TableBench: A Capability-Based Table Benchmark for Large Language Models

Anonymous ACL submission

Abstract

The rapid advancement of techniques in large language models (LLMs) for processing tabular data necessitates improvements in evaluation benchmarks. However, most of existing table benchmarks offer evaluation from a singular task-based perspective, failing to provide a comprehensive and meticulous assessment of the LLMs' table-related capabilities. To address this gap, we introduce TableBench, a capability-based benchmark tailored to evaluate the performance of LLMs on tabular data. Our framework intricately outlines 10 essential capabilities required from the point a model receives a table-related input to the generation of an output, with each capability tested across 6 table formats. We evaluate 20 models using TableBench and observe that GPT-4 and GPT-4o achieve the highest scores, while phi3-small outperform other open-source models of similar scale. Drawing from our evaluation, we present a series of valuable insights, which can serve as a pivotal reference for future table-related LLM research.

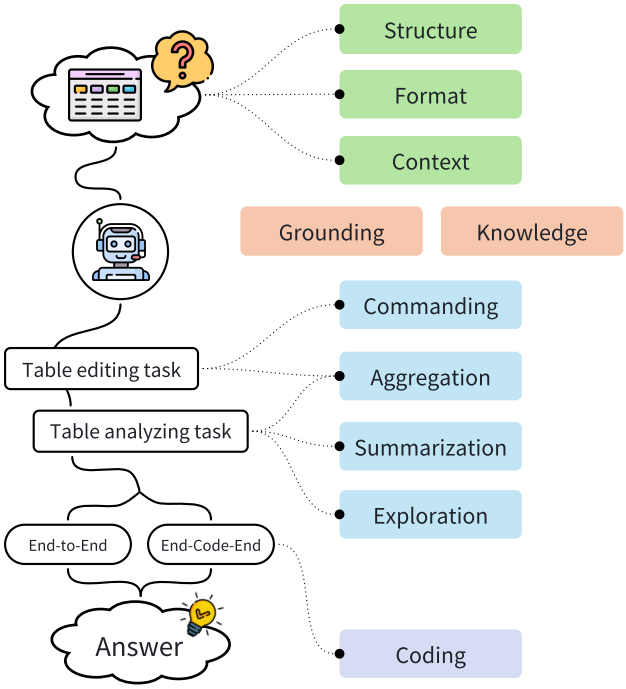


Figure 1: The Capability-Based Framework of TableBench.

1 Introduction

Tables, characterized by the two-dimensional inherent structure, are essential for storing, organizing and presenting large amounts of data, with widespread applications across diverse domains such as finance, medicine, business, education, etc. The pervasive utilization of tables has spurred the exploration of advanced techniques on large language models (LLMs) to effectively process tabular data, ranging from prompt engineering (Sui et al., 2024), model finetuning (Li et al., 2023; Zhang et al., 2023a) to LLM-powered agent (Li et al., 2024a; Hu et al., 2024).

The advancement of LLM techniques necessitates the improvement of evaluation benchmarks. Various existing datasets targeting diverse table-related tasks have been proposed. Specifically, a significant portion of these datasets is dedicated to the TableQA task, exemplified by datasets such as WikiTQ (Pasupat and Liang, 2015), HybridQA (Chen et al., 2020b), HiTab (Cheng et al., 2022), FeTaQA (Nan et al., 2022), and FinQA (Chen et al., 2021). Each of these datasets emphasizes distinct dimensions of evaluation. For instance, FinQA sharpens its focus on the financial domain, while HiTab introduces the additional challenge of comprehending hierarchical table structures. In terms of fact verification task, datasets such as TabFact (Chen et al., 2020a) and FEVEROUS (Aly et al., 2021) are tailored to validate factual accuracy derived from tables. There are also many other table tasks include Table-to-Text (Parikh et al., 2020; Nan et al., 2021), Text-to-SQL (Zhong et al., 2017; Yu et al., 2018), Table Interpretation (Deng et al., 2020), etc. In addition to datasets tailored for in-

dividual table tasks, there have been efforts to synthesize datasets across various table tasks, such as UnifiedSKG (Xie et al., 2022), TableInstruct (Zhang et al., 2023a), TableGPT (Li et al., 2023), etc. While existing datasets have evaluated have assessed LLMs from different angles, they are predominantly task-based. Task-based benchmarks primarily assess a model's performance on specific tasks, which often involves a combination of capabilities. Merely assigning a score to a task fails to provide a nuanced analysis of the individual capabilities necessary for that task. This highlights the absence of a holistic capability-based benchmark for evaluating LLMs in handling tabular data.

In response to this need, we introduce **TableBench**, a capability-based benchmark crafted to evaluate the LLMs' performance on tabular data. We design TableBench based on a framework that outlines the key capabilities required for an LLM to process tabular data, organized into a four-stage process. **Stage 1** involves understanding the table's structure, format, and input size. **Stage 2** focuses on locating relevant information and understanding its semantic context. **Stage 3** varies depending on the task, with editing tasks requiring aggregation and commanding capabilities, while analysis tasks require aggregating information, summarizing and exploring the underlying patterns. **Stage 4** is where the model provides final outputs, including the use of coding capabilities for additional processing. Each stage and capability is carefully evaluated through 500 curated questions per capability, forming the basis of TableBench.

Our contributions are summarized as follows:

- We introduce **TableBench**, a systematic and comprehensive capability-based benchmark to assess LLMs' capabilities on handling tabular data.
- We evaluate closed-source, generic open-source and table-specific models (20 in total) on our proposed benchmark to provide a thorough and detailed assessment.
- We provide valuable insights into LLMs' table-related capabilities. This analysis serves as a valuable reference for subsequent research and development efforts in enhancing the tabular capabilities of LLMs.

2 Related Work

2.1 Table-related models

The pervasive utilization of tables has simulated the efforts to develop powerful models to handle tabular data. Conventional approaches for processing tabular data predominately involve customizing specialized model architectures, often including special designs of attention mechanisms, positional embeddings and learning objectives for pretraining stage (Deng et al., 2020; Yin et al., 2020; Wang et al., 2021; Herzig et al., 2020; Liu et al., 2022). These modifications aims to tailor the models to capture structural nuances of tabular data. However, a notable shift in paradigm has occurred with the emergence of LLMs like GPT-4 (OpenAI, 2023b) , GPT-3.5 (OpenAI, 2023a), and Llama2 (Touvron et al., 2023). Initially designed as generic language models, they exhibit surprising capability to process tabular data and simulate new approaches to tackle table-related tasks (Lu et al., 2024). Some research leverages the powerful in-context learning of LLMs by curating suitable prompts to understand table data (Sui et al., 2024), or utilizing external programming tools to facilitate Chain-of-thought (CoT) (Wei et al., 2022) inference. Furthermore, finetuning open-source models such as Llama (Zhang et al., 2023b), CodeLlama (Zhang et al., 2024) or building LLM-powered agents equipped with external tools also receive great attentions. These ideas have infused new vitality into the field.

2.2 Table-related datasets

The progress in LLM techniques requires high-quality benchmarks across diverse table-related tasks. Typical table tasks include TableQA (Pasupat and Liang, 2015; Chen et al., 2020b; Nan et al., 2022; Cheng et al., 2021), Fact Verification (Chen et al., 2020a; Aly et al., 2021), Table-to-Text (Parikh et al., 2020; Nan et al., 2021), Text-to-SQL (Zhong et al., 2017; Yu et al., 2018), Table Interpretation (Deng et al., 2020), etc., which are facilitated by various carefully curated datasets. While these table datasets were released relatively early, they laid a solid foundation for subsequent datasets and have become an indispensable part of the field.

