

TableBench: A Comprehensive and Complex Benchmark for Table Question Answering

Xianjie Wu¹, Jian Yang^{1*}, Linzheng Chai¹, Ge Zhang², Jiaheng Liu¹, Xeron Du², Di Liang³, Daixin Shu¹, Xianfu Cheng¹, Tianzhen Sun¹, Tongliang Li⁴, Zhoujun Li^{1*}, Guanglin Niu¹

¹Beihang University

²M-A-P

³Fudan University

⁴Beijing Information Science and Technology University
{wuxianjie, jiaya, lizj}@buaa.edu.cn

Abstract

Recent advancements in large language models (LLMs) have markedly enhanced the interpretation and processing of tabular data, introducing previously unimaginable capabilities. Despite these achievements, LLMs still encounter significant challenges when applied in industrial scenarios, particularly due to the increased complexity of reasoning required with real-world tabular data, underscoring a notable disparity between academic benchmarks and practical applications. To address this discrepancy, we conduct a detailed investigation into the application of tabular data in industrial scenarios and propose a comprehensive and complex benchmark TableBench, including 18 fields within four major categories of table question answering (TableQA) capabilities. Furthermore, we introduce TABLELLM, trained on our meticulously constructed training set TableInstruct, achieving comparable performance with GPT-3.5. Massive experiments conducted on TableBench indicate that both open-source and proprietary LLMs still have significant room for improvement to meet real-world demands, where the most advanced model, GPT-4, achieves only a modest score compared to humans.

Code — <https://github.com/TableBench/TableBench>

Introduction

Recent studies have shown the potential of large language models (LLMs) on tabular tasks such as table question answering (TableQA) (Zhu et al. 2021; Zhao et al. 2023; Hegselmann et al. 2023; Li et al. 2023b; Zhang et al. 2024b; Lu et al. 2024) by adopting in-context learning and structure-aware prompts (Singha et al. 2023), suggesting that a well-organized representation of tables improves the interpretation of tabular. Tai et al. (2023) notes that eliciting a step-by-step reasoning process from LLMs enhances their ability to comprehend and respond to tabular data queries. Furthermore, Zha et al. (2023) investigates the use of external interfaces for improved understanding of tabular data.

Traditionally, adapting language models for tabular data processing entailed modifying their architectures with specialized features such as position embeddings and attention

*Corresponding author.

Year	Gallons Consumed	Fuel Expense	Average Price	Operating Expense Percentage	Available Seat Miles
2018	4,137	\$9,307	\$2.25	24%	67
2017	3,978	\$6,913	\$1.74	20%	66
2016	3,904	\$5,813	\$1.49	18%	65



Fact Checking

Multi-hop Fact Checking

QUESTION:

Does higher fuel consumption in 2017 and 2018 correspond to a higher fuel expenses in total?



Numerical Reasoning

Multi-hop Numerical Reasoning

QUESTION:

How much overall operating expenses increased in 2018 compared to 2017?



Data Analysis

Trend Forecasting

QUESTION:

Estimate what the total operating expense might be in 2019 based on available data.



Visualization

Chart Generation

QUESTION:

Please create a line graph based on the years and the corresponding "operating expenses" data.

Figure 1: Typical challenges from four major categories (fact checking, numerical reasoning, data analysis, visualization) in TableBench.

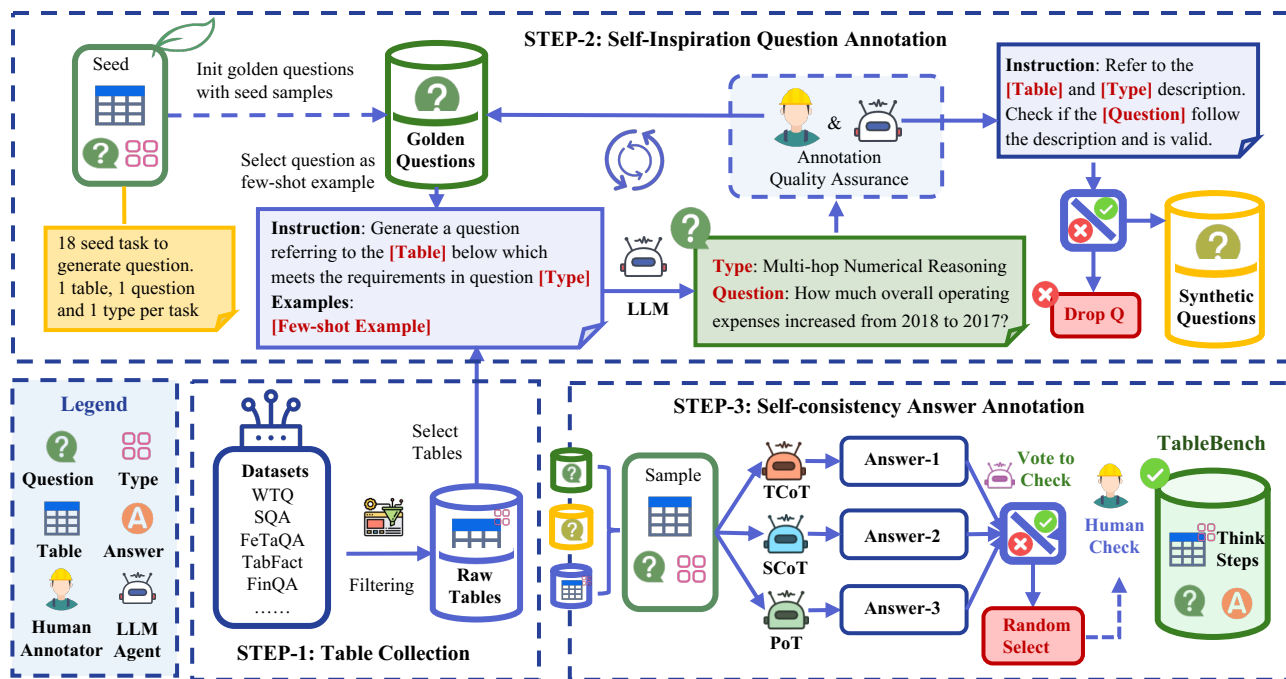


Figure 2: A comprehensive overview of the annotation framework.

