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. With the advancements in large language models (LLMs), the ability to interact with tables through natural language input has become a reality. In this paper, we present TableGPT, a unified fine-tuned framework that enables LLMs to understand and operate on tables using external function 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 tabular representations, which are vectorized representations of tables. This is the first successful attempt to extract vector representations from tables and incorporate them into LLMs. 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 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. Yet, interpreting and understanding these tables can pose a significant challenge. For years, people have struggled to find ways to decode the information in tables without being entangled in complicated Excel formulas or the inscrutable labyrinth of code. Two significant perspectives motivate us to address these challenges: technical advancement and real-world application.

From a technical standpoint, Generative Pre-trained Transformers (GPTs) [17, 18, 5, 15, 1] or Large Language Models (LLMs) [6, 21, 19, 22] have revolutionized the landscape of natural language processing. Their ability to generate human-like text has opened new vistas of possibilities. However, despite the commendable progress, there are certain areas where GPTs fail to measure up. Specifically, their ability to effectively read and interpret tables [7, 13, 23]. To illustrate, consider a table featuring multiple layers of information, or one that requires the understanding of inter-cell relationships to

Technical report preprint. Work in progress.

extract the intended knowledge. GPTs often falter in such complex scenarios due to their inherent limitations. The token number limit often results in an incomplete understanding of larger tables, while their training data and objectives do not explicitly include effective table reading and comprehension.

Moreover, the realm of multimodality has attracted significant research, with most efforts focusing on the integration of vision [8, 12], speech [10], and natural language. Regrettably, tables, which serve as a significant data representation modality, have been conspicuously overlooked.

The second motivating aspect pertains to real-world production. Current workflows are plagued by a high degree of uncertainty. Traditional chain-of-command structures are often ill-equipped to handle the complex demands of data analysis in tables. In such cases, solutions like NL2SQL [24, 14] and NL2VBA [] have been proposed, aiming to convert natural language into SQL or VBA commands, respectively. Unfortunately, both solutions have their drawbacks and have not proven to be universally effective []. Code, which is fundamentally unstructured, adds another layer of complexity, making post-processing a challenging task.

This paper introduces TableGPT, our attempt to tackle these challenges head-on. We seek to unify tables, natural language, and commands into a single model, making data interpretation and manipulation more intuitive and user-friendly. Our proposed command set is not only easier to control but also reduces the uncertainty that often accompanies traditional methods of handling table data.

By rethinking the interaction of tables, natural language, and commands, TableGPT is designed to push the boundaries of what is possible in data analysis, marking an important step forward in our pursuit of making data more accessible and understandable. To sum up, the main contributions are listed as follows.

1. We propose a novel fine-tuned LLM, TableGPT, specifically designed for table analysis. By unifying tables, natural language, and commands into one model, TableGPT comprehends tabular data, understands user intent through 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 feel to data analysis.
2. For the first time, we present a vector representation learning for tables, specifically created for an LLM. 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 holistic and enhanced comprehension of tables.
3. We construct an efficient framework for domain data fine-tuning. It allows for adaptation of various pre-trained LLMs using minimal domain data to cater to different scenarios. This is particularly beneficial in real-world applications where access to large amounts of specific domain data might be challenging. Further, our framework supports private deployment, offering robust data privacy protections. This aspect is critical in today's age where data privacy and protection are paramount.

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 [19], Phoenix [6], ChatGLM [21], Ziya [11], and Baichuan [2]. In designing TableGPT, we opted to use Phoenix [6] 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 [3, 13, 23] 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.