Some new benchmarks have been proposed to meet the increasing demands for more diverse evaluation. For instance, Text2Analysis incorporates advanced analysis tasks that extend beyond SQL-compatible operations and require more compli-

cated analysis (He et al., 2024). WikiTableEdit complicates the table manipulation task by introducing both regular and irregular table editing by natural language instruction (Li et al., 2024b). Additionally, Table-GPT (Zha et al., 2023) and Table-Instruct (Zhang et al., 2023b) collect and synthesize a wide range of table datasets to diversify the evaluation dimensions. Although originally designed to evaluate their own fine-tuned models, they still offer valuable references for evaluating generic LLMs.

While existing benchmarks have provided evaluations of LLMs across various dimensions, they are predominantly task-based. Task-based benchmarks primarily focus on evaluating a model's performance on specific tasks, which often involve a mixture of capabilities. Simply providing a score for a task does not allow for a detailed analysis of the specific capabilities required for that task. This highlights the absence of a holistic capability-based benchmark for evaluating the performance of LLMs in handling tabular data.

3 TableBench

3.1 Framework

We systemically design TableBench based on a capability-based framework (see Figure 1). It intricately delineates the required capabilities involved in the process from an LLM receiving a table-related input to providing output. We divide it into a 4-stage process and evaluate corresponding capabilities.

Stage 1 is where a model receives a user query and a table as the input and starts to recognize the table. This stage involves the understanding of table structure (*structure understanding*), table format (*format understanding*), and the table length (*large-table understanding*). These understanding capabilities enable the model to distinguish table from sequential natural language.

Stage 2 begins by the model searching for and identifying the regions where the answers to the user query might locate, demonstrating its *grounding capability*. It then proceeds to understand the semantic contents and relationships within these areas, showcasing its (*knowledge capability*).

Stage 3 depends on the specific user demands. From the user's perspective, the demands for tabular data can be primarily categorized into table editing and analyzing. In editing tasks, the model is required to group the located information from stage 2 (*aggregation capability*) and execute specific actions or commands to manipulate the table (*commanding capability*). For table analysis tasks, the model first synthesize the extracted information or summarize grouped contents (*summarization capability*) before delving deeper to explore significant insights (*exploration capability*).

Stage 4 is where the model provides the final outputs. In the context of tabular data, a notable aspect is the utilization of code in addition to directly output the answer from the table. Therefore, during the output phase, the *coding capability* of the model also requires evaluation.

3.2 Benchmark Construction

Structure Understanding This category assesses the capability of an LLM on identifying the organizational layout of tables. The model is required to answer questions related to table boundary, header, rows/column number, and supplementary information given a serialized table input. It aims to test whether an LLM can accurately interpret the foundational layout of a table.

Data source: This category contains two types of data sources. The first source is the annotated table regions of VEnron2, VEUSUS and VFUSE, as introduced in TableSense (Dong et al., 2019). These table regions are used to construct a boundary detection task, where, given a table with addresses (e.g., A1, B2, C3), the model needs to output the range of the table, such as "A5:G8". For the second data source, we use the same combination of three datasets as Gol et al. (2019): DeEx[1], SAUS[2], and CIUS[3] (DSC for short) to construct structure-related questions. DSC contains columns/rows numbers and cell-level annotations, which we utilize to require model to answer the numbers of columns/rows and identify whether the given table contains a header or extra information.

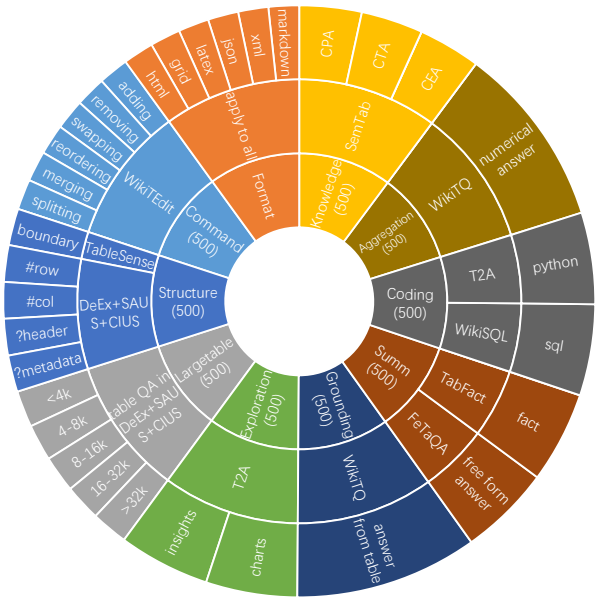
Metric: We use accuracy as the metric to evaluate this capability.

Format Understanding This category is essential to reveal the capability of an LLM to recognize and digest various table formats. One table can have various storage formats, each presenting different levels of information compression (Sui et al., 2024). We follow Sui et al. (2024) to convert a table

[1]https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/

[2]https://dbgroup.eecs.umich.edu/project/sheets/datasets.html

[3]https://ucr.fbi.gov/crime-in-the-u.s



Capability	Metric	Input	Output	Avg. Len
Structure	Accuracy	T+Q	Answer	2,244
Largetable	Accuracy	T+Q	Answer	33,239
Commanding	TEDS	T+I	Table	1,117
Knowledge	Accuracy	T+Q	Answer	605
Grounding	Accuracy	T+Q	Answer	1,686
Aggregation	Accuracy	T+Q	Answer	1,822
Summarization	Score	T+Q T+S	Text Boolean	1,361
Exploration	Accuracy	T+Q	Answer	1,343
Coding	Pass Rate	T+Q	Code	1,893
Format	/	/	/	/

Figure 2: The Components of TableBench. Pie chart on the left side details the datasets used to evaluate each capability and their subcategories. Table on the right side summarizes the metrics, inputs, outputs and average data length. T,Q,I,S are abbreviations for Table, Question, Instruction, and Statement.

into different formats but go further by applying them to a greater number of models and datasets. We uniformly convert all the tables into 2d-lists, which enables convenient transformation to various formats. In our experiments, we typically select six widely used formats: JSON, HTML, XML, LaTeX, Markdown, and Grid. But our code is capable of supporting more conversions of other different formats. Since we apply these formats to all of our other capability tests, no specific datasets and metrics are used.

Knowledge Understanding This category intends to assess the capability of an LLM to understand the semantic structure and meaning of table contents. It requires a model to link the elements of the table to semantic tags from Knowledge Graph (KG).

Data source: We use WikidataTables (Cutrona et al., 2020) dataset in SemTab 2023[4]. The dataset encompasses three tasks, namely CEA (Cell Entity Annotation), which involves annotating each target cell with an entity from Wikidata; CTA (Column Type Annotation), which entails annotating each entity column with items from Wikidata as its type; and CPA (Column Property Annotation), which requires annotating each column pair with a property from Wikidata. For the output of those tasks, the dataset employs the IDs of either an entity or a property. To evaluate the model's capacity for understanding knowledge in tables rather than just memorizing IDs, we convert the IDs into their corresponding english names and aliases, transforming the output of the task into the names of entities or properties.

Metric: The overall metric is defined as the average accuracy of CEA, CPA, and CTA accuracy. For the CEA and CPA tasks, a prediction is considered correct when the predicted value matches one of the ground truth entity or property names or aliases. As for the CTA task, we utilize approximate accuracy, which is defined the same as SemTab[5]. Formula details are in Appendix C.2.

Large-table Understanding This category focuses on evaluating the capability of an LLM to manage large tables that may contain vast amounts of cells. It is design to evaluate whether a model can handle scalability issues and whether the performance remains robust with large tables. In this part, we collect tables from 5 token lengths: <4k, 4-8k, 8-16k, 16-32k and >32k, each with 100 pieces of data.

Data source: We utilize DCS (same as Struc-

[4]https://www.cs.ox.ac.uk/isg/challenges/sem-tab/

[5]https://sem-tab-challenge.github.io/2023/

ture Understanding part) as the source of data and classify them into different lengths. To exclusively assess the model's comprehension of large tables while decoupling it from other capabilities, we have processed the dataset as follows:

- To decouple from the understanding of complex structures, we select the column-major simple tables from the dataset for evaluation. This involves removing metadata, notes, and other supplementary information from the tables, filtering out those that are primarily column-based and possess only the first row as the header for experiment.
- To decouple from the ability to comprehend complex questions, we generate simple queries with fixed answers to test large table capability. These primarily include questions about the number of rows in the table, the number of columns, and the content of the header in the n-th column.

