

TABLELLM: Enabling Tabular Data Manipulation by LLMs in Real Office Usage Scenarios

Xiaokang Zhang^{1*}, Jing Zhang¹, Zeyao Ma^{1*}, Yang Li^{1*}, Bohan Zhang^{1*}, Guanlin Li^{1*}, Zijun Yao², Kangli Xu², Jinchang Zhou², Daniel Zhang-Li², Jifan Yu², Shu Zhao³, Juanzi Li², Jie Tang²

¹ School of Information, Renmin University of China, China

² Computer Science, Tsinghua University ³ Computer Science, Anhui University, China

{zhang2718,zhang-jing,zeyaoma}@ruc.edu.cn,{juanzi,jietang}@tsinghua.edu.cn

{yaozj20,xukl21,zhoujc21,zlenn21,yujf21}@mails.tsinghua.edu.cn,zhaoshuzs2002@hotmail.com

ABSTRACT

We introduce TABLELLM, a robust large language model (LLM) with 13 billion parameters, purpose-built for proficiently handling tabular data manipulation tasks, whether they are embedded within documents or spreadsheets, catering to real-world office scenarios. We propose a distant supervision method for training, which comprises a reasoning process extension strategy, aiding in training LLMs to understand reasoning patterns more effectively as well as a cross-way validation strategy, ensuring the quality of the automatically generated data. To evaluate the performance of TABLELLM, we have crafted a benchmark tailored to address both document and spreadsheet formats as well as constructed a well-organized evaluation pipeline capable of handling both scenarios. Thorough evaluations underscore the advantages of TABLELLM when compared to various existing general-purpose and tabular data-focused LLMs. We have publicly released the model checkpoint, source code, benchmarks, and a web application for user interaction¹.

KEYWORDS

Large language model, Tabular data manipulation

1 INTRODUCTION

A substantial amount of data is routinely structured in tabular formats, a format widely embraced across various industries for different purposes. For instance, they enable bank employees to monitor transactions and detect fraud, assist human resources in managing employee information efficiently, and facilitate government agencies in conducting censuses and surveys for policy-making. While tabular data is ubiquitous, specific table-related tasks can be laborious, error-prone, and require specialized skills. Automating these tasks offers significant benefits to both academic and industrial sectors, attracting considerable interest [4, 10].

Conventional methods for processing tabular data predominantly focus on adapting language model architectures, incorporating elements like position embeddings, attention mechanisms, and learning objectives to encode the inherent structural attributes of tabular data [13, 14, 26, 41]. However, a shift in paradigm has occurred with the rise of large language models (LLMs) like GPT-4 [28], GPT-3.5 [29], and PaLM2 [2]. Recent research emphasizes crafting precise prompts that integrate crucial partial information from provided tabular data and leveraging external programming languages like SQL and Python. This approach facilitates a step-by-step

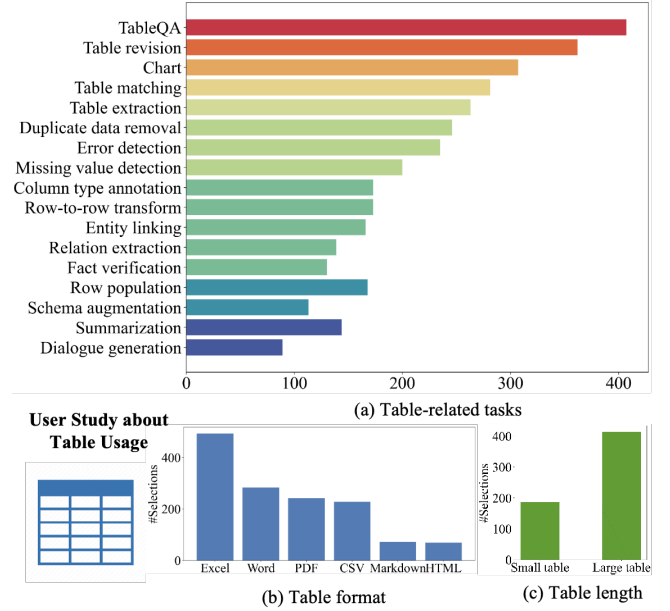


Figure 1: Illustration of the user study about (a) table-related tasks (tableQA, table revision, chart, table matching, duplicate data removal, etc.); (b) table formats (Excel, Word, etc.); (c) table length (Small: < 50 rows, Large: ≥ 50 rows).

Chain-of-thought (COT) [36] inference process by close-sourced LLMs [7, 19, 23, 40, 46]. The availability of open-source LLMs, exemplified by Llama [34], enables the fine-tuning of models for tabular data processing, as demonstrated by TableLlama [44].

Numerous previous studies have focused on improving a model’s reasoning capabilities for table question answering (tableQA) [7, 14, 19, 24, 26, 40, 41, 44, 46]. Moving beyond tableQA, some of these endeavors have also tackled diverse table-related tasks, including fact verification [19, 26, 40, 44, 46], column type annotation [24, 44], table matching [24], schema augmentation [24, 44], and more. However, many of these efforts, while valuable in an academic context, may fall short of reflecting the complete realism of individuals working with tabular data in real-world office scenarios.

To capture authentic insights from office users, we conduct an extensive user study utilizing a questionnaire focused on various tasks related to tables. The questionnaire is distributed to 507 participants across diverse professions, aiming to capture their specific

¹<https://tablellm.github.io/>

*Work was done when interned at Zhipu AI.

requirements in real-world office scenarios. For further details on the user study, please refer to Section 3. The results, depicted in Figure 1, reveal a clear preference among respondents for tasks related to tableQA, table revision, chart creation, table matching. Notably, tables in Excel/CSV and Word/PDF formats, as well as long tables, emerged as the preferred choices among participants.

Challenges. Compared to academically-focused table tasks, real-world office use of tabular data presents two primary challenges. **(1) Diverse Operations:** user preferred tasks involve a wide range of operations, including query, update, merge, and chart, which go beyond the query operations in tableQA. **(2) Unique Processing Approaches for Different Formats:** Word/PDF documents often contain contextual textual data alongside tabular information, allowing for hybrid querying. Excel/CSV spreadsheets, on the other hand, contain regular and long tables, enabling more intricate operations like update and merge.

While existing works either focus on leveraging LLMs’ ability to derive answers directly from their internal parameters, particularly suitable for document-embedded tabular data, or specialize in writing and executing code to obtain answers from spreadsheet tabular data, they each have limitations. The former struggles with long tables and diverse operations in spreadsheets, while the latter fails to handle hybrid queries involving both text and tabular data. In summary, existing works have yet to effectively address both types of tabular data simultaneously, meeting the requirements of real-world office usage.

Our Solution. We present TABLELLM, specifically designed to handle a wide array of table operations encountered in spreadsheet and document usage scenarios, named tabular data manipulation in real office usage scenario. To facilitate model training, we introduce a distant supervision method that complements the reasoning process of existing benchmarks, aiding in training LLMs to understand reasoning patterns more effectively. Additionally, we validate the automatically generated questions and answers through a cross-way validation strategy, ensuring data quality. We also provide a theoretical analysis of the effectiveness of cross-way validation compared to single-answer sampling and same-way validation. Utilizing this distant supervision training data, we fine-tune CodeLlama (13B) [33], resulting in the development of TABLELLM. This model adeptly handles tabular data embedded within documents through an inner-parameter-driven approach and spreadsheet-embedded tabular data via a code-driven method.

A rigorous performance assessment is conducted, involving the collection of primary tableQA test instances from existing benchmarks and the creation of additional table manipulation instances by an annotation team. Given the complex evaluation process under the two scenarios, we design a meticulous evaluation method that considers query, update, merge and chart operations with distinct metrics. **TABLELLM proves to be on par with GPT-3.5 and even outperforms the most capable commercial LLM GPT-4 in the spreadsheet-embedded scenario.**

Impact and Beneficial Groups In the realm of tabular data processing research, our contributions encompass: (1) Addressing a practical problem of tabular data manipulation in real-world office usage scenarios. (2) Presenting techniques that extend reasoning

processing and integrate a cross-way validation strategy to enhance the quality of distant supervision training data. Theoretical proof is provided for the effectiveness of cross-way validation. We firmly believe that TABLELLM holds significant potential to create a positive impact on both industrial developers and users, owing to the following contributions: (3) Delivering a high-quality open-source LLM tailored for tabular data manipulation in both 7B and 13B, thereby enhancing accessibility and fostering collaboration within the community. (4) Offering an online application service to facilitate convenient usage and improve the overall user experience.

2 RELATED WORK

We review table tasks, including basic analysis tasks represented by tableQA, table manipulation, and advanced table data analysis.

TableQA-represented Basic Analysis. Beyond the primary tableQA task, various research endeavors tackle basic table analysis tasks like fact verification, column type annotation, schema augmentation, data-to-text, and more [8, 12–14, 19, 24, 26, 37, 39–41, 44, 46]. These tasks commonly involve tables extracted from the web, typically of relatively short length and interspersed with textual content.

Representation Learning. Many traditional methods, such as TaBERT [41], TAPAS [14], TableGPT [13], Tableformer [39], MATE [12], TUTA [35], Tabbie [18], TABT5 [1], TAG-QA [48] and TURL [8], emphasize intricate encoder design, incorporating various positional encodings and dense/sparse attention mechanisms to represent tables. These methods also integrate reconstruction losses at token, cell, and column levels. TAPEX [26] and GraPPa[42] additionally integrate SQL execution as a pre-training task.

Finetuning LLM. As LLM capabilities progress, researchers are shifting focus from intricate table encoding to gathering ample data for training unified LLMs capable of handling multiple table-related tasks. For instance, TableLlama [44] and TAT-LLM [51] fine-tune the Llama2 (7B) model on various table-related benchmarks. UnifiedSKG [37] further integrates structured data-related benchmarks, like knowledge graph question answering, into the tuning process of the T5 (3B) [32] model to address tasks requiring structured data.

Prompting LLM. Due to the closed-source nature of GPT series LLMs and the high cost associated with fine-tuning these models, researchers have focused on designing effective prompts for GPT series LLMs to enable tableQA analysis tasks [3, 7, 19, 20, 40, 46, 47]. The approach typically involves a multi-step inference process, breaking down the main question into subquestions, invoking external tools like SQL and Python to address these subquestions. For instance, DATER [40] employs SQL, StructGPT [19] and Chain-of-Table [3] use self-defined interfaces/actions, and ReAcTable [46] employs SQL for querying tabular data and Python for handling string manipulation tasks. In contrast to incorporating tool execution results directly into the LLM, Binder [7] integrates the LLM’s generated results back into SQL/Python.

Table Manipulation. A new research direction aims to enhance table manipulation capabilities, particularly focusing on tasks such as insert, update, and delete operations within spreadsheet formats like Excel and CSV, as well as databases [11, 23, 31, 45]. Such tasks often involve working with lengthy and regular tables, making it practical to utilize LLMs alongside tools to address them. For instance, DB-GPT [38], ChatDB [15], C3 [11] and Din-SQL [31]

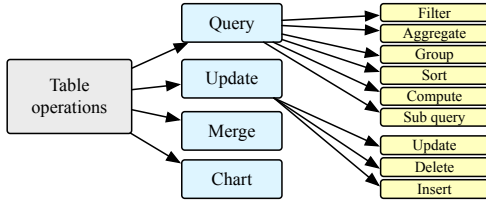


Figure 2: Common operations for tabular data manipulation.

translate questions into SQL queries. SheetCopilot [23] and DataCopilot [45] develop their atomic interfaces based on Excel’s embedded functions and various programming languages like C++ and Python, allowing LLMs to invoke them.

Advanced Analysis. Recently, researchers have redirected their focus towards advanced table data analysis tasks [16, 22]. These tasks involve intricate operations such as correlation analysis, feature engineering, and machine learning. The predominant methods in this area enable LLMs to generate Pandas code, which offers comprehensive support for advanced data analysis, facilitating the handling of these advanced analysis tasks.

Summary. The majority of research still focuses on TableQA-represented basic analysis tasks, with some beginning to explore table manipulation and advanced analysis. Direct inference from LLM parameters is common for basic tasks, while inferring code/APIs and executing their results is favored for table manipulation and advanced analysis. However, current research seldom considers user scenarios. SQL-invoked tasks suit database administrators, while advanced analysis is for data analysts requiring in-depth pattern analysis. For office usage, people prefer both QA tasks on document-embedded tables and manipulation tasks on spreadsheet-embedded tables. Existing methods fall short of fully supporting these needs.

3 USER STUDY AND PROBLEM DEFINITION

3.1 User Study

To gather authentic insights from office users, we conduct an extensive user study involving a questionnaire, focusing on two key aspects: (1) inherent characteristics of tables and (2) exploration of table-related tasks. The questionnaire covers usage frequency of different tables, preferred length, and format options like Excel, CSV, Word, PDF, Markdown, and HTML. For tasks, we design 17 frequently mentioned ones, drawing inspiration from TableLlama [44] and Table-GPT [24]. We’ve distributed the questionnaire to 507 diverse participants, including teachers, students, university administrators, marketing professionals, HR personnel, and R&D specialists. They’re asked about preferred table lengths, formats, and frequently required tasks.