mechanisms to grasp structural nuances of tables. However, the introduction of LLMs like GPT-4, GPT-3.5 (Brown et al. 2020; OpenAI et al. 2024), and PaLM2 (Anil et al. 2023) has heralded a new approach focused on the art of crafting precise, information-rich prompts that seamlessly integrate table data, coupled with leveraging external programming languages like SQL, Python, or other languages (Wang et al. 2024; Chai et al. 2024), which facilitates more sophisticated chain-of-thought (Wei et al. 2022) (CoT) reasoning processes across both proprietary and open-source LLM platforms, including Llama. Such advancements have propelled the fine-tuning of models for tabular data-specific tasks, showcased by initiatives like StructLM (Zhuang et al. 2024), enhancing capabilities in table structure recognition, fact verification, column type annotation, and beyond. However, the existing benchmark might not entirely resonate with the practical challenges, especially complex reasoning requirements encountered by professionals routinely navigating tabular data in real-world settings. *Therefore, there is a huge need for creating a benchmark to bridge the gap between the industrial scenarios and the academic benchmark.*

To better evaluate the capability of LLMs in Table QA, we introduce TableBench, a comprehensive and complex benchmark covering 18 subcategories within four major categories of TableQA abilities, as illustrated in Figure 1. First, We systematically analyze real-world challenges related to table applications and define task complexity based on the required number of reasoning steps. Based on the analysis, we introduce a rigorous annotation workflow, integrating manual and automated methods, to construct TableBench. Subsequently, We create a massively TableQA instruction corpora TableInstruct, covering three distinct reason-

ing methods. Textual chain-of-thought (TCoT) utilizes a textual reasoning approach, employing a series of inferential steps to deduce the final answer. Symbolic chain-of-thought (SCoT) adopts symbolic reasoning steps, leveraging programming commands to iteratively simulate and refine results through a *Think then Code* process. Conversely, program-of-thought (PoT) generates executable code, using lines of code as reasoning steps within a programming environment to derive the final result. Based on open-source models and TableInstruct, we propose TABLELLM as a strong baseline to explore the reasoning abilities of LLMs among tabular data, yielding comparable performance with GPT-3.5. Furthermore, we evaluate the performance of over 30 LLMs across these reasoning methods on TableBench, highlighting that both open-source and proprietary LLMs require substantial improvements to meet real-world demands. Notably, even the most advanced model, GPT-4, achieves only a modest score when compared to human performance.

The contributions are summarized as follows:

- We propose TableBench, a human-annotated comprehensive and complex TableQA benchmark comprising 886 samples across 18 fields, designed to facilitate fact-checking, numerical reasoning, data analysis, and visualization tasks.
- We introduce TableInstruct, a massive TableQA instruction corpus covering three distinct reasoning methods. TABLELLM, trained on TableInstruct, serves as a robust baseline for TableBench.
- We systematically evaluate the interpretation and processing capabilities of more than 30 models on our crafted TableBench and create a leaderboard to evaluate

Properties	Value
<i>Basic Insight</i>	
Unique Tables	3681
Question Length(Avg)	20.30
Answer Length (Avg)	8.52
Columns Per Table	6.68
Rows Per Table	16.71
Ratio of Numerical Cells	65.74%
Average Reasoning Steps	6.26
<i>Question Categories</i>	
Fact Checking	Match-Based Fact Checking Multi-hop Fact Checking
Numerical Reasoning	Arithmetic Calculation Comparison Aggregation Ranking Counting Time-based Calculation Multi-hop Numerical Reasoning Domain-Specific
Data Analysis	Descriptive Analysis Anomaly Detection Statistical Analysis Correlation Analysis Causal Analysis Trend Forecasting Impact Analysis
Visualization	Chart Generation
TableBench Size	886
TableInstruct Size	19,661

Table 1: Data statistics of TableBench

them on four main tasks. Notably, extensive experiments suggest that comprehensive and complex TableQA evaluation can realistically measure the gap between leading language models and human capabilities in real-world scenarios.

Construction of TableBench

To bridge the gap between academic benchmarks and industrial scenarios, we comprehensively analyze tabular data applications in real-world contexts, categorizing these problems into four major categories and 18 specific subcategories. We define the complexity of these tasks based on the reasoning steps required for problem-solving and provide detailed guidelines for defining and decomposing these steps, which are rigorously followed during the annotation process. Additionally, we introduce an annotation framework that combines manual and automated methods to enhance annotation efficiency, as illustrated in Figure 2. Finally, we propose two high-quality corpora: TableBench, a comprehensive and complex benchmark consisting of 886 samples, and TableInstruct (20K samples in total), massive instruction corpora designed to instruct LLMs with various reasoning methods.

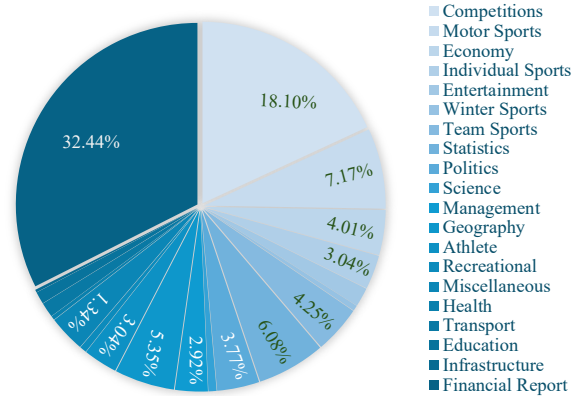


Figure 3: Topic distribution across tables

Tabular Data Collection

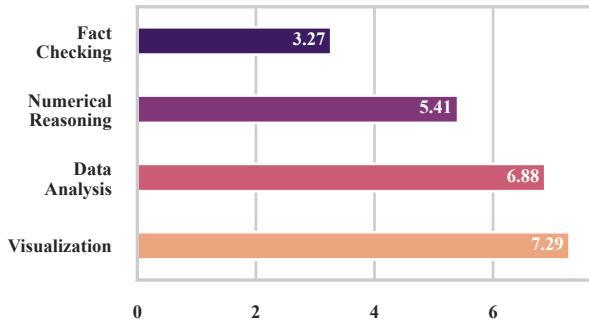
We collect raw tabular data from existing datasets, including typical datasets such as WTQ (Pasupat and Liang 2015), SQA (Iyyer, Yih, and Chang 2017), TabFact (Nan et al. 2022), FeTaQA (Nan et al. 2022), FinQA (Chen et al. 2021c), AIT-QA (Katsis et al. 2022), etc. To align closely with the "reasoning complexity of questions" dimension in real-world tabular problems, we do not specifically design for the complexity of the tables themselves, such as structural complexity or large-sized tables. Instead, we adopt a moderate complexity in tabular data. We select tables based on topics and size, ensuring each contains at least 8 rows and 5 columns. We focus on tables with significant numerical values to emphasize numerical reasoning, thereby ensuring depth in numerical computation reasoning. Ultimately, we collect 3681 tables covering 20 major topics: finance, competition, sports, science, etc.

