Exploration Lab IITK at SemEval-2025 Task 8: Multi-LLM Agent QA over Tabular Data

Aditya Bangar **Ankur Kumar** **Shlok Mishra** **Ashutosh Modi**
Indian Institute of Technology Kanpur (IIT Kanpur)
Kanpur, India
{adityavb21, ankurk21, shlokm21}@iitk.ac.in, ashutoshm@cse.iitk.ac.in

Abstract

This paper presents our Multi-LLM Agentic System that helps solve the problem of tabular question answering as posed in the SemEval Task- 8:Question Answering over Tabular Data. Our system incorporates a Agentic Workflow where we assign each agent a role along with the context from other agent to better help resolve the ambiguity. As the user poses their question along with the dataframe, we firstly try to infer the types of the columns from the dataframe and also the expected answer type given the question and the column types, then the planner agent gives out a plan that tells us about the steps that we have to take to get the answer, each step is written such that it helps us write one line of python code. Then we call the cod- ing agent which attempts to write the code given the information from the previous agents. Then we do a debugging pass through a debugging agent which tries to resolve the issue given the previous context and finally deliver the answer if the code runs error free. Our system achieved 14th place on the overall open source models track.

1 Introduction

Question Answering (QA) over tabular data is a Natural Language Processing (NLP) task that involves extracting aanswers from structured tabular formats. This task is crucial as many financial reports, scientific data, and government records, exist in tabular form. Automating QA over tables enhances information retrieval and decision-making processes. The task covered in this work involves evaluating the performance of language models on tabular QA on *Databench (full tables)* benchmark, following the benchmark and methodologies outlined in the task overview paper (Grijalba et al., 2024).

The problem involves addressing challenges like diverse query intents, table structures, and complex answer types (Boolean, categorical, numerical, and lists). Key sub-problems include query intent disambiguation, efficient data retrieval, and reasoning over large tables and multi-turn interactions. The broader impact of this research lies in its potential to scale across diverse data sources. Our system utilizes *OpenAI's API*, accessed via *Python*, to interact with **Deepseek's R1** open-source model.

Through this competition, we gained valuable insights into the strengths and limitations of traditional Transformer-based models in *question answering over tabular data*. We ranked **fourteenth** among all submissions, highlighting both our achievements and areas for improvement. We explored various reasoning-based approaches and found that breaking the problem into smaller, more manageable subtasks significantly enhanced performance. By assigning each subtask—such as interpreting the question, extracting relevant information, searching and querying with code, and debugging errors—to specialized models, we distributed the workload effectively and provided each model with a focused area of operation. This multi-agent approach not only streamlined the process but also improved precision by reducing the cognitive burden on any single model. We observed that the model struggles with semantic meaning of some tricky questions, which requires additional reasoning before execution which suggests potential areas for improvement.

The implementation is publicly available at ***this Github repository***.

2 Background

2.1 Problem Definition

Tabular question answering (QA) involves generating accurate answers to natural language queries based on structured tabular data. Formally, given a table T and a natural language question Q, the goal is to produce an answer A:

$$(T, Q) \rightarrow A$$

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 2165–2169

Example: Consider table T:

Name	Age
Alice	30
Bob	25

Given question Q: "What is Alice's age?", the expected answer A is "30".

2.1.1 Terminology and Definitions

- **Table** (T): A structured tabular dataset where each column C_i has a specific data type (e.g., numerical, categorical).
- **Question** (Q): A natural language query for extracting information from table T.
- **Answer** (A): The response derived from table T that satisfies question Q.

2.1.2 Our Approach

We propose a multi-agent, self-correcting framework for Question Answering over Tabular Data, addressing the limitations of direct table encoding seen in models such as TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020), and StruBERT (Trabelsi et al., 2022). Our approach leverages modern Large Language Models (LLMs) (Fang et al., 2024) to iteratively refine understanding, code generation, and debugging.