The user study findings, depicted in Figure 1, indicate a clear preference among participants for tasks such as tableQA, table revision, chart creation, and table matching, followed by the data cleaning related tasks, including error detection, duplicate data removal, and missing value detection. Notably, table extraction, focused on format conversion, is in demand but can be handled efficiently by non-LLM tools, thus not considered LLM-related tasks. There’s relatively less demand for tasks like column type annotation, entity

linking, and fact verification. Additionally, tables in Excel/CSV and Word/PDF formats, along with long tables (typically in Excel/CSV format), emerged as the preferred choices among participants. We also present the complete questionnaire in Appendix A.2.

3.2 Problem Definition

Tabular Data refers to data organized in a table or grid format, with rows and columns facilitating efficient organization and access. Each row typically represents a different record, while each column represents a different attribute of the record. On top of it, **document-embedded tabular data** is tabular data integrated into documents, often in compact formats within Word or PDF files, accompanied by textual content for context and explanation, while **spreadsheet-embedded tabular data** refers to tables within spreadsheets, typically in Excel or CSV files, presented in regular and extensive formats.

Operation Definition. In the user study, tableQA tasks are addressed by query operations, table revision tasks by update operations, table matching tasks by merge operations, and chart generation tasks are regarded as standalone operations. Furthermore, data cleaning tasks such as error detection, duplicate removal, and missing value detection can be handled using update operations. In summary, tabular data manipulation tasks can be categorized into four primary operations: query, update, merge, and chart, as detailed in Figure 2. The “query” operation selects desired data, encompassing filter, aggregate, group, sort, compute, and subquery functionalities, effectively addressing most tableQA instances. The “update” operation modifies or deletes existing data and adds new data. “Merge” operation combines two tables into one. Lastly, the “chart” operation visualizes table content using graphical representations such as bar, pie, or line charts. These operations serve as a guide for generating supervision data, as discussed in Section 4.1.

PROBLEM 1. *Tabular Data Manipulation in Real Office Usage Scenarios* focuses on developing an LLM that can perform a range of query, update, merge, and chart operations with tabular data embedded in documents and spreadsheets.

For document-embedded tabular data, querying specific information is the primary user need, while for spreadsheet-embedded tabular data, users often require querying, data modification, and chart creation. Tasks for document-embedded data suit the LLM’s inner parameters due to its text and tabular data handling proficiency, whereas spreadsheet-embedded tasks demand a more intricate, code-driven approach for effective manipulation.

4 TABLELLM

The overview design of TABLELLM is shown in Figure 3, which consists two primary aspects: **(1) Distant Supervision Data Construction.** The development of distant supervision data involves the integration of both existing benchmark training data and new questions and answers generated from available tabular data. To enhance the training of LLMs, we suggest expanding the reasoning processes within benchmark data. This includes text-based reasoning for queries on document-embedded tabular data and code-based reasoning for manipulations of spreadsheet-embedded tabular data. Additionally, to assure the quality of the automatically generated

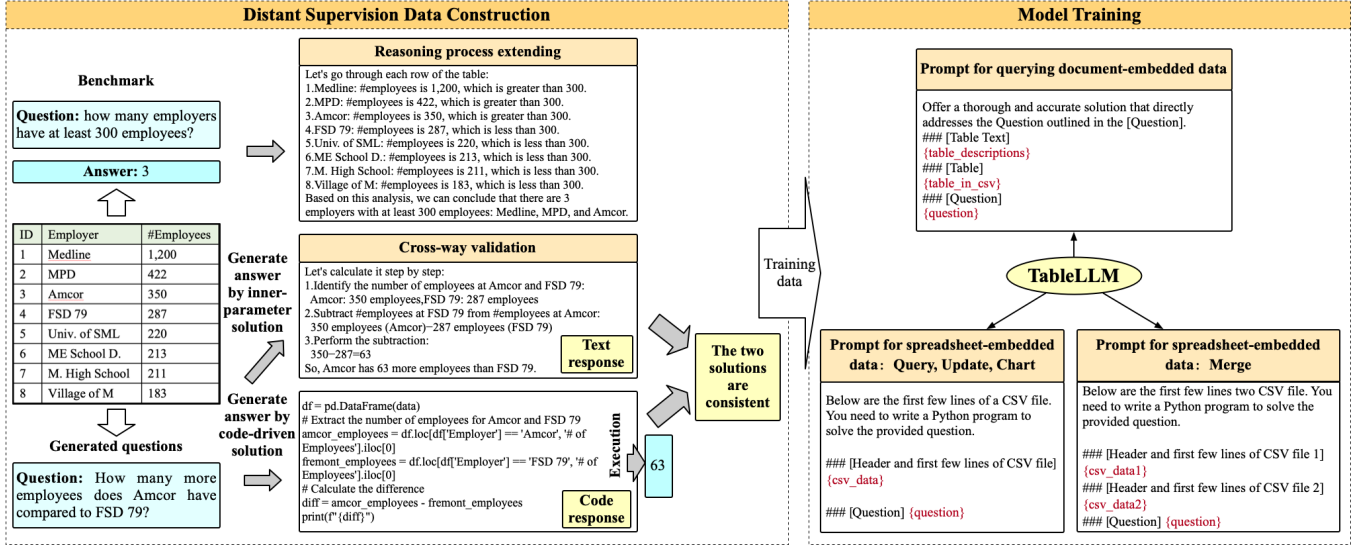


Figure 3: Overview illustration of TABLELLM. The construction of distant supervision data involves two key steps: (1) expanding the reasoning processes based on (question, answer) pairs from existing benchmarks, and (2) cross-way validation of generated (question, answer) pairs. Model training necessitates unique prompts tailored to operations in different scenarios.

training data, we introduce a cross-way validation strategy. This strategy utilizes diverse solution methods for cross-validation, ensuring the reliability and accuracy of the data; (2) **Model Training**. The training of the model utilizes distinct prompts for document-embedded and spreadsheet-embedded tabular data.

4.1 Distant Supervision Data Construction

Extending Reasoning Process for Existing Benchmarks. While existing benchmarks offer ample training data for tableQA, the simple short answers provided by individual instances fall short for tackling complex tabular data manipulation tasks, which often demand intricate reasoning processes to derive answers effectively. Therefore, we augment existing benchmarks by enriching their reasoning processes to facilitate the training of LLMs.

Primarily, to address queries on document-embedded tabular data, we gather training data from widely-adopted tableQA benchmarks including WikiTQ [30], FeTaQA [27], and TAT-QA [50]. Inspired by CoT [36], We extend the provided short answers by presenting GPT-3.5 with the (question, answer) pairs and instructing it to enhance the reasoning process. This augmentation is represented in textual form, rather than as code, to align with the nature of queries involving hybrid text and tabular data inputs. We conjecture that expanding on the reasoning process beyond the short answers during training could enhance the reasoning ability of LLMs. Notably, for WikiTQ and FeTaQA that solely provide tabular data, we supplement them by generating table descriptions using GPT-3.5. Due to the inner-parameter-driven technique employed, we impose a constraint on the length of input tables, limiting them to a token count of fewer than 500. To validate the quality of these text-based reasoning processes, we utilize CritiqueLLM [21], an LLM model specialized in rating, to assess the consistency between the reasoning process and the answers provided in the benchmarks.

Furthermore, to handle queries on spreadsheet-embedded tabular data, we compile training data from two Text2SQL benchmarks: WikiSQL [49] and Spider [43]. Given that spreadsheet-embedded tabular data manipulation primarily involves pure tabular data inputs and complex table manipulations, it aligns more with code-driven techniques. Thus, we select training instances from WikiSQL and Spider, as they correspond to SQL queries. However, instead of directly using the provided SQL queries, we expand pandas code as the reasoning process for each (question, answer) pair by Deepseek [5], a recent powerful code LLM, as Pandas offers greater flexibility to support functionalities such as chart beyond table query, update, and merge. We ensure the quality of the generated code by validating that the executed outcomes align with the provided answers in the benchmarks. Note for Spider, in line with our focus on single-table operations typical in office scenarios, we exclude multi-table queries and those whose SQL queries yield null results, to better reflect real-world applications.

Automatically Generating Training Data by Cross-way Validation. While the training data derived from existing benchmarks is of high quality, the variety of questions and answers, especially the table update, merge, and chart operations they offer is limited. To address this, we introduce a cross-way validation strategy for automatically generating new questions and answers using only the provided tabular data. The detailed process is as follows:

(1) **Question Generation.** We select 5,177 tables from WikiTQ, 5,000 from TAT-QA, and 4,019 from FeTaQA with less than 500 tokens to simulate document-embedded tabular data. For each table, GPT-3.5 generates questions involving single or multiple table query operations, as depicted in Figure 2. GPT-3.5 also creates contextual table descriptions for WikiTQ and FeTaQA-sourced tables, while TAT-QA tables retain their original text context. Furthermore, we select 1,300 long tables from GitTables [17]. For each table, we

generate 20 questions involving various table manipulation operations, as illustrated in Figure 2. Existing benchmarks typically focus on table query operations, so update, chart, and merge operations are all generated. For query, update, and chart operations, we prompt GPT-3.5 for question generation. However, for the merge operation, given its well-defined nature, we directly construct templates to generate the merge question. Appendix A.7 provides the prompts and templates used for question generation.

(2) **Answer Generation and Cross-way Validation.** For questions based on document-embedded tabular data, we employ GPT-3.5 for both answer generation through inner-parameter and code-driven solutions. The generated code is executed to produce an answer, which serves as the reference. CritiqueLLM [21] is used to evaluate the alignment between the inner-parameter-inferred answer and this reference, thereby improving answer quality. Such inner-parameter-driven and code-driven techniques offers diverse solutions, constituting a form of cross-way validation.

This cross-validation approach is inspired by ensemble learning [9], which combines multiple weak learners to create a strong learner. Building on this concept, we conduct an improved theoretical inference to ensure the quality of automatically generated data. Let’s denote Y_a as the event that the first response is correct, Y_b as the event that the second response is correct, Y as the event that both responses are correct, and E as the event that the two responses are consistent. Based on these definitions, we can establish the following theorem:

Theorem 4.1. (1) If A and B are drawn from the same distribution such that $P(Y_a) = P(Y_b) = p > 1/2$, then consistency checking outperforms single inference, i.e., $P(Y|E) \geq P(Y_a)$.
(2) If A and B are further drawn from independent distributions, the effect will be superior (in terms of expectation).

This theorem suggests that when the model’s probability of providing correct answers exceeds 1/2, employing consistency verification is decisively more effective than direct inference. Moreover, in terms of expected performance, utilizing cross-validation with two independent distributions surpasses consistency checks with a single distribution. The proof is provided in Appendix A.3.

Then for questions on spreadsheet-embedded tabular data, we employ GPT-3.5 to generate a pandas code solution, which is then followed by the generation of an alternative code solution using GPT-3.5 again. The accuracy of the executed outcomes from the first code are verified by comparing them with the outcomes of the second code. Given the potential diversity of two coding solutions resulting in the same answers, this dual-coding strategy can be regarded as stemming from different distributions. Thus it also functions as a cross-way validation method, ensuring the reliability of the solutions.

4.2 Model Training

In the scenario of document-embedded tabular data, the input for LLMs includes both the text and the entire content of the table. However, in the case of spreadsheet-embedded tabular data, due to the typically extensive length of the table, only the header and a subset of rows are provided as input to the LLM. The prompt for the merge operation, involving two tables, is distinct and specifically designed. Figure 3 illustrates the specific prompts.

Given the prompt x as input, we enable LLMs to generate either the textual or code solution, collectively denoted as y . Subsequently, the training loss function is defined as:

$$\mathcal{L}(x, y) = - \sum_{i=1}^{|y|} \log \text{TABLELLM}(y_i | x, y_{j < i}),$$

where $\text{TABLELLM}(y_i | x, y_{j < i})$ represents the probability of generating the i -th token of y given x and the preceding tokens of y . This loss function resembles the standard language model loss but exclusively considers the loss computed on y . We hybridize the document-embedded and spreadsheet-embedded training data in a 1:1 ratio, thoroughly shuffle them, and then partition them into batches for training.

The trained single model addresses two types of data sources. Given that code models tend to excel in reasoning-intensive tasks compared to text models [25], combining the two data sources can enhance text-level reasoning with code-level reasoning. Moreover, a single model could alleviate deployment pressure.

4.3 Model Deployment as Web Application

We recently launch our TABLELLM as a web application², with a screenshot shown in Figure 5. The typical workflow is as follows: Users begin by uploading their tabular data embedded in documents (with support for Word and PDF formats) and spreadsheets (supporting Excel and CSV formats). The system utilizes Grobid³ to parse PDF files and python-docx⁴ for Word files, converting them into CSV format for web visualization. Users then enter queries or instructions in the query box. Depending on the type of uploaded document or spreadsheet, appropriate prompts from Figure 3 guide the TABLELLM to generate answers. The response could be a table, a chart, or a textual answer. Additionally, the application offers a feature for merging two tables, where users can upload two spreadsheets and specify the merging conditions in the query box⁵.

