TableGPT: Towards Unifying Tables, Nature Language and Commands into One GPT

Liangyu Zha[1,2]   **Junlin Zhou**[1,2]   **Liyao Li**[1,2]   **Rui Wang**[1,2]   **Qingyi Huang**[3]
Saisai Yang[3]   **Jing Yuan**[3]   **Changbao Su**[3]   **Xiang Li**[3]   **Aofeng Su**[3]   **Tao Zhang**[3]
Chen Zhou[3]   **Kaizhe Shou**   **Miao Wang**   **Wufang Zhu**   **Guoshan Lu**   **Chao Ye**
Yali Ye   **Wentao Ye**   **Yiming Zhang**   **Xinglong Deng**   **Jie Xu**
Haobo Wang[4]   **Gang Chen**[4]   **Junbo Zhao**[4]
[1]directional lead   [2]joint first author   [3]equal contribution   [4]project lead

Zhejiang University

Abstract

Tables are prevalent in real-world databases, requiring significant time and effort for humans to analyze and manipulate. The advancements in large language models (LLMs) have made it possible to interact with tables using natural language input, bringing this capability closer to reality. In this paper, we present TableGPT, a unified fine-tuned framework that enables LLMs to understand and operate on tables using external functional commands. It introduces the capability to seamlessly interact with tables, enabling a wide range of functionalities such as question answering, data manipulation (e.g., insert, delete, query, and modify operations), data visualization, analysis report generation, and automated prediction. TableGPT aims to provide convenience and accessibility to users by empowering them to effortlessly leverage tabular data. At the core of TableGPT lies the novel concept of global tabular representations, which empowers LLMs to gain a comprehensive understanding of the entire table beyond meta-information. By jointly training LLMs on both table and text modalities, TableGPT achieves a deep understanding of tabular data and the ability to perform complex operations on tables through chain-of-command instructions. Importantly, TableGPT offers the advantage of being a self-contained system rather than relying on external API interfaces. Moreover, it supports efficient data process flow, query rejection (when appropriate) and private deployment, enabling faster domain data fine-tuning and ensuring data privacy, which enhances the framework's adaptability to specific use cases.

1   Introduction

The vast and intricate world of data is often encapsulated in tables, being a foundation for data-driven decision-making in a wide spectrum of applications, including financial analysis, supply chain management, and healthcare analytics. It enables stakeholders to analyze trends, patterns, and relationships, leading to informed business decisions, process improvements, and resource optimization. For years, data scientists have struggled to process tables using complicated Excel formulas or handcrafted programming. Consequently, there has been an urgent need to understand and interpret tabular data in a more efficient fashion.

In the field of natural language processing, Generative Pre-trained Transformers (GPTs) [21, 22, 2, 19, 18] or Large Language Models (LLMs) [4, 32, 24, 33] have **revolutionized** the paradigm of language data mining. Following this line of works, researchers have also explored large models for various modalities like vision [6, 12], and speech [8]. From a technical standpoint, their ability to generate

Table 1: Comparisons with previous command-using LLMs for tabular data. (See details in Sec 3.2)

Methods / Properties	ChatExcel [25]	SheetCopilot [16]	Data-Copilot [34]	TableGPT (ours)
Nature Language Operations	✓	✓	✓	✓
Visualization	✗	✓	✓	✓
Analysis & Report	✗	✗	✓	✓
Prediction	✗	✗	✓	✓
Chain-of-command	✗	✗	✓	✓
Base Model	Unknown	API	API	Fine-tuned
Vague Input Rejection	✗	✗	✗	✓
Private Deployment	✗	✗	✗	✓

human-like text has opened new vistas of possibilities for processing tabular data. Nevertheless, it is non-trivial to directly employ the vanilla ChatGPT [18] model in the tabular area for two reasons: (i)-**Global Table Understanding**: the GPTs are known to suffer from the limited token length and thus, they can not read a whole large table, making them hard to understand the global tabular information. (ii)-**Generalized to Tabular Domain:** Second, their training processes are tailored for natural languages and thus, they are less generalizable when handling tabular data.

There have been several works [35, 17, 16] developed to integrate natural language for tabular data analysis. NL2SQL (Nature language to SQL) [35, 17] is a long-standing research topic that converts natural language to SQL commands that manipulate the relational database. Recently, SheetCopilot [16] explored languages to VBA (Visual Basic for Applications, an embedded script language for Microsoft Excel) command such that benefit from a rich set of spreadsheet software functionalities. However, we found that both solutions demonstrate unsatisfactory performance. We speculate that these forms of programming code, which is fundamentally unstructured, adds another layer of complexity, making automated post-processing almost insurmountable.

