DeepSearch: LLM-powered Data Acquisition for Machine Learning

Kaiyu Li
Wilfrid Laurier University
Waterloo, Canada
kli@wlu.ca

Zhongxin Hu
York University
North York, Canada
zxh@yorku.ca

Yuxin Gao
University of Toronto
Toronto, Canada
yuxin@uoft.ca

Yuyang Wu
Wilfrid Laurier University
Waterloo, Canada
wuxx7310@mylaurier.ca

ABSTRACT

In this paper, we present a novel yet practical problem—target-driven dataset acquisition (TDDA)—which seeks to acquire suitable training data for machine learning tasks based on a natural language (NL) query. This setting poses several practical challenges: users often lack clarity about which features are needed, face incomplete metadata, and must navigate complex join relationships across multiple tables. As a result, they are unable to formulate a precise query that integrates all steps of the TDDA data science pipeline.

To address this challenge, we propose DeepSearch, an end-to-end framework that solves the TDDA problem by leveraging the capabilities of large language models (LLMs). DeepSearch orchestrates the entire process by (1) inferring plausible features, (2) mapping them to actual columns within the data lake, (3) identifying joinable tables, (4) and ultimately constructing the final training dataset. Experiments on both synthetic and real-world scenarios demonstrate the effectiveness of DeepSearch. Drawing from our preliminary results and observations, we highlight several promising directions for future research. We hope this work sparks greater interest in addressing this important yet underexplored problem.

VLDB Workshop Reference Format:
Kaiyu Li, Zhongxin Hu, Yuxin Gao, and Yuyang Wu. DeepSearch: LLM-powered Data Acquisition for Machine Learning. VLDB 2025 Workshop: The 2nd International Workshop on Data-driven AI (DATAI).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/AIeasy/DataSearchTool.

1 INTRODUCTION

The focus of machine learning (ML) has shifted from solely designing effective models to also acquiring high-quality training datasets [15–17, 23]. To address this, platforms such as data lakes [3, 6] and data marketplaces [8] have emerged to help users search for relevant datasets. However, despite these systems, acquiring datasets that are readily usable for training ML models remains challenging due to several practical issues.

C1: Uncertainty in Feature Specification. Before acquiring the dataset for a machine learning task, users often struggle to anticipate which features the dataset should include, making it hard to formulate precise queries in either natural language or SQL [18, 24]. This is typically due to limited domain knowledge or insufficient prior exploration [12]. For example, in a customer churn prediction task, while the target variable such as a customer's churn status is clear, the relevant input features that influence it are not always obvious to data analysts.

C2: Limited Understanding of Metadata. In poorly maintained data lakes, schema documentation is often incomplete [22]. In some data marketplaces, schema details may not be disclosed until the purchase is completed due to trading policies [25]. Even when schemas are provided, their large scale, structural complexity, and inconsistency across datasets make it difficult for users to align them with their needs [1, 19]. For example, consider a social science researcher aiming to train an ML model to predict student dropout risk. While she may have a clear idea of the relevant features, navigating a data lake can still be challenging. She might not know whether to search for a table named *student_records* or *academic_status*, or what each table contains. Besides, without knowing how dropout status is labeled (e.g., *status* = 'inactive' or *completion_flag* = 0), or which features are available (e.g., *attendance*, *age*, and *grade*), it's hard to form a precise query.

C3: Optimal Join Plan Selection. In data lakes or marketplaces with many tables, there are often multiple joinable tables, resulting in a large number of possible join plans for a given query [6]. Without physically performing the joins, generating the final dataset, training a model, and evaluating its performance, it is difficult to assess the quality of a join plan. As a result, identifying the optimal join strategy that produces the best training dataset for a machine learning task is challenging, since it involves exploring numerous computationally expensive join combinations. For example, when predicting student dropout using tables such as transcripts, course enrollments, and financial aid records, it's often unclear which tables should be joined. The inclusion or exclusion of certain tables can significantly affect both the size and quality of the resulting dataset, making it difficult to determine the optimal join plan.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