We open the application for trial to a diverse group including teachers, students, administrators from universities, marketing professionals, human resources personnel, and research and development specialists. They are encouraged to provide feedback by clicking “Thumbs up” or “Thumbs down”. So far, we have collected 2,000 use cases from users, with 1,560 involving spreadsheet-embedded scenarios (1,869 for single table operations and 131 for double table operations) and 440 for document-embedded scenarios. Among these, we have received 1,473 feedbacks with 1,293 “Thumbs up” and 180 “Thumbs down”, closely aligning with the performance metrics reported in Table 2. We conduct an error analysis in Appendix A.4 for further improvement.

5 EXPERIMENT

5.1 Test Set Creation

We collect test sets from established benchmarks for document-embedded query tasks, including WikiTQ [30], FeTAQA [27], and TAT-QA [50]. Additionally, we incorporate the OTT-QA [6] dataset,

²<https://tablellm.github.io/>

³<https://github.com/kermitt2/grobid>

⁴<https://pypi.org/project/python-docx/>

⁵Currently, the system is configured to support the merging of two tables only.

Table 1: Benchmark (test set) statistics

Scenario	Name	Description	Size
Document -embedded	WikiTQ	<500 tokens & add text	633
	FeTaQA	<500 tokens & add text	753
	TAT-QA	<500 tokens	800
	OTT-QA	<1000 tokens	1,987
Spreadsheet -embedded	WikiSQL	Remove vague questions	1,000
	Spider	Choose single table	512
	Our created	Query/Update/Merge/Chart	1,200
Both	TABLELLM-bench	-	6,885

which features distinct tables and questions compared to our training data, to evaluate the generalization capabilities of our model. For spreadsheet-embedded table tasks, we utilize test sets from WikiSQL [49] and Spider [43], which align with our query operation requirements. We extract the table, question, and answer from each instance, omitting the SQL statement. To ensure quality, we select clearly phrased questions with accurate answers from WikiSQL and discard instances with vague questions or flawed SQL resulting in multiple potential answers or incorrect results.

As no benchmarks exist for table update, merge, and chart operations, we create test set through human annotation. We choose 50 long tables from InfiAgent-DABench [16], ensuring they are entirely distinct from our training data. Following the process outlined in Section 4.1, we generate questions and answers. An annotator team verifies and corrects the generated content, including answers, codes, and operation types. They execute the code to ensure functionality and check if the answers align with the questions, making adjustments as needed. Since initial questions lack linguistic diversity, we utilize five prominent models from Huggingface for rewriting to enhance variety. This results in a composition of 10% original questions and 90% rewritten ones, with each model contributing to 18% of the rewrites. Human annotators also manually review the questions to retain essential information and avoid irrelevant additions. Table 1 displays the benchmark statistics.

5.2 Evaluation Approach

Given the diverse range of operation types in our dataset, we have adopted a categorized evaluation approach to assess the performance of models across different operations:

- **Query operations:** For the answers obtained through code execution, we conduct an exact match comparison between the model’s output and the ground truth answers to determine correctness. However, for answers directly inferred via inner-parameters, we rely on CritiqueLLM [21] to assign a score from 1 to 10 by comparing the model’s output with the ground truth answers, with a score threshold of 7 considered correct. This is because the generated answers are often lengthy and challenging to precisely match. We also conduct a meta evaluation on CritiqueLLM’s rating scores by humans, obtaining 3% false positive percentage and 4.25% false negative percentage, which highlights the reliability of CritiqueLLM. Details about the meta evaluation is provided in Appendix A.9.

- **Update and merge operations:** As these operations directly modify tables, we require the model’s output to be the complete modified table. We then perform an exact match comparison between the model’s output and the ground truth answers to determine correctness.
- **Chart operations:** Assessing charting operations is challenging through direct answer comparison. Instead, we compare code output by the model with the corresponding code from the ground truth answer. CritiqueLLM is once again employed to compare the model’s output code with the ground truth code, using a score threshold of 5 for evaluation.

Based on the correctness determination, we assess accuracy.

5.3 Comparison Methods

The comparison methods are categorized into four types:

- **Pre-training and fine-tuning LLMs:** This category encompasses models like TaPas [14] (based on BERT) TAPEX [26] (based on BART), and TableLlama [44] (based on Llama2 (7B)).
- **General LLMs:** This group includes GPT-3.5 [29], GPT-4 [28], and Llama2 (13B) [34].
- **Coding-specific LLMs:** This category contains LLMs tailored for coding tasks, including CodeLlama [33] and DeepSeek [5].
- **Prompt-driven LLMs:** This group includes StructGPT [19], ReAcTable [46], Binder [7], and DATER [40], focusing on creating sophisticated prompts to guide LLMs in processing tabular data.

Baseline Selection Principle. We compare with methods featuring fully-maintained codes runnable under Linux Server, thus excluding ReAcTable [46] and SheetCopilot [23]. DataCopilot [45] is also not considered due to its self-designed interfaces limited to certain areas like finance. DIN-SQL [31] and C3 [11] are excluded as they focus on generating SQL and rely on databases. Daagent [16], designed for advanced data analysis, is also excluded as its functionalities do not align with the intended scope of our assessment.

Implementation. (1) TaPas and TAPEX have individual checkpoints trained on WikiTQ and WikiSQL. We assess their performance in document-embedded tabular data scenarios using the WikiTQ-trained versions and in spreadsheet-embedded tabular data scenarios using the WikiSQL-trained versions. As for TableLlama, we evaluate its single checkpoint directly. (2) For both general and coding-specific LLMs, we provide customized prompts for scenarios involving the processing of document-embedded and spreadsheet-embedded tabular data, as detailed in Appendix A.8. (3) Prompt-driven LLMs follow their established prompts. StructGPT, for instance, designs distinct prompts for WikiTQ, WikiSQL, and Spider. We standardize StructGPT’s prompts for WikiTQ, TAT-QA, FeTaQA, OTT-QA, and WikiSQL, aligning them with the prompts used for WikiTQ. Meanwhile, both Binder and DATER use a single unified set of prompts across all benchmarks. (4) TABLELLM is trained using both CodeLlama (7B) and CodeLlama (13B) versions. During inference with our TABLELLM, we consistently apply the same set of prompts used during its training phase. The generated distant supervision data is presented in Table 7 in Appendix A.6.

Table 2: Overall evaluation in both document-embedded and spreadsheet-embedded tabular data scenarios (%)

Model	Document-embedded tabular data				Spreadsheet-embedded tabular data			Average accuracy	Inference times
	WikiTQ	TAT-QA	FeTaQA	OTT-QA	WikiSQL	Spider	Our created		
TaPEX	38.55	–	–	–	83.90	15.04	–	45.83	1
TaPas	31.60	–	–	–	74.20	23.05	–	42.95	1
TableLlama	24.01	22.25	20.47	6.39	43.70	–	–	23.36	1
Llama2-Chat (13B)	48.82	49.63	67.73	61.50	–	–	–	56.92	1
GPT-3.5	58.45	<u>72.13</u>	71.18	60.80	81.70	67.38	77.08	69.82	1
GPT-4	74.09	77.13	78.35	69.50	84.00	69.53	77.83	75.78	1
CodeLlama (13B)	43.44	47.25	57.24	49.72	38.30	21.88	47.58	43.63	1
Deepseek-Coder (33B)	6.48	11.00	7.12	7.44	72.50	58.40	73.92	33.84	1
StructGPT (GPT-3.5)	52.45	27.53	11.80	13.96	67.80	84.80	–	43.06	3
Binder (GPT-3.5)	61.61	12.77	6.85	5.13	78.60	52.55	–	36.25	50
DATER (GPT-3.5)	53.40	28.45	18.26	13.03	58.20	26.52	–	32.98	100
TABLELLM (7B)	58.77	66.88	72.64	<u>63.11</u>	<u>86.60</u>	82.62	<u>78.83</u>	72.68	1
TABLELLM (13B)	<u>62.40</u>	68.25	<u>74.50</u>	62.51	90.70	<u>83.40</u>	80.83	<u>74.66</u>	1

* Underline represents the runner up.

5.4 Overall Experimental Results

Effectiveness. Table 2 displays the overall evaluation in two scenarios. “–” in the table indicates that the method does not support the dataset or that the tested accuracy is too low. The results show that **TABLELLM generally surpasses others in the spreadsheet-embedded scenario and is on par with GPT-3.5 in the document-embedded scenario**. Detailed findings include:

(1) **TaPEX and TaPas show limited performance due to their small model sizes.** These two pre-training and fine-tuning models, utilizing BART and BERT respectively, only demonstrate relatively strong performance on WikiSQL and WikiTQ benchmarks when using their respective trained versions.

(2) **StructGPT, Binder, and DATER’s varying performance across datasets suggests a limitation in the generalization capability of prompt-driven LLMs.** While these models, which generate prompts for tabular data QA tasks, consistently perform well in the WikiTQ benchmark, their performance weakens on other datasets. StructGPT stands out in the Spider benchmark due to its customized prompts tailored for this specific dataset.

(3) **DeepSeek (33B) excels in the spreadsheet-embedded tabular data scenario.** This superior performance is attributed to DeepSeek’s extensive optimization for coding capabilities, enabling proficient code generation for processing spreadsheet-embedded tabular data. However, this specialization in coding proficiency comes at the expense of other abilities, such as direct answer inference from inner parameters.

(4) **Our TABLELLM outperforms both GPT-3.5 and GPT-4 in the spreadsheet-embedded scenario.** Moreover, in our created benchmark with entirely distinct tabular data and questions from the training data, TABLELLM achieves an impressive 80.83% accuracy, showcasing robust generalization ability. Conversely, in the document-embedded scenario, TABLELLM matches GPT-3.5 but slightly trails GPT-4, possibly due to the scenario’s demand for extensive commonsense reasoning with text data, where TABLELLM could benefit from enhanced training in text QA. It’s noteworthy

Table 3: Effect of diverse training data sources (%)

Train data	Document-embedded		Spreadsheet-embedded	
	WikiTQ	TAT-QA	Spider	Our created
CodeLlama (13B)	43.4	47.3	21.9	47.6
Original train data	49.9	53.4	–	–
Extended train data	53.7	62.6	82.0	52.2
Generated train data	51.5	59.8	63.7	80.1
Mixed data	54.7	63.5	84.2	80.9

Table 4: Effect of cross-way validation strategy (%)

Validation strategy	WikiTQ	TAT-QA
Self-check validation	49.4	55.8
Same-way validation	49.6	58.2
Cross-way validation	51.5	59.8

that OTT-QA features entirely different tabular data and questions from the training data, where TABLELLM (7B) surpasses GPT-3.5 by 2.31% accuracy, further demonstrating its generalization prowess.

Efficiency. All methods, except prompt-driven LLMs, require only one inference process per instance. However, Binder necessitates a one-step inference for each instance, requiring 50 samples per step for self-consistency validation. DATER requires four-step inferences for each instance, with self-consistency validation at each step, totaling 100 inferences per instance. StructGPT requires three inferences per question.

5.5 Ablation Studies on Training Data

Effect of Diverse Training Data Sources. We analyze the influence of different training datasets by comparing five distinct training configurations:

- **CodeLlama (13B):** The base version without any training.

- **With original training data of existing benchmarks:** Train CodeLlama using 2,000 training instances from TAT-QA and WikiTQ, then evaluate on corresponding test sets.
- **With extended training data of existing benchmarks:** Train on 2,000 training instances from WikiTQ/TAT-QA, supplemented with extended reasoning process for each instance. Train on 2,000 training instances from Spider, supplemented with extended code, and evaluate on both Spider and our created test sets.
- **With generated training data:** Train on 2,000 generated instances based on WikiTQ/TAT-QA’s tabular data, then test on corresponding test sets. Train on 2,000 generated code-outputted instances based on GitLab’s tabular data, and evaluate on Spider and our created test sets.
- **With mixed data:** Train with a mix of 2,000 extended and 2,000 generated instances from TAT-QA/WikiTQ, then evaluate on corresponding test sets. Train with a mix of 2,000 extended Spider training instances and 2,000 generated code-outputted instances, and evaluate on both Spider and our created test sets.

The results presented in Table 3 demonstrate the effectiveness of incorporating extended reasoning processes, showcasing a performance boost of 3.8% and 9.2% on WikiTQ and TAT-QA respectively, compared to using solely original training data (i.e., question and answer pairs). This improvement is primarily attributed to the inclusion of detailed textual explanations of results, aiding LLMs in recognizing reasoning patterns. Furthermore, the addition of generated data yields an additional 1.6% and 6.4% enhancement in performance over the original training data on WikiTQ and TAT-QA, respectively, emphasizing the value of including answers with reasoning processes. Notably, the combination of both extended and generated training data leads to a significant 4.8% and 10.1% increase in performance relative to using only the original data, highlighting the advantages of integrating diverse data sources. The results observed on Spider and our created test sets further corroborate the benefits of extended and generated training data.

