Saama Technologies at SemEval-2025 Task 8: Few-shot prompting with LLM-generated examples for question answering on tabular data

Hwanmun Kim, Kamal Raj Kanakarajan, Malaikannan Sankarasubbu
Saama Technologies
{hwan.kim, kamal.raj, malaikannan.sankarasubbu}@saama.com

Abstract

For SemEval 2025 Task 8, addressing tabular data question answering, we introduce a novel few-shot prompting system that guides large language models (LLMs) to generate Python code representing the reasoning process. Our system automatically creates a library of exemplar code snippets from training data, which are then used for few-shot prompting. Crucially, we incorporate a selection prompt to choose the best candidate code from multiple LLM-generated options, improving robustness and accuracy. Our system achieved competitive results, ranking 17th in the Open Model track and 25th overall. Ablation studies demonstrate the effectiveness of our exemplar generation and code selection strategies. We conclude with a discussion of limitations and promising avenues for future research.

1 Introduction

Recent rapid advancement of Large Language Models (LLMs) has innovated a wide range of natural language processing (NLP) tasks Zhao et al. (2023); Hadi et al. (2023), as the extended context size enables LLMs to handle much more complicated tasks. Yet, question answering on tabular data requires more strategic approaches as even extended context size cannot handle full size of large tables effectively. SemEval 2025 Task 8[1] Os'es Grijalba et al. (2025) deals with this problem by providing the challenge for the recently developed DataBench dataset. Through this dataset, SemEval 2025 Task 8 provides hundreds of questions and associated tables written in English along with answers and related information.

In this paper, we explain our approach to SemEval 2025 Task 8 in detail. To provide LLMs guides to steps to deduce the answer from the table, we generated example python codes out of the provided questions and tables in the training data. With these examples, we performed few-shot prompting followed by a selection prompt to choose the most proper code to answer the question among multiple candidate codes.

We applied our optimal approach, identified through extensive experimentation, to the competition test set. Our submitted system achieved a rank of 17th in the Open Model track and 25th overall on the SemEval 2025 Task 8 leaderboard. Ablation studies were conducted to evaluate the contribution of each system component. Further analysis of our example generation method revealed a need for improvement in handling boolean-type questions. Finally, we propose several directions for future research to enhance system performance.

2 Background

As in many other NLP tasks, question answering using LLMs has shown notable progress recently thanks to a variety of approaches including such as chain-of-thought prompting Wang et al. (2023), retrieval-augmented-generation Fan et al. (2024), knowledge distillation Sutanto and Santoso (2024), and hybrid approaches Daull et al. (2023).

Among different kinds of question answering tasks, question answering on tabular data is distinguished from other question answering tasks by the fact that its provided tabular data are structured and can be arbitrarily long. To deal with these problems, most approaches rely on non-natural languages such as SQL or Python to represent logical steps deducing a subtable or an answer from the given table Jin et al. (2022); Zhang et al. (2023); Cao et al. (2023); Kong et al. (2024); Zhang et al. (2024); Lu et al. (2024); Zhu et al. (2024). To utilize the most relevant information from the table, many systems adopt mechanisms to select relevant subtables Zhang et al. (2023); Kong et al. (2024); Sun et al. (2016). As there are multiple approaches with different advantages, selection agents that choose the

[1] https://jorses.github.io/semeval/

best approach among multiple candidates are often used to increase the performance of the system Gao et al. (2024); Pourreza et al. (2024).

2.1 Task description

SemEval 2025 Task 8 aims to develop a question answering system for tabular data. The challenge provides multiple datasets and questions about these datasets where each question only deals with a single dataset at a time. The challenge is composed of two subtasks; the subtask 1 can provide datasets of any size and the subtask 2 provides tables of maximum 20 rows each.

2.1.1 Dataset

SemEval 2025 Task 8 uses the DataBench dataset Grijalba et al. (2024) for development and evaluation. DataBench datset used for this challenge is composed of 3 splits: train, dev, and test.