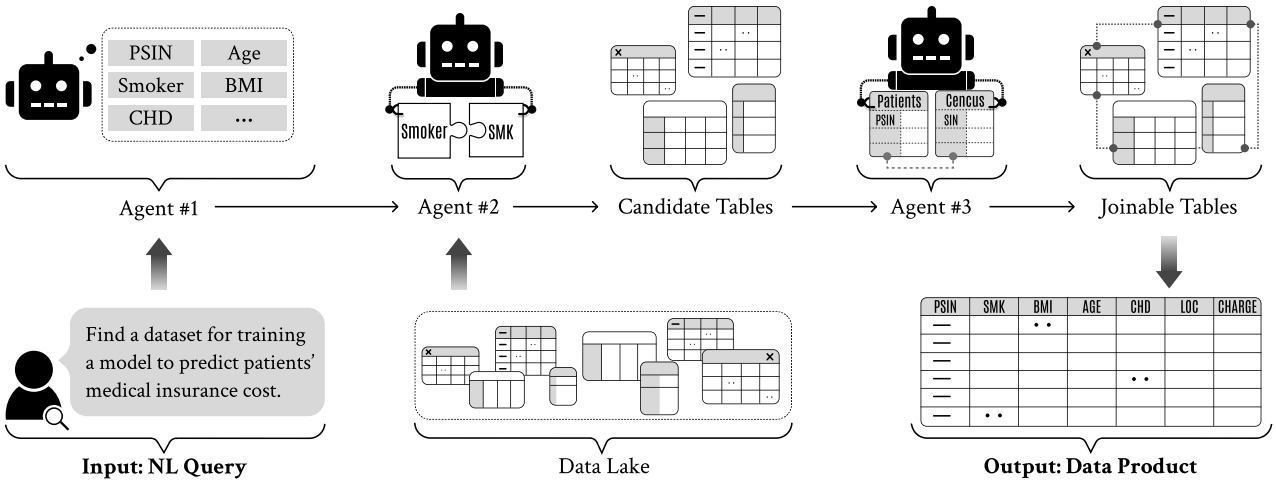

Figure 1: Example of TDDA Using DeepSearch.

This paper introduces the problem of acquiring training data based solely on a natural language query that specifies the target variable for a supervised learning task (regression or classification). We define this task as target-driven dataset acquisition (TDDA), which is constrained by three key challenges: **C1**, **C2**, and **C3**.

Our Proposal. The emergence of large language models (LLMs) enables the interpretation of user intent and the identification of relevant tables from a data lake or marketplace based on prompt semantics [27]. However, directly feeding the user query and all available tables into an LLM is impractical due to context window limitations [21]. Moreover, general-purpose LLMs are currently not capable of executing the full TDDA pipeline automatically, which involves complex steps like feature selection and join plan optimization [14]. Therefore, we propose breaking down the TDDA pipeline into manageable sub-tasks, each handled by a dedicated LLM agent, and integrating them into a cohesive end-to-end workflow.

We propose **DeepSearch**, an end-to-end framework for TDDA tasks. To address **C1**, we introduce an LLM-powered agent that conducts feature conjecture by generating candidate features potentially relevant to the query. To address **C2**, we employ a second agent that maps these candidate features to actual columns in the data lake, while also mitigating potential hallucinations introduced during the conjecture step. The agent identifies the candidate tables that includes the matched columns, and discovers joinable relationships among them. Finally, to solve **C3**, we apply a greedy heuristic to efficiently construct an efficient join plan, producing the resulting dataset—referred to as the *data product*.

Contributions. We introduce the novel and practical task of TDDA in the context of a cold-start setting or a data lake that has not been carefully curated, and we propose an end-to-end framework to address this challenge. Preliminary experiments on both synthetic and real-world datasets using state-of-the-art large language models demonstrate the effectiveness of our approach in targeted scenarios and offer insights to guide future research. To the best of our knowledge, this is the first work to formally study the TDDA problem. Based on our empirical findings, we outline several promising directions for future investigation.

2 TARGET-DRIVEN DATASET ACQUISITION FOR MACHINE LEARNING

2.1 Problem Definition

Data Model. Without loss of generality, we focus on the TDDA problem within a data lake. A data lake $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ consists of n relational tables, where each table D_i contains multiple rows and columns. These tables may originate from different departments or institutions. Some tables can be joined via primary-foreign key relationships, while others allow row-level alignment across tables D_i and D_j using shared identifiers such as unique IDs. For instance, student information tables in a university database can be joined with government census data using social insurance numbers.