Question Annotation

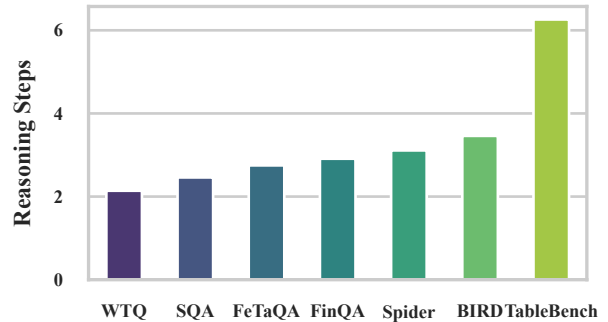
We opt to manually construct a more complex set of questions to mitigate the data leak risk in LLMs rather than modi-

Dataset	Fact Checking	Numerical Reasoning	Data Analysis	Visualization
WTQ	✓	✗	✗	✗
SQA	✓	✗	✗	✗
TabFact	✓	✗	✗	✗
FeTaQA	✓	✗	✗	✗
FinQA	✗	✓	✗	✗
AIT-QA	✗	✓	✗	✗
WikiSQL	✓	✓	✗	✗
Spider	✓	✓	✗	✗
Bird	✗	✓	✓	✗
Text2Analysis	✗	✗	✓	✓
TableBench	✓	✓	✓	✓

Table 2: Comparison with existing datasets in categories.



(a) Reasoning steps of various question categories in TableBench



(b) Reasoning steps comparison across different datasets

Figure 4: Comparison with existing datasets in reasoning complexity.

ifying existing datasets. We introduce a self-inspiration question generation mechanism to construct questions across different categories. Firstly, We meticulously craft one seed question and a detailed definition for each category, forming the initial question seed corpus. Subsequently, we incorporate these initial seed questions as examples into a meticulously designed prompt to guide GPT4-based agents in generating questions that adhere to specific category constraints. We limit the output to five questions in the initial rounds. These questions are manually annotated to identify new patterns and added to the seed corpus. We continuously select representative questions into the question seed corpus to promote benchmark qualities, eventually maintained at 50 questions, serving as the test set questions for TableBench. Upon reaching 50 questions per category, we conduct manual annotations on a sample basis (30%), with the remaining questions validated by another GPT-4 agent through a question verification process, eventually serving as the questions for TableInstruct.

Answer Annotation

We design a self-consistency mechanism for annotating answers based on a given table and question. During the answer generation phase, we utilize three LLM agents, each employing a distinct reasoning method (TCoT, SCoT, and PoT) to generate responses. We introduce a voting mechanism to assess the answers generated by the different agents. We preliminarily reserve the results if the voting system identifies a valid consistency among all agents. These preliminary results are then subjected to manual review and modification to produce the final answer and its associated reasoning details. Additionally, to minimize bias in answers generated by LLMs, we enforce a strict format for all answers, retaining only the essential and accurate content, thereby avoiding any preference for model-specific answer styles. For answers excluded due to inconsistencies, particularly those stemming from questions deemed too complex for LLMs to generate an adequate response, we randomly select 30% of the filtered data for manual annotation and subsequently incorporate them into the dataset. Notably, We manually annotate all answers in the TableBench with no

omissions and carefully scrutinize each.

Dataset Statistic

Topics TableBench primarily consists of numerical tables, with the largest portions derived from financial reports and data from competitive events, as illustrated in Figure 3,

Question Categories Drawing from real-world scenarios and user demands for tabular data, we devise four primary question categories: fact-checking, numerical reasoning, data analysis, and visualization, encompassing 18 sub-categories, thoroughly illustrating the various challenges encountered in TableQA scenarios, as shown in Table 1. Compared to existing datasets, TableBench covers a wider range of question types, as shown in Table 2, with a particular focus on data analysis and chart generation capabilities, which are notably lacking in previous datasets.

Reasoning Steps We define the complexity of the dataset by calculating the number of reasoning steps required to solve the problem. Figure 4 illustrates that the overall complexity of the benchmark is significantly greater than that of existing datasets, especially concerning questions related to data analysis and visualization.

TABLELLM

Problem Definition

Table question answering (Table QA) can be formulated as follows: Given a semi-structured table \mathcal{T} , comprised of \mathcal{R} rows and \mathcal{C} columns, the objective is to generate an answer \mathcal{A} to a question \mathcal{Q} utilizing the information contained within \mathcal{T} , where \mathcal{A} is a set of values or entities denoted as $\{a_1, a_2, \dots, a_k\}$, where $k \in \mathbb{N}^+$.

Reasoning Methods

In-context learning (ICL) (Dong et al. 2022) refers to strategies that optimize input for LLMs (\mathcal{M}) to generate practical outputs with a task-specific instruction (\mathcal{I}) and a few output examples (\mathcal{E}). We introduce distinct reasoning methods to fully assess the reasoning capabilities of LLMs

Textual Chain-of-Thought (TCoT) TCoT (Wei et al. 2022) refers to a reasoning process in which LLMs incrementally derive a series of intermediate steps or sub-goals through textual prompts before generating the final answer. These intermediate steps constitute a "thought chain" that ultimately leads the model to the correct outcome. Formally, the method is:

$$\mathcal{M}(\mathcal{T}, \mathcal{Q}, \mathcal{E}) \rightarrow \{r_1, r_2, \dots, r_k, \mathcal{A}\} \quad (1)$$

where r_k represents the k -th reasoning step.

Symbolic Chain-of-Thought (SCoT) SCoT implements a methodology that utilizes Python-based instruction to facilitate logical reasoning, comprising three primary steps repeated until a definitive conclusion is derived: **STEP-1**: Analyzing the available information to determine the next move. **STEP-2**: Generating instructions using Python programming language commands. **STEP-3**: Simulating the outcomes by executing the instructions and analyzing the results. The entire steps can be formally framed as follows:

$$\mathcal{M}(\mathcal{T}, \mathcal{Q}, \mathcal{E}) \rightarrow \{(r_{a_1}, r_{p_1}, r_{s_1}), \dots, (r_{a_k}, r_{p_k}, r_{s_k}), \mathcal{A}\} \quad (2)$$

where r_{a_k} is the analyzing step, r_{p_k} is the program commands generating step, and r_{s_k} is the result simulation step.