In this work, we develop TableGPT that pushes the boundaries of what is possible in data analysis empowered by LLM techniques, marking an important step forward in our pursuit of making data more accessible and understandable. Our TableGPT framework unifies tables, natural language, and commands into a single GPT model, making data interpretation and manipulation more intuitive and user-friendly. By rethinking the interaction of tables, natural language, and commands, we integrate several core components into TableGPT:

- **Global Table Representation:** We make the first attempt to develop a global representation learning paradigm for tables that encodes the whole table into one vector. By jointly training the LLM and a table encoder on vast amounts of text and table data, we equip the encoder to adequately capture the global information in the input table. This enables the LLM to perceive and understand the table data effectively, thereby providing a more global and enhanced comprehension of tables.

- **Chain-of-Command:** We introduce this concept to emphasize the essential idea of a structured and hierarchical execution of tasks. Just like a well-coordinated organization where each directive is cascaded from a higher level to its lower counterpart, TableGPT follows a similar chain of commands, breaking down complex tasks into simpler ones and executing them step-by-step. Moreover, it fosters the ability to refuse ambiguous or inappropriate commands, much like an actual data scientist, instead of blindly following any potential erroneous instruction, thereby improving the interaction between humans and LLM systems in the field of data science. Our proposed command set is not only easier to control but also reduces the uncertainty that often accompanies traditional methods of handling table data.

- **Domain-aware Fine-Tuning:** To foster the ability to adapt to specific domains of tables and corresponding textual materials, domain-aware fine-tuning hinges on customizing training in a way that the model generates text embodying similar stylistic and logical elements found in a given domain, thereby augmenting its understanding of specific domain table data. To make this approach scalable and feasible, we have also developed a data processing pipeline that yields notable improvements with only a small amount of data,
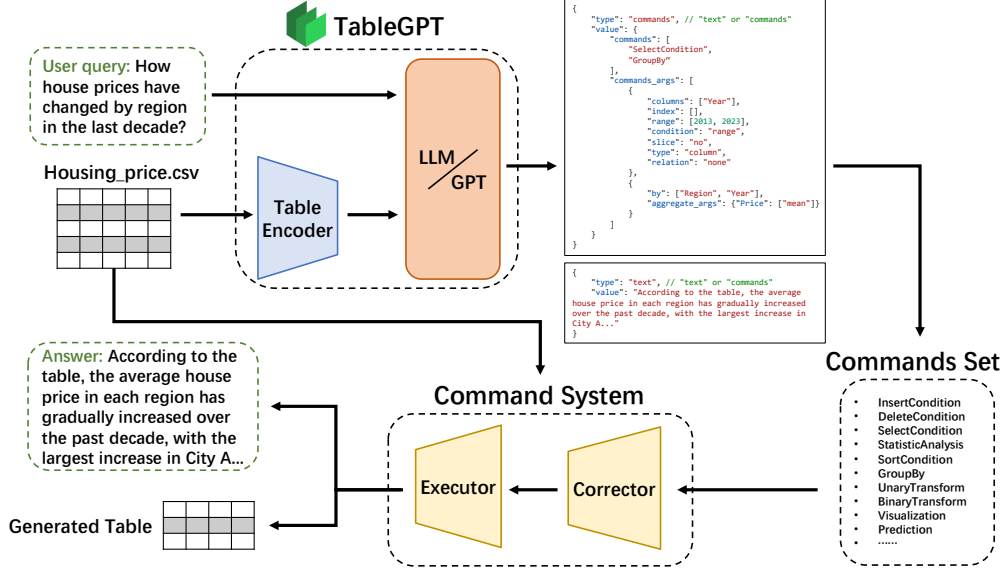
Figure 1: An architecture of TableGPT framework.

hence alleviating the resource-demanding aspect of training LLMs and supporting private deployment.

From a real-world production standpoint, the unstructured code outputted by NL2SQL poses significant challenges for preemptive checks and error corrections. Hence, we advocate for the use of structured command sequences, simplifying post-processing. Data-Copilot [34] also embraces this command-based approach with self-instruct [28], but its reliance on API-called native LLMs to comprehend tabular data's processing and analysis logic directly presents limitations. Given the intrinsic data variability and task-specificity of tabular data, we believe an effective product should be custom-built for tabular data while maintaining general applicability to broader downstream tasks. This conviction underscores the imperative of introducing a LLM specifically pre-trained for tabular data.

To sum up, this work presents a pioneering TableGPT framework, which is a unified, well-fledged holistic solution, enabling efficient tabular data processing, analysis and visualization, driven all by natural languages. We summarize several important advantages of TableGPT as follows:

- **Language-driven EDA:** TableGPT understands user intent from natural language, dissects the desired actions, and executes external commands on the table. It subsequently returns the processed results in both tabular and textual explanations to the user. This novel approach simplifies the way users engage with table data, bringing an intuitive instantiation to Exploratory Data Analysis (EDA).