Metric: We use accuracy as the metric to evaluate this capability.

Commanding Table commanding refers to the capability of an LLM to manipulate table(s) following various user commands. We aim to explore the model's intrinsic commanding capabilities, thus requiring the model to directly return the edited result without relying on external tools or code.

Data source: We select WikiTableEdit (Li et al., 2024b) as the data source for testing this capability. Each data instance in this dataset consists of a triplet denoted as (instruction, source table, and target table), which requires the model to return the modified table directly. It covers a wide range of operations such as adding, removing, swapping, reordering, merging, and splitting.

Metric: We use Tree-Edit-Distance-based Similarity (TEDS) (Zhong et al., 2019) as the metric for evaluation, which is designed to measure both the structure and the cell content similarity between the prediction and ground truth. The score is normalized between 0 and 1 where 1 means perfect matching. It is original designed for evaluating HTML-format tables, but we modify the code to support multiple formats. Detailed formula can be found in Appendix C.1.

Coding This category requires developing code to manipulate and process table data effectively, aiming to assess the coding proficiency of an LLM in handling and processing tabular data. Given a serialized table input, the model is required to writing Python code to automate data transformations, merging, filtering, and other processing tasks. Furthermore, it encompasses tasks related to generating SQL queries for manipulating information from tables.

Data source: For coding with table, there are currently two mainstream types of datasets – Python and SQL. For Python, we use the Text2analysis (He et al., 2024) dataset, which is designed for exploring advanced analysis through Python code generation on tables. We specifically select the rudimentary analysis task to test the model's coding capabilities. For SQL, we choose the WikiSQL (Zhong et al., 2017) dataset, which involves parsing queries and corresponding tables into SQL statements.

Metric: The overall metric is defined as the average pass rate of Text2Analysis and WikiSQL pass rate. Pass rate of WikiSQL is the execution accuracy in (Zhong et al., 2017). The pass rate for Text2analysis is similar to (He et al., 2024), representing the proportion of samples where the code correctly passes out of the total number of samples.

Grounding This category aims at testing the capability of an LLM in locating and extracting the specific information given a user question. It requires the model to index, search, locate and return the exact match answer of the question from the table.

Data source: We utilize the WikiTQ (Pasupat and Liang, 2015) dataset, which is a TableQA dataset. To evaluate grounding capability, we filter out examples where the result is a specific cell in a table, using these as evaluation samples.

Metric: We use accuracy as the metric to evaluate this capability.

Aggregation This category is design to reveal the capability of an LLM to perform numerical reasoning. It requires the model to synthesize information across multiple cells or rows/columns to conduct numerical calculation and inference.

Data source: We utilize the WikiTQ (Pasupat and Liang, 2015) dataset, which is a tableQA dataset. To evaluate aggregation capability, we filter out examples where the result is numerical but not a specific cell in a table, using these as evaluation samples.

Metric: We use accuracy as the metric to evaluate this capability.

Summarization This task is centered around testing the capability of an LLM on distilling high-level summaries from structured data. It requires the model to provide a one-sentence description given a question or determine whether a summary can be derived from the table.

Data source: We utilize free form TableQA and fact verification tasks to evaluate summarization capability. For free form TableQA task, generating a free-form answer, which is expressed as a single sentence, requires the model not only to retrieve the correct information but also to summarize it to produce a complete and coherent sentence. We utilize FetaQA (Nan et al., 2022) dataset, which is a TableQA dataset with free form answer. For the fact verification task, the given statement is pre-summarized information, and model needs to query and summarize information from the table and then compare it with the given statement to determine the result. We utilize TabFact (Chen et al., 2020a) dataset.

Metric: The overall metric is defined as the average score of BLEU for FetaQA and accuracy for TabFact.

Exploration This category aims at evaluating the capability of an LLM to navigate through and draw valuable insights from tabular data. It requires the model to uncover patterns, trends, and insights from organized table data.

Data source: We utilize basic insights and chart generation tasks in Text2Analysis (He et al., 2024) dataset, which is designed for exploring advanced analysis through Python code generation on tables. To more accurately evaluate the model's exploration capability rather than coding capability, we transform the dataset into one that directly generates basic insights and chart results. In other words, the output of the task is the execution results of the code in the original dataset.

Metric: We use accuracy as the metric to evaluate this capability.

3.3 Data Statistics

The statistics of TableBench are shown in Figure 2. TableBench covers a wide range of table tasks which are re-classified into different capabilities. For each capability, we meticulously select 500 data samples to evaluate the LLMs' performance, resulting in a total of 4,500 questions. In terms of average input length, most of the data falls within 2k tokens. We specially prepare a set of large tables with an average length of 33,239 and the longest data length reaching 680k.

4 Experiments

A thorough set of experiments are conducted on TableBench. We evaluate 20 models on all the capability tests with 6 different formats. We report the best format results in Table 1 and put the specific results of each format in Appendix F. We detail the selected models and experiment setups in this section.

4.1 Models

Our evaluation encompasses closed-source models, open-source models and table-specific models (20 models). We provide a detailed assessment of these models on TableBench. Below is a list of all the models we select: **(1) Closed-source models:** GPT-3.5, GPT-4, GPT-4o. **(2) Table-specific models:** TableLLM (Zhang et al., 2024), TableLlama (Zhang et al., 2023a). **(3) Open-source models:** Llama2 (Touvron et al., 2023), Llama3 (Touvron et al., 2023), Qwen2 (Bai et al., 2023), Mistral (Jiang et al., 2023), Mixtral (Jiang et al., 2024), Yi1.5 (AI et al., 2024), Phi3 (Abdin et al., 2024), DeepSeek (DeepSeek-AI, 2024). Specific model versions are presented in Appendix 3.

4.2 Setups

For each capability testing, we carefully design the prompt to optimize the model performance under one-shot setting. Detailed prompt templates for each category can be found in Appendix D.

We use OpenCompass (Contributors, 2023) as the evaluation framework and vLLM (Kwon et al., 2023) to accelerate inference speed.

5 Key Insights

In this section, we will delve deeper into the evaluation results and provides some insights of table-related capabilities of LLMs.

Evaluation results The evaluation results of all selected models on TableBench are presented in Table 1 and Figure 3. It is not surprising to find that closed-source models outperform most of the open-source and table-specific models in terms of overall scores, where GPT-4 and GPT-4o receive the same highest score of 0.57. Llama-3-70B-Instruct is the best-performance open-source model which reaches 0.52. However, it is surprising to find that

Table 1: LLMs Evaluation on **TableBench** with Their Highest Score among All Formats. #1 and #2 represent the fine-tuned table formats of TableLlama and TableLLM respectively. For each metric in each section, the **bold** number indicates the highest performance.