We focus on a challenging scenario where metadata, table schemas, and information about joinable tables are unavailable. This situation commonly arises when a data lake is in its early stages or has not been carefully curated [13, 22]. It's important to note that, in this work, we limit our scope to relational tables and leave the exploration of other data formats as future work.

NL Query. In this paper, we focus on a practical scenario where a non-expert user submits a natural language (NL) query to discover training data for a supervised learning task (i.e., regression or classification). The user is uncertain about which features or columns are needed, lacks knowledge of the dataset's statistical requirements, and is either unable or unwilling to perform data engineering tasks such as feature engineering, identifying joinable tables in a large data lake $\mathcal{D}$, or writing SQL queries [9, 12]. The only input provided is a NL query Q that specifies the prediction target. For example, as shown in Figure 1, a user might input: "Find a dataset for training a model to predict patients' medical insurance cost," and expect the system to return a ready-to-use training dataset in tabular form.

Definition 2.1 (Target-Driven Dataset Acquisition for ML). The Target-Driven Dataset Acquisition (TDDA) for ML problem takes as input a data lake $\mathcal{D}$ without metadata or schema details, and an ambiguous NL query Q that specifies only the target variable for a supervised learning task. The objective is to produce a complete

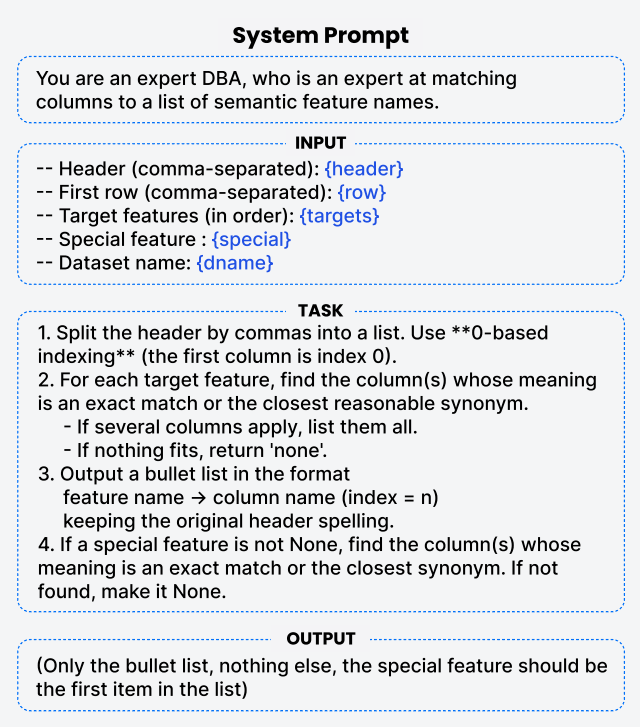

Figure 2: The Prompt Template for Columns Matching.

table that includes the given target variable along with appropriate independent variables, ready for training a machine learning model.

2.2 DeepSearch Framework

A naive approach to the TDDA problem is to feed the user query and all available table information into a single prompt, allowing an LLM to identify relevant columns, locate the corresponding tables, determine joinable relations, and generate the final dataset. However, this strategy is impractical. LLMs are prone to hallucination, often misinterpreting table semantics and selecting irrelevant columns. Existing models also struggle with complex data workflows involving multiple tables and intricate join logic [10, 20]. Additionally, LLMs are limited by context window constraints (e.g., GPT-4o's 128k-token limit [21]), making it infeasible to handle large-scale schema information in one pass.

To address these challenges, we decompose the TDDA task into smaller, well-defined sub-tasks, each handled by a specialized agent within a modular pipeline. In this paper, we propose DeepSearch, a framework designed to operationalize this decomposition, as illustrated in Figure 1. The DeepSearch workflow consists of four key stages: Feature Conjecture (Agent 1), Candidate Table Matching (Agent 2), Joinable Tables Discovery (Agent 3), and Join Plan Selection using a heuristic greedy algorithm.