Our methodology uses the *Deepseek's* latest **Deepseek R1** model in a multi-agent setting. We use *OpenAI*'s API to call the model at various substeps of the Query task, mainly plan generation, relevant column extraction, query code generation, and debugging. First, we extract dataframe metadata, including column names, types, statistical metrics, and expected answer types. This, along with the question and dataframe, is sent to the *Planner Agent*, which formulates a solution strategy. The plan is passed to the *Relevant Column Agent*, which identifies essential columns and refines the context for the *Coder Agent*. The *Coder Agent* then generates and executes Python query code, verifying success based on error-free execution and correct answer type.

If execution fails due to errors or incorrect output, the *Debugger Agent* modifies the code and retries execution. This process is repeated up to three times to enhance accuracy. If all attempts fail, the system outputs *NULL*. This structured approach balances efficiency and reliability in query resolution.

2.2 Related Works

The survey by Fang et al. (Fang et al., 2024) provides an in-depth review of tabular question answering (QA) using large language models (LLMs). This work not only summarizes existing methodologies but also categorizes the research into several key areas that address the multifaceted challenges of working with tabular data. It explains how different approaches tackle issues such as clarifying ambiguous user intents, retrieving relevant segments from tables, managing sequential interactions, and enabling autonomous query answering.

Specifically, the survey examines **Query Intent Disambiguation** techniques, which focus on clarifying the user's query (Zha et al., 2023; Deng et al., 2022). It further reviews **Search & Retrieval** methods aimed at extracting the most pertinent portions of a table (Zhao et al., 2024; Sundar and Heck, 2023) and discusses strategies for handling **Multi-Turn Settings** in sequential interactions (Sui et al., 2024; Ye et al., 2023; Liu et al., 2023). In addition, the survey explores **Autonomous Tabular Question Answering** through multi-agent approaches (Zhu et al., 2024; Ye et al., 2024, 2023) and highlights the role of **Few-shot and Zero-shot Learning** in efficiently adapting models with minimal labeled data (Ye et al., 2024). These distinct research directions collectively underscore the evolving landscape of tabular QA using LLMs.

2.3 Corpus/Data Description

This project utilizes *DataBench*, a benchmark dataset designed for evaluating question answering over tabular data in structured CSV-style files.

Dataset Overview: DataBench comprises 65 publicly available tabular datasets across five domains: Health, Business, Social Networks and Surveys, Sports and Entertainment, and Travel and Locations. The dataset includes a total of 3,269,975 rows and 1,615 columns.

- **Domain Taxonomy:** Table 1 shows a breakdown of DataBench by domain, including the number of datasets, rows, and columns.
- **Column Types:** Table 2 illustrates the range of column data types found in DataBench, from simple numeric fields to more complex list structures.

2.4 Questions and Answers Generation

DataBench includes 1,300 hand-crafted question-answer pairs (20 per dataset) spanning five types:

Table 1: DataBench Domain Taxonomy

Domain	Datasets	Rows	Columns
Business	26	1,156,538	534
Health	7	98,032	123
Social	16	1,189,476	508
Sports	6	398,778	177
Travel	10	427,151	273
Total	**65**	**3,269,975**	**1615**

Table 2: Column Types in DataBench

Type	Columns	Example
Number	788	55
Category	548	Apple
Date	50	1970-01-01
Text	46	A red fox ran...
URL	31	`google.com`
Boolean	18	True
List[Number]	14	[1, 2, 3]
List[Category]	112	[sam, dee, lia]
List[URL]	8	[`ggu.uk`, `abc.in`]

Boolean, **Category**, **Number**, **List[Category]**, and **List[Number]**.
Examples:

- *Question*: "What's the oldest passenger's class?" *Answer*: First
- *Question*: "Who are the passengers under 30?" *Answer*: [Lil Lama, Cody Lama]

3 System Overview

We explored various approaches to Question Answering over Tabular Data, including transformer-based models and LLM-based coding paradigms. While models such as TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020), and StruBERT (Trabelsi et al., 2022) showed promise, their direct encoding of table context was insufficient for our analytical needs.