					Total					
Model	**Structure**	**Large-table**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**	**Total**
Proprietary Models										
GPT-3.5	0.43 (json)	0.10 (json)	0.54 (html)	0.95 (xml)	0.40 (grid)	0.63 (latex)	0.35 (grid)	0.44 (md)	0.54 (md)	0.49
GPT-4	0.47 (md)	0.23 (json)	0.64 (json)	**0.97** (xml)	0.38 (xml)	0.75 (json)	0.54 (json)	**0.54** (json)	**0.58** (json)	**0.57**
GPT-4o	**0.51** (md)	**0.24** (json)	**0.66** (html)	0.94 (json)	**0.46** (md)	**0.76** (xml)	**0.59** (md)	**0.54** (xml)	0.43 (html)	**0.57**
Table-Specific Models										
TableLlama	0.08 (#1)	0.01 (#1)	0.31 (#1)	0.06 (#1)	0.01 (#1)	0.37 (#1)	0.17 (#1)	**0.46** (#1)	0.15 (#1)	0.18
TableLLM-7B	0.24 (#2)	**0.03** (json)	0.35 (#2)	0.15 (html)	**0.27** (html)	0.30 (#2)	0.17 (#2)	0.35 (#2)	0.17 (#2)	0.23
TableLLM-13B	**0.27** (#2)	**0.03** (json)	**0.38** (xml)	**0.80** (html)	**0.27** (html)	**0.47** (json)	**0.22** (#2)	0.36 (#2)	**0.24** (latex)	**0.34**
Open-Source Models										
Llama-2-7B-Chat	0.27 (json)	0.01 (json)	0.27 (latex)	0.70 (latex)	0.13 (latex)	0.33 (json)	0.13 (latex)	0.34 (latex)	0.04 (grid)	0.25
Mistral-7B-Instruct-v0.2	0.27 (json)	0.07 (html)	0.34 (xml)	0.74 (html)	0.30 (md)	0.43 (grid)	0.18 (md)	0.33 (grid)	0.16 (json)	0.31
Mistral-7B-Instruct-v0.3	0.42 (html)	0.10 (html)	0.43 (xml)	0.90 (xml)	0.41 (xml)	0.47 (xml)	0.25 (latex)	0.43 (html)	0.38 (md)	0.42
Qwen-2-7B-Instruct	0.39 (html)	0.09 (latex)	0.45 (xml)	0.89 (xml)	0.33 (xml)	0.57 (json)	0.32 (latex)	0.41 (xml)	0.60 (html)	0.45
Phi-3-Small-8k-Instruct	0.36 (json)	0.06 (json)	0.40 (grid)	0.91 (xml)	0.36 (md)	0.58 (json)	0.33 (md)	**0.57** (html)	**0.61** (md)	0.46
Phi-3-Small-128k-Instruct	0.35 (latex)	0.07 (json)	0.29 (grid)	0.90 (xml)	0.37 (xml)	0.45 (html)	0.16 (json)	0.38 (grid)	0.57 (md)	0.39
Deepseek-7B-chat	0.27 (latex)	0.02 (json)	0.33 (latex)	0.80 (xml)	0.19 (grid)	0.34 (json)	0.15 (latex)	0.35 (json)	0.43 (latex)	0.32
Llama-3-8B-Instruct	0.37 (xml)	0.05 (json)	0.37 (xml)	0.85 (xml)	0.41 (latex)	0.51 (xml)	0.29 (json)	0.43 (grid)	0.56 (md)	0.43
Yi-1.5-9B-Chat	0.35 (json)	0.07 (latex)	0.46 (md)	0.83 (md)	0.32 (json)	0.51 (latex)	0.24 (json)	0.46 (json)	0.56 (md)	0.42
Llama-2-13B-chat	0.30 (json)	0.01 (json)	0.33 (xml)	0.78 (md)	0.18 (md)	0.35 (json)	0.15 (json)	0.37(xml)	0.06 (grid)	0.27
Phi-3-Medium-4k-Instruct	0.30 (json)	0.03 (json)	0.48 (json)	0.84 (md)	**0.44** (md)	0.56 (md)	0.30 (md)	0.46 (latex)	0.53 (xml)	0.44
Yi-1.5-34B-Chat	0.36 (json)	0.07 (json)	0.50 (json)	0.84 (json)	0.27 (md)	0.60 (json)	0.32 (md)	0.49 (html)	0.49 (html)	0.45
Mixtral-8x7B-Instruct	0.44 (html)	**0.16** (html)	0.43 (html)	0.83 (html)	0.39 (html)	0.53 (html)	0.33 (html)	0.41 (latex)	0.16 (grid)	0.41
Llama-3-70B-Instruct	**0.46** (md)	0.14 (json)	**0.56** (xml)	**0.93** (xml)	0.35 (json)	**0.68** (grid)	**0.46** (html)	0.49 (md)	**0.61** (md)	**0.52**

Phi3-Small-8k-Instruct not only surpasses models with similar parameters, such as Mistral-7B-Instruct-v0.3 and Qwen-2-7B-Instruct, but also outperforms larger LLMs like Mixtral-8x7B-Instruct and Yi-1.5-34B-Chat. Also, the table-specific models provide unsatisfactory results.

Format understanding capability We conduct holistic experiments of applying different formats to all the capability tests. Generally, it can be observed that different models exhibit varying preferences for formats, and the formats of the best results in various capability tests also differs. However, We have observed that JSON is the preferred format for large-table capability testing, as 14 out of 20 models deliver the best results with this format compared to the other five. And XML and HTML format shows advantages in knowledge understanding testing. Besides, it is found that models within the same series exhibit similar format preferences, such as GPT-series, llama-series, mistral-series, phi-series in largetable capability testing.

Largetable capability analysis Largetable testing poses great challenges to the tested models as is shown in their relatively low scores. We specially select 5 models that claim to support long context ranging from 16k to 131k, and provide the accuracy of their answers to input with varying table lengths (see Table 2). For questions of less than 4k length, the models can answer a a portion of questions correctly, with GPT-4o achieving an accuracy of 56%. However, for data exceeding 4k in length, the performance of all models, except for GPT-4o, deteriorates significantly, with most models unable to provide correct answers for data exceeding 32k in length. Our questions primarily focus on tasks such as determining the number of columns or rows, or outputting the column name of a specified cell. The models lose their largetable understanding capability as the table length increases, even though most of the data falls within the context window of the models. This indicates that there is much room for improvement of extending model context window while maintaining the model's table processing capability.

Table 2: The Results (Accuracy) of Five Long Context Models in Largetable Testing of TableBench. "Avg" is the average score over five lengths. The highest average score is bold, and the highest score in each length is underlined.

Model	<4k	4-8k	8-16k	16-32k	>32k	Avg
Yi-1.5-34B (16k)	0.32	0.00	0.00	0.01	0.00	0.07
Mistral-7B (32k)	0.24	0.08	<u>0.16</u>	0.01	0.00	0.10
GPT-4o (128k)	<u>0.56</u>	<u>0.55</u>	0.06	0.01	0.00	**0.24**
Phi-3-7B (128k)	0.15	0.03	0.03	0.04	<u>0.01</u>	0.07
Qwen-2-7B (131k)	0.26	0.05	0.06	<u>0.07</u>	0.00	0.09

Dependencies among different capabilities In Table 1, we observe dependencies among a model's various capabilities. Primarily, a positive correlation is observed between the commanding capability and the model's comprehension of table structures. Broadly speaking, as the understanding capability of table structures improves, the commanding capability will improve correspondingly.

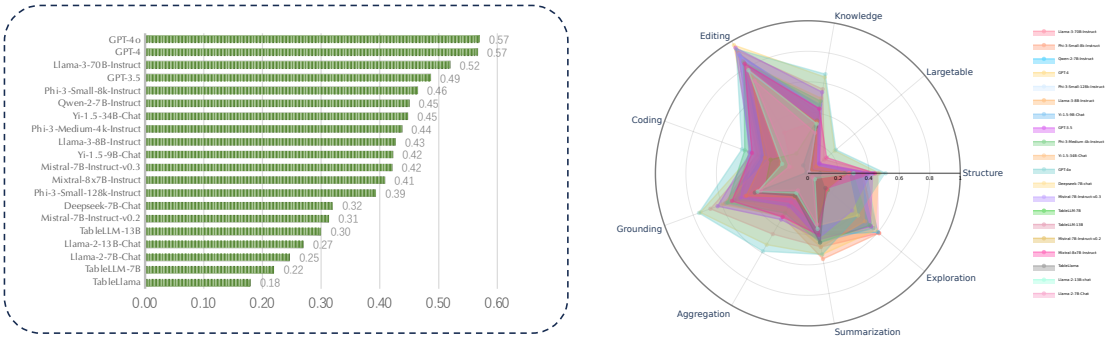


Figure 3: Total Evaluation Scores (on the left figure) and Evaluation Scores of Each Capabilities (on the right figure).