Step 1: Feature Conjecture. To improve reliability and reduce hallucinations in the early stage of the pipeline, we design the first step of DeepSearch to decouple feature conjecture from schema inspection and table matching. In this step, Agent 1 receives the user query Q and, using a carefully crafted prompt, generates a list of plausible features relevant to the user's objective. The prompt deliberately excludes any table schema to prevent the model from forming spurious associations between irrelevant columns in the data lake and the target variable. Instead, the LLM relies solely on the query and its reasoning capabilities to hypothesize meaningful features. The resulting list of candidate features is then passed to Agent 2 for candidate table mapping. For example, as shown in Figure 1, given the query "Find a dataset for training a model to predict patients' medical insurance cost," Agent 1 may propose features such as Patient Social Insurance Number (PSIN), Age, Smoking Status (Smoker), Number of Children (CHD), and BMI. This setup ensures that the system grounds its reasoning in task semantics before interacting with the data lake.

Step 2: Candidate Table Matching. After generating the list of potential features, Agent 2 identifies relevant tables in the data lake $\mathcal{D}$ that may contain these features. A natural approach is to employ a retrieval-augmented generation (RAG) mechanism to search for candidate columns [7]. While RAG leverages semantic embeddings for retrieval, in practice, it can struggle to identify relevant columns when names are ambiguous or lack clear semantic cues, particularly in poorly maintained data lakes without metadata [2]. This limitation is evidenced in our preliminary experiments. For example, as shown in Figure 1, the column SMK encodes smoking status but shows little linguistic resemblance to the word "smoker," making it difficult to match even for human annotators.

Given that the data lake is assumed to be poorly maintained and lacks reliable schema metadata, conventional schema-based methods are unsuitable for this task. To overcome this limitation, we employ an in-context few-shot learning approach that enables the LLM to infer the semantics of each column using example-driven prompts. Specifically, we provide the model with randomly sampled rows from the original tables to help it interpret the meaning of each column in context. The prompt template used for this process is shown in Figure 2. At the end of this stage, DeepSearch identifies a set of m candidate tables containing potentially relevant columns. Among these, tables that include the target variable are designated as *target tables*, while the remaining tables, named *independent tables*, may contribute complementary features. These candidate tables are then forwarded to Agent 3.

Step 3: Joinable Tables Discovery. Given the m candidate tables, Agent 3 is tasked with identifying potential joinable relationships among them. Similar to Agent 2, we employ an in-context few-shot learning approach to infer joinability between tables. Specifically, for each pair of candidate tables, we provide the model with the column names and two randomly sampled rows from each table to help assess whether a meaningful join relationship exists between them. For example, as illustrated in Figure 1, a hospital patient information table and a government census table may be joinable via the social insurance number column.

At the end of this step, the candidate tables are organized into a graph $G = (V, E)$, where each vertex $v_i \in V$ represents a candidate table. For any $v_i, v_j \in V$, an edge $(v_i, v_j) \in E$ exists if the corresponding two tables are determined to be joinable. We denote the subset of vertices corresponding to all the target tables, i.e., tables that contain the target variable, as $V_{\text{target}} \subseteq V$.

Step 4: Join Plan Selection The final step involves identifying an optimal strategy for selecting and joining a subset of the m candidate tables to construct a training dataset that maximizes

predictive performance. After formalizing the candidate tables and their joinable relationships as a graph $G = (V, E)$, with $V_{\text{target}} \subseteq V$ denoting the set of target tables, a join plan is represented by a connected subgraph of G. Only the subgraphs that include at least one node from V_{target} are considered valid, as the presence of the target variable is essential to define a supervised learning task. Each valid subgraph is associated with a utility score, i.e., the performance of a machine learning model trained on the corresponding joined dataset. The objective is to find the subgraph that includes at least one target table and maximizes this utility. However, enumerating all possible join plans, training models on the resulting datasets, and evaluating their performance is computationally infeasible particularly in large data lakes where the candidate table graph can be extremely large. Therefore, in our preliminary investigation of the TDDA problem, we propose a heuristic utility function based on the number of relevant features (identified by Agent 1) covered by the joined tables. We formally define the problem as follows.

Definition 2.2 (Optimal Join Plan Selection for TDDA). Given a graph $G = (V, E)$ representing the candidate tables and their joinable relationships in a TDDA problem, and $V_{\text{target}} \subseteq V$ denoting the target tables, the objective is to identify a connected subgraph $P \subseteq G$ such that $P \cap V_{\text{target}} \neq \emptyset$ and $\bigcup_{v \in P} C(v)$ is maximized, where $C(v)$ denotes the number of relevant features contained in the table corresponding to vertex v.