Program-of-Thoughts (PoT) PoT (Chen et al. 2022) offers a novel approach to numerical reasoning tasks by distinctly delineating computation from reasoning. PoT decomposes the problem into programming commands \mathcal{P} and utilizes a language interpreter, like Python, to compile and execute the resultant code. In contrast to SCoT, PoT enhances reasoning capabilities by actually executing generated code (\mathcal{P}) within a programming environment to output results, thereby implementing reasoning through structured code steps. The method can be formulated as:

$$\mathcal{M}(\mathcal{T}, \mathcal{Q}, \mathcal{E}) \rightarrow \mathcal{P} \rightarrow \mathcal{A} \quad (3)$$

Supervised Fine-Tuning

We train TABLELLM by finetuning all parameters of baseline LLMs to learn from the TableInstruct. The training objective \mathcal{L}_{all} can be described as:

$$\mathcal{L}_{all} = - \sum_{n=1}^N \mathbb{E}_{q^{R_n}, a^{R_n} \sim \{D^{R_n}\}_{n=1}^N} \left[\log P(a^{R_n} | q^{R_n}; \mathcal{M}) \right] \quad (4)$$

where q^{R_n} and a^{R_n} are the table-related question and answer from the dataset D^{R_n} of reasoning method R_n , respectively. N is the number of reasoning methods.

Experiments

Implementation Details

We meticulously design uniform style prompt templates to implement distinct reasoning methods to ensure the fairness of the evaluation. Furthermore, we impose formatting constraints on the outputs of LLMs and parse the final answers from the outputs to prevent any extraneous information from affecting the evaluation results. For open-source

models, we operate within the *transformer* environment on multiple A100 GPUs. For proprietary models, we employ official APIs to interact with exclusive LLMs. We conduct supervised finetuning of various open-source LLMs on the designated training set (TableInstruct). We utilize a cosine annealing scheduler, setting the initial learning rate at $2e^{-5}$, and conduct training over three epochs. Optimization is performed using the Adam optimizer, with a batch size of 512 and a maximum sequence length of 4096.

LLMs

We evaluate 34 models with sizes ranging from 7B to 110B parameters, including general/code LLMs, open-source/proprietary models, and SFT (Ouyang et al. 2022) models. For open-source LLMs, we evaluate on Llama2s (Touvron et al. 2023), Llama3s (Grattafiori et al. 2024), Llama3.1s, CodeLlamas (Rozière et al. 2024), CodeQwen1.5-7B-Chat, Qwen1.5s (Bai et al. 2023), Qwen2s (Yang et al. 2024), Mistral-7B-Instruct-v0.2 (Jiang et al. 2023), Deepseek-Coders (Guo et al. 2024), StructLLMs (Zhuang et al. 2024), MAP-Neo-7B-Instruct (Zhang et al. 2024a), WizardLM-13B-V1.2 (Xu et al. 2023). For proprietary LLMs, we perform evaluation on GPTs (Brown et al. 2020; OpenAI et al. 2024) (GPT-3.5-Turbo, GPT4-Turbo, GPT4-o), Qwen-Max (Yang et al. 2024), GLM-4 (GLM et al. 2024), Yi-Large (AI et al. 2024) and Deepseek models (DeepSeek-AI et al. 2024) (Chat-V2, Coder-V2). Furthermore, we finetune TABLELLM based on CodeQwen-7B, DeepSeekCoder-7B, Llama3-8B, Llama3.1-8B, and Qwen2-7B to further explore the Table QA capabilities of LLMs.

Automatic Evaluation Metrics

we adopt Rouge-L (Lin 2004) to assess the quality of the generated answers by measuring the n-gram overlap with reference answers. In the PoT method, we enforce a specific format for the executable code outputs and evaluate the final answer with the ROUGE-L metric, ensuring alignment with other reasoning methodologies. Specifically, in the task of chart generation, we parse and execute code derived from LLM responses and establish rigorous test cases to assess the accuracy of the generated charts, with a particular focus on the precision of y-axis fields, employing the pass@1 metric (Chen et al. 2021a) for evaluation.

Main Results

Table 3 showcases the main results of over 30 advanced advanced LLMs on the TableBench. GPT-4 outperforms other models in numerous tasks, demonstrating superior performance across complex reasoning scenarios. Particularly in numerical computation and analytical tasks, GPT-4 maintains a commendable level of performance. TABLELLM finetuned on the open-source models with TableInstruct achieves a performance level comparable to GPT-3.5, significantly validating the effectiveness of our training data. Despite these advancements, humans still surpass all LLMs in these tasks. Nevertheless, certain advanced LLMs, especially those employing proprietary approaches, demonstrate