This correlation stems from the fact that table operations frequently involve modifications to rows and columns, necessitating a clear grasp of the table's structure for accurate manipulation.

Furthermore, from a holistic perspective, the aggregation capability and summarization capability exert influence on the model's exploration capability. Models with high proficiency in these two abilities tend to exhibit enhanced exploration capabilities. This phenomenon suggests that when addressing queries requiring the provision of insights, trends, or the identification of hidden patterns, a model requires the support from integrating and summarizing related information. However, a few models displayed subpar performance in the exploration capability test. Through error analysis, we determine that this is partly attributable to the models' inadequate capabilities and partly due to their failure to effectively adhere to the prompt instructions. This lead to output answers that could not be recognized, ultimately resulting in them being labeled as incorrect.

Table-specific models' capabilities We evaluation three table specific models on TableBench: TableLlama, TableLLM-7B, and TableLLM-13B. TableLlama is a fine-tuned version of Llama2 (7B) trained on TableInstruct (Zhang et al., 2023b), and TableLLM is based on CodeLlama. In Figure 5, we apply different formats, including the models' original ones, to evaluate the models. It is observed that TableLlama is sensitive to the table formats and finetuning prompts. After testing with our standardized six formats, TableLlama-7B only receives a score of 0.06. However, when using the original prompts and table formats, several scores of model's capabilities greatly improve with overall score from 0.06 to 0.18. A similar pattern is observed with the TableLLM-7B model, likely indicating smaller parameter models tend to have a stronger preference for fine-tuned formats. Based on the fine-tuning dataset provided in the TableLlama paper, it is further evident that fine-tuned models exhibit a shift in capabilities towards the requirements of the fine-tuning dataset, resulting in a loss of generalization. However, this loss of capability can be mitigated as the model's parameter size increases. For instance, when the parameter count of TableLLM increased to 13B, the model's performance using the standardized format is comparable to, if not better than, its performance on its specific format, particularly excelling in the commanding capability.

6 Conclusion

In this paper, we introduce TableBench, a new capability-based benchmark crafted to assess the performance of LLMs on tabular data. Our framework intricately outlines 10 essential capabilities required from the point a model receives a table-related input to the generation of an output, with each capability tested across 6 table formats. We conduct comprehensive evaluations on 20 models using TableBench and observe that GPT-4 and GPT-4o achieve the highest scores, while phi3-small outperform other open-source models of similar scale. Based on the evaluation, we find that the JSON format is an optimal choice for LLMs to process large tables and those models that claim to be able to process long context window do not perform well on tabular data. We also observe dependencies among different capabilities. Our benchmark and conclusions can serves as a pivotal reference for future table-related LLM research.

Limitations

The primary limitation of this work lies in the unexplored capabilities. For instance, in the current editing and analysis tasks, we have mainly explored the performance on column-major simple tables and have not addressed more complex structured tables, such as those with merged cells or multi-tiered headers. This limitation is also reflective of the field of tabular tasks, where there is limited exploration of complex structured tables in downstream tasks. There are few appropriate datasets and benchmarks available that facilitate such exploration. This represents a future direction that the community in this field can collectively investigate.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Qin Cai, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Yen-Chun Chen, Yi-Ling Chen, Parul Chopra, Xiyang Dai, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Victor Fragoso, Dan Iter, Mei Gao, Min Gao, Jianfeng Gao, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Ce Liu, Mengchen Liu, Weishung Liu, Eric Lin, Zeqi Lin, Chong Luo, Piyush Madan, Matt Mazzola, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Xin Wang, Lijuan Wang, Chunyu Wang, Yu Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Haiping Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Jianwei Yang, Ziyi Yang, Yifan Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2024. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. Finqa: A dataset of numerical reasoning over financial data. *Proceedings of EMNLP 2021*.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2021. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

V. Cutrona, F. Bianchi, E. Jimenez-Ruiz, and M. Palmonari. 2020. Tough tables: Carefully evaluating entity linking for tabular data. In *The Semantic Web - ISWC 2020*, Lecture Notes in Computer Science, pages 328–343. Springer International Publishing.

Accepted paper will be published by Springer as part of the Lecture Notes in Computer Science series.

DeepSeek-AI. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: table understanding through representation learning. *Proceedings of the VLDB Endowment*, 14(3):307–319.

Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. Tablesense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 69–76.

Majid Ghasemi Gol, Jay Pujara, and Pedro Szekely. 2019. Tabular cell classification using pre-trained cell embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 230–239. IEEE.

Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, et al. 2024. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18206–18215.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, et al. 2024. Infiagent-dabench: Evaluating agents on data analysis tasks. *arXiv preprint arXiv:2401.05507*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and ZHAO-XIANG ZHANG. 2024a. Sheetcopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *Preprint*, arXiv:2310.09263.

Zheng Li, Xiang Chen, and Xiaojun Wan. 2024b. Wikitableedit: A benchmark for table editing by natural language instruction. *Preprint*, arXiv:2403.02962.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Weizheng Lu, Jiaming Zhang, Jing Zhang, and Yueguo Chen. 2024. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

OpenAI. 2023a. `https://chat.openai.com.chat`.

OpenAI. 2023b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

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

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, Tao Zhang, Chen Zhou, et al. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023a. Tablellama: Towards open large generalist models for tables. *Preprint*, arXiv:2311.09206.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023b. Tablellama: Towards open large generalist models for tables. *Preprint*, arXiv:2311.09206.

Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao, Juanzi Li, and Jie Tang. 2024. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *Preprint*, arXiv:2403.19318.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2019. Image-based table recognition: data, model, and evaluation. *arXiv preprint arXiv:1911.10683*.

A Model Versions

B Extra Experiments

C Metric Details

C.1 TEDS

Tree-Edit-Distance-Distance-Based (TEDS) metric is proposed by Zhong et al. (2019) to measure the semantic and structural similarity between two

Model	Version
GPT-3.5	gpt-3.5-turbo-0125
GPT-4	gpt-4-1106-preview
GPT-4o	gpt-4o-2024-05-13
TableLlama	osunlp/TableLlama
TableLLM-7B	RUCKBReasoning/TableLLM-7b
TableLLM-13B	RUCKBReasoning/TableLLM-13b
Llama-2-7B-Chat	meta-llama/Llama-2-7b-chat-hf
Llama-2-13B-Chat	meta-llama/Llama-2-13b-chat-hf
Llama-3-8B-Instruct	meta-llama/Meta-Llama-3-8B-Instruct
Llama-3-70B-Instruct	meta-llama/Meta-Llama-3-70B-Instruct
Mistral-7B-Instruct-v0.2	mistralai/Mistral-7B-Instruct-v0.2
Mistral-7B-Instruct-v0.3	mistralai/Mistral-7B-Instruct-v0.3
Mixtral-8x7B-Instruct	mistralai/Mixtral-8x7B-Instruct-v0.1
Phi-3-Small-8k-Instruct	microsoft/Phi-3-small-8k-instruct
Phi-3-Small-128k-Instruct	microsoft/Phi-3-small-128k-instruct
Phi-3-Medium-4k-Instruct	microsoft/Phi-3-medium-4k-instruct
Qwen-2-7B-Instruct	Qwen/Qwen2-7B-Instruct
Yi-1.5-9B-Chat	01-ai/Yi-1.5-9B-Chat
Yi-1.5-34B-Chat	01-ai/Yi-1.5-34B-Chat
Deepseek-7B-Chat	deepseek-ai/deepseek-llm-7b-chat

Table 3: The Huggingface/Offical Model Version Names of All the Tested Models.

