadata* and *on-the-fly metadata*, respectively.

Inevitably, there will be some metadata that we cannot include in the attached metadata. First, some metadata is not associated with a single table. For example, a business glossary or an ontology of the domain apply to all the data in the data lake. Second, some metadata is simply too large to add to the LLM's context or even impractical to compute. For example, when the METADATA REASONER is considering a set of tables, it may need to decide whether the tables can be joined via some path. Pre-calculating all the possible join paths in a data lake is often impractical. Meanwhile, if the data values are hard to summarize, listing them all would saturate the context window and degrade reasoning performance.

Hence, we designed the METADATA REASONER to have access to a set of specialized tools that provide access to the on-the-fly metadata. We describe the capabilities of these tools to the agent, and we instruct the agent to use these tools if it cannot infer the required information from the attached metadata.

The optimal partition between attached and on-the-fly metadata is dependent on the context in which we are applying METADATA REASONER. This allocation hinges on several variables: metadata availability, fragment size, latency constraints, and the agent's capacity to orchestrate multiple tools. Ultimately, this partitioning strategy is a critical determinant of overall system performance.

3.5.1 Attached metadata

The attached metadata needs to enable the agent to decide whether the table satisfies the requirements of the task Q, which includes information about its schema, temporal and spatial scope, the granularity of the data, and the other tables it refers to. In particular, we include:
Schema Information: The physical schema of each candidate table, including the table name, column names, data types, constraints and any existing human-authored documentation.
Semantic Enrichment via Statistical Synthesis: We perform statistical profiling for each table. A data profiler calculates descriptive metrics such as value ranges (min/max), categorical cardinality (top-K values), and null-count ratios. These statistics are then processed by an LLM to generate natural language descriptions to save token cost. Note that all of this enrichment is done ahead of time on the catalog $\mathcal{T}$. An example is in Appendix A.2.

3.5.2 On-the-fly metadata

We provide the following tools:

- `column_profiler()`: Granular column profiling is invoked when summarized descriptions lack the depth necessary for final data selection. It retrieves high-resolution signals such as exact distinct value counts, mean/median distributions, and frequency histograms.

- `data_finder()`: Value-level grounding is strictly a query-time requirement, as the specific entities of interest (e.g., unique user IDs or country codes) are only known once the query is received and the candidate tables are established. This tool provides deterministic evidence of data presence.

- `joinability_check()`: Just-in-time joinability analysis is essential because providing all possible pairwise joinability information to the LLM in advance is computationally wasteful. Instead, the system retrieves signals regarding foreign key overlap and referential alignment only for the specific candidate tables identified during the reasoning loop.

4 Experimental Setup

This section details the datasets we used, how we enhanced them, the baselines, and our evaluation metrics.

4.1 Datasets

We evaluate the METADATA REASONER using two primary benchmarks to capture different facets of enterprise data challenges.

4.1.1 KramaBench

For real-world assessment, we utilize KramaBench (Lai et al., 2025), a benchmark specifically designed to simulate data lake environments across six diverse domains: astronomy, legal, environment, biomedical, wildfire, and archeology. Although KramaBench was originally intended to evaluate end-to-end data-to-insight workflows, we adapt its analytic tasks, data sources, and the reference data sources to specifically test the data discovery phase. This benchmark presents significant challenges to autonomous agents, demanding cross-table analytical reasoning and navigation of high-cardinality search spaces. Most notably, in the astronomy domain, it contains approximately 1,500 tables.

The original data sources are organized as directories of heterogeneous files (e.g., raw CSVs, spreadsheets, or specialized scientific formats). In order to simulate an enterprise data warehouse environment, first we transform them into a collection of tables. To achieve this, we recursively traverse the data directories, applying specialized parsers tailored to file types (e.g., parsing fixed-width `.tle` orbital data or inferring headers from `.omni2` formats) to extract both the tabular data and associated descriptive text. The extracted tables are then standardized into a schema-aware CSV format, with column names validated for database compatibility. Crucially, unstructured descriptions, whether embedded in file headers or located in separate `.txt` metadata files, are consolidated into user provided table description field. Table 1 summarizes the dataset characteristics, illustrating the scale and complexity of each dataset.

Table 1: KramaBench Dataset Statistics. "Avg. #col." stands for the average number of columns per table in a domain. "Avg. #tables" stands for the average number of ground truth tables per task.

Domain	#Table	Avg. #col.	#Task	Avg. #table
Astronomy	1498	28.6	12	17.2
Legal	134	2.5	30	16.8
Environment	36	10.3	20	7.9
Biomedical	29	35.4	9	2.0
Wildfire	21	12.1	21	1.7
Archeology	5	16.2	12	1.5

Table 2: Synthetic BIRD Financial data lake Statistics.

	Base	Splits	Duplicates	Low Quality	#Tables
Base	8	0	0	0	8
Synth	8	30	76	38	152

4.1.2 Synthetically-Scaled BIRD

To systematically stress-test the system's resilience against the complexities of real-world enterprise environments, we synthetically created a data lake with many tables that are related to each other and with different table qualities. We started with the BIRD dataset (Li et al., 2023) that was originally designed for Text-to-SQL tasks and provides natural language questions paired with clean database schemas and ground-truth SQL queries. As we describe in Section 4.2, we created two distinct versions of a financial database: a clean version containing only the original base tables, and a messy version that included redundant views, tables with inconsistent data quality, and tables that were created from the original ones by vertical and horizontal fragmentation. Table 2 summarizes the statistical characteristics and scale of both. We utilized 106 original analytical questions associated with BIRD's `financial` database to evaluate the data selection performance.