- **Unified Cross-modal Framework:** Innovatively, we devise a global table encoder for understanding the whole table. TableGPT is able to fully understand the user query, meta-knowledge, and whole tabular data, which leads to much more reliable execution commands for table manipulation.

- **Generalization and Privacy:** By domain-aware fine-tuning, our TableGPT can better handle data variability of tables and generalize to different domains. Further, our framework supports private deployment, offering robust data privacy protections. This aspect is critical in the modern age where data privacy and protection are just paramount.

3

2 TableGPT

2.1 Model Design

The development of TableGPT begins with the foundation provided by pre-trained LLMs. The advancements in the field of natural language processing have led to the development of a number of exceptional open-source LLMs, such as LLaMa [24], Phoenix [4], ChatGLM [32], Ziya [9], and Baichuan [11]. In designing TableGPT, we opted to use Phoenix [4] with 7B parameters as our base model for fine-tuning, owing to its excellent capabilities in handling both Chinese and English languages. This choice is not, however, exclusive. Our model design supports adaptation with other LLMs, providing versatility and flexibility in its implementation.

What sets TableGPT apart from its predecessors [25, 16, 34] is the novel approach to its fine-tuning process. We performed the fine-tuning on a vast corpus, comprising 2T tokens of textual data and 0.3M tables. This corpus offers a diverse landscape for the model to learn from, including but not limited to user query-command sequence pairs and publicly available domain-specific data for table analysis reports.

The overall architecture of TableGPT is shown in Figure 1. When a user inputs a table and a query, these are received by TableGPT, which consists of a table encoder and an LLM. The table encoder serves to extract vector representations from the input table. These representations, coupled with the text query, are then fed into the LLM for inference. The LLM discerns the user's query intent and generates an output that includes both a command sequence and a textual reply. The command sequence undergoes error correction in the command system's corrector before it is fed into the executor for execution. The final output, provided to the user, includes the manipulated table and a textual reply. This streamlined process delivers efficient, reliable responses to table data queries, enhancing user experience and simplifying data analysis.

2.2 Global Representation of Table

The rapid development of large language models (LLMs) has seen them interfacing with a multitude of modalities such as vision, and audio. For instance, the integration of vision and LLMs has led to models like CLIP [20] (Contrastive Language–Image Pretraining) from OpenAI that connects images and text through shared latent space. The combination of audio and LLMs gave rise to models like Wave2Vec [1] and Tacotron [29] that employ the representation of audio in the form of spectrograms to generate or understand speech.

Despite these advancements, the exploration of LLMs interfacing with tabular data remains limited. The question of how to enable LLMs to comprehend and interpret tables is essential. Some studies have attempted to convert sample rows of table data directly into a sentence-like text description [7], while others have attempted to artificially define a global representation of table data through the template-based extraction of column names, industry background, and other metadata schema [34]. However, these approaches only extract partial information from table data for LLMs, consequently overlooking the global information and industry background inherent in the data.

Notably, for the tables, it is required to embed the whole table into one single vector, instead of generating sample-wise embedding. This can be non-trivial and challenging because, unlike images, videos, and audio, table data is inherently a highly abstract structured data type. Furthermore, it possesses a dual permutation invariance structure where shuffling rows or columns does not affect the information contained within the table, a distinct contrast to images and audio, which carry inductive bias in adjacent positions or sequences. Moreover, tables from different domains vary in size and format, such as having different numbers of discrete and continuous columns, making it challenging to extract features from diverse tables using a unified neural network architecture [31].

Yet, it remains an open problem to effectively extract global representations from tables for LLMs to achieve comprehensive table understanding. To this end, we present a Cascaded Table Encoder that jointly extracts knowledge from metadata and whole numerical entries.

Cascaded Table Encoder. Consider the approach of an experienced data scientist encountering a table. They typically examine the structure of the table data, such as the table headers and distribution of feature columns, to understand the meaning of different cells based on their position, without focusing too much on the numeric information of each cell. Following this biologically plausible

4

approach, we propose a novel cascading table encoder. It divides the information in the table data into two main parts. The first part learns the metadata representation of the table, such as schema, industry background, and the meanings of column names, which can help LLMs understand the global information of the table structure. The second part learns the numerical information representation of the table, such as the distribution and trends of values in different columns, helping LLMs understand the global information of the table numbers like human experts.