- **Train split**: 988 questions over 49 tables
- **Dev split**: 320 questions over 16 tables
- **Test split**: 522 questions over 15 tables. Participants submit the results on this split.

There is no overlap of tables between different splits. Each question belongs to one of 5 types: boolean, number, list of number, category, list of category. In the train split and the dev split, this type information and the columns used to answer each question are included. For the test split, only the question and the table are provided.

2.1.2 Evaluation Metric

The evaluation metric for SemEval 2025 Task 8 is the accuracy of the result, which is defined as the ratio of the results matching with answers over the total number of questions. When determining whether a result matches with the answer, the order within the list is ignored. For numeric results, each number is rounded up to the second decimal.

3 System overview

Our approach is consisted of three steps. First, we generate example python codes using zero-shot prompting on the LLM with the training data. Second, we do few-shot prompting on the LLM using the example codes generated in the first step to generate candidate python codes for the input question. Finally, we run a selection prompt to select the most suitable code to answer the given question.

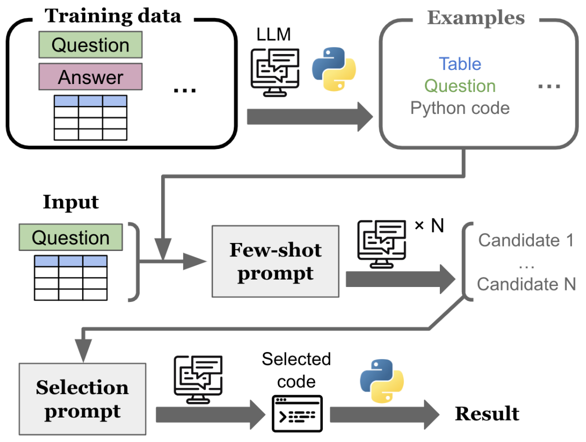

Figure 1: Overview of our few-shot prompting system.

By executing this selected code, we obtain the final result for the given question. This entire process is summarized in Fig. 1.

3.1 Example generation with LLM

To guide the LLM to properly answer the question based on the given training data, the most straightforward way is to utilize in-context learning via few-shot prompts. While one may try a few-shot prompt composed of the question and answer pairs given in the training data, such prompt lacks any detailed explanation how one can reach to the result. Moreover, it may be almost impossible to present all the information in the table if the table size is too large compared to the context size of the LLM. Therefore, one should use example logical steps to guide LLM to replicate its own logical steps to deduce the result from the table. In our approach, we used python codes as our logical steps.

To turn the entire training split into example codes to be used in few-shot prompting, we first performed zero-shot prompting on the training data along with a simplified table. The zero-shot prompt we used can be found in Appendix B. For the simplified table, we randomly sampled 10 rows from the original table and presented them as in Appendix A. Our table presentation is inspired by Zhang et al. (2023). Once the python code is generated, we executed it and compare the execution result with the given answer of the training data, then accepted it as a valid example if the result matches with the answer. To collect as much examples as possible, we repeated the zero-shot prompting 10 times per each different temperature $T = 0.2, 0.3, 0.4, 0.5$ and aggregated all the valid examples. As a result, we collected the example set of 839 examples out of 988 questions in the train-

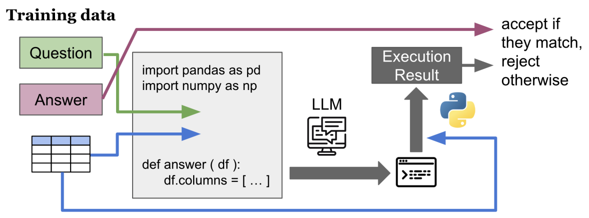

Figure 2: Example generation process.

ing data. The entire example generation process is illustrated in Fig. 2.

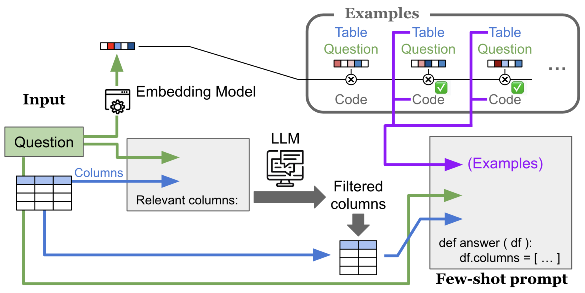

Figure 3: Few-shot prompt for candidate code generation.