4.2 Synthetic data lake creation

To simulate the redundancy, fragmentation, and quality degradation typical of enterprise data lakes, we developed a data creation pipeline that applied transformation on the BIRD dataset to create a massive repository of tables. This expansion was achieved through three primary noise-injection operations: generating horizontally partitioned sub-tables, creating redundant variants, and intentionally degrading data integrity by introducing broken foreign keys, duplicates, and missing entities. The details are in Appendix B.1. To fully track the data creation pipeline, we recorded detailed lineage information for all derived tables. We incorporated this lineage information into the metadata of each table, making it available to METADATA REASONER and all other testing methods.

4.3 Baselines

We compare the Full METADATA REASONER (MR) against an enterprise vector-search engine and three ablation variants:

- **Vector Search (Baseline)**: This baseline serves as a non-agentic control. First, it utilizes vector embeddings to retrieve semantically close candidate metadata chunks, applying a 0.7 cosine distance threshold to immediately discard irrelevant results. Finally, a semantic ranker re-orders the candidates to prioritize contextual accuracy. We use this vector-search engine as the semantic search tool for our METADATA REASONER because of its high performance.

- **Pneuma** (Balaka et al., 2025): Pneuma utilizes hybrid full-text and vector search over LLM-generated metadata, followed by an LLM-based judge to re-rank the top-K tables. We implemented it using Gemini-3-Flash, and feeding the same user-provided documentations as for METADATA REASONER.

We use `text-embedding-005` for metadata encoding in all methods. All METADATA REASONER variants utilize Gemini-3-Flash as the backbone model and are implemented using the ADK framework [1].

- **MR-Search**: This utilizes the discrimination-oriented metadata for search engine within the agentic workflow. The agent may perform multiple search iterations, but only consumes the responses from the search engine.

- **MR-Search+Tools**: This leaves out the attached metadata for the retrieved tables in the LLM prompt. It tests the system's ability to dynamically use on-the-fly metadata for reasoning.

- **MR-Search+Attached**: This incorporates the attached table content summarization. But it does not use the tools to fetch on-the-fly metadata.

- **Full MR (MR-Search+Attached+Tools)**: The complete architecture that autonomously fetches the exact fragments of metadata.

4.4 Evaluation methods and metrics

Evaluation with ground truth table sets. Standard rank-based metrics (e.g., Precision@K) are ill-suited for agentic systems that output variable-length sets. We therefore measure selection accuracy using a set-aware ***F1-Score***, a harmonic mean of precision and recall that captures the two core objectives of our task goal. Specifically, we utilize Recall to evaluate the sufficiency (or coverage) of the selected data sources, and Precision to evaluate their minimality (or conciseness). To ensure a rigorous comparison with ranking-based baselines like vector search, we evaluate their performance across various retrieval thresholds K and report the best achieved F1-score. We also track the average number of steps, i.e., ***Average #step***, per task to measure the computational cost and efficiency of the reasoning process.

[1] ADK:https://google.github.io/adk-docs/

Table 3: Average recall, precision, F1, and the number of steps over six KramaBench domains across different methods. "Rec." stands for "Recall" and "Prec." stands for "Precision".

Method	Rec. (%)	Prec. (%)	F1 (%)	#Steps
Vector Search (Tool)	-	-	50.77	-
Pneuma	-	-	45.12	-
MR-Search	75.15$_{\pm 2.91}$	79.08$_{\pm 2.63}$	73.43$_{\pm 2.59}$	13.51$_{\pm 1.21}$
MR-Search+Attached	80.83$_{\pm 3.00}$	84.13$_{\pm 3.73}$	79.66$_{\pm 2.80}$	**8.36**$_{\pm 0.83}$
MR-Search+Tools	79.96$_{\pm 4.11}$	79.72$_{\pm 3.12}$	76.65$_{\pm 3.43}$	24.14$_{\pm 2.09}$
Full MR	**84.44**$_{\pm 2.29}$	**86.63**$_{\pm 2.65}$	**83.16**$_{\pm 2.49}$	10.10$_{\pm 1.07}$

Note: Bold values indicate the best performance across all the methods.

Table 4: F1 score (%) comparison across six domains. We report the mean scores over five different runs and the corresponding standard deviations. The F1 scores for the vector search baseline represent the best F1 scores achieved across various choices of top K results.

Domain	Vector Search	Pneuma	MR-Search	MR-Search+ Attached	MR-Search+ Tool	Full MR
Astronomy	32.80	27.70	64.38$_{\pm 2.96}$	67.68$_{\pm 1.94}$	70.34$_{\pm 2.78}$	**72.31**$_{\pm 3.72}$
Legal	38.45	24.94	63.09$_{\pm 2.51}$	61.27$_{\pm 2.75}$	60.80$_{\pm 3.50}$	**64.40**$_{\pm 1.53}$
Environment	45.56	33.93	79.73$_{\pm 0.76}$	85.49$_{\pm 1.22}$	81.77$_{\pm 3.24}$	**87.95**$_{\pm 3.08}$
Biomedical	50.37	58.15	74.88$_{\pm 5.04}$	77.63$_{\pm 6.31}$	82.70$_{\pm 6.36}$	**87.98**$_{\pm 3.35}$
Wildfire	58.25	42.70	65.93$_{\pm 1.05}$	89.80$_{\pm 3.20}$	69.84$_{\pm 2.41}$	**90.21**$_{\pm 1.91}$
Archeology	79.17	83.33	92.59$_{\pm 3.20}$	**96.11**$_{\pm 1.36}$	94.44$_{\pm 2.27}$	**96.11**$_{\pm 1.36}$
Average F1	50.77	45.12	73.43$_{\pm 2.59}$	79.66$_{\pm 2.80}$	76.65$_{\pm 3.43}$	**83.16**$_{\pm 2.49}$