We consider the rows and columns of the table as elements of a set and learn the overall representation of the entire set. We use a modified set transformer [15] as the backbone of the table encoder. The set transformer [15], originally designed for dealing with permutation invariant problems, aligns well with the inherent structure of tabular data. We enhance it with an attention mechanism [26] that can capture the interdependencies between different rows or columns of the table, enabling the model to understand the relations between different parts of the table data.

This encoder is pre-trained on ten thousand table datasets using a masked table modeling approach, similar to the masked language modeling used in BERT [5] but adapted to tabular data. The learned table representation not only can be used for table understanding but also can enhance the predictive performance of downstream classifiers.

Our proposed method presents a significant step forward in the integration of tables, natural language, and commands into LLMs. It provides a comprehensive approach for extracting global representations from tables and enables LLMs to understand and manipulate.

2.3   Chain-of-Command

In recognition of the fact that Large Language Models (LLMs) like GPT can struggle with numerical reasoning, prone to computational errors and hallucinations [10], our approach does not require them to operate and calculate within the tables in their latent space. Instead, we provide a series of pre-packaged function commands for LLMs to call upon. LLMs, understanding the global representation of the table and user input, generate a sequence of commands for the backend system to execute, resulting in a modified table. Compared to the SQL statements generated by text2SQL [35, 17], these command sequences are more easily examined and error-located by the backend parsing system, while SQL statements can be challenging to diagnose and correct for specific errors.

However, user queries are often vague and complex, and we can only encapsulate and provide some basic table operation commands. Teaching the LLM to deconstruct complex and vague queries is crucial. For example, a user's query for a specified object column could be a synonym or translation of a column in the original table, or the user may only have a vague intent and cannot express the demand clearly.

The Chain-of-thought [13, 30] approach emphasizes breaking down complex reasoning into a series of intermediate steps. We introduce the concept of Chain-of-command (CoC), an approach that enhances the chain-of-thought by providing a mechanism for step-by-step instructions associated with these intermediate steps. For instance, when a user asks, "Show me the five movies with the highest profit margin," the LLM first checks if a profit margin column exists in the table. If not, it generates arithmetic instructions to calculate the profit margin using box office and cost data; next, it executes instructions to sort by profit margin in descending order and slice to select the top five movies. When user queries are too vague, like "Give me some numbers," the LLM might struggle to decompose and could refuse execution, instead, it would ask the user for more specific intent.

The aim of the Chain-of-command is to enhance LLM's reasoning capabilities and robustness when operating table data. This approach involves translating user inputs into a sequence of intermediate command operations, enabling LLMs to manipulate tables more accurately and efficiently symbolically. The ability to manipulate symbolic instructions is particularly valuable for real-world applications involving complex and accurate interactions with historical data, such as record-keeping and data analysis in management environments.

To enhance the performance and stability of our approach, we constructed a substantial dataset of command chain instructions while fine-tuning LLMs to adapt to commands, and employed contextual learning to provide prompts for multiple steps in the command chain sequence. A strong and accurate command chain process allows LLMs to better reason about table data and handle more complex scenarios.

The Chain-of-command approach has three main advantages. First, it enables LLMs to execute complex table instructions accurately, thereby enhancing their multi-hop reasoning capabilities for table operations. Second, by breaking down complex operations into a series of intermediate table operations, the chain-of-command method enhances the LLM's ability to handle complex multi-table interactions. Lastly, it enables LLMs to refuse overly vague instructions and ask users for more specific intent. This approach allows LLMs to handle edge cases and unexpected scenarios better, making it a promising method for real-world applications.

2.4 Domain Data Processing Pipeline

Despite the broad knowledge and dialogue capabilities of large language models (LLMs) due to extensive pre-training on a diverse corpus, their performance often falls short in addressing the nuanced language styles and logic of specific industries. This is primarily due to the lack of exposure to proprietary domain data during their training phase. To mitigate this issue, we have developed an efficient domain data processing pipeline [3].

Motivated by the goal to streamline the fine-tuning process of LLMs with minimal computational overhead and accelerated model iteration, our pipeline is designed to harness the power of active learning [23]. Through this, we curate a select set of fine-tuning examples from the domain data, allowing LLMs to achieve superior fine-tuning results with a reduced number of examples. This strategic utilization of resources expedites the model's learning process, thereby speeding up its iteration.

Additionally, we have fortified the document retrieval capabilities of LLMs. We utilize technologies like vector databases [27] and LangChain [14] to facilitate the retrieval of pertinent information from a plethora of proprietary documents, further enriching the context that LLMs learn from.

In essence, our pipeline serves as a catalyst for the rapid and cost-effective adaptation of LLMs to the data needs of various specific industries. This pipeline not only addresses the challenges of industry-specific language styles and logic but also empowers LLMs to handle commands that interact with tables, integrating the realms of natural language, tables, and commands.