	Fact Checking			Num-Reasoning			Data Analysis			Visualization			Overall		
	TCoT	SCoT	PoT	TCoT	SCoT	PoT	TCoT	SCoT	PoT	TCoT	SCoT	PoT	TCoT	SCoT	PoT
Human Performance	94.3			87.1			82.1			86.3			85.91		
<i>Open-source In Context Learning Methods</i>															
Llama2-7B	34.99	27.47	3.61	6.70	4.63	3.95	14.31	12.49	1.56	0.00	0.00	0.00	12.36	9.95	2.76
CodeLlama-7B	33.06	12.34	19.44	5.43	2.99	13.31	16.16	17.06	1.79	0.00	0.00	0.00	12.30	9.28	8.85
Gemma-7B	27.63	10.07	21.62	6.78	2.91	10.45	20.33	11.76	6.74	0.00	0.00	2.00	13.96	6.97	9.81
Mistral-7B	50.45	40.56	6.25	8.73	5.77	2.60	21.99	21.12	1.19	0.00	0.00	0.00	17.86	15.11	2.35
Deepseek-Coder-7B	22.92	27.48	48.98	6.45	5.61	34.66	18.73	20.72	18.17	8.00	18.00	18.00	13.10	14.58	28.89
CodeQwen1.5-7B	30.56	32.94	0.00	6.24	5.68	0.00	27.04	22.47	0.00	2.00	0.00	0.00	16.80	14.85	0.00
Qwen1.5-7B	56.08	53.53	39.23	11.30	10.99	20.40	24.77	22.96	7.66	0.00	0.00	0.00	20.70	19.65	16.29
Qwen2-7B	57.70	57.52	0.00	16.09	16.65	0.76	24.02	21.50	0.38	0.00	4.00	2.00	22.77	22.26	0.60
StructLM-7B	47.72	64.06	13.54	9.55	19.97	11.48	19.59	23.83	4.38	0.00	0.00	0.00	17.06	25.21	8.30
MAP-Neo-7B	32.70	33.22	0.00	7.23	6.46	0.00	21.85	14.38	0.44	0.00	0.00	4.00	15.26	12.03	0.40
Llama3-8B	38.32	72.53	13.94	22.02	17.33	19.50	30.15	30.75	9.31	0.00	0.00	10.00	25.71	27.59	14.43
Llama3.1-8B	47.89	36.29	30.38	11.26	13.77	17.24	15.78	14.82	8.86	8.00	0.00	8.00	16.76	15.81	14.88
Llama2-13B	48.47	32.69	3.03	15.83	6.79	4.48	22.04	17.16	3.19	0.00	0.00	0.00	20.86	13.25	3.61
StructLM-13B	26.28	64.49	1.04	12.30	17.38	0.00	20.70	18.41	0.28	0.00	0.00	0.00	16.35	21.94	0.21
WizardLM-13B	53.93	46.01	8.33	13.79	16.52	14.79	22.61	20.16	3.73	0.00	0.00	4.00	20.75	20.23	9.12
Qwen1.5-14B	40.83	61.92	44.38	10.29	15.01	28.20	22.99	29.24	10.33	2.00	8.00	2.00	18.03	25.14	21.48
Qwen1.5-32B	64.99	67.86	49.01	19.13	21.15	34.01	24.27	28.29	17.43	4.00	8.00	8.00	25.38	28.30	27.79
Deepseek-Coder-33B	48.27	54.34	33.12	9.41	12.69	32.60	9.09	21.70	19.97	0.00	0.00	24.00	13.01	19.92	27.20
CodeLlama-34B	64.39	58.28	5.90	13.10	13.30	4.20	19.23	15.28	0.53	0.00	0.00	2.00	20.24	18.19	2.88
StructLM-34B	19.10	30.21	27.74	15.36	9.03	14.45	20.74	17.92	5.38	0.00	0.00	2.00	16.93	14.37	11.61
Mixtral-8x7B	54.54	56.01	35.86	16.80	16.05	26.23	24.69	25.67	13.96	2.00	0.00	6.00	23.14	23.24	21.32
Qwen1.5-72B	71.27	67.03	33.16	19.01	16.68	20.85	26.63	27.33	13.03	2.00	8.00	14.00	26.66	25.80	18.65
Qwen2-72B	72.50	71.13	56.37	36.97	31.81	41.33	32.20	31.85	22.36	20.00	14.00	12.00	38.13	35.14	33.91
Qwen1.5-110B	74.87	69.80	53.55	29.81	23.33	36.83	27.34	29.32	18.38	14.29	12.00	24.00	32.81	30.10	30.77
Llama3-70B	73.88	75.44	60.64	37.64	28.87	36.59	37.47	34.06	26.11	4.00	6.00	10.00	39.59	34.48	33.59
Llama3.1-70B	76.32	77.65	59.05	44.89	38.93	34.04	35.88	33.87	23.15	26.00	6.00	34.00	43.85	39.22	32.52
<i>Close-source In Context Learning Methods</i>															
GPT-3.5-Turbo	59.95	75.68	60.92	23.45	23.16	42.09	34.40	32.54	30.25	10.00	4.00	38.00	30.85	31.41	39.34
Qwen-Max	70.48	68.21	50.42	32.83	25.06	32.80	27.87	30.98	19.41	18.00	8.00	30.00	34.26	31.04	29.39
Yi-Large	71.41	66.08	13.19	40.18	23.20	15.25	29.22	22.59	5.97	26.00	26.00	6.00	38.57	27.82	10.90
GLM-4	67.93	73.59	31.49	34.01	26.18	25.46	30.47	31.54	25.34	8.00	14.00	22.00	34.80	32.76	25.92
Deepseek-Chat-V2	72.41	69.89	57.48	50.07	38.96	45.96	38.07	34.44	30.37	40.00	24.00	46.00	47.22	39.63	41.07
Deepseek-Coder-V2	73.90	70.17	63.00	47.22	38.24	47.26	33.09	31.82	31.56	40.00	26.00	44.00	44.26	38.57	42.75
GPT-4-Turbo	75.92	77.62	70.08	53.01	44.31	49.31	41.03	36.52	34.63	62.00	32.00	48.00	51.32	44.26	45.69
GPT-4o	72.63	71.01	62.31	54.46	42.20	47.83	38.90	34.65	30.03	56.00	38.00	54.00	50.39	42.22	42.92
<i>Open-Source Fine-Tuning Methods</i>															
TableLLM _{CodeQwen-7B}	62.90	66.94	4.86	24.86	14.90	12.04	31.49	30.52	16.08	36.00	26.00	36.00	32.16	27.13	14.21
TableLLM _{Dpsk-Coder-7B}	69.23	63.15	7.12	35.87	21.20	14.61	31.75	29.62	21.04	36.00	18.00	30.00	37.76	28.80	17.19
TableLLM _{Llama3.1-8B}	68.15	65.17	25.67	30.51	17.86	28.64	33.47	30.20	19.77	24.00	18.00	44.00	35.29	27.70	25.76
TableLLM _{Llama3-8B}	62.13	64.46	15.07	29.42	16.73	12.68	30.21	29.45	17.53	26.00	20.00	28.00	32.89	26.97	15.83
TableLLM _{Qwen2-7B}	71.05	62.34	10.59	37.25	19.95	10.34	32.60	31.95	18.76	24.00	20.00	26.00	38.32	29.09	14.54

Table 3: The main results of advanced LLMs on TableBench are presented alongside human performance. All methods involving code generation and computation, particularly in the chart generation task, execute code only once to derive the final answer. The overall results represent a weighted average of performance across different categories.

potential in these scenarios. However, complex reasoning environments on tabular data still remain challenges.

Category Analysis Experimental results in Table 3 reveal that most models perform commendably in fact-based reasoning tasks, indicating their proficiency in this area. However, challenges arise in numerical reasoning tasks due to the complexity of mathematical computations, especially complex calculations such as aggregation, which require multiple intermediate steps to reach the final answer. Data analysis tasks necessitate more intricate and comprehensive analytical skills, such as using correlation coefficients to analyze model relationships and employing linear regression functions to predict future trends, thereby imposing higher demands on the overall reasoning abilities of LLMs. The task of chart generation poses the greatest challenge, requiring significant coding skills and strict adherence to instructions. Notably, smaller-sized models exhibit significant deficiencies in chart generation tasks, highlighting their limitations in utilizing code to handle complex tasks.

Reasoning Methods Analysis As illustrated in Table 3, those methods incorporating reasoning steps demonstrate a clear advantage on TableBench compared to methods that derive conclusions directly. The TCoT method exhibits stable and superior performance across various dimensions. The PoT method delivers commendable results in purely numerical computations, particularly in chart generation, but falls short in textual reasoning. We investigate the factors contributing to the suboptimal performance of the PoT method and find that the code execution success rate constrains the performance, as we only conduct a single generation and execute the code without employing any strategy for code correction. Even for the best-performing GPT4-Turbo, the executable code ratio is only 78.67%. This indicates that the PoT method requires LLMs with significant code-generation capabilities and instruction-following ability. However, it also underscores the substantial potential of the PoT method. Conversely, the SCoT method adapts effectively in scenarios requiring a combination of numerical and textual reasoning, such as analytical tasks, achieving a balanced yet modest overall performance. The performance of SCoT falls short of expectations due to its reliance on simulated outcomes rather than executing actual code.