3.2 Few-shot prompting

Once example python codes are generated, we perform few-shot prompting on the LLM to generate candidate python codes to answer the input questions. For that, we sample a few examples from the example set for each prompt. To choose the closest examples to the given question, we evaluate 1536-dimensional embedding vectors of the given question and all questions in the example set using `gte-Qwen2-1.5B-instruct` model[2]. Then we select examples based on the cosine similarity between these embedding vectors.

For succinct presentation of tables, we filtered the columns of the given table so that only columns relevant to the given input question can be presented. For this process, we utilized another few-shot prompts, inspired by Talaei et al. (2024), to select relevant columns for the given question. Details of this process is described in Appendix D.

With the chosen examples and the input data, we created few-shot prompt as presented in Appendix C. The over process is summarized in Fig. 3.

3.3 Selection prompt

Although we guide the LLM with our few-shot prompt, there is always some fluctuation of generated results for the stochastic nature of the LLM. To regularize this, we generate multiple candidate python codes for each input question using the prompts in Section 3.2, and run another prompt to select which candidate code is most appropriate to answer the given question. Our selection prompt is inspired by Gao et al. (2024), and a template is presented in Appendix E.

4 Experimental setup

4.1 Hardware

All of our text generation and embedding evaluation on LLMs were performed on a 4$\times$ Quadra RTX 8000 (48GB VRAM) card.

4.2 LLMs used for prompting

`Qwen2.5-72B-Instruct` model[3] was used to run the zero-shot prompts for example generation. `Qwen2.5-coder-32B-Instruct` model[4] was used to run the few-shot prompts for candidate code generation. `Qwen2.5-coder-7B` model[5] was used to run the column filtering prompts for the input questions and the selection prompts. All prompts were run with temperature $T = 0$ unless otherwise noted.

5 Results

We present the results of our experiments in Table 1. All experiments were performed at temperature $T = 0$. Here we report the average performance of 10 repeated experiments. For the challenge competition, we submitted the results from Exp 8.

6 Discussions

To see how effective each component of our approach is, we may compare different experiment results listed in Table 1.

- Comparison of Exp 1 $\sim$ Exp 3 shows that `Qwen2.5-coder-32B-Instruct` performed the best in our setting. This code-specific model outperformed a generic-purposed model with more parameters.
- Comparison of Exp 2 and Exp 4 shows that the table presentation improves the accuracy by the margin of 12.5 %p.

[2]https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct

[3]https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

[4]https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct

[5]https://huggingface.co/Qwen/Qwen2.5-Coder-7B

Exp #	**Experiment**	**Model size**	**Split**	**Accuracy** (%)
1	Zero-shot	7B	dev	61.0
2	Zero-shot	32B	dev	78.8
3	Zero-shot	72B	dev	77.5
4	Zero-shot, without table presentation	32B	dev	66.3
5	Few-shot ($n_{\text{ex}} = 5$)	32B	dev	81.9
6	Few-shot ($n_{\text{ex}} = 5$), without column filtering	32B	dev	81.2
7	Few-shot ($n_{\text{ex}} = 5$) with selection prompt ($n_{\text{cand}} = 3$)	32B	dev	**82.1**
8	Few-shot ($n_{\text{ex}} = 5$) with selection prompt ($n_{\text{cand}} = 3$)	32B	test	**78.35**

Table 1: Experiment results. All accuracies are average value of 10 repeated experiments except Exp 8. Model size 7B / 32B / 72B indicates the model `Qwen2.5-coder-7B` / `Qwen2.5-coder-32B-Instruct` / `Qwen2.5-72B-Instruct`, respectively.