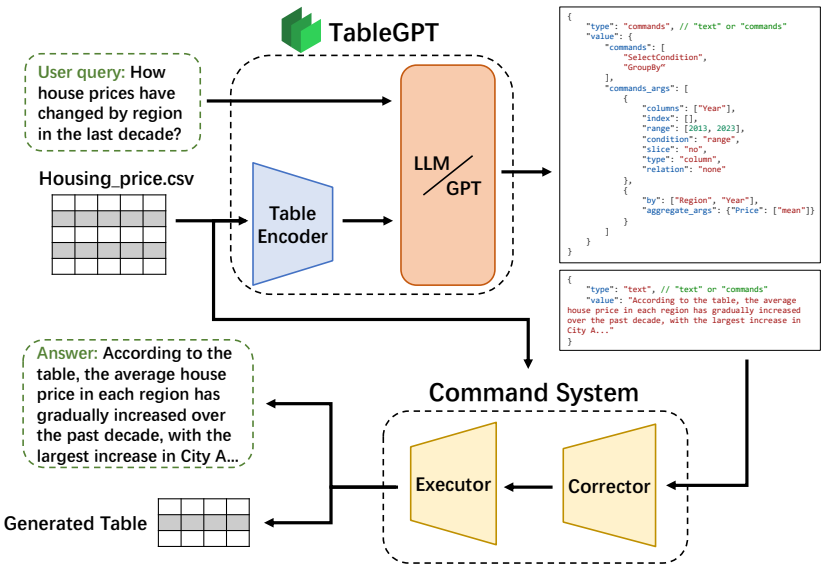

Figure 1: An architecture of TableGPT framework.

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 a 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 Vectorized 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 [16] (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 [4] and Tacotron [20] 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 (e.g., TabLLM [9]), 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 (e.g., Data-Copilot [23]). However, these approaches only extract partial information from table data for LLMs, consequently overlooking the holistic information and industry background inherent in the data.

Different from other modalities that ultimately extract vector representations for interfacing with LLMs, the effective extraction of vector representations from tables for LLMs to achieve table understanding remains an open question.

The extraction of vectorized representations from tables is 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 (as seen in XTab).

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 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 as the backbone of the table encoder. The set transformer, originally designed for dealing with permutation invariant problems, aligns well with the inherent structure of tabular data. We enhance it with an attention mechanism 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 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 vectorized 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, 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 vector 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, 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 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.

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. 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 and LangChain 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, SheetCopilot, and Data-Copilot. These approaches typically rely on using prompts to invoke pre-defined external commands through inference API of LLMs, such as OpenAI API. 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.

Table 1: Comparison with previous Tool-using LLMs

Methods	ChatExcel	SheetCopilot	Data-Copilot	**TableGPT** (ours)
Nature Language Operations	✓	✓	✓	✓
Visualization	✗	✓	✓	✓
Analysis & Report	✗	✗	✓	✓
Prediction	✗	✗	✓	✓
Chain-of-command	✗	✗	✓	✓
Base Model	Unknown	API	API	Fine-tuned
Rejection	✗	✗	✗	✓
Private Deployment	✗	✗	✗	✓

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. This model is the first attempt to capture vector representations of tables in LLM. It brings a more comprehensive understanding and analysis capability of tables.

Figure 2: Cases of TableGPT.