Evaluation without ground truth table sets. To evaluate table selection on the synthetic dataset without manual re-labeling, we assess the returned tables by tracking their lineage back to the required base tables parsed from the original Gold SQL queries. As detailed in the appendix B.2.2, a selected table is strictly scored as correct only if it maps to a required base table, completely excludes any injected noise or quality degradation, and perfectly matches necessary partition filter constraints.

5 Experimental Evaluation

In this section, we evaluate the METADATA REASONER across both the real-world KramaBench environment and the synthetically scaled BIRD dataset. We organize our evaluation around five core Research Questions (RQ) to assess the system's end-to-end accuracy, its impact on downstream analytic tasks, its robustness against enterprise data messiness, the effectiveness of combining attached and on-the-fly metadata, and the optimal metadata descriptions for semantic search (detailed in Appendix C.2).

5.1 RQ1: End-to-end data selection performance

We first evaluate whether the METADATA REASONER can effectively identify the accurate combination of data sources.

5.1.1 Results on KramaBench

Table 3 shows that across the six domains of KramaBench, the Full MR reaches an average F1-score of 83.16%, substantially outperforming the vector search baseline, which achieves the best F1 score of 50.77% and the LLM-powered hybrid retrieval system, Pneuma (45.12%). By reasoning over the attached metadata to narrow the search space before invoking tools, the METADATA REASONER achieves a peak F1-score with a manageable step count of 10.10.

Based on the domain-specific F1 scores in Table 4, the performance gap is most significant in the Astronomy domain (largest-scale with 1,498 tables). The Full MR achieves an F1-score of 72.31%, more than doubling the performance of vector search (32.8%) and Pneuma (27.70%). METADATA REASONER maintains robust accuracy across all domains, achieving its highest relative gain in the Wildfire domain comparing different METADATA REASONER variants, where the addition of content-driven attached metadata and on-the-fly metadata boosts the F1-score from 65.93% to 90.21%.

Detailed failure mode analysis and case studies (provided in Appendix C.1) reveal that METADATA REASONER successfully returned the correct set of tables, even when they were ranked as low as 5th and 11th by vector search.

Table 5: Data selection accuracy of METADATA REASONER evaluated on the clean Base (8 tables) and the messy Synthetic (152 tables) BIRD financial data lakes across different query types. The category "n Tables" indicates queries Q span exactly n tables. For comparison, the baseline accuracy of the Top-10 retrieved tables from vector search is also reported.

$\mathcal{Q}$ Type	#Q	Recall (%)		Precision (%)		F1 Score (%)	
		Base (8)	Synth (152)	Base (8)	Synth (152)	Base (8)	Synth (152)
1 Table	10	100.	100.	95.0	90.0	96.7	93.3
2 Tables	49	99.9	96.9	90.1	88.1	93.1	91.2
3 Tables	29	85.1	80.5	92.2	88.8	87.8	83.5
4 Tables	17	75.0	83.8	100.	88.7	85.4	75.3
>4 Tables	1	80.0	60.0	100.	75.0	88.8	66.6
Overall	106	91.2	87.4	92.8	88.4	90.7	86.5
Top10 Retrieved		-	58.1	-	29.7	-	37.2

5.1.2 Results on the synthetic dataset

In Table 5, we measured the system's end-to-end data selection accuracy on the synthetically scaled BIRD data lake. The METADATA REASONER demonstrates strong and robust performance in data selection across both clean and messy data environments, outperforming the standard vector search baseline. On the clean Base dataset with 8 tables, the method achieves an impressive overall F1 score of 90.7%, which holds up remarkably well at 86.5% even when subjected to the redundant variants and horizontal partitions of the Synthetic dataset (152 tables). In contrast, the baseline vector search retrieval (Top-10 tables) on the Synthetic dataset yields an F1 score of only 37.2%, suffering from both low precision and moderate recall.

Analysis by query complexity reveals a degradation in performance as queries span a larger number of ta-

Table 6: Distribution of selected table types from METADATA REASONER (MR) and top K search results in the synthetic messy data lake (%). The data catalog $\mathcal{T}$ contains 152 tables. METADATA REASONER successfully selected all the high-quality tables.

Table Types	$\mathcal{T}$	MR	Vector Search Top1	Top2	Top5	Top10
base	5.3	36.7	39.6	25.0	17.0	13.4
split	19.7	0.3	28.3	24.5	21.3	22.1
_prod (base)	5.3	43.7	2.8	13.2	10.4	8.8
_prod (split)	19.7	19.3	14.2	17.9	22.1	21.5
Ratio of Good Tables ↑		**100**	84.9	80.6	70.8	65.8
_test	12.5	0.0	4.7	6.2	10.2	10.7
_stg	12.5	0.0	0.9	1.9	3.6	6.6
_subset	7.2	0.0	7.6	9.9	11.1	10.3
_dups	7.2	0.0	1.9	1.4	1.5	2.6
_broken_fk	6.6	0.0	0.0	0.0	1.5	3.1
_nulls	4.0	0.0	0.0	0.0	1.3	0.9
Ratio of Bad Tables ↓		**0**	15.1	19.4	29.2	34.2

bles, particularly within the messy Synthetic environment. For simpler queries requiring 1 to 2 tables, the method maintains high accuracy with F1 scores exceeding 91% on both datasets. However, as the number of join operations increase, performance drops; for queries spanning 4 or more tables, the F1 score falls to 75.3% and 66.6% on the Synthetic dataset, respectively. This downward trend is more pronounced on Synthetic than on Base (which maintains 85.4% and 88.8% F1 for those categories).