- Comparison of Exp 2 and Exp 6 indicates that the few-shot approach is better than the zero-shot approach by the accuracy margin of 2.4 %p.
- Comparison of Exp 5 and Exp 6 shows that column filtering increases the accuracy by the margin of 0.7 %p.
- Comparison of Exp 5 and Exp 7 implies that selection prompt can slightly boost the accuracy by the margin of 0.2 %p.

n_{ex}	**Accuracy** (%)
3	81.7
4	80.9
5	**81.9**
6	81.6

Table 2: Accuracy of the system for different n_{ex}. All other conditions of the experiment is identical to Exp 4.

n_{cand}	**Accuracy** (%)
2	81.4
3	**82.1**
4	81.6
5	82.0
6	81.8
7	81.9

Table 3: Accuracy of the system for different n_{cand}. All other conditions of the experiment is identical to Exp 5.

Moreover, Table 2 and Table 3 shows how the accuracy of the system changes as the number of few-shot examples (n_{ex}) and the number of candidates for the selection prompt (n_{cand}) change. Based on these experiments, we selected $n_{\text{ex}} = 5$ and $n_{\text{cand}} = 3$ for our best performing system.

Question type	**Exp 2**	**Exp 7**
Boolean	85.2	85.3
Category	86.7	90.6
Category, list	67.2	71.4
Number	80.6	84.2
Number, list	74.2	78.6

Table 4: Accuracy (%) of each question type

To see how our few-shot prompting approach performs on each question type, we compared the accuracies of each question type on Exp 2 and Exp 7. This comparison shows that our approach outperforms the baseline zero-shot approach by the accuracy margin of $3.6 \sim 4.4$%p in all other question types except the boolean type. For the boolean type, our approach shows similar performance (accuracy margin of 0.1%p) with the baseline approach. We believe this behavior is related to how we generate and collect examples for the few-shot prompts. In cases of numerical and text fields, it is very rare to produce the right answer from the example code with wrong logic. However, example codes for the boolean type intrinsically has the danger of having wrong logic while giving the right answer as there are only two possible outcomes (true of false) for this type.

7 Conclusion

We presented a novel few-shot prompting system for tabular data question answering in SemEval 2025 Task 8. Our approach leverages LLM-generated Python code as reasoning guides and incorporates a selection prompt to enhance output quality. This system achieved a 17th-place ranking in the Open Model track and 25th overall. Our analysis highlighted the contributions of individual components and identified key areas for future research, particularly in handling boolean queries.

Limitations

As discussed in Section 6, our method for the generation and collection of example codes shows limited performance on boolean-type questions, and improving this limitation would be an interesting subject for the future research. While our approach entirely relied on the python codes, many other approaches in the literature uses SQL as the means to convey the logic of question answering for the tabular data. To utilize the full capacity of both approaches, one may combine python codes and SQL queries in a single system for better performance. It is also noteworthy that our selection prompt is the simplest zero-shot prompt. Therefore, there is some room to improve this selection process by providing examples, introducing some logical reasoning for selection, or fine-tuning with synthesized training data.

References

Yihan Cao, Shuyi Chen, Ryan Liu, Zhiruo Wang, and Daniel Fried. 2023. Api-assisted code generation for question answering on varied table structures. *arXiv preprint arXiv:2310.14687*.

Xavier Daull, Patrice Bellot, Emmanuel Bruno, Vincent Martin, and Elisabeth Murisasco. 2023. Complex qa and language models hybrid architectures, survey. *arXiv preprint arXiv:2302.09051*.

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*, pages 6491–6501.

Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, et al. 2024. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. *arXiv preprint arXiv:2411.08599*.

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.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. Opentab: Advancing large language models as open-domain table reasoners. *arXiv preprint arXiv:2402.14361*.

Weizheng Lu, Jing Zhang, Ju Fan, Zihao Fu, Yueguo Chen, and Xiaoyong Du. 2024. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*.

Jorge Os'es Grijalba, Luis Alfonso Ure na-L'opez, Eugenio Mart'inez C'amara, and Jose Camacho-Collados. 2025. SemEval-2025 task 8: Question answering over tabular data. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, Vienna, Austria. Association for Computational Linguistics.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *arXiv preprint arXiv:2410.01943*.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 771–782.