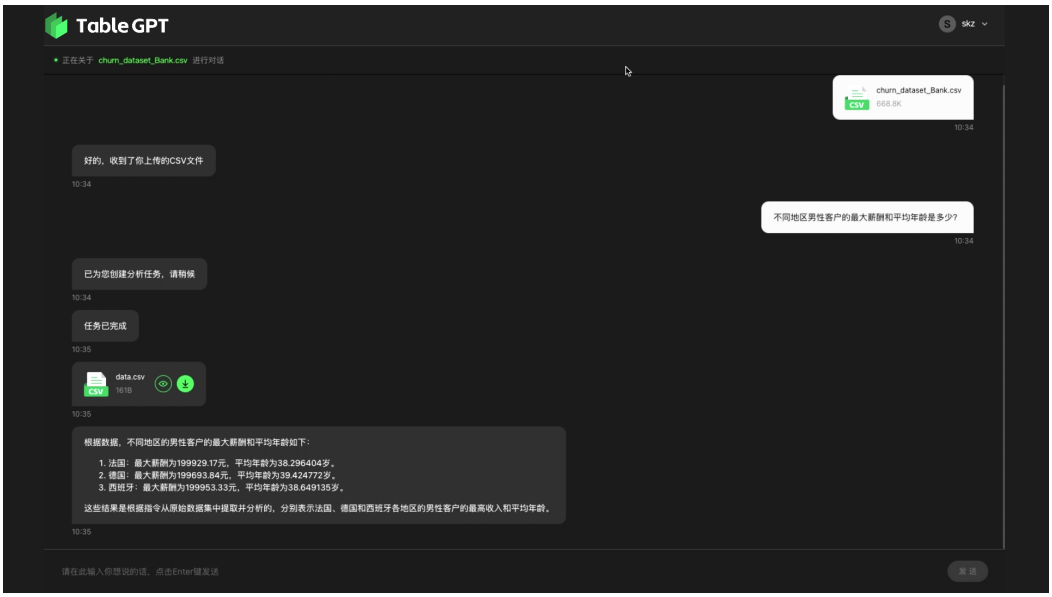

Figure 3: Cases of TableGPT.

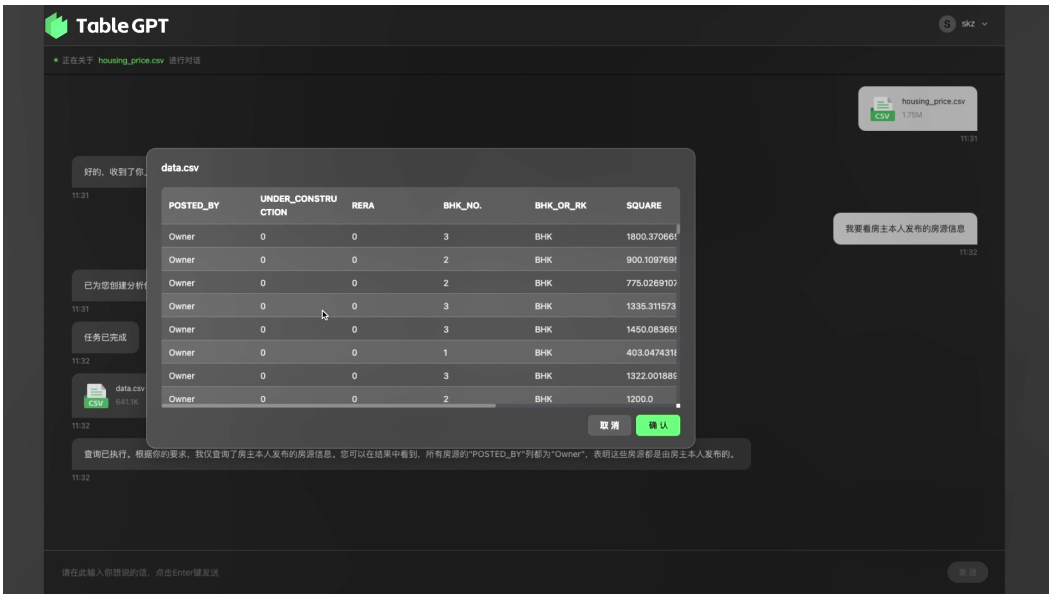

Figure 4: Cases of TableGPT.