5.2 RQ2: Performance boost on downstream analytical tasks

To assess the impact of table selection on downstream analytical accuracy, we conducted an Text-to-SQL experiment within the synthetic BIRD Financial data lake environment. We compared two distinct input tables for a simple-prompt SQL generator using Gemini-3-Pro: (1) the top-10 ranked tables from the vector search, and (2) the set of recommended datasets selected by the METADATA REASONER. Both configurations provided identical metadata signals to the SQL generator, including the same schemas, metadata, the natural language question, and supporting evidence. For the METADATA REASONER configuration, we further augmented the prompt with the output *justification* logic.

Our results indicate that using the tables produced from METADATA REASONER achieved an overall execution accuracy of 64.15% (68/106), outperforming the results with the top-10 retrieval set, which yielded an accuracy of 54.72% (58/106). The primary failure of the top-10 retrieved tables baseline is that it misses essential entities or attributes. This occurs because the over-retrieval of similar-looking duplicates and low-quality variants for a single target dimension crowds out other required data.

5.3 RQ3: Robustness against data messiness

To understand why traditional retrieval engines fail in enterprise environments, we analyzed the specific types of tables selected by the systems in our noisy synthetic data lake. The data lake was evenly split between high-quality tables (i.e., base tables, valid partitions, and clean production copies) and noise injected tables (i.e., the staging _stg, testing _test, duplicated _dups, and noise injected _broken_fk, _nulls, _subset tables).

In Table 6, we calculate the table type distribution over different table types. The numbers in the $\mathcal{T}$ column represent the proportion of each table type out of the absolute total catalog count (152 tables). The numbers in method columns (MR and Vector Search Top Ks) are average ratios calculated per query and then averaged over the full 106 test queries. We note that our METADATA REASONER achieves perfect discrimination, maintaining a 100% ratio of good tables and preserving a 0% selection rate for any bad tables. In contrast, vector search exhibits high vulnerability to the injected noises. While vector search achieves competitive scores at top-1 retrieval, its selection of bad tables grows sequentially with the retrieval budget, rising to 34.2% failure rate at $K = 10$.

5.4 RQ4: Ablation of the attached and on-the-fly metadata

We conducted an ablation study to evaluate the effectiveness of the attached and on-the-fly metadata.

The performance boost from the attached metadata: The introduction of attached table content summarization metadata provides the single most significant boost to both accuracy and efficiency. In Table 3, we can see that adding the attached metadata to the initial agent prompt (*MR-Search* → *MR-Search+Attached*) improves the F1-score from 73.43% to 79.66% while reducing the average execution steps from 13.51 to 8.36.

The on-the-fly metadata trade-off: The on-the-fly metadata (via the use of tools like `data_finder` and `joinability_check`) provides the highest overall accuracy. However, as shown in Table 4, the *MR-Search+Tool* variant, without the warm start provided by attached metadata, the agent defaults to exhaustive trial-and-error, with steps nearly tripling to 24.14 (see Table 3) as it attempts to get more metadata signals without sufficient prior context.

6 Conclusion and Further Work

Moving beyond simple retrieval, METADATA REASONER introduces agentic selection—a framework that automates the discovery of joinable, structurally sound data within complex enterprise ecosystems. Future work includes improving performance on more complex queries and tasks, adapting the system to encompass unstructured data, and enhancing the system with web search tools to complement private catalogs with public data from the open web.

7 Limitations

While the METADATA REASONER framework significantly improves agentic data selection, it possesses a few limitations. First, the current system lacks upfront catalog-specific knowledge, such as business glossaries or domain ontologies, which restricts it to catalog-agnostic planning and may limit its initial interpretation of highly specialized user tasks. Second, the framework currently focuses on structured and semi-structured tabular data, meaning it does not yet seamlessly integrate unstructured enterprise data or supplement private catalogs with public datasets from the open web. Integrating overarching semantic models and extending the reasoner's capabilities to unstructured domains represent critical areas for future work.

8 Ethical considerations

The deployment of autonomous LLM-driven agents for enterprise data discovery introduces several ethical and security implications that must be managed responsibly prior to real-world application.

Data Privacy: The agent autonomously profiles data using tools and searches for specific values. Without strict, user-level access controls, it risks exposing Personally Identifiable Information (PII) or other sensitive enterprise data during its search.

Bias and Hallucination: The system relies on an LLM to summarize data. If the LLM hallucinates or introduces bias into these summaries, it could systematically skew the agent's data selection and compromise downstream analytics.

Accountability: Although the system provides a natural-language justification for its table selections, safe real-world deployment requires strict auditing and logging of the agent's exact search trajectories and tool invocations to maintain transparency.

References

Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging llms for tabular data representation and retrieval in an end-to-end system. *Proceedings of the ACM on Management of Data*, 3(3):1–28.

Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *2020 ieee 36th international conference on data engineering (icde)*, pages 709–720. IEEE.

Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *The world wide web conference*, pages 1365–1375.

Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272.

Yeounoh Chung, Gaurav T Kakkar, Yu Gan, Brenton Milne, and Fatma Ozcan. 2025. Is long context all you need? leveraging llm's extended context for nl2sql. *arXiv preprint arXiv:2501.12372*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Debrup Das, Sam O' Nuallain, and Razieh Rahimi. 2025. Rader: Reasoning-aware dense retrieval models. *arXiv preprint arXiv:2505.18405*.

AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration*. Morgan Kaufmann.

Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023a. Table discovery in data lakes: State-of-the-art and future directions. In *Companion of the 2023 International Conference on Management of Data*, SIGMOD '23, page 69–75, New York, NY, USA. Association for Computing Machinery.

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

Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE.

Yanjie Fu, Dongjie Wang, Wangyang Ying, Xinyuan Wang, Xiangliang Zhang, Huan Liu, and Jian Pei. 2025. Autonomous data agents: A new opportunity for smart data. *arXiv preprint arXiv:2509.18710*.

Alon Y. Halevy. 2001. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294.

Madelon Hulsebos, Wenjing Lin, Shreya Shankar, and Aditya Parameswaran. 2024. It took longer than i was expecting: Why is dataset search still so hard? In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics*, pages 1–4.

Tengjun Jin, Yuxuan Zhu, and Daniel Kang. 2025. Elt-bench: An end-to-end benchmark for evaluating ai agents on elt pipelines. *arXiv preprint arXiv:2504.04808*.

Wojciech Kosiuk, Xingyu Ji, Yeounoh Chung, Fatma Özcan, and Madelon Hulsebos. 2026. Fine-grained table retrieval through the lens of complex queries. *arXiv preprint arXiv:2603.07146*.

Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating matching techniques for dataset discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 468–479. IEEE.

Christos Koutras, Jiani Zhang, Xiao Qin, Chuan Lei, Vasileios Ioannidis, Christos Faloutsos, George Karypis, and Asterios Katsifodimos. 2025. Omnimatch: Joinability discovery in data products. *Proceedings of the VLDB Endowment*, 18(11):4588–4601.

Eugenie Lai, Gerardo Vitagliano, Ziyu Zhang, Sivaprasad Sudhir, Om Chabra, Anna Zeng, Anton A Zabreyko, Chenning Li, Ferdi Kossmann, Jialin Ding, and 1 others. 2025. Kramabench: A benchmark for ai systems on data-to-insight pipelines over data lakes. *arXiv preprint arXiv:2506.06541*.

Aristotelis Leventidis, Martin Pekár Christensen, Matteo Lissandrini, Laura Di Rocco, Katja Hose, and Renée J Miller. 2024. A large scale test corpus for semantic table search. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1142–1151.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357.

Zhuoming Li, Yichen Gong, Yelong Shen, and Xing Xie Zhang. 2024. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. *arXiv preprint arXiv:2402.11193*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. In *Transactions of the Association for Computational Linguistics*, volume 12, pages 157–173.

Zhaoyang Liu, Zezheng Lai, Gaojie Zhang, Renjie Zhang, Keqing Chen, Xiao Wang, Yujie Zhu, Shaogang Cao, Jiacheng Chen, Yixiao Ge, and 1 others. 2024b. Controllm: Augmenting language models with tools by planning, customization, and interaction. In *Proceedings of the 2024 International Conference on Machine Learning (ICML)*.

Meixiu Long, Duolin Sun, Dan Yang, Junjie Wang, Yue Shen, Jian Wang, Peng Wei, Jinjie Gu, and Jiahai Wang. 2025. Diver: A multi-stage approach for reasoning-intensive information retrieval. *arXiv preprint arXiv:2508.07995*.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825.

Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. 2023. Dataset discovery and exploration: A survey. *ACM Comput. Surv.*, 56(4).

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and 1 others. 2025. Reasonir: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models are easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*. PMLR.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, and 1 others. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *arXiv preprint arXiv:2407.12883*.

Zixin Wei, Yucan Guo, Jinyang Li, Xiaolin Han, Xiaolong Jin, and Chenhao Ma. 2025. Revisiting task-oriented dataset search in the era of large language models: Challenges, benchmark, and solution. *arXiv preprint arXiv:2512.15363*.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. 2025. Easytool: Enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972.

Weinan Zhang, Junwei Liao, Ning Li, Kounianhua Du, and Jianghao Lin. 2024. Agentic information retrieval. *arXiv preprint arXiv:2410.09713*.

Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*, pages 847–864.

Yizhang Zhu, Liangwei Wang, Chenyu Yang, Xiaotian Lin, Boyan Li, Wei Zhou, Xinyu Liu, Zhangyang Peng, Tianqi Luo, Yu Li, and 1 others. 2025. A survey of data agents: Emerging paradigm or overstated hype? *arXiv preprint arXiv:2510.23587*.

Yuan Zhuang, Yifei Li, Ling Chen, and Wei Wang. 2024. Toolnet: Connecting llms with massive tools via graph-based propagation. *arXiv preprint arXiv:2403.00839*.