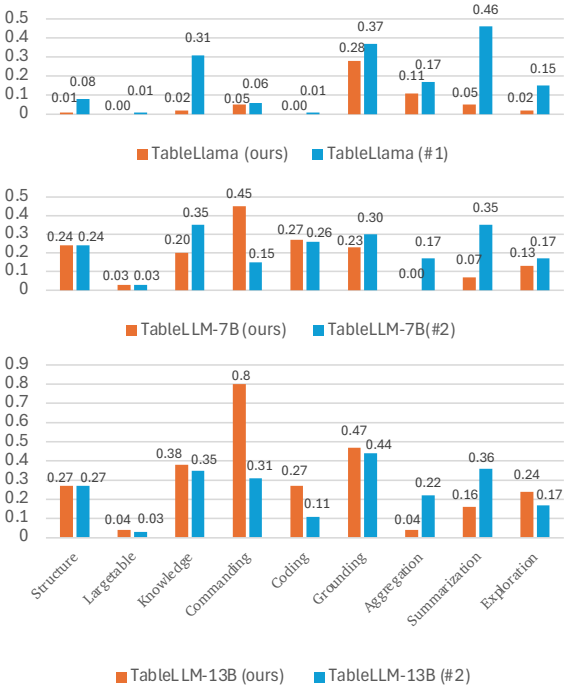


Figure 4: Comparison of Evaluation Results on Our Formats and the Table-Specific Model Formats.

tables. It takes the cost of insertion and deletion operations as 1. When the edit is substituting a node n_o with n_s, the cost is 1 if either n_o or n_s is not `td` (table cells in HTML). When both n_o and n_s are `td`, the substitution cost if 1 if the column span or the row span of n_o and n_s is different. Otherwise, the substitution cost is the normalized Levenshtein similarity (Levenshtein et al., 1966) ($\in$ [0,1]) between the content of n_o and n_s. Therefore, TEDS between two trees is computed as

$$TEDS(T_a, T_b) = 1 - \frac{EditDist(T_a, T_b)}{max(|T_a|, |T_b|)}$$

where $EditDist$ denotes tree-edit distance, and $|T|$ is the number of nodes in T.

C.2 CTA approximate accuracy

For CTA task in knowledge understanding category, we utilize approximate accuracy as the metric, which is defined as:

$$Approximate\ Accuracy = \frac{\sum_{a \in all\ samples} g(a)}{\#all\ samples}$$

$$g(a) = \begin{cases} 1.0, & \text{if } a \text{ is a perfect annotation (PA)} \\ 0.8^{d(a)}, & \text{if } a \text{ is an ancestor of PA and } d(a) < 5 \\ 0.7^{d(a)}, & \text{if } a \text{ is a descendent of PA and } d(a) < 3 \\ 0, & \text{otherwise} \end{cases}$$

where, # denotes the number, $d(a)$ is the depth to the perfect annotation. E.g., $d(a) = 1$ if a is a parent of the perfect annotation, and $d(a) = 2$ if a is a grandparent of the perfect annotation.

D Prompt Templates

E Format Examples

F Different Format Results

TableBench - Structure Understanding

Instruction: Given a table, please answer the question based on the table. Do NOT add any explanations. Return the final result as the JSON format: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Question: Does this table have a header? Return the final result as the JSON format: {'prediction': '<answer>'}.

Answer: {'prediction': 'yes'}

Table: <serialized input table>

Answer:

TableBench - Largetable Understanding

Instruction: Given a table, please answer the question based on the table. Do NOT add any explanations. You only need to output a number in a JSON format: {'prediction': '<answer>'}, such as {'prediction': '3'}

Table: <example serialized table>

Question: How many columns are there in the table? Return the final result as the JSON format: {'prediction': '<answer>'}.

Answer: {'prediction': '3'}

Table: <serialized input table>

Answer:

Table 4: Results of All Models on TableBench in XML Format.

XML									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.32	0.07	0.54	0.95	0.39	0.60	0.31	0.42	0.48
GPT-4	0.43	0.11	0.61	0.97	0.38	0.72	0.48	0.52	0.51
GPT-4o	0.43	0.11	0.65	0.94	0.42	0.76	0.54	0.54	0.38
Table-Specific Models									
TableLlama	0.00	0.00	0.00	0.00	0.00	0.04	0.03	0.01	0.02
TableLLM-7B	0.24	0.02	0.14	0.14	0.25	0.21	0.00	0.07	0.08
TableLLM-13B	0.18	0.01	0.38	0.65	0.25	0.41	0.03	0.09	0.08
Open-Source Models									
Llama-2-7B-Chat	0.15	0.01	0.22	0.58	0.07	0.31	0.12	0.30	0.01
Llama-2-13B-Chat	0.19	0.01	0.33	0.77	0.16	0.32	0.14	0.37	0.04
Llama-3-8B-Instruct	0.37	0.03	0.37	0.85	0.39	0.51	0.28	0.41	0.53
Llama-3-70B-Instruct	0.38	0.09	0.56	0.93	0.34	0.67	0.42	0.49	0.60
Mistral-7B-Instruct-v0.2	0.31	0.05	0.34	0.63	0.25	0.39	0.14	0.30	0.12
Mistral-7B-Instruct-v0.3	0.39	0.05	0.43	0.90	0.41	0.47	0.21	0.41	0.32
Mixtral-8x7B-Instruct	0.41	0.10	0.37	0.80	0.38	0.52	0.29	0.35	0.10
Phi-3-Small-8k-Instruct	0.32	0.04	0.34	0.91	0.34	0.54	0.27	0.45	0.59
Phi-3-Small-128k-Instruct	0.28	0.03	0.27	0.90	0.37	0.40	0.13	0.36	0.50
Phi-3-Medium-4k-Instruct	0.24	0.02	0.46	0.80	0.33	0.54	0.25	0.44	0.53
Qwen-2-7B-Instruct	0.31	0.07	0.45	0.89	0.33	0.56	0.30	0.41	0.56
Yi-1.5-9B-Chat	0.25	0.04	0.43	0.79	0.29	0.48	0.22	0.46	0.49
Yi-1.5-34B-Chat	0.33	0.04	0.49	0.56	0.25	0.57	0.30	0.47	0.47
Deepseek-7B-Chat	0.19	0.01	0.32	0.80	0.14	0.31	0.10	0.29	0.42

TableBench - Commanding

Instruction: I want you to act as a data analytics scientist. I will give you a table, a user query, and you need to generate Python code to answer the user query. You should follow the rules:

1. Return the Python code, and wrap the code in ```` ```python ...``` ```` to make it a code block.
2. There is already a DataFrame variable for `table` in the environment, so you should not reassign table=pd.read_html(...) in the code.
3. Stores the final result in the variable result. I will directly utilize the 'result' variable for assessing the subsequent code outcomes.
4. The result must be one variable being one of pd.DataFrame, List[int], List[float], List[str], int, float, str, dict.
5. For Pivot/groupby, Aggregation, Filter, Sort, Add virtual field, Calculation operations, you should use the pandas library.
6. For forecasting, you should use the greykite or prophet library.
7. For chart generation:
7.1 You should use the matplotlib library.
7.2 You should assign a !!!result!!! dictionary: {"x_fields": field_name, "y_fields": [field_name1, field_name2], "chart_type": chart_type} chart_type choose from lineChart, barChart, scatterChart, pieChart.
7.3 You should print the result at the end.
8. For basic insight, there are 7 types: Rank, RankLast, Attribution, Trend, Monotonicity, Outlier, Unimodality.
8.1 There are two descriptions of insight, and you can choose one of them or their values to answer the query:
+ is_insight: bool, whether the insight is true
+ property: dict, the property of the insight. For "Rank", key is "First" and value is str; for "RankLast", key is "Last" and value is str; for "Attribution", key is "Dominator" and value is str; for "Trend", key is "IsIncreasing" and value is bool; for "Monotonicity", key is "Trend" and value is "Increasing" or "Decreasing"; for "Outlier", key is "TemporalOutlierLocation" and value is List[str]; for "Unimodality", key is "Location" and value is List[str].
8.2 The result of your answer to the query needs to refer to the description. The result must be a direct answer to the query, not a direct dict of the above descriptions. The result can be True, "Increasing", "Decreasing" and so on.