The problem is a classical Connected Maximum Coverage Problem, which has been proven to be NP-hard [4]. To address this computational challenge, we adopt a greedy heuristic for efficient approximation. Specifically, we begin with the candidate table that achieves the highest R^2 score, indicating its strong predictive power for the target variable. From this starting point, we perform a breadth-first search (BFS) to iteratively join neighboring tables that contribute additional relevant features, continuing until no further new relevant features are found. While this approach does not guarantee an optimal solution, it significantly reduces computation time and yields practical results in our settings.

3 EXPERIMENT

3.1 Setup

3.1.1 Datasets and Queries. We used two data lakes: one sourced from Kaggle, comprising a mix of synthetic and real-world data, and another from the benchmark dataset Lakebench [6], which consists of government statistical data. The characteristics of the datasets and associated queries are summarized in Table 1.

(1) **Kaggle Dataset [11].** We started by selecting 20 tables from Kaggle, with 10 designated for classification tasks and 10 for regression tasks. To simulate a more realistic data lake environment, we adopted the data fragmentation strategy from Lakebench [6], randomly shuffling the column order within each table and horizontally splitting them by columns. This process produced 120 fragmented tables, among which 414 valid joinable table pairs were identified based on shared keys or identifiers. Based on the original dataset descriptions and usage contexts, we composed 6 NL queries for classification tasks and 14 for regression tasks.

(2) **OpenS Dataset [6].** We used a data lake from the LakeBench benchmark, which included multiple open datasets used for join and union-based queries. Due to the difficulty of identifying joinable tables suitable for machine learning, we focused on a single data lake: the Open Statistics dataset from Singapore government reports (OpenS). OpenS contains 1,534 tables, with 54,805 joinable table pairs. After performing the joins, we dropped tables with fewer than 500 rows. We then applied R^2 regression to assess the suitability of the joined tables for machine learning, filtering out those with an R^2 score below 0.9. Finally, three graduate students manually reviewed the remaining tables for their potential in supervised learning. After removing duplicate queries, we curated 8 valid regression queries and 3 valid classification queries.

Table 1: Characteristics of datasets and queries.

Dataset	#Tables	#Edges	#Queries (Classification / Regression)
Kaggle	120	414	6 /14
OpenS	1534	54805	3 / 8

3.1.2 Method Selection. We explored several baseline methods, such as using BM25 to index table columns. However, these approaches were unable to retrieve the correct tables for the TDDA task and failed to produce valid data products. As a result, we report only the performance of DeepSearch under different settings as our preliminary investigation.

3.1.3 Metrics. We first evaluated the success rate of DeepSearch in returning a dataset suitable for training the requested machine learning model. Next, to evaluate the quality of the returned datasets, we measured the performance of the trained models—reporting the R^2 score for regression tasks and the F1 score for classification tasks. Additionally, we calculated column recall (*ColRecall*), defined as the proportion of ground-truth columns that are present in the returned dataset. We also assessed relevance (*Relevance*) by measuring the proportion of tables in the candidate pool (returned by Agent 2) that are relevant to the query, reflecting the effectiveness of fuzzy table matching. Finally, we recorded the execution time.

3.1.4 LLM models. We used GPT-4 [21], DeepSeek-R1 [5], and Gemini-1.5 [26] in our experiments, representing a diverse set of state-of-the-art LLMs. GPT-4 is known for its strong performance and benchmark-setting capabilities. DeepSeek-R1 offers a competitive open-source alternative focused on transparency. Gemini-1.5 was selected for its advanced reasoning abilities.

3.1.5 Machine. The experiments were conducted on an Ubuntu system featuring an Intel Core i9-10850K CPU, 32 GB of RAM, and an NVIDIA RTX 3080 GPU.

3.2 Results

The experimental result are shown in Table 2. Overall, DeepSearch performs well across various metrics—including success rate, R^2 for regression, F1 score for classification, column recall (ColRecall), and table relevance (Relevance)—and maintains reasonable execution time (a few minutes) when using the GPT-4 model on the Kaggle dataset. However, its performance diminishes when applied to the OpenS dataset or when alternative models are used.