A Metadata Examples

A.1 An example of the generated discrimination-oriented description

To illustrate, consider the discrimination-oriented descriptions generated via this two-stage process for tables `0309geomag_forecast` and `0310geomag_forecast`. While these tables share an identical schema, they differ in temporal ranges. Consequently, our process creates a prompt template that maintains necessary semantic context (e.g., *NOAA geomagnetic Kp index forecast*), but prioritizes highlighting their specific temporal boundaries.

```
Table: 0309geomag_forecast
"NOAA geomagnetic Kp index forecast specifically
for March 11, March 12, and March 13. The data
provides floating-point forecast values at 3-hour
intervals in Universal Time (UT)."

Table: 0310geomag_forecast
"NOAA geomagnetic Kp index forecast specifically
for March 10, March 11, and March 12..."
```

A.2 An example for the attached metadata

The metadata associated with the table `0309geomag_forecast` is shown below.

```
Table: 0309geomag_forecast
"""Table Description:
NOAA Kp index forecast 10 Mar - 12 Mar
This table provides a geomagnetic forecast from
March 10th to March 12th...Each row corresponds
to a specific time interval. The columns
represent the forecasted geomagnetic activity
for each day.

Schema:
- Time_UT: STRING (NULLABLE) - "...time ranges,
e.g., '00-03UT'."
- Mar_10: FLOAT (NULLABLE) - "Forecasted geomag-
netic activity value for March 10th...ranging
from 3.0 to 5.0, and an average of 3.75..."
- Mar_11: FLOAT (NULLABLE) - "...for March
11th...from 3.33 and 3.67."
- Mar_12: FLOAT (NULLABLE) - "...for March
12th...from 2.0 to 3.67."
```

B Detailed Experimental Setup

B.1 Synthetic data lake creation

To simulate the redundancy, fragmentation, and quality degradation typical of enterprise data lakes, we developed a data creation pipeline that transformed clean base tables from the BIRD dataset into a massive repository of tables. This expansion was achieved through three primary noise-injection operations:

Partitioned Tables. We used an LLM to identify logical partitioning keys (e.g., categorical regions or temporal buckets) and horizontally split the base tables into multiple sub-tables. This forces the agent to compute unions of views to retrieve complete entities. For example, horizontally splitting a base `card` table along a categorical

type key yields three sub-tables, `card_type_classic`, `card_type_junior` and `card_type_gold`.

Redundant Tables. We duplicated tables to simulate different data lifecycle stages, creating production, staging, and testing variants. A production table was created as an exact, clean copy of the original table to represent the highly authoritative source. A staging table was created by duplicating the base table and injecting noises, either (1) injecting row-level duplicates by appending a random 10% sample of existing records, or (2) applying a mask replacing values with *NULL* at a 5% probability rate. A testing table was created by taking a random 10% continuous rows/fraction sample subset of the original table. This forces the system to reason about data lineage to select the correct view.

Low-quality Tables. We intentionally degraded data integrity by generating tables with broken foreign keys, injected duplicates, and missing entities. We broke foreign keys by replacing 10% of the row references with *NULL* and another 5% of references with an out-of-bounds index (e.g., 99999999) for the identified columns ending with `_id` to simulate orphan records. We created duplicates by randomly sampling 10% of rows and appending them back to the table. We introduced data incompleteness by generating random subsets, which were created by extracting a 20% random row sample from the original table.

To fully track the data creation pipeline, we recorded detailed lineage information for all derived tables. For the partitioned tables, we stored a human-readable explanation of the partition rule (e.g., "Split partition of seasons where year is 1970"), the original source table name (e.g., "season"), the partition column (e.g., "year"), and the split data type ("timestamp" or "categorical"). These new sub-tables follow a systematic naming convention in the format of `{source}_{column}_{value}`. For the redundant tables, in addition to appending lifecycle suffixes (i.e., `_prod`, `_stg`, and `_test`), we injected explicit table descriptions such as "Clean PROD version", "Sample TEST version", and "Dirty STG version", while preserving a reference to the base. Finally, for the low-quality tables, each variant retained its source table reference alongside a specific degradation description (e.g., "Broke FK columns in {column}", "Injected random NULLs into {table}", "Duplicated rows to {table}", and "A subset of {table}"). These tables were correspondingly labeled with the suffixes `_broken_fk`, `_nulls`, `_dups`, and `_subset`, respectively. We incorporated this lineage information into the metadata of each table, making it available to METADATA REASONER and all other testing methods.

B.2 Evaluation Metric Calculation

B.2.1 Evaluation with ground truth table sets

Given that LLM-based agents generate a variable-length set of tables rather than a fixed-size ranked list, we adopt the ***F1-Score*** as our primary metric to measure accuracy between the reference table set T_{ref} and the predicted table set T_{pred}. This set-aware metric measures the system's ability to find all necessary tables and avoid irrelevant sources:

- **Recall**: $\frac{|T_{ref} \cap T_{pred}|}{|T_{ref}|}$ (Measures coverage),
- **Precision**: $\frac{|T_{ref} \cap T_{pred}|}{|T_{pred}|}$ (Measures conciseness),
- **F1-Score**: $2 \times \frac{Precision \times Recall}{Precision + Recall}$.

Additionally, we track the average number of steps, i.e., ***Avg. #step***, per task to measure the computational cost of the reasoning process.

B.2.2 Evaluation without ground truth table sets

For the synthetic BIRD dataset, we lack direct ground-truth reference tables to evaluate the correctness of selections within the derived data lake. Fortunately, because we retain complete lineage information mapping the derived tables back to their base tables, we can leverage the existing Gold SQL queries from the clean database to evaluate table selection performance, effectively bypassing the need for manual re-labeling. This approach resolves the challenges of redundant variants and horizontally partitioned tables through the following three steps:

1. Parse Gold Query: Extract all base tables and column-value constraints (e.g., from WHERE clauses) from the original Gold SQL.