Effect of Cross-way Validation. We examine the effectiveness of our proposed cross-way validation method, which assesses the consistency between direct answer generation and code generation solutions during the automatically generating training data process. We compare it against two other validation methods: **Same-way validation**, which generates two direct answers by the same inner-parameter technique of GPT-3.5 and assesses their alignment using CritiqueLLM, and **Self-check validation**, which enables GPT-3.5 to generate one textual solution and self-check its answer. According to the results presented in Table 4, our cross-way validation method outperforms the other methods. This superior performance is attributed to its use of two distinct responses, leading to more reliable validation results.

5.6 Training Strategy Investigation

We investigate three training aspects: data size, the ratio between document- and spreadsheet-embedded data, and shuffling data strategy. We explore the following variants:

- **Data size:** Options include 1k, 2k, 5k, 10k, 20k, and 40k instances.
- **Data ratio:** The proportion of document- to spreadsheet-embedded data, explored in ratios of 0:10, 2:8, 4:6, 5:5, 6:4, 8:2, and 10:0.

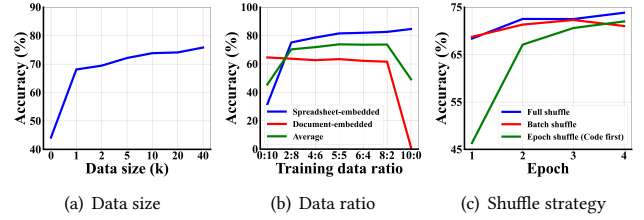


Figure 4: Effects of data size, ratio, and shuffle strategies.

- **Shuffle strategy:** Three approaches – full shuffle (instance-level shuffling), batch shuffle (keeping instances within a batch of the same type and shuffling batches), and epoch shuffle (keeping instances within an epoch of the same type and shuffling epochs).

The default configuration for our experiments is 10k training data instances, a 5:5 data ratio, and full shuffle. When evaluating one factor, the default settings are maintained for the other factors.

Figure 4 illustrates the accuracies of TABLELLM under various training data settings. As depicted in Figure 4(a), performance gains follow a log-linear relationship with the training data size, motivating us to stop early at 40K, which offers a cost-effective balance. Figure 4(b) indicates that a 5:5 ratio yields balanced performance across both data types. Lastly, Figure 4(c) demonstrates that full shuffle and batch shuffle lead to faster convergence than epoch shuffle because epoch lacks a sufficient mixture of the two data sources.

6 CONCLUSION

Our pioneering study introduces a TABLELLM (13B) tailored for tabular data manipulation in real office scenarios. We gather actual requirements from office settings and identify document-embedded and spreadsheet-embedded scenarios. Ensuring high-quality data through extended reasoning processes and cross-way validation on automatically generated training data, the resulting TABLELLM performs comparably to GPT-3.5 and even surpasses GPT-4 in the spreadsheet-embedded scenario. We anticipate that our published dataset, model checkpoint, and code will offer a cost-effective solution for researchers and developers aiming to enhance LLM capabilities for tables and develop diverse table-related applications.

REFERENCES

- [1] Ewa Andrejczuk, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with tabt5. *arXiv preprint arXiv:2210.09162* (2022).
- [2] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. 2023. PaLM 2 Technical Report. *CoRR abs/2305.10403* (2023). <https://doi.org/10.48550/arXiv.2305.10403> arXiv:2305.10403
- [3] Anonymous. 2024. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=4L0xnS4GQM>
- [4] Gilbert Badaro, Mohammed Saeed, et al. 2023. Transformers for Tabular Data Representation: A Survey of Models and Applications. *TACL* (2023).
- [5] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiuqi Du, Zhe Fu, et al. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954* (2024).
- [6] Wenhui Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. 2021. Open Question Answering over Tables and Text. *Proceedings of ICLR 2021* (2021).
- [7] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding Language Models in Symbolic Languages. *ICLR* (2023).
- [8] 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 (2020), 307–319.
- [9] Thomas G Dietterich et al. 2002. Ensemble learning. *The handbook of brain theory and neural networks* 2, 1 (2002), 110–125.
- [10] Haoyu Dong, Zhoujun Cheng, et al. 2022. Table Pre-training: A Survey on Model Architectures, Pre-training Objectives, and Downstream Tasks. In *IJCAI*.
- [11] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. *CoRR abs/2307.07306* (2023). arXiv:2307.07306
- [12] Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. 2021. MATE: Multi-view Attention for Table Transformer Efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 7606–7619.
- [13] Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. TableGPT: Few-shot Table-to-Text Generation with Table Structure Reconstruction and Content Matching. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 1978–1988. <https://doi.org/10.18653/v1/2020.coling-main.179>
- [14] Jonathan Herzig, Paweł 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*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 4320–4333. <https://doi.org/10.18653/v1/2020.acl-main.398>
- [15] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory. *arXiv preprint arXiv:2306.03901* (2023).
- [16] Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. InfiAgent-DABench: Evaluating Agents on Data Analysis Tasks. arXiv:2401.05507 [cs.CL]
- [17] Madelon Hulsebos, Çağatay Demiralp, and Paul Groth. 2023. Gittables: A large-scale corpus of relational tables. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–17.
- [18] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584* (2021).
- [19] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9237–9251. <https://doi.org/10.18653/v1/2023.emnlp-main.574>
- [20] Ziqi Jin and Wei Lu. 2023. Tab-CoT: Zero-shot Tabular Chain of Thought. *arXiv preprint arXiv:2305.17812* (2023).
- [21] Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2023. CritiqueLLM: Scaling LLM-as-Critic for Effective and Explainable Evaluation of Large Language Model Generation. arXiv:2311.18702 [cs.CL]
- [22] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 18319–18345.
- [23] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. SheetCopilot: Bringing Software Productivity to the Next Level through Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=tfyr2zRvOK>
- [24] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. TableGPT: Table-tuned GPT for Diverse Table Tasks. arXiv:2310.09263 [cs.CL]
- [25] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Anyanya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. arXiv:2211.09110 [cs.CL]
- [26] 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*. <https://openreview.net/forum?id=O50443AsCP>
- [27] Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoolkopf, Riley Kong, Xiangru Tang, Muthiah Mutuma, Ben Rosand, Isabel Trindade, Rensree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. FeTaQA: Free-form Table Question Answering. *Transactions of the Association for Computational Linguistics* 10 (2022), 35–49.
- [28] OpenAI. 2023. GPT-4 Technical Report. *CoRR abs/2303.08774* (2023). <https://doi.org/10.48550/arXiv.2303.08774> arXiv:2303.08774
- [29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html
- [30] 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)*, Chengqing Zong and Michael Strube (Eds.). Association for Computational Linguistics, Beijing, China, 1470–1480. <https://doi.org/10.3115/v1/P15-1142>
- [31] Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015* (2023).
- [32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. <http://jmlr.org/papers/v21/20-074.html>
- [33] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D’efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. *ArXiv abs/2308.12950* (2023). <https://api.semanticscholar.org/CorpusID:261100919>
- [34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shriti Bhoale, 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, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). <https://doi.org/10.48550/arXiv.2307.09288> arXiv:2307.09288
- [35] 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*. 1780–1790.
- [36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af07b31abca4-Abstract-Conference.html
- [37] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 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*. Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 602–631. <https://doi.org/10.18653/v1/2022.emnlp-main.39>
- [38] Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, et al. 2023. DB-GPT: Empowering Database Interactions with Private Large Language Models. *arXiv preprint arXiv:2312.17449* (2023).
- [39] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust Transformer Modeling for Table-Text Encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 528–537. <https://doi.org/10.18653/v1/2022.acl-long.40>
- [40] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large Language Models Are Versatile Decomposers: Decomposing Evidence and Questions for Table-Based Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (<conf-loc>, <city>Taipei</city>, <country>Taiwan</country>, </conf-loc>)* (SIGIR '23). Association for Computing Machinery, New York, NY, USA, 174–184. <https://doi.org/10.1145/3539618.3591708>
- [41] 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*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 8413–8426. <https://doi.org/10.18653/v1/2020.acl-main.745>
- [42] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845* (2020).
- [43] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium.
- [44] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023. TableLlama: Towards Open Large Generalist Models for Tables. *arXiv:2311.09206* [cs.CL]
- [45] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2023. Data-Copilot: Bridging Billions of Data and Humans with Autonomous Workflow. *arXiv preprint arXiv:2306.07209* (2023).
- [46] Yunjia Zhang, Jordan Henkel, Avriella Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2023. ReActTable: Enhancing ReAct for Table Question Answering. *arXiv:2310.00815* [cs.DB]
- [47] Bowen Zhao, Changkai Ji, Yuejie Zhang, Wen He, Yingwen Wang, Qing Wang, Rui Feng, and Xiaobo Zhang. 2023. Large Language Models are Complex Table Parsers. *arXiv preprint arXiv:2312.11521* (2023).
- [48] Wenting Zhao, Ye Liu, Yao Wan, Yibo Wang, Zhongfen Deng, and Philip S Yu. 2023. Localize, retrieve and fuse: A generalized framework for free-form question answering over tables. *arXiv preprint arXiv:2309.11049* (2023).
- [49] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).

- [50] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 3277–3287. <https://doi.org/10.18653/v1/2021.acl-long.254>
- [51] Fengbin Zhu, Ziyang Liu, Fuli Feng, Moxin Li, and Tat-Seng Chua. 2024. TAT-LLM: A Specialized Language Model for Discrete Reasoning over Tabular and Textual Data. *arXiv preprint arXiv:2401.13223* (2024).

A APPENDIX

A.1	Web Application	10
A.2	The Survey Details	10
A.3	Verification of Cross-way Validation	11
A.4	Error Analysis	13
A.5	Training Environment and Settings	13
A.6	Training Data for TABLELLM	13
A.7	Prompts for Automatically Generating Dataset	14
A.8	Prompts for Baselines	14
A.9	Meta Evaluation of CritiqueLLM	14

A.1 Web Application

We present a screenshot of the web application deployed with our proposed TABLELLM in Figure 5, where Figure 5(a) shows an instruction with the update operation and Figure 5(b) shows an instruction with the chart operation.

A.2 The Survey Details

We conduct a survey among universities and enterprises to assess users’ needs for tabular data-related tasks in real office environments. We obtain a total of 507 valid responses, representing various roles, including research and development specialists (36.69%), teachers (14.40%), administrators (14.00%), students (12.62%), marketing professionals (4.14%), and human resources personnel (1.97%).

The survey results, depicted in Figure 1, reveal: (a) Participants’ demands for various table-related tasks, with TableQA (80.28%) being the most sought-after, followed by Table revision (71.40%), which involves creating, updating, and deleting tables. Tasks with demand exceeding 200 include TableQA, Table revision, Chart creation, Table matching, Duplicate data removal, Error detection, Missing value detection, and Table extraction. (b) Participants prefer Excel, Word, PDF, and CSV formats, and (c) long tables with more than 50 rows.

Below is the complete survey on table usage:

- (1) What is your occupation?
 - A Student
 - B Teacher
 - C Administrator
 - D Human Resources Professional
 - E Marketing Professional
 - F Research and Development Specialist
 - G Others[Fill in the Blank]
- (2) In your daily work, how often do you work with tables (such as Excel, CSV, or direct access to databases)?
 - A Rarely use (less than once a day on average)

- B Occasionally use (1 to 5 times per day)
 C Frequently use (5 to 20 times per day)
 D Is my work theme (use more than 20 times a day)
- (3) In your normal work, What are the sizes of tables that you typically work with?[Multiple choice question]
 A Tables under 50 rows.
 B Tables over 50 rows.
- (4) What types of tables do you typically encounter and handle in your daily work?[Multiple choice question]
 A Excel
 B Word
 C HTML
 D CSV
 E PDF
 F Markdown
 G Others[Fill in the Blank]
- (5) Which Table manipulation tasks do you need to use in your work?[Multiple choice question]
 A TableQA, e.g.,
 – Find the number of people with grade above 90;
 – Group them according to 90-100 points, 80-90 points, 60-80 points and 60 points, and count the number of people in each score segment;
 – Find all conferences held in Jiangsu in the second half of 2023;
 B Table revision, e.g.,
 – Sort by height column;
 – Convert the Date column to Month/Day/year format;
 – Insert a column “total score”, representing the weighted sum of 60% and 40% from the first to the third column;
 – Delete the “normal score” column;
 C Chart, e.g.,
 – Draw statistical drawings, such as line diagrams, column charts, pie charts;
 D None.
- (6) Which Table cleaning tasks do you need in your work?[Multiple choice question]
 A Missing value detection, e.g.,
 – Detect missing values and fill in the mean value of the corresponding column;
 B Error detection, e.g.,
 – Check a cell whose format is not “month/day/year” and convert it;
 C Delete duplicate data, e.g.,
 – To filter duplicate data by name, only keep the first row with the same name;
 D None.
- (7) Which Table analysis tasks do you need in your work?[Multiple choice question]
 A Column type annotation, e.g.,
 – Given some examples of a column like 1,000 RMB, 1,500 RMB, and 2,000 RMB, name the column;
 B Entity linking, e.g.,
 – Given a candidate combination of column names, assign an appropriate column name to each column in the table;
 C Row-to-row transform, e.g.,
 – Predict the rating of the fourth team based on the “win-loss rating” of the first three teams;
 D Fact verification, e.g.,
 – Based on the content of the table, determine whether “profit growth in the first quarter of 2023 is 10%” is true;
 E None.
- (8) Which Table-to-Text tasks do you need in your work?[Multiple choice question]
 A Summarization, e.g.,
 – Generate a title for the table;
 B Dialogue generation, e.g.,
 – Given the table and the history of the conversation, generate the next conversation;
 C None.
- (9) Which Table augmentation tasks do you need in your work?[Multiple choice question]
 A Row population, e.g.,
 – Given “name”, “age”, “height”, generate specific row data;
 B Schema augmentation, e.g.,
 – Given “Date”, “Growth rate”, “Net income”, expand the other columns;
 C None.
- (10) Do you need Table matching at work? (For example, merge two tables as required)
 A Yes, I need Table matching.
 B No, I don’t need Table matching.
- (11) Do you need Table extraction at work? (For example, organize the Markdown format table into Excel, extract a table from the web page to Excel)
 A Yes, I need Table extraction.
 B No, I don’t need Table extraction.
- (12) Do you have tabular tasks that do not fit into the above categories? If yes, please give an example.[Fill in the Blank]