Table: <example serialized table>

Question: Who is the tallest boy in class A?

Answer: <modified table>

Table: <serialized input table>

Question: <question>

Answer:

TableBench - Knowledge Understanding

Instruction: Given a table and a target cell, please annotate target cell with an entity of Wikidata. One cell can be annotated by one entity. Return the entity label name, and do !!!not!!! return the entity id. Return the final result as the JSON format without extra explanation: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Target row id: 3
Target column id: 1
Target cell value: ESO 568-22

Answer: {'prediction': '3'}

Table: <serialized input table>

Target row id: <row id>
Target column id: <col id>
Target cell value: <value>

Answer:

TableBench - Grounding

Instruction: Given a table, please answer the question based on the table. Do NOT add any explanations. If there is more than one answer, separate them with commas(,). Return the final result as the JSON format without extra explanation: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Question: how long did it take the 5th place swimmer to finish?

Answer: {'prediction': '43.12'}

Table: <serialized input table>

Question: <question>

Answer:

TableBench - Summarization

Instruction: Given a table, please answer the question with description based on the table. Do NOT add any explanations. Return the final result as the JSON format: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Table page title: Scott Pye
Table section title: Career results
Question: What was the best achievement of Scott Pye in 2012?

Answer: {'prediction': 'In 2012, Scott Pye placed 2nd with Triple Eight Race Engineering, driving a Holden VE Commodore in the 2012 V8 Supercar Series.'}

Table: <serialized input table>

Table page title: Scott Pye
Table section title: Career results
Question: <question>

Answer:

TableBench - Aggregation

Instruction: Given a table, please answer the question based on the table. Do NOT add any explanations. If there is more than one answer, separate them with commas(,). Return the final result as the JSON format without extra explanation: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Question: how many races in the 1950s were not called by bryan field

Answer: {'prediction': '5'}

Table: <serialized input table>

Question: <question>

Answer:

TableBench - Exploration

Instruction: Given a table, please answer the question based on the table. Do NOT add any explanations. If there is more than one answer, separate them with commas(,). Return the final result as the JSON format without extra explanation: {'prediction': '<answer>'}, such as {'prediction': 'China'}.

Table: <example serialized table>

Question: ???

Answer: {'prediction': '5'}

Table: <serialized input table>

Question: <question>

Answer:

Table 5: Results of All Models on TableBench in JSON Format.

JSON									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.43	0.10	0.52	0.47	0.38	0.61	0.33	0.42	0.54
GPT-4	0.42	0.23	0.64	0.91	0.29	0.75	0.54	0.54	0.58
GPT-4o	0.47	0.24	0.64	0.91	0.43	0.75	0.57	0.53	0.42
Table-Specific Models									
TableLLM	0.01	0.00	0.00	0.00	0.00	0.20	0.06	0.03	0.02
TableLlama-7B	0.24	0.03	0.09	0.01	0.25	0.15	0.00	0.07	0.07
TableLlama-13B	0.27	0.04	0.35	0.59	0.24	0.47	0.03	0.11	0.22
Open-Source Models									
Llama-2-7B-Chat	0.27	0.01	0.21	0.58	0.11	0.33	0.09	0.33	0.01
Llama-2-13B-Chat	0.30	0.01	0.29	0.73	0.18	0.35	0.15	0.22	0.00
Llama-3-8B-Instruct	0.37	0.05	0.37	0.45	0.31	0.51	0.29	0.42	0.47
Llama-3-70B-Instruct	0.40	0.14	0.55	0.86	0.35	0.64	0.44	0.48	0.58
Mistral-7B-Instruct-v0.2	0.37	0.05	0.29	0.31	0.27	0.38	0.13	0.29	0.16
Mistral-7B-Instruct-v0.3	0.41	0.06	0.41	0.67	0.30	0.46	0.22	0.41	0.34
Mixtral-8x7B-Instruct	0.42	0.11	0.35	0.53	0.36	0.52	0.30	0.36	0.06
Phi-3-Small-8k-Instruct	0.36	0.06	0.30	0.74	0.33	0.58	0.30	0.47	0.61
Phi-3-Small-128k-Instruct	0.30	0.07	0.25	0.73	0.35	0.38	0.16	0.35	0.54
Phi-3-Medium-4k-Instruct	0.30	0.03	0.48	0.83	0.29	0.53	0.28	0.45	0.50
Qwen-2-7B-Instruct	0.38	0.06	0.44	0.80	0.12	0.57	0.26	0.39	0.57
Yi-1.5-9B-Chat	0.35	0.06	0.42	0.75	0.32	0.50	0.24	0.46	0.50
Yi-1.5-34B-Chat	0.36	0.07	0.50	0.84	0.19	0.60	0.30	0.48	0.47
Deepseek-7B-Chat	0.23	0.02	0.33	0.28	0.16	0.34	0.09	0.35	0.40

Table 6: Results of All Models on TableBench in Latex Format.

Latex									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.40	0.09	0.54	0.79	0.39	0.63	0.33	0.42	0.54
GPT-4	0.44	0.2	0.6	0.83	0.37	0.73	0.51	0.52	0.49
GPT-4o	0.48	0.21	0.62	0.82	0.41	0.75	0.57	0.54	0.38
Table-Specific Models									
TableLLM	0.00	0.00	0.01	0.02	0.00	0.17	0.09	0.04	0.02
TableLlama-7B	0.24	0.03	0.13	0.42	0.24	0.21	0.00	0.07	0.13
TableLlama-13B	0.23	0.01	0.36	0.63	0.23	0.40	0.01	0.11	0.24
Open-Source Models									
Llama-2-7B-Chat	0.27	0.01	0.27	0.70	0.13	0.32	0.13	0.34	0.00
Llama-2-13B-Chat	0.27	0.01	0.28	0.71	0.17	0.35	0.06	0.24	0.03
Llama-3-8B-Instruct	0.37	0.03	0.36	0.72	0.41	0.51	0.27	0.41	0.55
Llama-3-70B-Instruct	0.45	0.08	0.53	0.77	0.32	0.67	0.44	0.47	0.60
Mistral-7B-Instruct-v0.2	0.35	0.05	0.33	0.61	0.28	0.39	0.16	0.30	0.06
Mistral-7B-Instruct-v0.3	0.40	0.06	0.39	0.72	0.38	0.43	0.25	0.42	0.35
Mixtral-8x7B-Instruct	0.39	0.10	0.38	0.66	0.38	0.53	0.31	0.41	0.07
Phi-3-Small-8k-Instruct	0.36	0.04	0.33	0.75	0.33	0.56	0.32	0.46	0.58
Phi-3-Small-128k-Instruct	0.35	0.05	0.24	0.73	0.34	0.28	0.12	0.35	0.47
Phi-3-Medium-4k-Instruct	0.26	0.03	0.45	0.74	0.34	0.53	0.29	0.46	0.49
Qwen-2-7B-Instruct	0.36	0.09	0.43	0.75	0.31	0.55	0.32	0.39	0.58
Yi-1.5-9B-Chat	0.33	0.07	0.43	0.73	0.30	0.51	0.22	0.45	0.52
Yi-1.5-34B-Chat	0.34	0.05	0.49	0.77	0.24	0.55	0.30	0.47	0.46
Deepseek-7B-Chat	0.27	0.01	0.33	0.70	0.15	0.32	0.15	0.33	0.43

Table 7: Results of All Models on TableBench in Grid Format.