	Auto Metric	GPT-4 Eval	Human Eval
GPT-3.5-Turbo	30.87	32.84	34.12
Qwen-Max	34.29	36.12	37.12
Yi-Large	38.56	43.12	41.23
GLM-4	34.82	38.60	39.21
Deepseek-Chat-V2	47.24	48.31	50.12
Deepseek-Coder-V2	44.92	44.92	46.13
GPT-4-Turbo	51.30	52.82	54.02
GPT-4o	50.53	54.18	53.19
PCC with Auto Metric	1.000	0.981	0.995

Table 4: We performed a consistency test of evaluation methods for advanced LLMs on TCoT performance

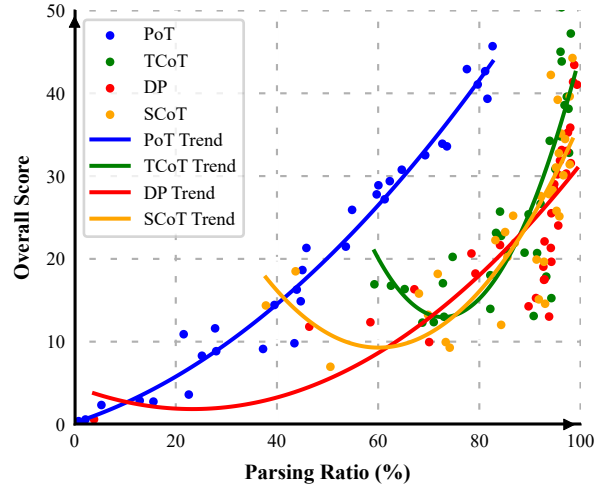


Figure 5: The impact of the parsing ratio on the overall score, where the parsing ratio is defined as the proportion of responses generated by the LLM that can be successfully parsed according to predetermined instructions.

Consistency of Evaluation Methods

Despite constraints imposed on the output format and the standardization of ground truth annotations, the ROUGE-L metric may not fully capture the real performance due to the inherent flexibility in the outputs of LLMs. Both GPT-4 and human judgment are conducted, as shown in Table 4, to assess this potential bias. The Pearson Correlation Coefficient (Cohen et al. 2009) (PCC) is adopted to analyze the consistency across different evaluation methods. The results, as presented in the table, indicate a high level of agreement among these evaluating methods, demonstrating that the constraints are effective and our metric accurately reflects the real performance of LLMs on the TableBench.

Further Analysis

Instruct Following Analysis

We observe that the performance trends of small-size LLMs across different reasoning methods differ from those observed in large-size models in Table 3. In further analysis, we introduce a comparison with non-reasoning methods, specifically focusing on Direct Prompting (DP), which provides solutions directly without intermediate reasoning steps. We find that the non-reasoning method (DP) performs better on small-size LLMs than reasoning-based methods. As shown in Figure 5, most models exhibit good instruction-following capabilities with the DP method due to the simpler instructions to follow. Conversely, small-size LLMs perform significantly worse with the PoT method, mainly due to their insufficient code generation capabilities, resulting in a lower rate of executable code generation. Additionally, the iterative symbolic reasoning steps required by the SCoT method pose considerable challenges for small-scale models.

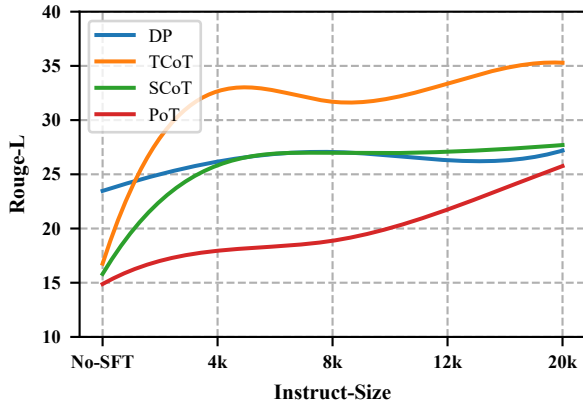


Figure 6: TableInstruct data efficient on TableLLM_{Llama3.1-8B}

In comparison to the DP, SCoT, and TCoT methods in Figure 5, the data points on the left side of the quadratic curve show that at low parsing ratios, the overall score increases as the parsing ratio decreases, suggesting that certain models (e.g., StructLLM), possess strong table understanding capabilities but exhibit weaker instruction-following abilities. This may be attributed to differences in the instruction format during instruction tuning compared to the format we employ. The right side of the quadratic curve reveals that despite the strong instruction-following performance of the DP method, the non-reasoning DP method faces a clear performance ceiling. In contrast, reasoning-based methods show significant potential for improvement. The curve of the PoT highlights the substantial potential of the PoT to enhance the overall score by increasing the parsing rate.

Data Efficiency of TableInstruct

In this section, we discuss the data efficiency of TableInstruct on the SFT process. We construct datasets of varying sizes by sampling from TableInstruct with sampling rates ranging from 0.2 to 0.6. Figure 6 visually depicts the relative performance at different sampling rates. Surprisingly, with only 60% of the samples, the model retains over 90% of the performance of the complete dataset. We observe that the Llama-3-8B model requires fewer than 4,000 samples to surpass the performance of Qwen1.5-70B on the dataset, demonstrating that the TableInstruct corpus significantly enhances tabular reasoning in smaller models. The full data provides the highest knowledge coverage, enabling the model to achieve optimal overall performance, comparable to GPT-3.5, with inference costs being only a fraction, indicating the high efficiency of TableInstruct.

Related Work

Table QA (Mueller et al. 2019; Jin et al. 2022) has grown substantially, driven by the development of robust datasets that engage advanced algorithms in the tasks of semantic comprehension (Huang et al. 2024; Li et al. 2023c, 2024b; Bai et al. 2023; Yang et al. 2024; Li et al. 2022a, 2024c,a). These datasets function as significant milestones for en-

hancing table-centric semantic understanding. WTQ (Pasupat and Liang 2015), SQA (Iyyer, Yih, and Chang 2017), and TabFact (Chen et al. 2020) set the cornerstone for Table QA research. They furnish benchmarks founded on question-answer pairs predicated on HTML tables sourced from Wikipedia. However, these datasets rely heavily on specific cell content from the table to formulate answers, which can not fully represent the multi-dimensional queries posed in real-world scenarios.