The performance difference between the Kaggle and OpenS datasets is primarily attributed to differences in metadata quality. Kaggle tables typically feature meaningful, descriptive titles and column names written in full English words, which facilitates

Table 2: Experimental results of DeepSearch.

	GPT-4				DeepSeek-R1				Gemini-1.5			
	Classification		Regression		Classification		Regression		Classification		Regression	
	Kaggle	OpenS	Kaggle	OpenS	Kaggle	OpenS	Kaggle	OpenS	Kaggle	OpenS	Kaggle	OpenS
Success Rate	0.667	0.667	0.857	0.75	0.667	0.333	0.571	0.25	0.667	0.333	0.428	0.375
R^2 (F1) / Ground Truth	0.656 / **0.974**	0.60 / **0.998**	0.647 / **0.778**	0.675 / **0.994**	0.612 / **0.974**	0.29 / **0.998**	0.325 / **0.778**	0.23 / **0.974**	0.601 / **0.974**	0.30 / **0.998**	0.38 / **0.778**	0.325 / **0.994**
Time (s)	308.1	338.3	315.9	259.6	92.6	502.35	104.5	554.33	658.59	2201.14	807.4	2409.12
ColRecall	0.558	0.28	0.718	0.42	0.5	0.33	0.47	0.11	0.41	0.333	0.23	0.21
Relevance	0.631	0.61	0.756	0.69	0.60	0.33	0.521	0.25	0.52	0.33	0.378	0.31

feature inference and table selection. In contrast, OpenS tables often lack informative titles, and their column names are usually obscure, like 'SG' or 'P', making it more challenging for the model to interpret their content.

Among the models tested, GPT-4 demonstrates the strongest performance, exhibiting a robust ability to understand and generalize in complex data acquisition tasks. Gemini-1.5, despite its strengths in multimodal reasoning and coding, proves less effective when it comes to data lake exploration and schema inference. DeepSeek-R1, designed primarily for conversational tasks, falls short in handling complex data science queries, leading to weaker performance in the TDDA benchmark.

Across all models and datasets, query execution times generally remain within a few minutes, which is reasonable given the current scale of data lakes. However, as the number of tables increases, the runtime tends to grow roughly linearly. Looking ahead, we believe that more advanced indexing techniques will be necessary to efficiently retrieve relevant tables instead of linearly scanning. Among the three models, DeepSeek-R1 was the fastest on Kaggle datasets, but this speed came at the expense of declined accuracy, particularly in tasks requiring complex reasoning.

We also conducted ablation studies to examine how DeepSearch's performance varies with different parameters. Increasing the number of example rows from each table which provides more context can enhance performance, as it allows the model to gather richer semantic cues about each column. In larger data lakes, identifying relevant candidate tables becomes more challenging than in some data lakes, which raises the likelihood of matching columns with potentially relevant names in irrelevant tables, causing DeepSearch may inadvertently include irrelevant tables, ultimately reducing the effectiveness of the final dataset.

4 CASE STUDY

In this section, we present three real-world cases from our experiments to illustrate how DeepSearch operates in practice and to highlight the challenges of TDDA for ML. The selected cases, drawn from the Kaggle datasets used in our evaluation, span three domains: real estate, medical insurance, and biology. Each example corresponds to a specific query, showcasing distinct performance.

Query 1: *Please find a dataset suitable for training a regression model to predict the house price.*

This case represents a perfect match. The agents successfully identified 5 candidate tables from the data lake, connected by 10 possible join relationships. All selected tables were relevant to the query, including the ground-truth table. Ultimately, DeepSearch selected the table with the highest R^2 score, which precisely matched the ground truth. The entire process took 466.61 seconds to complete the TDDA workflow.

Query 2: *Identify a data set for training a model to predict patient medical insurance costs using their health-related information.*

This case demonstrates a failure in performance. DeepSearch retrieved 19 candidate tables, though only 6 of them were actually relevant to the query (including the ground truth). Finally, the system incorrectly selected unrelated tables concerning housing prices to construct the final data product. This failure stems from the LLM's overly general feature conjecture, where it mistakenly associated price- and tax-related features with patient medical payments. This example highlights the importance of providing richer context in the query to guide the LLM more effectively, particularly in large data lakes with many potentially misleading tables.