A.3 Verification of Cross-way Validation

We use Y_a to denote that the first response A is correct, Y_b to denote that the second response B is correct, Y to denote that both responses are correct, and E to denote that the two responses are consistent. We will prove the following:

Theorem A.1. *If A and B come from the same distribution D , $P(Y_a) = P(Y_b) = p > 1/2$, then the consistency check is better than single inference, that is, $P(Y|E) \geq P(Y_a)$.*

Theorem A.2. *If $P(Y_a) = P(Y_b) = p$, A and B are sampled from independent distributions D_A and D_B respectively, the outcome will improve (in terms of expected value),*

$$E[P(Y|E)|Y_a \sim D, Y_b \sim D] \leq E[P(Y|E)|Y_a \sim D_A, Y_b \sim D_B].$$

Lemma A.1. If $\frac{1}{2} \leq p \leq 1$, then $\frac{p^2}{p^2 + (1-p)^2} \geq p$.

PROOF.

$$\begin{aligned} \frac{p^2}{p^2 + (1-p)^2} - p &= \frac{p^2 - p((1-p)^2 + p^2)}{(1-p)^2 + p^2} \\ &= \frac{p(-2p^2 + 3p - 1)}{(1-p)^2 + p^2} \\ &= \frac{p(1-p)(2p-1)}{(1-p)^2 + p^2} \\ &\geq 0. \end{aligned}$$

□

Lemma A.2. If $x_1 + x_2 + x_3 + \dots + x_k = S$ and x_1, x_2, \dots, x_k are non-negative numbers, then

$$x_1^2 + x_2^2 + x_3^2 + \dots + x_k^2 \geq \frac{S^2}{k}.$$

PROOF. According to the Cauchy-Schwarz inequality, we have

$$\begin{aligned} (x_1^2 + x_2^2 + x_3^2 + \dots + x_k^2) &= \frac{1}{k} (1 + 1 + 1 + \dots + 1) (x_1^2 + x_2^2 + x_3^2 + \dots + x_k^2) \\ &\geq \frac{1}{k} (x_1 + x_2 + x_3 + \dots + x_k)^2 \\ &= \frac{S^2}{k}. \end{aligned}$$

□

Lemma A.3. We define \bar{x} to represent the mean of a set of numbers x_1, x_2, \dots, x_n , that is, $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$.

$$\sum_{i=1}^n x_i y_i = \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) + n\bar{x}\bar{y}.$$

PROOF.

$$\begin{aligned} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) &= \sum_{i=1}^n x_i y_i - \sum_{i=1}^n (x_i) \bar{y} - \sum_{i=1}^n (y_i) \bar{x} + \sum_{i=1}^n \bar{x} \bar{y} \\ &= \sum_{i=1}^n x_i y_i - \bar{y} \sum_{i=1}^n (x_i) - \bar{x} \sum_{i=1}^n (y_i) + n\bar{x}\bar{y} \\ &= \sum_{i=1}^n x_i y_i - n\bar{y}\bar{x} - n\bar{x}\bar{y} + n\bar{x}\bar{y} \\ &= \sum_{i=1}^n x_i y_i - n\bar{x}\bar{y}. \end{aligned}$$

Moving terms to the other side of the equation, thus proving. □

PROOF OF THEOREM A.1. According to Bayes' theorem,

$$P(Y|E) = \frac{P(YE)}{P(E)} = \frac{P(YE)}{P(YE) + P(\bar{Y}E)}.$$

Since the two samplings are independent, we have

$$P(Y) = P(Y_a Y_b) = P(Y_a) \cdot P(Y_b) = p^2,$$

$$P(EY) = P(E|Y) \cdot P(Y) = P(Y) = p^2,$$

$$\begin{aligned} P(E\bar{Y}) &= P(E\bar{Y}_a \bar{Y}_b) + P(EY_a \bar{Y}_b) + P(E\bar{Y}_a Y_b) \\ &= P(E\bar{Y}_a \bar{Y}_b) \\ &= P(\bar{Y}_a \bar{Y}_b) P(E|\bar{Y}_a \bar{Y}_b) \\ &\leq P(\bar{Y}_a \bar{Y}_b) \\ &= (1-p)^2. \end{aligned}$$

Then:

$$P(Y|E) = \frac{P(YE)}{P(E)} = \frac{P(YE)}{P(YE) + P(\bar{Y}E)} \geq \frac{p^2}{p^2 + (1-p)^2} \geq p.$$

□

PROOF OF THEOREM A.2.

$$P(Y|E) = \frac{P(YE)}{P(YE) + P(\bar{Y}E)} = \frac{1}{1 + \frac{P(\bar{Y}E)}{P(YE)}}.$$

In order to increase the value above, with the numerator fixed at 1, we need to reduce the denominator, which is as follows:

$$\begin{aligned} \frac{P(\bar{Y}E)}{P(YE)} &= \frac{P(E|\bar{Y}) \cdot P(\bar{Y})}{P(E|Y) \cdot P(Y)} \\ &= \frac{P(E|\bar{Y}) \cdot P(\bar{Y})}{P(Y)} \\ &= P(E|\bar{Y}) \cdot \frac{P(\bar{Y})}{P(Y)} \\ &= P(E|\bar{Y}) \cdot \frac{1 - P(Y)}{P(Y)}. \end{aligned}$$

In the above process, $P(Y)$ represents the probability that the model provides two correct responses, which still equals to

$$P(Y) = P(Y_a Y_b) = P(Y_a) \cdot P(Y_b) = p^2.$$

Therefore, we should minimize $P(E|\bar{Y})$, meaning that if the generated responses are all incorrect, then the probability of them being consistent should be minimized as soon as possible.

Now we only need to consider the probability of the model not producing the correct answer, i.e., the probability of making errors. Assuming there are a total of k types of errors generated by the model, $e_1, e_2, e_3, \dots, e_k$, then $P_a(e_1) + P_a(e_2) + P_a(e_3) + \dots + P_a(e_k) = P_b(e_1) + P_b(e_2) + P_b(e_3) + \dots + P_b(e_k) = 1 - p$. Then,

$$\overline{P_a(e)} = \overline{P_b(e)} = \frac{\sum_{i=1}^k P_a(e_i)}{k} = \frac{\sum_{i=1}^k P_b(e_i)}{k} = \frac{1-p}{k}.$$

If A and B come from the same distribution D , then $P_a(e_i) = P_b(e_i)$, we can obtain

$$P(E|\bar{Y}) = (P_a(e_1))^2 + (P_a(e_2))^2 + \dots + (P_a(e_k))^2.$$

According to Lemma A.2 above, we see that the equation for the same distribution is greater than or equal to $\frac{(1-p)^2}{k}$. Since for each distribution D , it is greater than or equal to this value, then its expected value should also be greater than this value. That is,

$$E[P(E|\bar{Y})|Y_a \sim D, Y_b \sim D] \geq \frac{(1-p)^2}{k}.$$

However, when A and B come from independent distributions D_A and D_B respectively, $P(E|\tilde{Y}) = P_a(e_1)P_b(e_1) + P_a(e_2)P_b(e_2) + \dots + P_a(e_k)P_b(e_k) = \sum_{i=1}^k P_a(e_i)P_b(e_i)$.

According to Lemma A.3,

$$\begin{aligned} & \sum_{i=1}^k P_a(e_i)P_b(e_i) \\ &= k \cdot \overline{P_a(e)} \cdot \overline{P_b(e)} + \sum_{i=1}^k \left((P_a(e_i) - \overline{P_a(e)})(P_b(e_i) - \overline{P_b(e)}) \right) \\ &= \frac{(1-p)^2}{k} + \sum_{i=1}^k \left((P_a(e_i) - \overline{P_a(e)})(P_b(e_i) - \overline{P_b(e)}) \right), \end{aligned}$$

Meanwhile,

$$\begin{aligned} & \sum_{i=1}^k \left((P_a(e_i) - \overline{P_a(e)})(P_b(e_i) - \overline{P_b(e)}) \right) \\ &= k \times \frac{\sum_{i=1}^k \left((P_a(e_i) - \overline{P_a(e)})(P_b(e_i) - \overline{P_b(e)}) \right)}{k} \\ &= k \times \text{Cov}(P_a(e), P_b(e)), \end{aligned}$$

(In the above formula, $\text{Cov}(X, Y)$ represents the covariance of two sets of numbers (X, Y)).

$$\begin{aligned} & E[P(E|\tilde{Y})|Y_a \sim D_A, Y_b \sim D_B] \\ &= \frac{(1-p)^2}{k} + k \cdot E[\text{Cov}(P_a(e), P_b(e))|Y_a \sim D_A, Y_b \sim D_B]. \end{aligned}$$

Given that D_A, D_B are independently distributed, the expected value of the covariance is 0.

$$E[P(E|\tilde{Y})|Y_a \sim D_A, Y_b \sim D_B] = \frac{(1-p)^2}{k}.$$

Compared with when A and B are from the same distribution, we can derive the following inequality:

$$\begin{aligned} & E[P(E|\tilde{Y})|Y_a \sim D, Y_b \sim D] \\ &\geq (1-p)^2/k \\ &= E[P(E|\tilde{Y})|Y_a \sim D_A, Y_b \sim D_B]. \end{aligned}$$

Using the corollary obtained from our previous application of Bayes' theorem, as $P(E|\tilde{Y})$ decreases, $P(Y|E)$ increases, we can ultimately derive the following result. Therefore,

$$E[P(Y|E)|Y_a \sim D, Y_b \sim D] \leq E[P(Y|E)|Y_a \sim D_A, Y_b \sim D_B].$$

□

A.4 Error Analysis

In the part of document-embedded tabular data, we analyze a sample of 170 of the results that are significantly different from ground truth. We classify these errors into four categories and display their corresponding frequencies in Table 5. Question Understanding Error (as exemplified in Figure 6) suggests a lapse in comprehending given question Computational Error (demonstrated in Figure 7) denotes an error occurring during comparison, calculation, or logical operations. Intermediate Answer (depicted in Figure 8) signifies that the model's response is only an intermediate solution and does

Table 5: Error analysis for Document-embedded data

Error Type	Size
Question Understanding Error	146
Computational Error	14
Intermediate Answer	7
Incomplete Answer	3

Table 6: Error analysis for Spreadsheet-embedded data

Error Type	Size
Question Understanding Error	171
Data Type Error	55
Unrunnable Code Error	4

not fulfill the requirements for a final answer. Incomplete Answer (portrayed in Figure 9) indicates that while there may be multiple standard answers, the model only provides a partial response.

In the part of Spreadsheet-embedded tabular data, we analyze a sample of 230 of the results that are significantly different from ground truth, include 64 samples in TableQA category, 126 samples in Table revision category (52 samples in Update category, 51 samples in Insert category, 23 samples in Delete category), 18 samples in Chart category, and 22 samples in Table matching category.

We present the distribution of each type of error in Table 6. Question Understanding Error (illustrated in Figure 10) indicates an error during question comprehension. Data Type Error (demonstrated in Figure 11) suggests that the model mishandles the data type within the dataframe. The Unrunnable Code Error (shown in Figure 12) denotes code generated by the model that does not adhere to Pandas syntax, resulting in code failure.

A.5 Training Environment and Settings

Our experiments are conducted using PyTorch 2.1.2 on a server running the CentOS Linux 7 operating system. The system is equipped with 8 NVIDIA A800 80GB GPUs, an Intel(R) Xeon(R) Platinum 8358 CPU, and 2048GB of RAM.