Acknowledging this incongruity, some datasets have been introduced to bridge the gap. ToTTo (Parikh et al. 2020), OTTQA (Chen et al. 2021b), and FeTaQA (Nan et al. 2022) step into the fore by providing free-form QA datasets. These datasets challenge models to generate answers that go beyond the table’s explicit content, thereby enhancing model performance to align with the free-form nature of real-world questions. FinQA (Chen et al. 2021c) and AIT-QA (Katsis et al. 2022) lay emphasis on numeric-focused queries. These datasets predominantly target financial tables, suggesting complex reasoning challenges that necessitate models to not only interpret but also to compute and extract nuanced information precisely. Further diversifying the landscape, datasets such as WikiSQL (Zhong, Xiong, and Socher 2017), Spider (Yu et al. 2018), and Bird (Li et al. 2023a) introduce logical expressions as supervisory signals to train Table QA models, discretizing reasoning capabilities through logic-based problem-solving. Despite the significant advancements made by LLMs in TableQA (Li et al. 2022b; Singha et al. 2023; Li et al. 2023b; Lei et al. 2023; He et al. 2023), there is still a critical need for benchmarks that reflect the reasoning complexity encountered in real-world tabular data scenarios. TableBench, a comprehensive and complex benchmark, incorporates real-world complexities into its evaluation scenarios, effectively addressing the limitations of existing benchmarks

Conclusion

In this work, we introduce TableBench, a comprehensive and complex benchmark designed to evaluate a broad spectrum of tabular skills. It encompasses 886 question-answer pairs across 18 distinct capabilities, significantly contributing to bridging the gap between academic benchmarks and real-world applications. We evaluate 30+ models with various reasoning methods on TableBench and provide a training set TableInstruct that enables TABLELLM to achieve performance comparable to ChatGPT. Despite these advancements, even the most advanced model, GPT-4, still lags significantly behind human performance on TableBench, underscoring the challenges of tabular tasks in real-world applications.

Limitations

We acknowledge the following limitations of this study: (1) This paper mainly focuses on the reasoning complexity of table questions, which does not extensively explore the inherent complexities of the tables themselves. (2) Tabular data in image formats, which are also prevalent in real-world applications, are not discussed in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, 62406033, U1636211, 61672081), and the State Key Laboratory of Complex & Critical Software Environment (Grant No. SKLCCSE-2024ZX-18).

References

- AI, ; ; Young, A.; Chen, B.; Li, C.; Huang, C.; Zhang, G.; Zhang, G.; Li, H.; et al. 2024. Yi: Open Foundation Models by 01.AI. arXiv:2403.04652.
- Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; et al. 2023. PaLM 2 Technical Report. arXiv:2305.10403.
- Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; et al. 2023. Qwen Technical Report. arXiv:2309.16609.
- Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-Shot Learners. *Proc. of NeurIPS*, 1877–1901.
- Chai, L.; Liu, S.; Yang, J.; Yin, Y.; Jin, K.; Liu, J.; Sun, T.; Zhang, G.; Ren, C.; Guo, H.; Wang, Z.; Wang, B.; Wu, X.; Wang, B.; Li, T.; Yang, L.; Duan, S.; and Li, Z. 2024. McEval: Massively Multilingual Code Evaluation. arXiv:2406.07436.
- Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; et al. 2021a. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.
- Chen, W.; Chang, M.; Schlinger, E.; Wang, W. Y.; and Cohen, W. W. 2021b. Open Question Answering over Tables and Text. In *Proc. of ICLR*.
- Chen, W.; Ma, X.; Wang, X.; and Cohen, W. W. 2022. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *TMLR*.
- Chen, W.; Wang, H.; Chen, J.; Zhang, Y.; Wang, H.; Li, S.; Zhou, X.; and Wang, W. Y. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *Proc. of ICLR*.
- Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.-H.; Routledge, B. R.; et al. 2021c. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *EMNLP 2021*, 3697–3711.
- Cohen, I.; Huang, Y.; Chen, J.; Benesty, J.; Benesty, J.; Chen, J.; Huang, Y.; and Cohen, I. 2009. Pearson correlation coefficient. *Noise reduction in speech processing*, 1–4.
- DeepSeek-AI; ; Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; Gao, H.; et al. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. arXiv:2401.02954.
- Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; and Sui, Z. 2022. A survey for in-context learning. *CORR*.
- GLM, T.; ; Zeng, A.; Xu, B.; Wang, B.; Zhang, C.; Yin, D.; Zhang, D.; Rojas, D.; Feng, G.; Zhao, H.; Lai, H.; et al. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. arXiv:2406.12793.
- Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.
- Guo, D.; Zhu, Q.; Yang, D.; Xie, Z.; Dong, K.; Zhang, W.; Chen, G.; Bi, X.; Wu, Y.; Li, Y. K.; Luo, F.; Xiong, Y.; and Liang, W. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence. arXiv:2401.14196.
- He, X.; Zhou, M.; Xu, X.; Ma, X.; Ding, R.; Du, L.; et al. 2023. Text2Analysis: A Benchmark of Table Question Answering with Advanced Data Analysis and Unclear Queries. arXiv:2312.13671.
- Hegselmann, S.; Buendia, A.; Lang, H.; Agrawal, M.; Jiang, X.; and Sontag, D. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *Proc. of AIS-TATS*, 5549–5581.
- Huang, S.; Ma, S.; Li, Y.; Huang, M.; Zou, W.; Zhang, W.; and Zheng, H. 2024. LatEval: An Interactive LLMs Evaluation Benchmark with Incomplete Information from Lateral Thinking Puzzles. In *Proc. of COLING*, 10186–10197.
- Iyyer, M.; Yih, W.-t.; and Chang, M.-W. 2017. Search-based neural structured learning for sequential question answering. In *ACL 2017*, 1821–1831.
- Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; et al. 2023. Mistral 7B. arXiv:2310.06825.
- Jin, N.; Siebert, J.; Li, D.; and Chen, Q. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, 174–186.
- Katsis, Y.; Chemmengath, S.; Kumar, V.; Bharadwaj, S.; Canim, M.; Glass, M.; Gliozzo, A.; Pan, F.; Sen, J.; Sankaranarayanan, K.; et al. 2022. AIT-QA: Question Answering Dataset over Complex Tables in the Airline Industry. In *Proc. of AACL*, 305–314.
- Lei, F.; Luo, T.; Yang, P.; Liu, W.; Liu, H.; Lei, J.; Huang, Y.; Wei, Y.; He, S.; Zhao, J.; and Liu, K. 2023. TableQAKit: A Comprehensive and Practical Toolkit for Table-based Question Answering. arXiv:2310.15075.
- Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; Zhou, X.; Ma, C.; Li, G.; Chang, K. C.; Huang, F.; Cheng, R.; and Li, Y. 2023a. Can LLM Already Serve as A Database Interface? A BIG Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proc. of NeurIPS*.
- Li, P.; He, Y.; Yashar, D.; Cui, W.; Ge, S.; Zhang, H.; Fainman, D. R.; Zhang, D.; and Chaudhuri, S. 2023b. Table-GPT: Table-tuned GPT for Diverse Table Tasks. arXiv:2310.09263.
- Li, Y.; Xu, Z.; Chen, S.; Huang, H.; Li, Y.; Jiang, Y.; Li, Z.; Zhou, Q.; Zheng, H.; and Shen, Y. 2023c. Towards Real-World Writing Assistance: A Chinese Character Checking Benchmark with Faked and Misspelled Characters. *CoRR*.
- Li, Y.; Xu, Z.; Chen, S.; Huang, H.; Li, Y.; Ma, S.; Jiang, Y.; Li, Z.; Zhou, Q.; Zheng, H.; and Shen, Y. 2024a. Towards Real-World Writing Assistance: A Chinese Character