Query 3: *Please help me find a dataset for training a classification model to predict prospective students' chances of acceptance of an MBA program.*

This query was successful, though it did not recover the ground-truth table. DeepSearch identified 8 candidate tables, 6 of which were relevant to the query, including the ground truth. Finally, it selected and joined two relevant but non-ground-truth tables to construct the final dataset. The resulting classification model achieved an F1 score of 0.9676, slightly below the ground-truth, indicating that while the output was not optimal, it was effective.

Inspiration. Based on our case-by-case analysis, we identify two key directions for future improvement. First, DeepSearch's success heavily depends on the quality of feature conjecture and relevant table matching. When too many irrelevant candidate tables are selected, the system may fail to construct a usable dataset. To address this, future work could explore improved prompting strategies to provide richer query context or incorporate more effective retrieval-augmented generation (RAG) techniques to enhance table selection.

Second, even when relevant candidate tables are correctly identified, DeepSearch does not always recover the ground-truth table, which can negatively impact model performance, even if the resulting dataset is usable. Addressing this challenge may require developing more robust methods for evaluating join plans.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose DeepSearch, an end-to-end framework for target-driven dataset acquisition for machine learning tasks. DeepSearch employs a multi-agent architecture that decomposes the data acquisition process into sub-tasks, each guided by carefully crafted prompts to a dedicated agent powered by large language models. As the first empirical study of this problem, our approach demonstrates effectiveness when the data lake is small and the semantics of the tables in it are clear. However, it struggles in complex scenarios—particularly when tasks involve diverse features or the data lake contains many confusing or irrelevant tables. Looking ahead, we identify several directions for future research.

Scalability to Large Data Lakes. Currently, DeepSearch performs adequately on small data lakes, with discovery time typically within a few minutes. However, this time grows linearly with the number of tables, since Agent 2 must scan all tables for candidate joins. Future work will explore indexing strategies to reduce retrieval time and improve scalability.

Optimal Joining Path Selection. At the final stage of DeepSearch, multiple join paths may yield valid datasets, but enumerating all possibilities is computationally expensive. We will investigate heuristics to estimate join selectivity and data utility without performing the joins. Besides, we will consider extend current work to support joins across heterogeneous data types, enabling richer feature construction, e.g., graphs or semi-structured sources.

Handling Data Quality Issues. Real-world datasets often contain missing values or inconsistent records, which can affect join quality and downstream model performance. Entity resolution is another challenge—for example, recognizing that "Wilfrid Laurier University" and "WLU" refer to the same entity. We aim to rethink data cleaning and resolution techniques under the setting of TDDA.

Domain-Specific Scenarios. In specialized domains such as healthcare, domain knowledge is critical for feature understanding and table matching. We plan to explore RAG techniques enhanced with domain-specific knowledge graphs and evaluate whether fine-tuned LLMs can better support domain-sensitive TDDA tasks.

Human-in-the-Loop Interaction. LLMs still face limitations in resolving task ambiguity. Future work will explore human-in-the-loop designs to incorporate human feedback into the TDDA pipeline, helping clarify intent and improve the accuracy.

REFERENCES

[1] Nour Alhammad, Alex Bogatu, and Norman W Paton. 2022. Towards Schema Inference for Data Lakes. *arXiv preprint arXiv:2206.03881* (2022).

[2] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven Failure Points When Engineering a Retrieval Augmented Generation System. arXiv:2401.05856 [cs.SE] https://arxiv.org/abs/2401.05856

[3] Chengliang Chai, Yuhao Deng, Yutong Zhan, Ziqi Cao, Yuanfang Zhang, Lei Cao, Yuping Wang, Zhiwei Zhang, Ye Yuan, Guoren Wang, et al. 2024. LakeCompass: An End-to-End System for Data Maintenance, Search and Analysis in Data Lakes. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4381–4384.

[4] Gianlorenzo D'Angelo and Esmaeil Delfaraz. 2025. Approximation Algorithms for Connected Maximum Coverage, Minimum Connected Set Cover, and Node-Weighted Group Steiner Tree. *arXiv preprint arXiv:2504.07725* (2025).