To overcome these limitations, we developed a multi-agent, self-correcting framework using modern LLMs (viz. DeepSeek-V3, DeepSeek-R1):

- **Understand the table and Plan**: We firstly extract some relevant information about the dataframe by writing functions that help infer the column types and also have an LLM call that determines the datatypes given the dataframe head and the query. This provides the LLM with a better understanding of both the data and the query via in-context learning. Then, we call the Planner Agent to devise an easy step-by-step plan that guides the coding agent to better interpret the query and dataframe for code generation.
- **Plan then Code**: Once the plan is obtained, another LLM call identifies the relevant columns required for answering the question, thereby clarifying the objective for the Coder Agent via in-context learning.
- **Output Analysis and Code Correction**: After receiving the code output from the Coder Agent, we execute the code and debug it using a Debugging Cycle involving the Debugging Agent. This cycle runs for a maximum of three attempts to optimize execution time. Figure 1 illustrates the workflow of our multi-agent code and output-based approach.

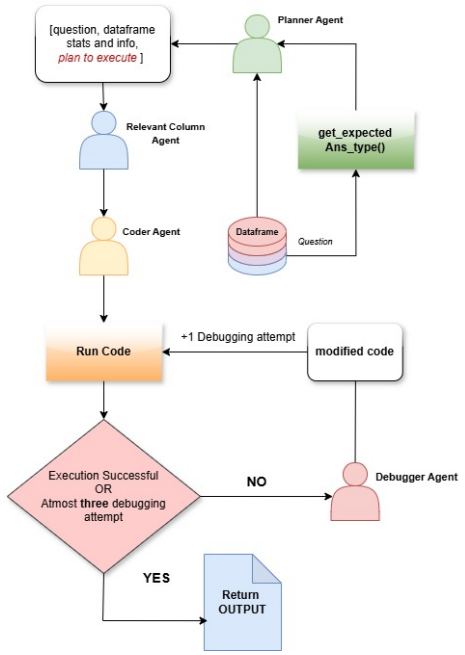

Figure 1: Workflow of the Multi-Turn Code and Output-Based Approach. The first agent generates code based on a plan, while the second agent evaluates and iteratively corrects the output.

Structured Outputs: We establish a defined structure for communication between the LLMs when certainty is required. For example, for inferring column types, determining the answer type, and generating code for extraction and execution, we employ intermediate LLM passes that produce outputs in a JSON structure.

4 Experimental Setup and Results

In our experiments, we evaluated our multi-agent, self-correcting framework for question answering over tabular data on the full DataBench dataset. The evaluation metric used was **accuracy**—the percentage of correctly answered queries based on the official evaluation script. All experiments were performed exclusively on the full dataset, in line with our design decisions, and the reported results are those obtained on the test set.

4.1 Baselines and Compared Systems

We compare our system against:

1. **Baseline Model:** The relevant baseline presented in the task paper (Grijalba et al., 2024) achieves an accuracy of 26% on the test set.
2. **Top Performer:** As reported in the official task ranking the best-performing system attained an accuracy of 95.02% on the full dataset.
3. **Our System:** Our multi-agent system, which leverages a sequence of specialized agents for planning, code generation, and debugging, achieves an accuracy of 79.69%. Despite this notable improvement over the baseline, our submission was ranked 14th in the overall task.

Table 3 summarizes the performance of these three systems.

Method	Accuracy (%)	Rank
Baseline (Task Paper)	26.00	33rd
Top Performer (Team TeleAI)	95.02	1st
Ours	79.69	14th

Table 3: Performance on the Full DataBench Dataset in the Open Source Models Track

4.2 Evaluation Metrics

We assess performance using:

- **Accuracy**: Percentage of correctly answered questions.
- **Leaderboard Ranking**: Our multi-turn framework is ranked on the SemEval leaderboard.