- Checking Benchmark with Faked and Misspelled Characters. In *ACL 2024*, 8656–8668.
- Li, Y.; Zhou, Q.; Li, Y.; Li, Z.; Liu, R.; Sun, R.; Wang, Z.; Li, C.; Cao, Y.; and Zheng, H. 2022a. The Past Mistake is the Future Wisdom: Error-driven Contrastive Probability Optimization for Chinese Spell Checking. In *Findings of ACL 2022*, 3202–3213.
- Li, Y.; Zhou, Q.; Li, Y.; Li, Z.; Liu, R.; Sun, R.; Wang, Z.; Li, C.; Cao, Y.; and Zheng, H. 2022b. The Past Mistake is the Future Wisdom: Error-driven Contrastive Probability Optimization for Chinese Spell Checking. In *Proc. of ACL Findings*, 3202–3213.
- Li, Y.; Zhou, Q.; Luo, Y.; Ma, S.; Li, Y.; Zheng, H.; Hu, X.; and Yu, P. S. 2024b. When LLMs Meet Cunning Questions: A Fallacy Understanding Benchmark for Large Language Models. *CoRR*.
- Li, Y.; Zhou, Q.; Luo, Y.; Ma, S.; Li, Y.; Zheng, H.; Hu, X.; and Yu, P. S. 2024c. When LLMs Meet Cunning Questions: A Fallacy Understanding Benchmark for Large Language Models. *CoRR*, abs/2402.11100.
- Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81.
- Lu, W.; Zhang, J.; Zhang, J.; and Chen, Y. 2024. Large Language Model for Table Processing: A Survey. *CoRR*.
- Mueller, T.; Piccinno, F.; Shaw, P.; Nicosia, M.; and Altun, Y. 2019. Answering Conversational Questions on Structured Data without Logical Forms. In *EMNLP-IJCNLP 2019*, 5902–5910.
- Nan, L.; Hsieh, C.; Mao, Z.; Lin, X. V.; Verma, N.; Zhang, R.; Kryściński, W.; Schoelkopf, H.; Kong, R.; Tang, X.; et al. 2022. FeTaQA: Free-form table question answering. *TACL 2022*, 35–49.
- OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; Avila, R.; Babuschkin, I.; Balaji, S.; Balcom, V.; et al. 2024. GPT-4 Technical Report. arXiv:2303.08774.
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Proc. of NeurIPS*, 27730–27744.
- Parikh, A.; Wang, X.; Gehrmann, S.; Faruqui, M.; Dhingra, B.; Yang, D.; and Das, D. 2020. ToTTo: A Controlled Table-To-Text Generation Dataset. In *EMNLP 2020*, 1173–1186.
- Pasupat, P.; and Liang, P. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *ACL 2015*, 1470–1480.
- Rozière, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Sauvestre, R.; Remez, T.; Rapin, J.; et al. 2024. Code Llama: Open Foundation Models for Code. arXiv:2308.12950.
- Singha, A.; Cambronero, J.; Gulwani, S.; Le, V.; and Parnin, C. 2023. Tabular Representation, Noisy Operators, and Impacts on Table Structure Understanding Tasks in LLMs. *CoRR*.
- Tai, C.-Y.; Chen, Z.; Zhang, T.; Deng, X.; and Sun, H. 2023. Exploring Chain of Thought Style Prompting for Text-to-SQL. In *Proc. of EMNLP*, 5376–5393.
- Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.
- Wang, B.; Ren, C.; Yang, J.; Liang, X.; Bai, J.; Chai, L.; Yan, Z.; Zhang, Q.-W.; Yin, D.; Sun, X.; and Li, Z. 2024. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. arXiv:2312.11242.
- Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Proc. of NeurIPS*, 24824–24837.
- Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; and Jiang, D. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. arXiv:2304.12244.
- Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; et al. 2024. Qwen2 Technical Report. arXiv:2407.10671.
- Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP 2018*, 3911–3921.
- Zha, L.; Zhou, J.; Li, L.; Wang, R.; Huang, Q.; Yang, S.; Yuan, J.; Su, C.; Li, X.; Su, A.; Zhang, T.; Zhou, C.; Shou, K.; Wang, M.; Zhu, W.; Lu, G.; Ye, C.; Ye, Y.; Ye, W.; Zhang, Y.; Deng, X.; Xu, J.; Wang, H.; Chen, G.; and Zhao, J. 2023. TableGPT: Towards Unifying Tables, Nature Language and Commands into One GPT. arXiv:2307.08674.
- Zhang, G.; Qu, S.; Liu, J.; Zhang, C.; Lin, C.; Yu, C. L.; Pan, D.; Cheng, E.; Liu, J.; Lin, Q.; et al. 2024a. MAP-Neo: Highly Capable and Transparent Bilingual Large Language Model Series. arXiv:2405.19327.
- Zhang, X.; Zhang, J.; Ma, Z.; Li, Y.; Zhang, B.; Li, G.; Yao, Z.; Xu, K.; Zhou, J.; Zhang-Li, D.; Yu, J.; Zhao, S.; Li, J.; and Tang, J. 2024b. TableLLM: Enabling Tabular Data Manipulation by LLMs in Real Office Usage Scenarios. arXiv:2403.19318.
- Zhao, B.; Ji, C.; Zhang, Y.; He, W.; Wang, Y.; Wang, Q.; Feng, R.; and Zhang, X. 2023. Large Language Models are Complex Table Parsers. In *Proc. of EMNLP*, 14786–14802.
- Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*.
- Zhu, F.; Lei, W.; Huang, Y.; Wang, C.; Zhang, S.; Lv, J.; Feng, F.; and Chua, T.-S. 2021. TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance. In *Proc. of ACL*, 3277–3287.
- Zhuang, A.; Zhang, G.; Zheng, T.; Du, X.; Wang, J.; Ren, W.; Huang, S. W.; Fu, J.; Yue, X.; and Chen, W. 2024. StructLM: Towards Building Generalist Models for Structured Knowledge Grounding. arXiv:2402.16671.