[5] DeepSeek-AI, Daya Guo, Qihao Zhu, Dejian Yang, Haowei Zhang, Junxiao Song, and Wenfeng Liang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

[6] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, et al. 2024. Lakebench: A benchmark for discovering joinable and unionable tables in data lakes. *Proceedings of the VLDB Endowment* 17, 8 (2024), 1925–1938.

[7] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[8] Raul Castro Fernandez, Pranav Subramaniam, and Michael J Franklin. 2020. Data market platforms: Trading data assets to solve data problems. *arXiv preprint arXiv:2002.01047* (2020).

[9] Amy K. Heger, Liz B. Marquis, Mihaela Vorvoreanu, Hanna Wallach, and Jennifer Wortman Vaughan. 2022. Understanding Machine Learning Practitioners' Data Documentation Perceptions, Needs, Challenges, and Desiderata. arXiv:2206.02923 [cs.HC] https://arxiv.org/abs/2206.02923

[10] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, et al. 2024. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679* (2024).

[11] Kaggle. n.d.. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com. Accessed: May 31, 2025.

[12] Wataru Kawabe and Yusuke Sugano. 2024. DuetML: Human-LLM Collaborative Machine Learning Framework for Non-Expert Users. *arXiv preprint arXiv:2411.18908* (2024).

[13] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J Miller. 2022. Integrating data lake tables. *Proceedings of the VLDB Endowment* 16, 4 (2022), 932–945.

[14] Dawei Li, Zhen Tan, and Huan Liu. 2025. Exploring large language models for feature selection: A data-centric perspective. *ACM SIGKDD Explorations Newsletter* 26, 2 (2025), 44–53.

[15] Kaiyu Li, Guoliang Li, Yong Wang, Yan Huang, Zitao Liu, and Zhongqin Wu. 2021. CrowdRL: An end-to-end reinforcement learning framework for data labelling. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 289–300.

[16] Kaiyu Li, Xiaohui Yu, and Jian Pei. 2025. CDA: Cost-Sensitive Data Acquisition for Incomplete Datasets. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 1551–1564.

[17] Kaiyu Li, Xiaohang Zhang, and Guoliang Li. 2018. A rating-ranking method for crowdsourced top-k computation. In *Proceedings of the 2018 International Conference on Management of Data*. 975–990.

[18] Rachel Lin, Bhavya Chopra, Wenjing Lin, Shreya Shankar, Madelon Hulsebos, and Aditya Parameswaran. 2025. AI-Assisted Dataset Discovery with DATASCOUT. (2025).

[19] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment* 12, 12 (2019), 1986–1989.

[20] Taiyu Oh, George Karagiannis, Mohsen Ghaffari, and Theodoros Rekatsinas. 2024. Thinking with Tables: Tabular Structures Enhance LLM Comprehension for Data-Analytics Requests. *arXiv preprint arXiv:2412.17189* (2024). https://arxiv.org/abs/2412.17189

[21] OpenAI. 2024. GPT-4o Overview. https://platform.openai.com/docs/models/gpt-4o-mini. Accessed: 2025-05-30.

[22] Pegdwendé N Sawadogo, Etienne Scholly, Cécile Favre, Eric Ferey, Sabine Loudcher, and Jérôme Darmont. 2019. Metadata systems for data lakes: models and features. In *New Trends in Databases and Information Systems: ADBIS 2019 Short Papers, Bled, Slovenia, September 8–11, 2019, Proceedings 23*. Springer, 440–451.

[23] Chen Shani, Jonathan Zarecki, and Dafna Shahaf. 2023. The lean data scientist: recent advances toward overcoming the data bottleneck. *Commun. ACM* 66, 2 (2023), 92–102.

[24] Aditi Singh, Akash Shetty, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. A Survey of Large Language Model-Based Generative AI for Text-to-SQL: Benchmarks, Applications, Use Cases, and Challenges. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 00015–00021.

[25] Markus Spiekermann. 2019. Data marketplaces: Trends and monetisation of data goods. *Intereconomics* 54, 4 (2019), 208–216.

[26] Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. https://arxiv.org/abs/2403.05530. arXiv:2403.05530.

[27] Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. 2024. Chat2data: An interactive data analysis system with rag, vector databases and llms. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4481–4484.