We set the learning rate to $2e-5$, the batch size per GPU to 4, and accumulate gradients over 4 steps, resulting in a total batch size of 128 across 8 GPU cards.

A.6 Training Data for TABLELLM

Table 7 presents the statistics of the constructed distant supervision data. To train TABLELLM (7B) and TABLELLM (13B), we initially experiment with 4K data, maintaining a 1:1 ratio between document-embedded and code-embedded data sources, which yield promising results on code-embedded test sets. However, given that the backbone model, CodeLlama, is primarily a code model, it may slightly compromise textual reasoning ability. To enhance performance on document-embedded test sets, we include additional training data for this scenario, resulting in a total of 73,157 training instances.

Table 7: Training data statistics

Scenario	Name	Description	Size
Document -embedded	WikiTQ (Extended)	<500 tokens & add text	4,811
	WikiTQ (Generated)	<500 tokens & add text	10,916
	FeTaQA (Extended)	<500 tokens & add text	3,061
	FeTaQA (Generated)	<500 tokens & add text	7,236
	TAT-QA (Extended)	<500 tokens	12,781
	TAT-QA (Generated)	<500 tokens	7,391
Spreadsheet -embedded	WikiSQL (Extended)	Remove vague questions	12,000
	Spider (Extended)	Choose single table	3,374
	Generated	Query/Update/Merge/Chart	11,587
Both	TABLELLM-bench	-	73,157

A.7 Prompts for Automatically Generating Dataset

The prompt presented in Figure 13 is for generating questions for both spreadsheet-embedded and document-embedded training data.

We use templates to automatically generate table merge instructions, which are presented in the Figure 14.

A.8 Prompts for Baselines

To enhance the model’s ability to generate python code in the specific format, we use the prompt shown in Figure 15 and Figure 16 to generate inference from Llama2-Chat (13B), GPT-3.5, GPT-4, CodeLlama (13B) and Deepseek (33B) on Spreadsheet-embedded tabular data. For Document-embedded tabular data, these models use the same prompt as the proposed TABLELLM, with prompts shown in Figure 3. For TableLlama, StructGPT, Binder and DATER, we use the prompts that they have already established.

A.9 Meta Evaluation of CritiqueLLM

We conduct a meta-evaluation of CritiqueLLM [21] through human annotations.

Initially, we sample 400 instances from our test sets. Among them, 200 instances are from the document-embedded test sets, with WikiTQ, TAT-QA, FeTaQA, and OTT-QA each comprising 50 instances, and 200 instances are from the spreadsheet-embedded test sets, with 150 instances for query operation and 50 instances for chart operation. Other operations, including update and merge, are not sampled because they are evaluated by exact match without the need for CritiqueLLM. For each instance in the test set, CritiqueLLM accepts the reference answer and the response of TABLELLM as input and outputs a score from 0 to 10 to reflect how well the assistant’s answer matches the reference answer. The prompt to indicate the scoring criteria to CritiqueLLM and instruct it to score is shown in Figure 17. A response obtaining a score higher than a threshold is considered correct. The threshold for document-embedded and spreadsheet-embedded tabular data is set at 7 and 5, respectively. Then we allow human annotators to score each response using the same scoring criteria as CritiqueLLM. Finally, we compare the human rating results with CritiqueLLM’s rating results and compute the proportion of false positive and false negative data, which refers

to the incorrect responses correctly judged by CritiqueLLM and the correct responses that are mistakenly predicted by CritiqueLLM.

The comparative analysis of outcomes from CritiqueLLM and human scoring results reveals that, within the document-embedded tabular data, CritiqueLLM exhibits a false positive rate of 1% and a false negative rate of 6%. For the spreadsheet-embedded tabular data, the false positive rate noted is 5%, with a false negative rate of 2.5%. Consequently, on the mixed test set, we obtain a 3% false positive rate and a 4.25% false negative rate. The congruence in the distribution of scores between CritiqueLLM and human evaluations substantiates the validity of employing the CritiqueLLM for assessing response quality.

We also analyze the reasons causing the errors of CritiqueLLM. The false positive instances are primarily due to that the response generated by TABLELLM is long, and the reference answer is a proper subset of the response. In this case, CritiqueLLM tends to give a high score to model’s responses regardless of the incorrect responses. Figure 18 and Figure 19 show representative examples of false positives for document-embedded and spreadsheet-embedded tabular data. False negatives are often caused by the model’s response not providing a specific answer at the beginning or end of the response, or the response is too long. Examples of false negatives are illustrated in Figure 20 and Figure 21.

In our method, we also employ CritiqueLLM for judging whether the extended reasoning process is consistent with the reference answers. Thus, false positive instances judged by CritiqueLLM potentially compromises the accuracy of model-generated responses. Fortunately, the ratio of such instances only account for 3%. In contrast, false negative instances do not detract from the model-generated answer’s quality, because these instances are excluded from the extended training data.

TableLLM: Manipulating Tables As the Way You Like

Single Table Operation Double Table Operation

- We will provide you a table and a list of possible questions to ask.
- You can choose one of the provided questions or create your own question to have a conversation with the table.
- You can also upload your own file containing table to start a conversation.

Upload your own file if you like

Drag and drop file here
Limit 200MB per file • CSV, XLSX, XLS, DOCX, PDF

Browse files

Provided table:

Refresh Table

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price	Last Price
0	GODREJIND	EQ	15-May-2017	564.6	581	584	568.5	578
1	GODREJIND	EQ	16-May-2017	578.55	581.45	589	572.25	583
2	GODREJIND	EQ	17-May-2017	584.8	583	594	576.85	584
3	GODREJIND	EQ	18-May-2017	588.6	582	588.85	571.2	572.25
4	GODREJIND	EQ	19-May-2017	574.6	581	585.8	567.55	579.8
5	GODREJIND	EQ	22-May-2017	578	584.45	586.75	562.35	562.35
6	GODREJIND	EQ	23-May-2017	565.95	576	581.55	555.55	558

Filter rows where 'Symbol' equals 'GODREJIND' and 'Series' is 'EQ'.

```
import pandas as pd

# Load the data from csv
df = pd.read_csv('data.csv')

# Filter rows where 'Symbol' equals 'GODREJIND' and 'Series' is 'EQ'
filtered_df = df[(df['Symbol'] == 'GODREJIND') & (df['Series'] == 'EQ')]

print(filtered_df)
```

	Symbol	Series	Date	Prev Close	Open Price	High Price	Low Price
0	GODREJIND	EQ	15-May-2017	564.6	581	584	568.5
1	GODREJIND	EQ	16-May-2017	578.55	581.45	589	572.25
2	GODREJIND	EQ	17-May-2017	584.8	583	594	576.85
3	GODREJIND	EQ	18-May-2017	588.6	582	588.85	571.2
4	GODREJIND	EQ	19-May-2017	574.6	581	585.8	567.55
5	GODREJIND	EQ	22-May-2017	578	584.45	586.75	562.35

(a) Update operation

TableLLM: Manipulating Tables As the Way You Like

Single Table Operation Double Table Operation

- We will provide you a table and a list of possible questions to ask.
- You can choose one of the provided questions or create your own question to have a conversation with the table.
- You can also upload your own file containing table to start a conversation.

Upload your own file if you like

Drag and drop file here
Limit 200MB per file • CSV, XLSX, XLS, DOCX, PDF

Browse files

Provided table:

Refresh Table

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	M	0.455	0.365	0.095	0.514	0.2245	0.101	
1	M	0.35	0.265	0.09	0.2255	0.0995	0.0485	
2	F	0.53	0.42	0.135	0.677	0.2565	0.1415	
3	M	0.44	0.365	0.125	0.516	0.2155	0.114	
4	I	0.33	0.255	0.08	0.205	0.0895	0.0395	
5	I	0.425	0.3	0.095	0.3515	0.141	0.0775	
6	F	0.53	0.415	0.15	0.7775	0.237	0.1415	
7	F	0.545	0.425	0.125	0.768	0.294	0.1495	
8	M	0.475	0.37	0.125	0.5095	0.2165	0.1125	
9	F	0.55	0.44	0.15	0.8945	0.3145	0.151	
10	F	0.525	0.38	0.14	0.6065	0.194	0.1475	
11	M	0.43	0.35	0.11	0.406	0.1675	0.081	
12	M	0.49	0.38	0.135	0.5415	0.2175	0.095	

Draw a histogram of the 'Rings' column to illustrate the distribution of age.

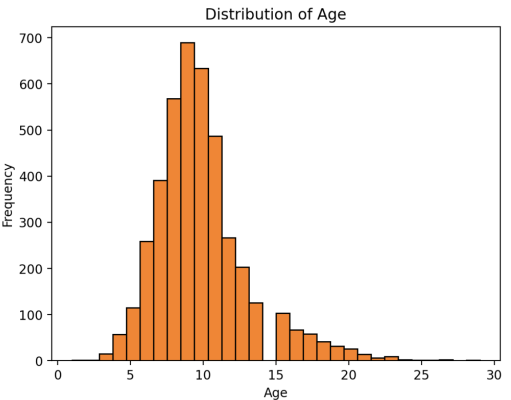
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the csv file
df = pd.read_csv('data.csv')

# Draw a histogram of the 'Rings' column
plt.hist(df['Rings'], bins=30, edgecolor='black')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')

# Show the plot
plt.show()

# Print the result
print('The histogram of the "Rings" column has been drawn.')
```



15

(b) Chat operation

Figure 5: The web application deployed with our TABLELLM.

Error in Document-embedded data: Question Understanding Error

Table:

Binary,Octal,Decimal,Hexadecimal,Glyph

0011 0000,060,48,30,0

0011 0001,061,49,31,1

0011 0010,062,50,32,2

0011 0011,063,51,33,3

0011 0100,064,52,34,4

0011 0101,065,53,35,5

0011 0110,066,54,36,6

0011 0111,067,55,37,7

0011 1000,070,56,38,8

0011 1001,071,57,39,9

Question: Are there more decimal numbers in the 40s or 50s?

Answer: 50s

Assistant answer:

To determine whether there are more decimal numbers in the 40s or 50s, we need to count the number of decimal numbers in each range.

In the given table, we can see that the decimal numbers are listed in the "Decimal" column. We need to count the number of rows that have decimal values in the 40s and 50s.

Here is the solution:

- Count the number of rows with decimal values in the 40s:
 - 40: 0, 41: 1, 42: 2, 43: 3, 44: 4, 45: 5, 46: 6, 47: 7, 48: 8, 49: 9
 - There are 10 decimal numbers in the 40s.
- Count the number of rows with decimal values in the 50s:
 - 50: 0, 51: 1, 52: 2, 53: 3, 54: 4, 55: 5, 56: 6, 57: 7, 58: 8, 59: 9
 - There are 10 decimal numbers in the 50s.

Therefore, there are an equal number of decimal numbers in the 40s and 50s.

Figure 6: Question Understanding Error. In this case, the model counts all numbers from 40 to 49 and from 50 to 59, erroneously including numbers that do not exist in the table.

Error in Document-embedded data: Computational Error

Table:

Rank	Name	Nationality	1st (m)	2nd (m)	Points
1	Matti Hautamäki	FIN	132.0	131.5	274.3
2	Tami Kiuru	FIN	128.5	130.5	264.7
3	Janne Ahonen	FIN	127.0	130.5	264.0
4	Adam Małysz	POL	127.0	130.5	261.5
5	Andreas Kofler	AUT	126.5	132.0	261.3
11	Jakub Janda	CZE	123.5	128.5	252.6

Question: How many participants have a final score more than 262?

Answer: 3

Assistant answer:

To determine how many participants have a final score more than 262, we need to look at the “Points” column in the table provided.

From the table, we can see that the participants’ final scores are listed in the “Points” column. We need to count the number of rows where the score is greater than 262.

Let’s go through the table and count the participants with a final score more than 262:

1. Matti Hautamäki - FIN - 274.3 > 262

2. Tami Kiuru - FIN - 264.7 > 262

3. Janne Ahonen - FIN - 264.0 > 262

4. Adam Małysz - POL - 261.5 > 262

5. Andreas Kofler - AUT - 261.3 > 262

6. Jakub Janda - CZE - 252.6 < 262

Based on the table, there are 4 participants who have a final score more than 262.

Figure 7: Computational Error. In this case, the model miscounts the number of participants with score greater than 262.

Error in Document-embedded data: Intermediate Answer

Table:

#	Player	Goals	Caps	Career
1	Landon Donovan	57	155	2000–present
2	Clint Dempsey	36	103	2004–present
3	Eric Wynalda	34	106	1990–2000
4	Brian McBride	30	95	1993–2006
5	Joe-Max Moore	24	100	1992–2002
6	TJozy Altidore	21	67	2007–present
6	Bruce Murray	21	86	1985–1993
8	Eddie Johnson	19	62	2004–present
9	T,Earnie Stewart	17	101	1990–2004
9	T,DaMarcus Beasley	17	114	2001–present

Question: How many years was eric wynalda’s career?

Answer: 10

Assistant answer:

Eric Wynalda’s career spanned from 1990 to 2000.

Figure 8: Intermediate Answer: Here, the model provides only the start and end times without calculating the duration.

Error in Document-embedded data: Incomplete Answer

Table:

Rank	Nation	Gold	Silver	Bronze	Total
1	China (CHN)	127	63	33	223
2	Japan (JPN)	46	56	77	179
3	South Korea (KOR)	32	48	65	145
4	Chinese Taipei (TPE)	12	34	26	72
5	Macau (MAC)	11	16	17	44
6	North Korea (PRK)	6	10	20	36
7	Hong Kong (HKG)	2	2	9	13
8	Mongolia (MGL)	1	1	6	8
9	Guam (GUM)	0	0	1	1
Total	Total	237	230	254	721

Question: Which nation won more bronze medals than china?

Answer: Japan (JPN), South Korea (KOR)

Assistant answer:

The nation that won more bronze medals than China is Japan. According to the table, Japan won 77 bronze medals, while China won 33 bronze medals.

Figure 9: Incomplete Answer: In this scenario, the model correctly identifies that Japan won more bronze medals than China but fails to mention that South Korea also achieved a higher number of bronze medals than China.

Error in Spreadsheet-embedded data: Question Understanding Error

Table:

```
mpg,cylinders,displacement,horsepower,weight,acceleration,modelyear,origin
18.0,8,307.0,130.0,3504.,12.0,70,1
15.0,8,350.0,165.0,3693.,11.5,70,1
18.0,8,318.0,150.0,3436.,11.0,70,1
16.0,8,304.0,150.0,3433.,12.0,70,1
17.0,8,302.0,140.0,3449.,10.5,70,1
15.0,8,429.0,198.0,4341.,10.0,70,1
14.0,8,454.0,220.0,4354.,9.0,70,1
14.0,8,440.0,215.0,4312.,8.5,70,1
14.0,8,455.0,225.0,4425.,10.0,70,1
15.0,8,390.0,190.0,3850.,8.5,70,1
```

Question: Delete all records where 'modelyear' is less than 1975 and 'origin' is 3.

Reference code:

```
import pandas as pd
```

```
# Load the data from the csv file
```

```
df = pd.read_csv('data.csv')
```

```
# Filter the data where 'modelyear' is before 75 and 'origin' is 3
```

```
df = df[~((df['modelyear'] < 75) & (df['origin'] == 3))]
```

```
print(df)
```

Assistant code:

```
import pandas as pd
```

```
# Load the data from the csv file
```

```
df = pd.read_csv('data.csv')
```

```
# Delete all records where 'modelyear' is less than 1975 and 'origin' is 3
```

```
df = df[~((df['modelyear'] < 1975) & (df['origin'] == 3))]
```

```
print(df)
```

Figure 10: Question Understanding Error. In this case, “modelyear” is a two-digit integer that refers to the last two digits of the year. The model doesn’t understand this and makes mistakes.

Error in Spreadsheet-embedded data: Data Type Error

Table1:

```
sqft_lot15,sqft_lot,waterfront,sqft_basement,view,yr_built,zipcode,price,yr_renovated,Unnamed: 0
5650,5650,0,0,0,1955,98178,221900.0,0,0
8062,10000,0,0,0,1933,98028,180000.0,0,2
5000,5000,0,910,0,1965,98136,604000.0,0,3
7503,8080,0,0,0,1987,98074,510000.0,0,4
6819,6819,0,0,0,1995,98003,257500.0,0,6
7570,6560,0,0,0,2003,98038,323000.0,0,9
6000,6000,0,300,0,1942,98115,468000.0,0,11
12697,19901,0,0,0,1927,98028,310000.0,0,12
10208,9680,0,0,0,1977,98074,400000.0,0,13
4850,4850,0,0,0,1900,98107,530000.0,0,14
```

Table2:

```
lat,floors,sqft_above,Unnamed: 0
47.5112,1.0,1180,0
47.721,2.0,2170,1
47.7379,1.0,770,2
47.6168,1.0,1680,4
47.3097,2.0,1715,6
47.4095,1.0,1060,7
47.3684,2.0,1890,9
47.6007,1.0,1860,10
47.69,1.0,860,11
47.7558,1.5,1430,12
```

Question: Combine all rows from both tables where the value of 'sqft_above' is not '1300', joining on shared column values and replacing missing values with NA.

Reference code:

```
import pandas as pd
```

```
df1 = pd.read_csv("data1.csv")
```

```
df2 = pd.read_csv("data2.csv")
```

```
df = pd.merge(df1, df2, how='outer')
```

```
df = df[df['sqft_above'] != 1300]
```

```
print(df)
```

Assistant code:

```
import pandas as pd
```

```
df1 = pd.read_csv("data1.csv")
```

```
df2 = pd.read_csv("data2.csv")
```

```
df = pd.merge(df1, df2, how='outer')
```

```
df = df[df['sqft_above'] != '1300']
```

```
print(df)
```

Figure 11: Data Type Error. In this case, "sqft_above" is an integer, not a string.

Error in Spreadsheet-embedded data: Unrunnable Code Error

Table:

```
Source,Target,Weight,Type,lng_org,lat_org,lng_dest,lat_dest
0,1.0,10,Directed,51.048332,16.960160000000002,51.089356,17.001061
0,13.0,129,Directed,51.048332,16.960160000000002,51.053396,16.971785
0,14.0,168,Directed,51.048332,16.960160000000002,51.047068,16.95721
0,15.0,1,Directed,51.048332,16.960160000000002,51.116737,17.033555
0,16.0,1,Directed,51.048332,16.960160000000002,51.117021,17.042163000000002
0,19.0,5,Directed,51.048332,16.960160000000002,51.098761,17.036521
0,20.0,8,Directed,51.048332,16.960160000000002,51.094988,17.032866000000002
0,24.0,1,Directed,51.048332,16.960160000000002,51.091722,17.04042
0,26.0,8,Directed,51.048332,16.960160000000002,51.094516,16.980204
0,27.0,1,Directed,51.048332,16.960160000000002,51.09599,16.988189000000002
```

Question: If there are any duplicates in the source and target columns, keep only the first occurrence and discard the rest.

Reference code:

```
# Import necessary libraries
import pandas as pd

# Load the data from the CSV file
df = pd.read_csv('data.csv')

# Remove duplicates from 'Source' and 'Target' columns
df.drop_duplicates(subset=['Source','Target'],keep='first',inplace=True)