3 Evaluation

3.1 Commands supported by TableGPT

To unleash the power of TableGPT, we have designed and supported a rich set of commands. Firstly, TableGPT enables natural language interaction with tables, empowering users to intuitively query, filter, sort, and aggregate data using everyday language. It also facilitates tasks such as data visualization and report generation, enhancing the interpretability and presentation of tabular information. Lastly, TableGPT facilitates automated decision-making processes, empowering users to make predictions, forecast trends, and estimate outcomes using table data and natural language instructions.

Note that when the intent of the user query is too vague, TableGPT will reject to generate commands and instead ask the user for more detailed intent. This is one of the benefits of chain-of-command, the ability to think about the rationality of commands like a human expert, rather than a rigid command translator.

3.2 Comparison with previous command-using LLMs

Several existing solutions attempt to combine tables and language models, such as ChatExcel [25], SheetCopilot [16], and Data-Copilot [34]. These approaches typically rely on using prompts to invoke pre-defined external commands through inference API of LLMs, such as OpenAI API[1]. In contrast, TableGPT takes a different approach by fine-tuning LLM specifically for table-related tasks. This key distinction allows us to harness the inherent capabilities of the LLM architecture while tailoring it to excel in table processing tasks. A detailed comparison of TableGPT with the previous command-using LLMs is shown in Table 1.

---

[1]https://openai.com/blog/openai-api

3.3 Case Study

We show some cases in Figure 2 - 8. More examples will be released soon.

4 Conclusion

We present TableGPT, a large language model designed for table analysis, unifying tables, nature language, and commands. It enables a variety of functions like answering questions, manipulating data, visualizing information, generating analysis reports, and making predictions. Technically, TableGPT addresses several major challenges in developing a natural language-driven framework for table data processing, including comprehensive table understanding, instruction chain generation, and domain-specific fine-tuning. We believe TableGPT has the potential to reshape the landscape of tabular data processing, accelerating the efficiency of table modeling and exploratory data analysis (EDA), and empowering various domains like finance, transportation, scientific research, etc.
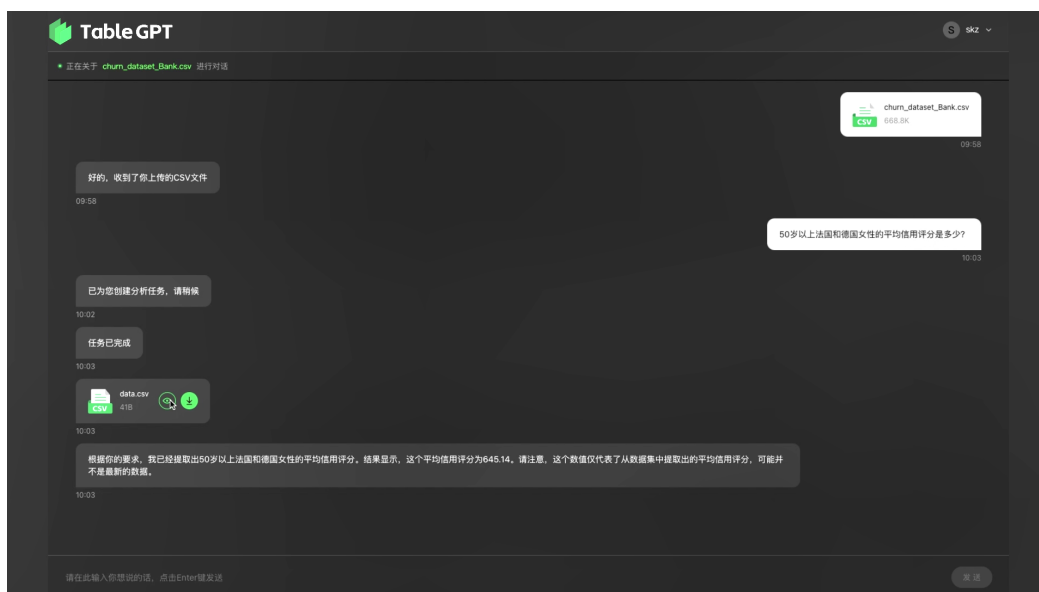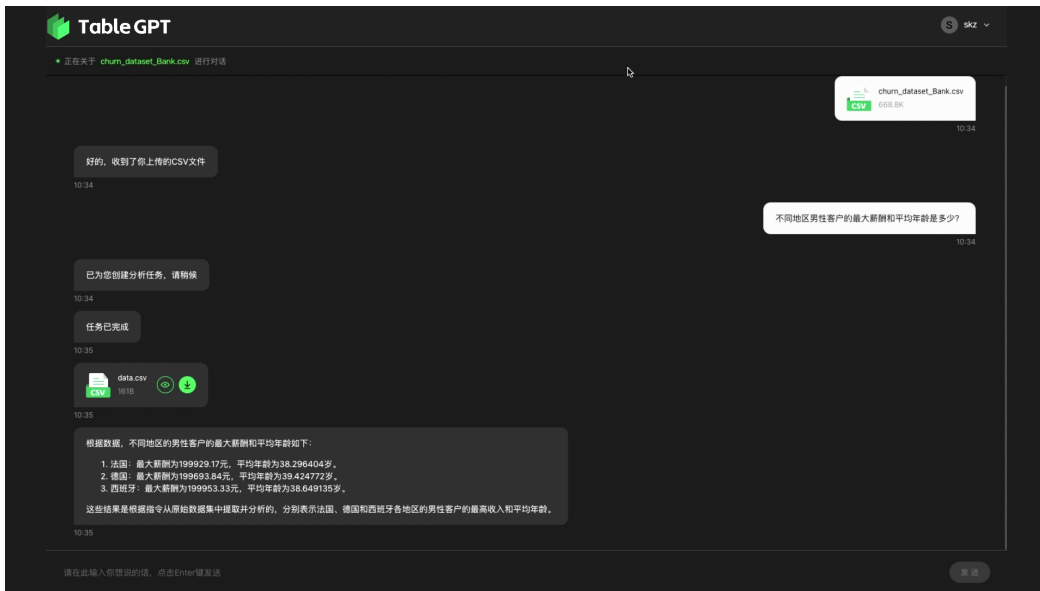