Patrick Sutanto and Joan Santoso. 2024. Llm distillation for efficient few-shot multiple choice question answering. *arXiv preprint arXiv:2412.09807*.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint arXiv:2405.16755*.

Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm. *arXiv preprint arXiv:2401.00426*.

Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, et al. 2024. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318*.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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. *Proceedings of the VLDB Endowment*, 17(12):3920–3933.

A Table presentation

In our prompts, we presented sampled tables several times. To demonstrate how we presented them, a sample m-row, n-column table is presented in Prompt 1. Note that any field over 27 characters are truncated up to first 27 characters and followed by "...". Also, any NaN field is replaced to NULL.

```
Input table (df):
[HEAD]: col_1 | ... | col_n
---
[ROW] 1: field_1_1 | ... | field_1_n
...
[ROW] m: field_m_1 | ... | field_m_n
```

Prompt 1: Table presentation

B Zero-shot prompt template

```
import pandas as pd
import numpy as np

"""
For the following input table, write a function
    to answer the question: (question)
(table presentation)
"""

def answer ( df ) -> (question type) :
    """
    Returns: (question)
    """
    df.columns = [(list of columns of the table)]
```

Prompt 2: Zero-shot prompt for code generation. Note that "-> (question type)" part is skipped if question type is not provided.

C Few-shot prompt template

```
import pandas as pd
import numpy as np

(zero-shot prompt from the first example)
(generated code from the first example)

...

(zero-shot prompt from the last example)
(generated code from the last example)

(zero-shot prompt from the input question, table)
```

Prompt 3: Few-shot prompt for code generation. Note that all zero-shot prompts in this template skip the lines importing packages. For the brevity of the example codes, comments added in the same line of actual code are omitted from the generated codes of examples.

D Column filtering

To filter the relevant columns for the input question from the input table, we used the few-shot prompts (Prompt 4) where the randomly sampled training data were used as examples. We set the number of examples to 6. We utilized the fact that the training data contain the information on the columns used to answer each question.

```
### You are a detail-oriented data scientist
    tasked with selecting relevant columns from
    a given database to answer the given
    question.

Database: (database name for table 1)
Columns: [(list of columns of table 1)]
Question: (question 1)
Relevant columns: [
    (used columns to answer question 1)]

...

Database: (database name for table 6)
Columns: [(list of columns of table 6)]
Question: (question 6)
Relevant columns: [
    (used columns to answer question 6)]

Database: (database name for input table)
Columns: [(list of columns of input table)]
Question: (input question)
Relevant columns:
```

Prompt 4: Few-shot prompt for column selection.

To regularize the fluctuating responses from the LLM and the bias of randomly sampled examples, we repeated the experiments 20 times (10 times at temperature 0.0 and 10 times at temperature 0.2) and aggregated all columns selected in the responses.

For column filtering of example data, we directly extracted column names from the generated codes instead of running LLMs for column selection.

In both cases of the example data and the input data, we manually added any column name that contains "id" or "name". We also added any column name that contains any selected column or the other way around.

E Selection prompt template

Prompt 5 is a template of our selection prompt in the case of $n_{\text{cand}} = 4$. Note that we have already excluded examples that cannot be executed without errors before feeding the examples for the prompt.

```
You are a data science expert.
For a given input table below and a question
    regarding this table, there are 4 candidate
    python functions to answer the question.
Your task is to compare these candidates and
    select the correct and reasonable candidate
    to answer the question.

(table presentation)

Question: (question)

[Candidate A]
import numpy as np
import pandas as pd
def candidate_A_solution(df: pd.DataFrame):
(python code for candidate 1)

...

[Candidate D]
import numpy as np
import pandas as pd
def candidate_D_solution(df: pd.DataFrame):
(python code for candidate 4)

Please output the selected candidate as "A" or "
    B" or "C" or "D".

Selected canddiate:
```

Prompt 5: Selection prompt. In this template prompt, number of candiates are set to 4.