2. Trace Lineage: For each returned table, consult the lineage map to identify its clean base ancestor and the specific operations used to generate it (e.g., horizontal splits or noise injections).

3. Verify Correctness: A returned table is strictly scored as correct if and only if: (1) Its associated base ancestor is required by the Gold SQL. (2) It excludes undesirable quality markers and noise (i.e., `_stg`, `_test`, `_subset`, `_dups`, `_broken_fk` or `_nulls` suffixes). See Table 6 for all types. (3) If it is a horizontally split partition, its split values perfectly correspond to the filter constraints demanded by the Gold SQL. For example, the selected table `district_A3_east_bohemia_prod` is correct if the gold SQL contains a matching filter condition `WHERE A3 = "East Bohemia"` on the `District` table.

Based on the correctly verified tables in the selected table set, we calculate the Recall, Precision and F1 scores as in Section 4.4.

C More experiment results

C.1 Failure mode analysis and case studies on Kramabench

As shown in Figure 3, ranking-based methods struggle with exact-set retrieval (Hits@20). High-cardinality domains like *Astronomy* (1,498 tables) suffer from extreme candidate noise, yielding the lowest sufficiency score (33%). Even moderate-sized domains like *Environment* (45%) and *Biomedical* (55%) exhibit low sufficiency because semantic ambiguity causes retrievers to fetch

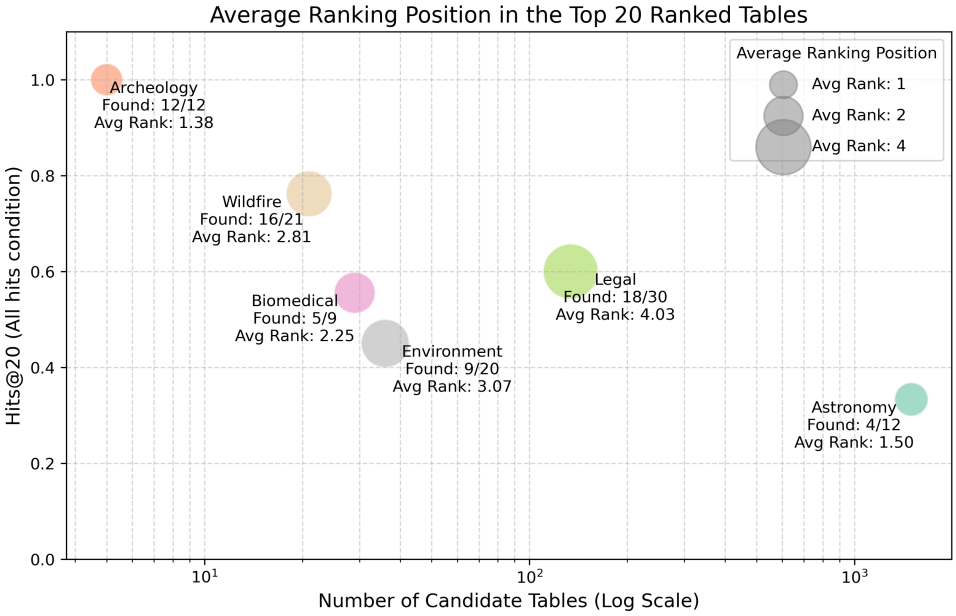

Figure 3: Average ranking position in the top 20 retrieved list from vector search. This bubble chart visualizes the search retrieval performance across six Kramabench domains, illustrating the relationship between dataset scale, retrieval success, and precision.

near-duplicates instead of the required exact set. Bubble size in Figure 3 visualizes precision based on the average retrieval position of correct tables for the examples that Hits@20 equals to 1. *Legal* exhibits the worst average position (4.03); while it solves 60% of queries, the scattered retrieval of relevant tables risks saturating the LLM's finite context window. Conversely, *Astronomy*'s low average rank (1.50) paired with its low hit rate reveals a bimodal retrieval profile: the system either fails to retrieve the target set entirely or identifies it with high confidence in the top-ranked slots.. Finally, *Archeology* that faces only 5 candidates satisfies the exact-set constraint with near-perfect positioning (1.38).

Case studies In Figure 4, we show an example from the Kramabench Legal dataset. This example demonstrates how the METADATA REASONER overcomes the retrieval limits of vector-based retrievers when handling a complex hypothetical query. The ranking-based retriever, relying on vector search (e.g., matching "2007 distribution"), can fail, with crucial tables like `2024_CSN_Report_Count` (containing 2007 report data) being ranked too low (Rank 11) because they focus on "counts" not "distribution". In contrast, the METADATA REASONER finds the right tables by 1) breaking the query into computable variables for search (Step 1&2): *Variable A: Total number of reports in 2007* and *Variable B: The percentage of "Auto Related" reports in 2024*; 2) Using tools (Step 3) to explicitly verify data presence before selection, such as confirming the table `2024_CSN_Report_Count` has historical 'Year = 2007' data and that `2024_CSN_Report_Categories` has the exact category string "Auto Related"; 3) Based on verified data, the METADATA REASONER selects the correct tables to build the specific formula needed: *Total 2007 reports* $\times$ *Percentage of 2024 "Auto Related" reports*, ensuring the correct data is retrieved and the complex query is fully answered.

C.2 RQ5: Effectiveness of discrimination-oriented search

Finally, we evaluate the very first step of our retrieval pipeline in isolation to validate our encoding strategy. When building the vector index, embedding the entire schema and generic content often causes similar tables to blur together in the vector space.