Figure 2: Cases of TableGPT.
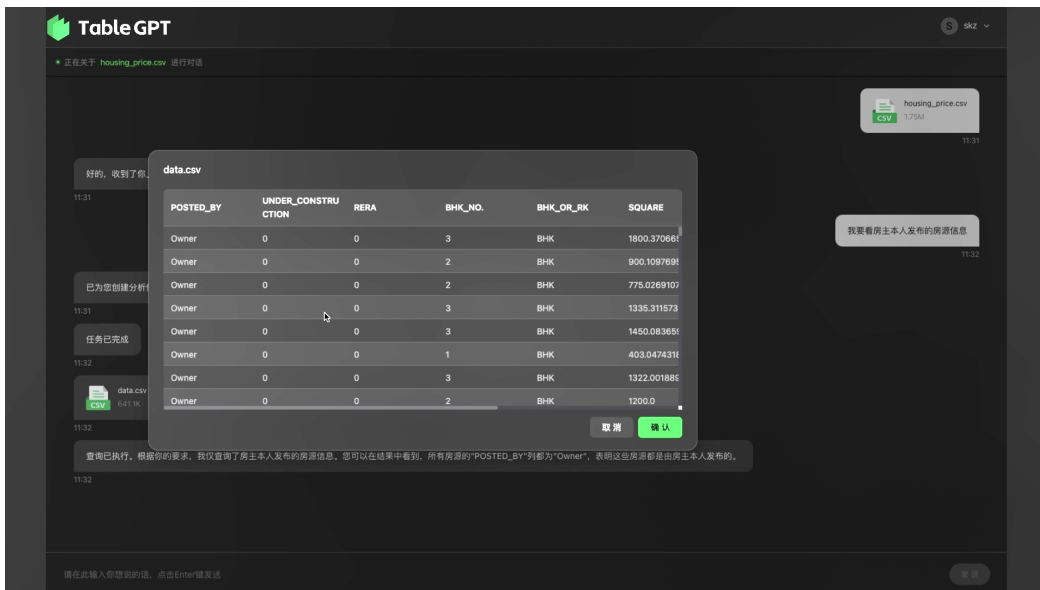
Figure 3: Cases of TableGPT.