5 Results

The experimental results highlight several key points:

- **Improved Accuracy:** Our system outperforms the baseline by a substantial margin, demonstrating the benefits of decomposing the problem into distinct subtasks handled by specialized agents.
- **Iterative Refinement:** The multi-agent framework—with dedicated agents for planning, relevant column identification, code generation, and debugging—plays a crucial role in handling complex queries over tabular data. The iterative debugging cycle ensures that errors in code generation are corrected promptly, leading to a higher likelihood of accurate outputs.
- **Ranking Implications:** Although our system achieves a high accuracy of 79.69%, the overall task ranking (14th) indicates that there remains significant room for improvement. Future works in this direction related to question ambiguity removal and iterative refinement through better understanding the semantics of the table in relation to the query will help improve the results.

These findings not only validate the effectiveness of our multi-agent approach on the full dataset but also provide a roadmap for future enhancements in tabular question answering systems.

6 Conclusion

The proposed 2-LLM agent framework combines intent-driven code generation with output-based refinement. Our Multi-Agent Code and Output-Based Approach outperforms baseline methods and transformer-based models in handling complex queries. Ranked third on the SemEval leaderboard, our framework sets a strong foundation for further work on prompt engineering, improved retrieval mechanisms, and domain-specific extensions.

Acknowledgments

We conducted this research at the Exploration Lab, IIT Kanpur, and the Department of Computer Science and Engineering at IIT Kanpur. We extend our heartfelt gratitude to all members for their invaluable support and guidance. We also thank the organizers and all contributors to the DataBench dataset for their contributions to the research community.

References

Yang Deng, Wenqiang Lei, Wenxuan Zhang, Wai Lam, and Tat-Seng Chua. 2022. PACIFIC: Towards proactive conversational question answering over tabular and textual data in finance. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6970–6984, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models(llms) on tabular data: Prediction, generation, and understanding – a survey. *Preprint*, arXiv:2402.17944.

Jorge Osés Grijalba, Luis Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with databench: A large-scale empirical evaluation of llms. In *Proceedings of LREC-COLING 2024*, Turin, Italy.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Tianyang Liu, Fei Wang, and Muhao Chen. 2023. Rethinking tabular data understanding with large language models. *Preprint*, arXiv:2312.16702.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. *Preprint*, arXiv:2305.13062.

Anirudh S. Sundar and Larry Heck. 2023. cTBLS: Augmenting large language models with conversational tables. In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 59–70, Toronto, Canada. Association for Computational Linguistics.

Mohamed Trabelsi, Zhiyu Chen, Shuo Zhang, Brian D. Davison, and Jeff Heflin. 2022. Strubert: Structure-aware bert for table search and matching. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 442–451. ACM.

Junyi Ye, Mengnan Du, and Guiling Wang. 2024. Dataframe qa: A universal llm framework on dataframe question answering without data exposure. *Preprint*, arXiv:2401.15463.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426. Association for Computational Linguistics.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, Tao Zhang, Chen Zhou, Kaizhe Shou, Miao Wang, Wufang Zhu, Guoshan Lu, Chao Ye, Yali Ye, Wentao Ye, Yiming Zhang, Xinglong Deng, Jie Xu, Haobo Wang, Gang Chen, and Junbo Zhao. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *Preprint*, arXiv:2307.08674.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. Docmath-eval: Evaluating math reasoning capabilities of llms in understanding long and specialized documents. *Preprint*, arXiv:2311.09805.

Jun-Peng Zhu, Peng Cai, Kai Xu, Li Li, Yishen Sun, Shuai Zhou, Haihuang Su, Liu Tang, and Qi Liu. 2024. Autotqa: Towards autonomous tabular question answering through multi-agent large language models. *Proc. VLDB Endow.*, 17(12):3920–3933.