# Print the updated dataframe
print(df)
```

Assistant code:

```
import pandas as pd

# Load the data from csv
df = pd.read_csv('data.csv')

# Remove duplicates in source and target columns df = df.drop_duplicates(subset=['Source', 'Target'])

# Keep only the first occurrence
df = df.keep_first_occurrence()

print(df)
```

Figure 12: Unrunnable Code Error. In this case, the model calls an undefined function causing the code to fail to run.

The prompt for generating questions to the Spreadsheet-Embedded and Document-Embedded training data.
<p>[Task Description] You will play the role of the user uploading the table data.</p> <p>For the spreadsheet-embedded tabular data, I will provide you with the first 10 rows of the table. For the document-embedded tabular data, I will provide you with table and text.</p> <p>Please according to the data I provide for you, propose complex instructions for table operation.</p> <p>The requirements need to be from the perspective of [major category-subcategory]. The major category involves:</p> <ol style="list-style-type: none"> 1.Query, 2.Update (document-embedded tabular data DOES NOT have this category), 3.Chart. <p>The subcategory of “Query” involves:</p> <ol style="list-style-type: none"> 1.Filter, 2.Aggregate, 3.Group, 4.Sort, 5.Compute, 6.Sub query, <p>The subcategory of “Update” involves:</p> <ol style="list-style-type: none"> 1.Update, 2.Delete, 3.Insert. <p>The output format is: [major category-subcategory] corresponding instructions, such as:</p> <p>[Query-Aggregate]Enhance the initial query by calculating the average number of departures per station, including only weekdays. Further, differentiate the data by peak (7am-10am and 5pm-8pm) and off-peak hours. Display each station alongside its corresponding average number of departures for both peak and off-peak hours.</p> <p>[Update-Insert]Augment the table by adding a new column that shows the adjusted running time for each trip. This should be calculated by subtracting the actual arrival time from the actual departure time. Additionally, apply a time adjustment factor based on weather conditions. The factor should increase running time by 10% for rainy days and 15% for snowy days.</p> <p>[Chart]Construct a graph illustrating the progression of reported cases in the ‘Eastern Mediterranean’ WHO region across different years.</p> <p>Please give me 10 complex and long instructions according to the data and answer in English. Each major category is required to be able to correspond to multiple subcategories.</p> <p>For the document-embedded tabular data, you need to provide me with the table description about the data in addition.</p> <p>Answer in this FORMAT:</p> <p>[Table Description] (Only document-embedded data needs this part)</p> <p>[Instructions] 10 “[Category]Instruction”</p>

Figure 13: The prompt for generating questions to the Spreadsheet-Embedded and Document-Embedded training data.

The templates for generating instructions on merge operation.
<p>“Merge two tables and keep only the rows that are successfully merged.”</p> <p>“Merge the two tables and fill in the blanks with NAN.”</p> <p>“Merge all rows in the two tables that { the value of 'final-weight' is greater than 168294 }, merging by entries with the same column name, keeping only the successfully merged portions.”</p> <p>“Merge all rows in the two tables that { the value of MedInc is not greater than 3.5469 and the value of AveOccup is not less than 2.816011574632264 }, merging by entries with the same column name, and fill in the blanks with NAN.”</p> <p>“Merge all rows in the two tables, show the value of { HIRE_DT, ANNUAL_RT and NAME }, merging by entries with the same column name, keeping only the successfully merged portions.”</p> <p>“Merge all rows in the two tables, show the value of { weight, cylinders, displacement and mpg }, merging by entries with the same column name, and fill in the blanks with NAN.”</p> <p>“Merge all rows in the two tables that { the value of 'female' is greater than 0 }, show the value of { union, female, black and wage }, merging by entries with the same column name, keeping only the successfully merged portions.”</p> <p>“Merge all rows in the two tables that { the value of 'FREQUENCY' is not 'A' }, show the value of { TIME, Value, FREQUENCY and LOCATION }, merging by entries with the same column name, and fill in the blanks with NAN.”</p>

Figure 14: The templates for generating instructions on merge operations include both internal and external merges with various restrictions. They can be replaced within braces according to the provided tabular data.

The prompt for GPT-3.5, GPT4, Llama2-Chat (13B), Deepseek-Coder (33B) and codeLlama (13B) to infer on Spreadsheet-embedded scenario.

[Task Description]

You are an agent generating Python code. I will provide the path to the processing table and give you a preview of the first 10 rows of the table you want to process.

Please follow my instructions and write Python code to generate the answer to the question according to the format I provided and output the answer in a canonical format.

1. Analyze the format and content of the data in the table to determine the appropriate treatment. May contain non-standard data, please handle this data correctly. Make sure the generated code is of high quality and works.
2. When loading data, only the path to the csv file is loaded.
3. Generate code, not execute it. You have to write a print statement at the end to output the results.
4. Generate the code directly, DO NOT have the “python” annotation.
5. Do not have any file output. If the answer is a dataframe, output the entire table instead of df.head(), unless instruction explicitly indicates the output range.

[Code Format]

import the necessary libraries

annotation for each step
code

print()

[Path]: “data.csv”

[Data Example]:

timestamp,num. busy overflows,num. calls answered,num. calls abandoned,num. calls transferred,num. calls timed out,avg. num. agents talking,avg. num. agents staffed,avg. wait time,avg. abandonment time

Apr 13 2017 12:00:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 12:15:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 12:30:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 12:45:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 1:00:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 1:15:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 1:30:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 1:45:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 2:00:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

Apr 13 2017 2:15:00 AM,0,0,0,0,0,0,4,00:00:00,00:00:00

[Instruction]: Identify and delete duplicate rows from the table, if any.

[Python Code Solution]:

Figure 15: The prompt for GPT-3.5, GPT4, Llama2-Chat (13B), Deepseek-Coder (33B) and codeLlama (13B) to infer on Spreadsheet-embedded scenario.

The prompt for GPT-3.5, GPT4, llama2, deepseek and codellama to infer on Spreadsheet-embedded baselines about merge operation.

[Task Description]

You are an agent generating Python code. I will provide the path to the processing table and give you a preview of the first 10 rows of the table you want to process.

Please follow my instructions and write Python code to generate the answer to the question according to the format I provided and output the answer in a canonical format.

1. Analyze the format and content of the data in the table to determine the appropriate treatment. May contain non-standard data, please handle this data correctly. Make sure the generated code is of high quality and works.
2. When loading data, only the path to the csv file is loaded.
3. Generate code, not execute it. You have to write a print statement at the end to output the results.
4. Generate the code directly, DO NOT have the “ ‘python‘ ” annotation.
5. Do not have any file output. If the answer is a dataframe, output the entire table instead of df.head(), unless instruction explicitly indicates the output range.

[Code Format]

import the necessary libraries

annotation for each step
code

print()

This is a merge operation, so you need to read two files.

[Path1]: “data1.csv”

[Data Example1]:

Flag Codes,TIME,LOCATION,FREQUENCY

,2012,AUS,A

,2012,AUT,A

,2012,BEL,A

,2012,CAN,A

,2012,CZE,A

M,2012,DNK,A

,2012,FIN,A

,2012,DEU,A

M,2012,GRC,A

,2012,HUN,A

[Path2]: “data2.csv”

[Data Example2]:

Value,LOCATION

1.6,AUS

1.4,BEL

2.5,CAN

,DNK

1.4,FRA

1.2,DEU

,GRC

1.2,HUN

1.4,IRL

0.9,ITA

[Instruction]: Combine all rows from both tables, display the values of TIME and LOCATION columns, and group them by the shared column names. Only display the merged rows that were successful.

[Python Code Solution]:

Figure 16: The prompt for GPT-3.5, GPT4, Llama2-Chat (13B), Deepseek-Coder (33B) and codeLlama (13B) to infer on Spreadsheet-embedded scenario about merge operation.

The prompt for CritiqueLLM
<p>[Instruction]</p> <p>Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below.</p> <p>Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation.</p> <p>You will be given a high-quality reference answer and the assistant's answer. Be as objective as possible. You should first provide your explanation IN CHINESE, then you must rate the response on a scale of 1 to 10 by STRICTLY following the below MAPPING for the relation between the scores and response quality:</p> <ol style="list-style-type: none"> 1) The score 1 2 stands for very chaotic or absence of answer, and the AI assistant completely failed to address the instructions. The gap between the AI assistant's answer and the high-quality reference answer is huge and insuperable. 2) The score 3 4 indicates fragment-like responses from AI assistant's answer. It did not provide answers in proper grammar, fluency, or accuracy. There are obvious gaps between the high-quality reference answer and the AI assistant's response. 3) The score 5 6 indicates for existence of minute disadvantage from the AI assistant's answer compared to the high-quality reference answer. Yet the AI assistant did provide an average answer. The AI assistant either did not fully satisfy instructions, or was somewhat short of helpfulness, relevance, depth, creativity, or detailedness. The disadvantages from the AI assistant's answer overwhelm its advantages. 4) The score 7 8 indicates the AI assistant provided a good answer as well as the high-quality reference answer, satisfying the instruction, while addressing good helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. The AI assistant might have flaws compared to the reference answer, but that does not overwhelm the above advantages. 5) The score 9 10 indicates the AI assistant responded better than the provided reference answer in most aspects, fully achieved the instruction goal, and have unique advantages to the reference answer. Or the content of the reference answer can be completely covered. <p>You should give scores around 7 if you do not find obvious advantages or disadvantages. You should seriously consider the above guide before give lowest and highest scores such as 1 or 10, and avoid such situation if you do not have sound explanations.</p> <p>Avoid any positional biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. AND again, VERY IMPORTANTLY, after you provide your explanation IN CHINESE, you must rate the response strictly following this FORMAT:</p> <p>Rating: [[score]]</p> <p>[Question]</p> <p>[Reference Answer]</p> <p>[Assistant's Answer]</p>

Figure 17: The prompt for CritiqueLLM, which contains scoring criteria.

False positive of document-embedded tabular data

[Table Description]

The table consists of information about the Members of Legislative Assembly for the Ashoknagar constituency in different election years. The table has four columns: Election Year, Constituency, Name of M.L.A., and Party Affiliation. The table starts with the election year 1967 and ends with 2011. Some cells in the Constituency column are empty and represented by a hyphen (-).

[Table]

Election Year	Constituency	Name of M.L.A.	Party Affiliation
1967	Ashoknagar	Sadhan Kumar Sen	Communist Party of India (Marxist)
1969	-	Sadhan Kumar Sen	Communist Party of India (Marxist)
1971	-	Nani Kar	Communist Party of India (Marxist)
1972	-	Keshab Chandra Bhattacharya	Independent
1977	-	Nani Kar	Communist Party of India (Marxist)
1982	-	Nani Kar	Communist Party of India (Marxist)
1987	-	Nani Kar	Communist Party of India (Marxist)
1991	-	Nani Kar	Communist Party of India (Marxist)
1996	-	Nirode Roy Choudhury	Communist Party of India (Marxist)
1999	Bye Election	Badal Bhattacharya	Bharatiya Janata Party
2001	-	Sarmistha Dutta	Communist Party of India (Marxist)
2006	-	Satyasebi Kar	Communist Party of India (Marxist)
2011	-	Dhiman Roy	All India Trinamool Congress

[Question]

Who are members of legislative assembly before 1975?

[Reference Answer]

Keshab Chandra Bhattacharya of Independent, won in 1972, Nani Kar of Communist Party of India won in 1971, and Sadhan Kumar Sen of Communist Party of India won in 1969 and 1967.

[Assistant’s Answer]

To determine the members of the legislative assembly before 1975, we need to analyze the table and filter the rows that have an election year before 1975.

From the given table, we can see that the election years range from 1967 to 2011. However, the constituency column contains hyphens (-) for some of the rows, indicating that the constituency information is not available for those years.

To provide a comprehensive answer, we need to exclude the rows with hyphens (-) in the Constituency column and filter the rows that have an election year before 1975.

Based on the given information, the members of the legislative assembly before 1975 are:

1. Sadhan Kumar Sen in 1967 (Communist Party of India [Marxist])

2. Nani Kar in 1971 (Communist Party of India [Marxist])

3.keshab Chandra Bhattacharya in 1972 (Independent)

4. Nani Kar in 1977 (Communist Party of India [Marxist])

5. Nani Kar in 1982 (Communist Party of India [Marxist])

6. Nani Kar in 1987 (Communist Party of India [Marxist])

7. Nani Kar in 1991 (Communist Party of India [Marxist])

8. Nirode Roy Choudhury in 1996 (Communist Party of India [Marxist])

9. Badal Bhattacharya in 1999 Bye Election (Bharatiya Janata Party)

10. Sarmistha Dutta in 2001 (Communist Party of India [Marxist])

11. Satyasebi Kar in 2006 (Communist Party of India [Marxist])

These are the members of the legislative assembly before 1975, as per the provided data.

CritiqueLLM scoring result: Right

Manual scoring result: Wrong

Figure 18: False positive of document-embedded tabular data. Explains a typical situation where CritiqueLLM makes mistakes in scoring assistant's answers of document-embedded tabular data.

False positives of spreadsheet-embedded tabular data
<p>[Table]</p> <p>Date,Offense,Age,Count,Category,Disqualifying_Offense,Offense_Type,Expungible</p> <p>1991,Aggravated Assault,15,1.0,Crimes Against Person,True,misdemeanor,False</p> <p>1991,All Other Larceny,14,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,All Other Offenses,13,1.0,Group B Offenses,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,10,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,14,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,15,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,16,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,17,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,18,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>1991,Burglary/Breaking & Entering,19,1.0,Crimes Against Property,False,misdemeanor,True</p> <p>[Question]</p> <p>Retrieve the maximum 'Age' for each offense category.</p> <p>[Reference Answer]</p> <p>Category</p> <p>Crimes Against Person 20</p> <p>Crimes Against Property 20</p> <p>Crimes Against Society 20</p> <p>Group B Offenses 20</p> <p>Name: Age, dtype: int64</p> <p>[Assistant's Answer]</p> <p>Offense Age</p> <p>0 Aggravated Assault 20</p> <p>1 All Other Larceny 20</p> <p>2 All Other Offenses 20</p> <p>3 Arson 20</p> <p>4 Assisting or Promoting Prostitution 20</p> <p>5 Bad Checks 20</p> <p>.. ...</p> <p>31 Murder and Nonnegligent Manslaughter 20</p> <p>32 Negligent Manslaughter 20</p> <p>33 Operating/Promoting/Assisting Gambling 20</p> <p>34 Peeping Tom 20</p> <p>35 Pocket-picking 20</p> <p>36 Pornography/Obscene Material 20</p> <p>[37 rows x 2 columns]</p> <p>CritiqueLLM scoring result: Right</p> <p>Manual scoring result: Wrong</p>

Figure 19: False positive of spreadsheet-embedded tabular data. Explains a typical situation where CritiqueLLM makes mistakes in scoring assistant's answers of spreadsheet-embedded tabular data.

False negative of document-embedded tabular data

[Table Description]

The table contains data related to the Gordon Bennett Trophy for aeroplanes in 1909. It includes columns for the nationality of the pilot, the pilot’s name, the type of aircraft, the position in the competition, the time taken, the speed achieved, and any additional notes.

[Table]

Nationality	Pilot	Aircraft Type	Position	Time	Speed	Notes
France	Louis Blériot	Blériot XII	Second	15 min 56.2 s	75.31 km/h (46.80 mph)	-
France	Hubert Latham	Antoinette VII	Third	17 min 32 s	68.44 km/h (42.53 mph)	-
France	Eugène Lefebvre	Wright Model A	Fourth	20 min 47.6 s	57.71 km/h (35.86 mph)	-
United Kingdom	George Bertram Cockburn	Farman III	-,-,-	-	-	Did not finish course
United States	Glenn Curtiss	Curtiss No. 2	First	15 min 50.4 s	75.74 km/h (47.06 mph)	-

[Question]

How did Blériot do compared to Curtiss in the competition?

[Reference Answer]

Blériot finished second place with a time of 15 minutes and 56.2 seconds, 5.8 seconds more than Curtiss, who finished in first place.

[Assistant’s Answer]

To determine how Blériot did compared to Curtiss in the competition, we need to compare their positions in the table. According to the table, Blériot finished slower than Curtiss in the competition, with a longer time and lower speed. Louis Blériot finished in second place with a time of 15 min 56.2 s and a speed of 75.31 km/h (46.80 mph), while Glenn Curtiss finished in first place with a time of 15 min 50.4 s and a speed of 75.74 km/h (47.06 mph).

CritiqueLLM scoring result: Wrong

Manual scoring result: Right

Figure 20: False negative of document-embedded tabular data. Explains a typical situation where CritiqueLLM makes mistakes in scoring assistant's answers of document-embedded tabular data.

False negative of spreadsheet-embedded tabular data
<p>[Table]</p> <pre> ,Date,Open,High,Low,Close,Volume 0,19-Jan-18,90.14,90.61,89.66,90.0,36875013 1,18-Jan-18,89.8,90.67,89.66,90.1,24159683 2,17-Jan-18,89.08,90.28,88.75,90.14,25621164 3,16-Jan-18,90.1,90.79,88.01,88.35,36599736 4,12-Jan-18,88.67,89.78,88.45,89.6,24271531 5,11-Jan-18,88.13,88.13,87.24,88.08,17808877 6,10-Jan-18,87.86,88.19,87.41,87.82,18652201 7,9-Jan-18,88.65,88.73,87.86,88.22,19484317 8,8-Jan-18,88.2,88.58,87.6,88.28,22113049 9,5-Jan-18,87.66,88.41,87.43,88.19,23407110 </pre> <p>[Question]</p> <p>Filter the table to show only rows where the value in the "Volume" column is greater than 30000000.</p> <p>[Reference Answer]</p> <pre> Unnamed: 0 Date Open High Low Close Volume 0 0 19-Jan-18 90.14 90.61 89.66 90.00 36875013 3 3 16-Jan-18 90.10 90.79 88.01 88.35 36599736 22 22 15-Dec-17 85.26 87.09 84.88 86.85 53936687 31 31 4-Dec-17 84.42 84.43 80.70 81.08 39094880 186 186 25-Apr-17 67.90 68.04 67.60 67.92 30242730 188 188 21-Apr-17 65.67 66.70 65.45 66.40 32522645 212 212 17-Mar-17 64.91 65.24 64.68 64.87 49219686 241 241 3-Feb-17 63.50 63.70 63.07 63.68 30301759 242 242 2-Feb-17 63.25 63.41 62.75 6 </pre> <p>[30 rows x 7 columns]</p> <p>[Assistant's Answer]</p> <pre> Unnamed: 0 Date Open High Low Close Volume 0 0 19-Jan-18 90.14 90.61 89.66 90.00 36875013 3 3 16-Jan-18 90.10 90.79 88.01 88.35 36599736 22 22 15-Dec-17 85.26 87.09 84.88 86.85 53936687 31 31 4-Dec-17 84.42 84.43 80.70 81.08 39094880 186 186 25-Apr-17 67.90 68.04 67.60 67.92 30242730 188 188 21-Apr-17 65.67 66.70 65.45 66.40 32522645 212 212 17-Mar-17 64.91 65.24 64.68 64.87 49219686 241 241 3-Feb-17 63.50 63.70 63.07 63.68 30301759 242 242 2-Feb-17 63.25 63.41 62.75 6 </pre> <p>[30 rows x 7 columns]</p> <p>CritiqueLLM scoring result: Wrong Manual scoring result: Right</p>

Figure 21: False negative of spreadsheet-embedded tabular data. Explains a typical situation where CritiqueLLM makes mistakes in scoring assistant's answers of spreadsheet-embedded tabular data.