Figure 4: Cases of TableGPT.
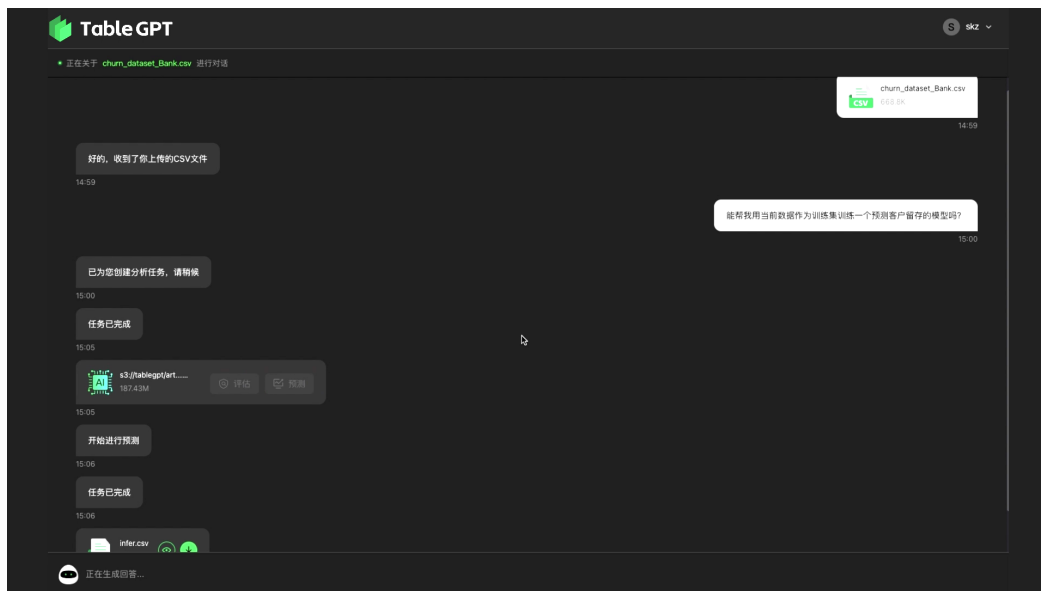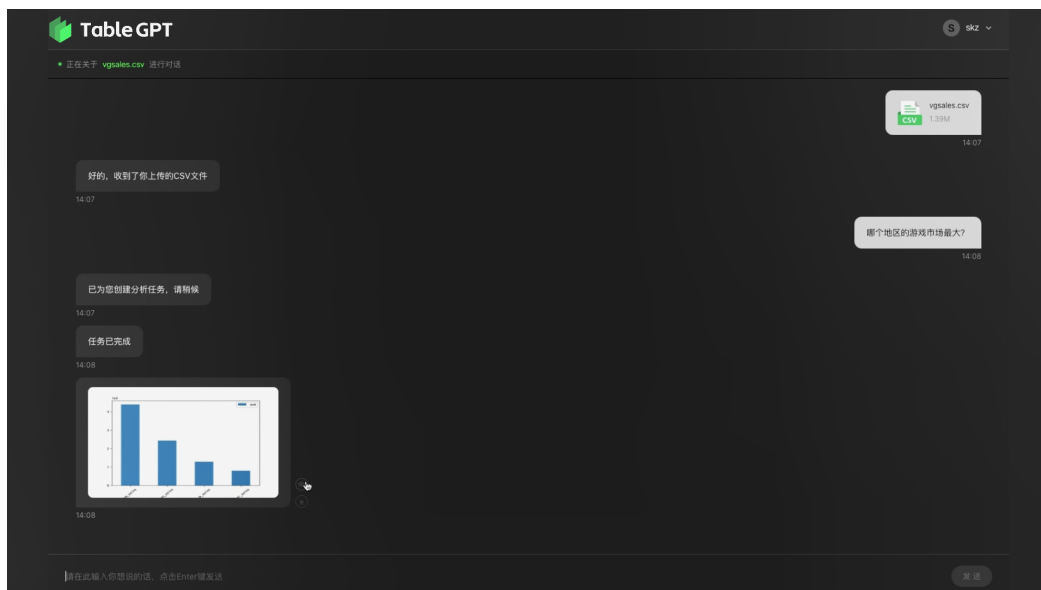
Figure 5: Cases of TableGPT.
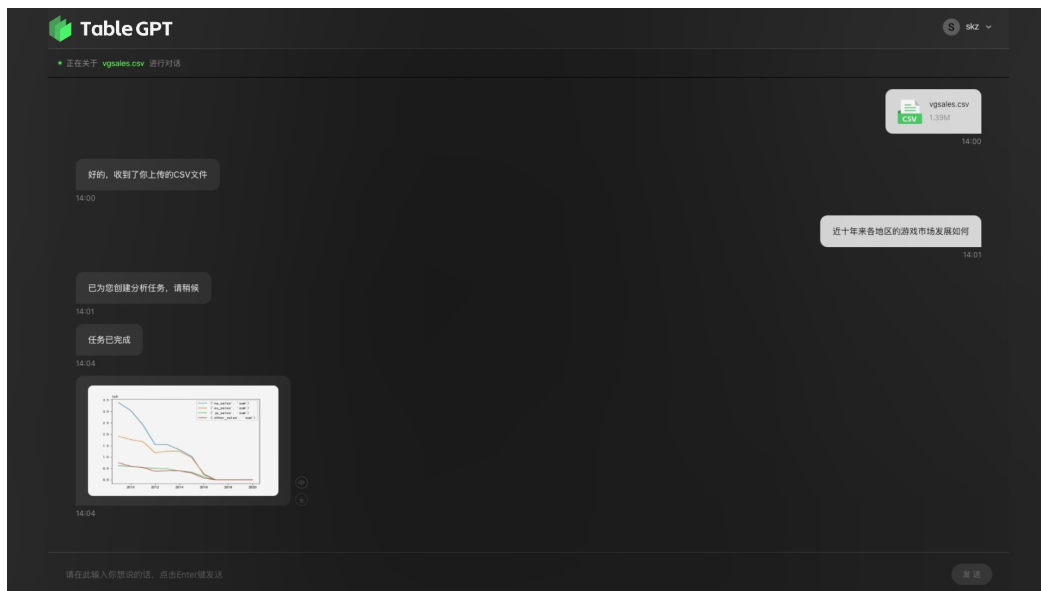