Grid									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.30	0.06	0.54	0.87	0.40	0.61	0.35	0.44	0.50
GPT-4	0.40	0.10	0.58	0.89	0.35	0.73	0.51	0.54	0.52
GPT-4o	0.50	0.12	0.63	0.90	0.41	0.76	0.55	0.54	0.37
Table-Specific Models									
TableLlama	0.00	0.00	0.02	0.03	0.00	0.28	0.08	0.05	0.02
TableLLM-7B	0.22	0.01	0.16	0.01	0.19	0.18	0.00	0.06	0.04
TableLLM-13B	0.23	0.01	0.36	0.00	0.23	0.39	0.02	0.10	0.19
Open-Source Models									
Llama-2-7B-Chat	0.18	0.00	0.25	0.41	0.07	0.31	0.12	0.31	0.04
Llama-2-13B-Chat	0.21	0.01	0.32	0.55	0.16	0.31	0.13	0.35	0.06
Llama-3-8B-Instruct	0.36	0.03	0.33	0.79	0.36	0.51	0.27	0.43	0.56
Llama-3-70B-Instruct	0.44	0.09	0.52	0.85	0.32	0.68	0.44	0.49	0.61
Mistral-7B-Instruct-v0.2	0.33	0.03	0.33	0.70	0.29	0.43	0.17	0.33	0.03
Mistral-7B-Instruct-v0.3	0.40	0.05	0.39	0.78	0.37	0.47	0.22	0.40	0.38
Mixtral-8x7B-Instruct	0.39	0.06	0.39	0.76	0.36	0.52	0.30	0.40	0.16
Phi-3-Small-8k-Instruct	0.33	0.03	0.40	0.77	0.34	0.55	0.30	0.46	0.58
Phi-3-Small-128k-Instruct	0.24	0.04	0.29	0.67	0.35	0.39	0.14	0.38	0.54
Phi-3-Medium-4k-Instruct	0.22	0.03	0.46	0.78	0.38	0.51	0.26	0.45	0.49
Qwen-2-7B-Instruct	0.38	0.03	0.42	0.83	0.30	0.54	0.27	0.41	0.57
Yi-1.5-9B-Chat	0.19	0.03	0.41	0.69	0.27	0.46	0.16	0.44	0.50
Yi-1.5-34B-Chat	0.19	0.04	0.49	0.72	0.23	0.52	0.26	0.46	0.48
Deepseek-7B-Chat	0.22	0.01	0.31	0.00	0.19	0.30	0.12	0.26	0.32

Table 8: Results of All Models on TableBench in HTML Format.

HTML									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.26	0.07	0.54	0.88	0.35	0.58	0.34	0.44	0.49
GPT-4	0.34	0.08	0.6	0.9	0.36	0.75	0.52	0.53	0.48
GPT-4o	0.45	0.11	0.66	0.9	0.42	0.74	0.54	0.53	0.43
Table-Specific Models									
TableLLM	0.00	0.00	0.00	0.01	0.00	0.10	0.03	0.02	0.01
TableLlama-7B	0.24	0.01	0.12	0.45	0.27	0.23	0.00	0.07	0.08
TableLlama-13B	0.23	0.01	0.38	0.80	0.27	0.41	0.03	0.16	0.23
Open-Source Models									
Llama-2-7B-Chat	0.11	0.00	0.19	0.66	0.05	0.30	0.11	0.30	0.01
Llama-2-13B-Chat	0.18	0.01	0.31	0.7	0.11	0.29	0.12	0.36	0.05
Llama-3-8B-Instruct	0.36	0.04	0.36	0.83	0.35	0.50	0.26	0.43	0.54
Llama-3-70B-Instruct	0.41	0.10	0.55	0.87	0.34	0.65	0.46	0.49	0.60
Mistral-7B-Instruct-v0.2	0.34	0.07	0.33	0.74	0.28	0.37	0.14	0.30	0.11
Mistral-7B-Instruct-v0.3	0.42	0.10	0.39	0.78	0.38	0.45	0.22	0.43	0.34
Mixtral-8x7B-Instruct	0.44	0.16	0.43	0.83	0.39	0.53	0.33	0.34	0.10
Phi-3-Small-8k-Instruct	0.28	0.04	0.24	0.85	0.32	0.41	0.30	0.57	0.59
Phi-3-Small-128k-Instruct	0.25	0.07	0.27	0.86	0.13	0.45	0.12	0.32	0.08
Phi-3-Medium-4k-Instruct	0.18	0.03	0.47	0.75	0.25	0.48	0.24	0.45	0.52
Qwen-2-7B-Instruct	0.39	0.06	0.42	0.85	0.32	0.55	0.30	0.41	0.60
Yi-1.5-9B-Chat	0.22	0.04	0.42	0.69	0.25	0.47	0.23	0.46	0.55
Yi-1.5-34B-Chat	0.23	0.06	0.48	0.45	0.20	0.52	0.31	0.49	0.49
Deepseek-7B-Chat	0.16	0.01	0.28	0.72	0.13	0.25	0.09	0.23	0.30

Table 9: Results of All Models on TableBench in Markdown Format.

Markdown									
Model	**Structure**	**Largetable**	**Knowledge**	**Commanding**	**Coding**	**Grounding**	**Aggregation**	**Summary**	**Exploration**
Proprietary Models									
GPT-3.5	0.40	0.07	0.52	0.89	0.39	0.62	0.34	0.44	0.54
GPT-4	0.47	0.19	0.61	0.92	0.36	0.74	0.52	0.54	0.56
GPT-4o	0.51	0.17	0.62	0.91	0.46	0.74	0.59	0.53	0.42
Table-Specific Models									
TableLlama	0.00	0.00	0.02	0.05	0.00	0.25	0.11	0.05	0.02
TableLLM-7B	0.24	0.01	0.20	0.03	0.24	0.19	0.00	0.07	0.06
TableLLM-13B	0.27	0.02	0.36	0.00	0.24	0.39	0.04	0.10	0.23
Open-Source Models									
Llama-2-7B-Chat	0.24	0.01	0.22	0.62	0.10	0.32	0.12	0.33	0.03
Llama-2-13B-Chat	0.24	0.01	0.30	0.78	0.18	0.32	0.11	0.36	0.04
Llama-3-8B-Instruct	0.36	0.05	0.31	0.84	0.36	0.51	0.26	0.42	0.56
Llama-3-70B-Instruct	0.46	0.09	0.52	0.88	0.34	0.66	0.43	0.49	0.61
Mistral-7B-Instruct-v0.2	0.34	0.01	0.32	0.68	0.30	0.38	0.18	0.29	0.05
Mistral-7B-Instruct-v0.3	0.38	0.02	0.38	0.81	0.38	0.42	0.24	0.42	0.38
Mixtral-8x7B-Instruct	0.40	0.06	0.34	0.79	0.37	0.52	0.29	0.40	0.07
Phi-3-Small-8k-Instruct	0.32	0.04	0.35	0.82	0.36	0.55	0.33	0.47	0.61
Phi-3-Small-128k-Instruct	0.33	0.03	0.26	0.83	0.36	0.38	0.15	0.37	0.57
Phi-3-Medium-4k-Instruct	0.26	0.03	0.45	0.84	0.44	0.56	0.30	0.45	0.52
Qwen-2-7B-Instruct	0.38	0.04	0.43	0.85	0.29	0.55	0.28	0.38	0.58
Yi-1.5-9B-Chat	0.32	0.05	0.46	0.83	0.30	0.50	0.23	0.45	0.56
Yi-1.5-34B-Chat	0.32	0.04	0.48	0.83	0.27	0.58	0.32	0.48	0.51
Deepseek-7B-Chat	0.23	0.01	0.31	0.00	0.17	0.33	0.14	0.31	0.36

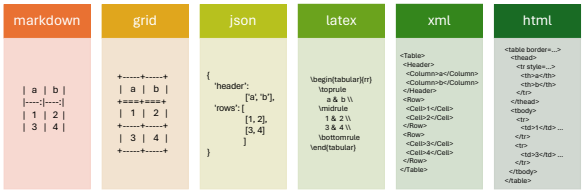


Figure 5: Format Examples.