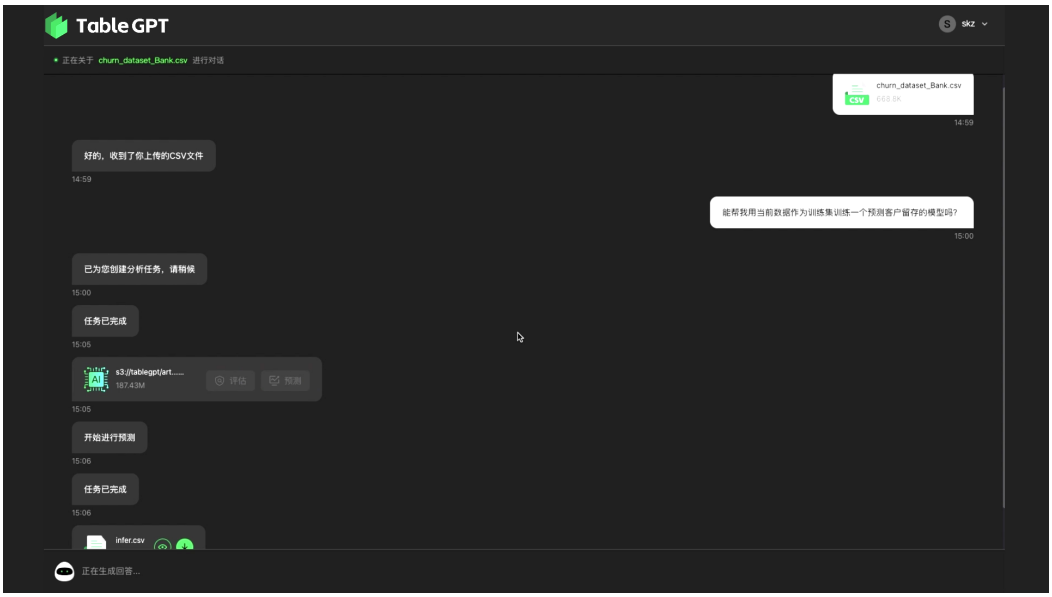

Figure 5: Cases of TableGPT.

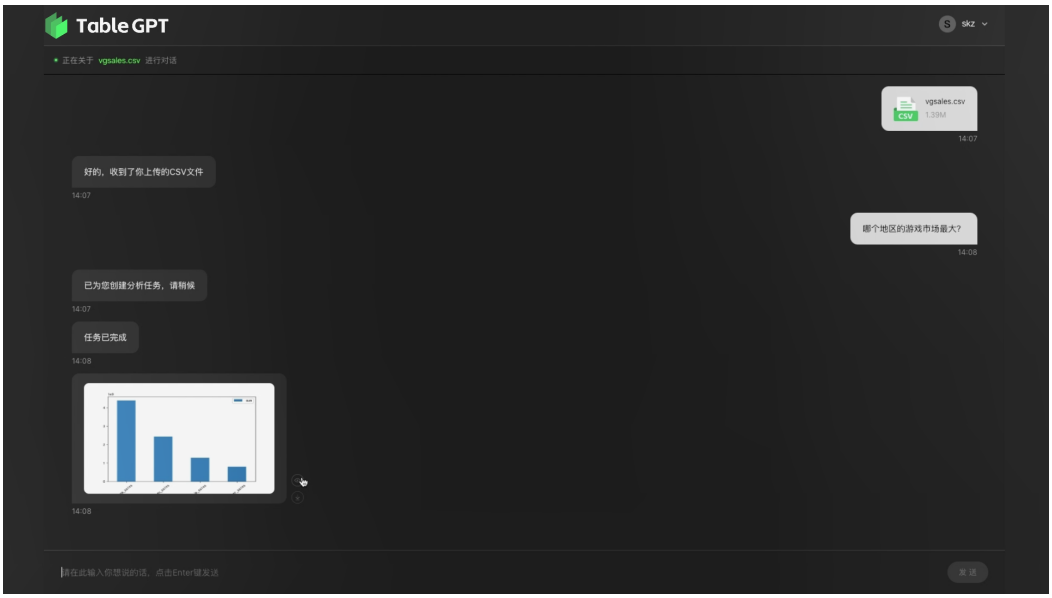

Figure 6: Cases of TableGPT.

Figure 7: Cases of TableGPT.

Figure 8: Cases of TableGPT.

References

[1] https://openai.com/blog/chatgpt, 2022.

[2] https://github.com/baichuan-inc/baichuan-7B, 2023.

[3] https://chatexcel.com/, 2023.

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

[5] 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.

[6] 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.

[7] Liying Cheng, Xingxuan Li, and Lidong Bing. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*, 2023.

[8] 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.

[9] 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.

[10] 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.

[11] IDEA-CCNL. Fengshenbang-lm. https://github.com/IDEA-CCNL/Fengshenbang-LM, 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] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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