Figure 6: Cases of TableGPT.
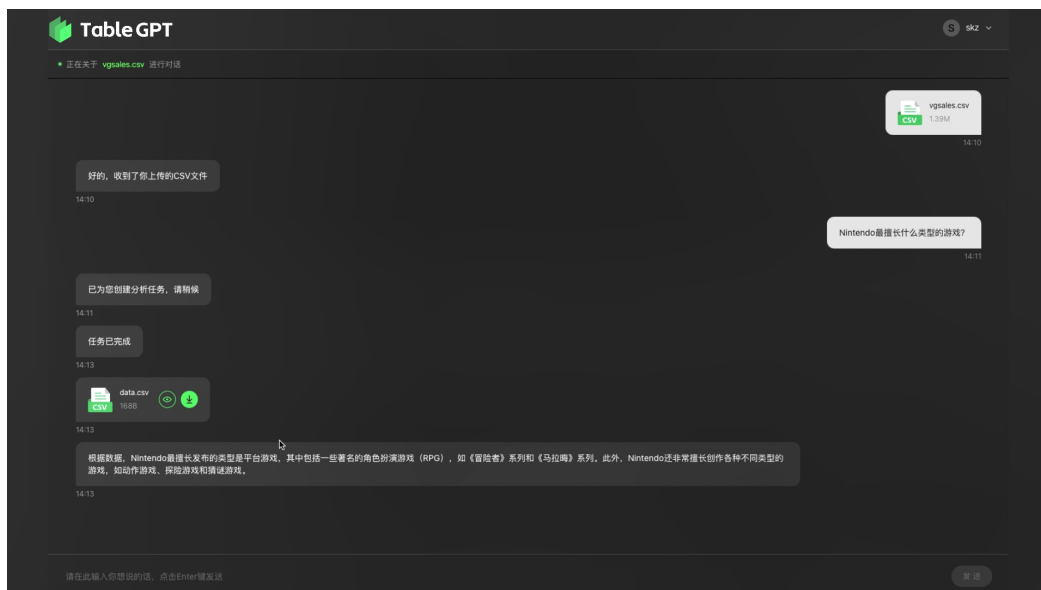
Figure 7: Cases of TableGPT.

Figure 8: Cases of TableGPT.

References

[1] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning, 2023.

[4] Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, et al. Phoenix: Democratizing chatgpt across languages. *arXiv preprint arXiv:2304.10453*, 2023.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[6] Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qian Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. Multimodal-gpt: A vision and language model for dialogue with humans. *arXiv preprint arXiv:2305.04790*, 2023.

[7] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[8] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*, 2023.

[9] IDEA-CCNL. Fengshenbang-lm. `https://github.com/IDEA-CCNL/Fengshenbang-LM`, 2023.

[10] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models, 2023.

[11] Baichuan Intelligence. Baichuan-7b. `https://github.com/baichuan-inc/baichuan-7B`, 2023.

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[13] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[14] LangChain. Langchain. `https://blog.langchain.dev/`, 2022.

[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[16] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. Sheetcopilot: Bringing software productivity to the next level through large language models. *arXiv preprint arXiv:2305.19308*, 2023.

[17] Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *arXiv preprint arXiv:2301.07507*, 2023.

[18] OpenAI. Chatgpt. `https://openai.com/blog/chatgpt`, 2022.

[19] OpenAI. Gpt-4 technical report, 2023.

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[21] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[23] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2021.

[24] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[25] Peking University. Chatexcel. `https://chatexcel.com/`, 2023.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[28] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[29] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis, 2017.

[30] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[31] Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. Ct-bert: Learning better tabular representations through cross-table pre-training. *arXiv preprint arXiv:2307.04308*, 2023.

[32] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*, 2022.

[33] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[34] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. *arXiv preprint arXiv:2306.07209*, 2023.

[35] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.