We compare three distinct metadata preparation methods:

Schema only (baseline): This method indexes the raw data structure, consisting of the original table name, schema information (column names and data types), and three random sampled rows for value grounding.

Table content-summarization: This method employs LLM-driven table content enrichment and schema as the method described in Section 3.5.1.

Discrimination-oriented metadata: This method utilizes the two-stage description generation pipeline detailed in Section 3.4.1. Instead of generating table description based on a single table content, this group-aware meta-prompting strategy generates descriptions highlighting the distinguishing table features within its semantic group.

To ensure a fair comparison, all three metadata representations utilized the same text-embedding model and underlying search algorithm. We evaluate retrieval accuracy using the `Recall@K` metric, which measures the system's ability to find all necessary tables within

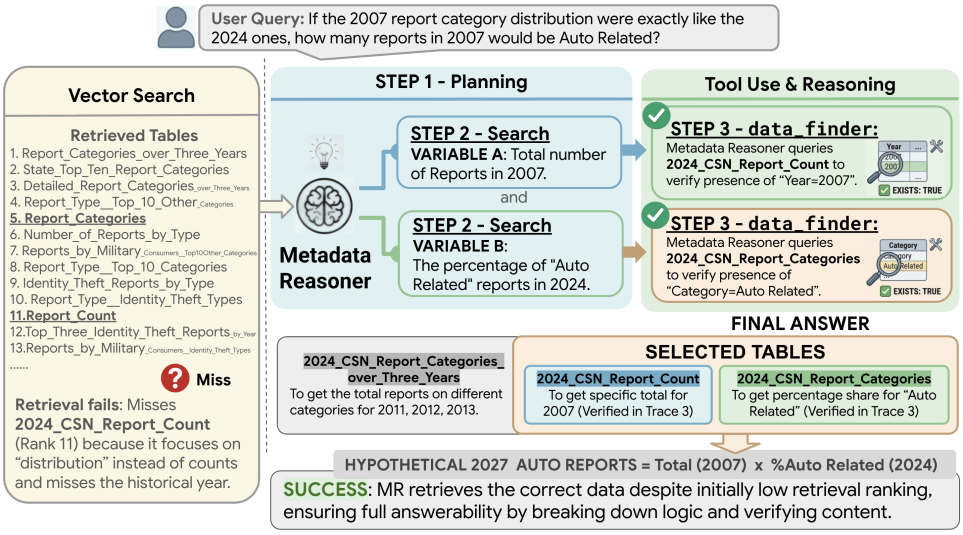

Figure 4: An example that METADATA REASONER selects the right tables with low retrieval ranks. The two ground truth tables rank 5 and 11 in the returned list in vector search. The METADATA REASONER breaks down the complex analytic query (Step 1) into searchable and computable variables for search (Step 2). It then uses tools to verify the data presence (Step 3), ensuring the precise and complete set of tables is selected for the final calculation.

Table 7: Recall scores (%) across six domains with different metadata representations at various K retrieval. "Rec@K" represents "Recall@K (%)".

Domain	Metadata	Rec@1	Rec@5	Rec@10
Astronomy (1498 tables)	Schema Only	14.85	32.58	**45.71**
	Table content	21.50	34.13	34.33
	Discrimination	**27.98**	**42.25**	43.71
Legal (134 tables)	Schema Only	24.44	47.06	51.91
	Table content	31.73	52.98	71.89
	Discrimination	**36.17**	**64.23**	**72.40**
Environment (36 tables)	Schema Only	23.44	54.89	65.21
	Table content	**30.29**	49.58	59.57
	Discrimination	29.29	**58.10**	**65.91**
Biomedical (29 tables)	Schema Only	16.67	37.04	51.85
	Table content	**41.67**	55.56	**74.07**
	Discrimination	31.48	**73.15**	73.15
Wildfire (21 tables)	Schema Only	37.30	74.60	95.24
	Table content	**51.59**	80.16	95.24
	Discrimination	**51.59**	**86.51**	**97.62**
Archeology (5 tables)	Schema Only	**66.67**	**100.**	-
	Table content	**66.67**	**100.**	-
	Discrimination	**66.67**	**100.**	-

Note: Bold values indicate the highest performance per domain.

the top K results.

As shown in Table 7, discrimination-oriented metadata consistently achieves the highest Recall@5, demonstrating significant advantages in high-cardinality environments. In the *Astronomy* domain (1,498 tables), it dramatically outperforms the schema-only baseline at Recall@1 (27.98% vs. 14.85%) and Recall@5 (42.25% vs. 32.58%). It similarly dominates across all thresholds in the *Legal* domain, where high semantic redundancy is common. These results prove that isolating differentiating variables via two-stage meta-prompting substantially enhances vector separability, particularly among structurally identical tables.

While discrimination-oriented metadata is broadly robust, the table content-driven strategy proves highly effective in specialized, data-dense domains like *Biomedical* and *Environment*. Because contextualized content representation is vital for capturing high-variance terminology (e.g., chemical compounds), this strategy achieves a peak Recall@1 of 41.67% in the *Biomedical* domain. In contrast, the schema-only baseline consistently fails to capture sufficient semantic context at low K values, trailing the best methods at Recall@1 by over 10% in *Legal* and 25% in *Biomedical*. This confirms that column headers alone are too generic to facilitate accurate semantic mapping in complex data landscapes.

This is an appendix.