FLEXTAF: Enhancing Table Reasoning with Flexible Tabular Formats

Anonymous ACL submission

Abstract

The table reasoning task aims to answer the question according to the given table. Currently, using Large Language Models (LLMs) is the predominant method for table reasoning. Most existing methods employ a fixed tabular format to represent the table, which could limit the performance since different instances and models suit different tabular formats. We prove the claim through quantitative analysis of experimental results, where different instances and models perform differently using various tabular formats. Building on this discussion, we propose FLEXTAF-Single and FLEXTAF-Vote to enhance table reasoning performance by employing flexible tabular formats. Specifically, (i) FLEXTAF-Single trains a classifier to predict the most suitable tabular format based on the instance and the LLM and utilize the format to reason. (ii) FLEXTAF-Vote integrates the results across different formats. Our experiments on WikiTableQuestions and TabFact bring average improvements of 2.3% and 4.4%, thereby validating the effectiveness of our methods[1].

1 Introduction

Table reasoning is a crucial task in natural language processing that aims to automatically extract and infer information from tables (Dong et al., 2022). In this task, a model needs to answer the question based on the table, with each question-table pair referred to as an instance. Given the superior commonsense and logical reasoning capabilities of Large Language Models (LLMs), researchers increasingly utilize them for table reasoning, which is the mainstream method (Chen, 2023; Zhang et al., 2024d). Therefore, we focus on how to solve the table reasoning task with LLMs in this paper.

Some previous works enhance table reasoning by designing prompts (Cheng et al., 2023; Zhang et al., 2024e; Lee et al., 2024), such as Chain-of-Table (Wang et al., 2024), which prompts the LLM to iteratively reason with natural language and programs. While previous methods provide a fixed tabular format in their prompts, Singha et al. (2023); Sui et al. (2024); Deng et al. (2024) argue that different table reasoning tasks have different most suitable tabular formats. However, ***existing works analyze from the task aspect*** without considering that solving different instances demands distinct reasoning skills (Shi et al., 2020; Chen et al., 2023; Liu et al., 2024a). For example, as shown in Figure 1, solving instance (a) involves sequential indexing, facilitated by the List format, while solving instance (b) requires identifying columns that satisfy specific conditions, making the Database format more suitable. Therefore, using a fixed tabular format for all instances could limit performance. Additionally, models vary in their reasoning capabilities (Bhandari et al., 2024; Zhang et al., 2024a), so the most suitable format could differ for models.

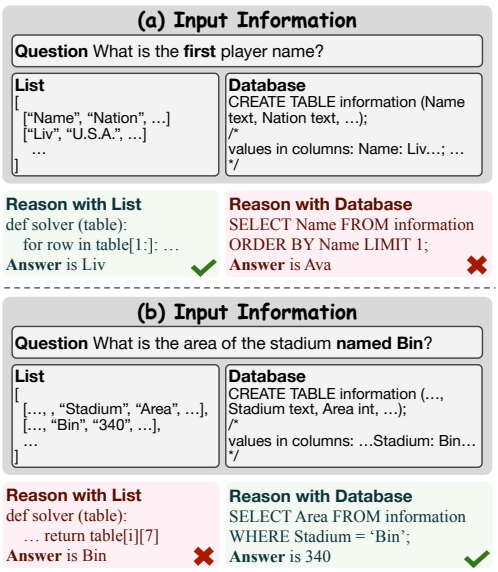

Figure 1: The table reasoning performance varies with tabular formats. The List format is convenient for sequential indexing, while the Database format facilitates the search for columns that meet specific conditions.

[1]Our code and prompts will be released upon acceptance.

Based on the above discussion, we focus on the impact of tabular formats on the table reasoning performance of LLMs from the following two aspects: (*i*) We claim that ***different instances and LLMs require distinct most suitable tabular formats*** [2] ; (*ii*) We propose to enhance table reasoning by ***predicting the most suitable tabular format or assembling the results from multiple formats***.

First, we discuss that the most suitable tabular formats depend on the instance and the LLM. We conduct exploratory experiments utilizing different tabular formats, instances, and LLMs. The results present that table reasoning performance varies significantly with different formats and LLMs. Based on the above analysis, we propose FLEXTAF, which includes FLEXTAF-Single and FLEXTAF-Vote, to enhance the table reasoning performance through flexible tabular formats. FLEXTAF-Single identifies the most suitable format based on the instance and the LLM by training the classifier and then utilize the format to reason. FLEXTAF-Vote determines the final answer by voting on results obtained from multiple formats. In comparison, although FLEXTAF-Single requires training data but only infers once, FLEXTAF-Vote is training-free but with expensive inference costs.

To demonstrate the effectiveness of our methods, we conduct experiments on WikiTableQuestions (Pasupat and Liang, 2015) and TabFact (Chen et al., 2020). Compared to the best results of using the fixed format with greedy decoding and self-consistency (Wang et al., 2023), FLEXTAF-Single and FLEXTAF-Vote show average improvements of 2.3% and 4.4% with comparable inference costs, confirming the effectiveness. Further analysis reveals that certain instances can only be resolved by a format, proving the most suitable tabular formats for different instances are distinct.

Our contributions are as follows:

1. We claim the most suitable tabular formats for different instances and LLMs are distinct.
2. We propose FLEXTAF, including FLEXTAF-Single and FLEXTAF-Vote, to enhance table reasoning by utilizing flexible tabular formats.
3. FLEXTAF-Single and FLEXTAF-Vote achieve average gains of 2.3% and 4.4% over the best results of using fixed formats with greedy decoding and self-consistency, demonstrating the effectiveness of our methods.

[2]We define **the most suitable tabular format** as the one that enables the model to solve a given instance correctly.

2 Preliminaries

We first claim that the most suitable tabular formats for different instances and LLMs are distinct. Since resolving different instances requires diverse abilities (Chen et al., 2023; Cheng et al., 2023; Chen et al., 2024b), it is necessary to tailor tabular formats for each instance accordingly. Additionally, the capabilities of different models vary (Cao et al., 2023; Bhandari et al., 2024; Zhang et al., 2024a), resulting in different tabular formats suitable for different LLMs. We further discuss from the perspective of experimental results in this section.

We select five popular formats, including Markdown, Dict, List, Pandas, and Database, following previous works (Singha et al., 2023; Cheng et al., 2023; Wang et al., 2024; Liu et al., 2024a). Descriptions of these formats are provided in Appendix A. We quantitatively analyze how tabular formats affect the table reasoning performance of LLMs from the aspects of the instance and the model.

2.1 Different Instances Suit Different Tabular Formats

Table 4 presents the percentage of instances that can only be correctly resolved by each tabular format. It can be observed that, even for all instances belonging to a dataset and the same table reasoning task, different instances suit different tabular formats. Specifically, existing some instances are suitable for only one tabular format, while other formats cannot accurately solve them. Furthermore, each tabular format is uniquely suited to certain instances, with some formats correctly solving over 20% of the instances exclusively. The results indicate that, for a given LLM, the most suitable tabular formats vary according to the specific instances.

2.2 Different Models Suit Different Tabular Formats

Table 1 demonstrates that the most suitable tabular format varies across models and the performance gap arises due to different formats. For instance, Llama3 (Meta, 2024) and `gpt-4o` (OpenAI et al., 2024) exhibit significantly better performance with the Markdown format compared to other formats, while DeepSeek-Coder (Guo et al., 2024) performs better with Dict and Database formats than with Markdown. We employ a Chi-square test to demonstrate significant differences in the distribution of the most suitable tabular formats across different models, as detailed in Appendix D.

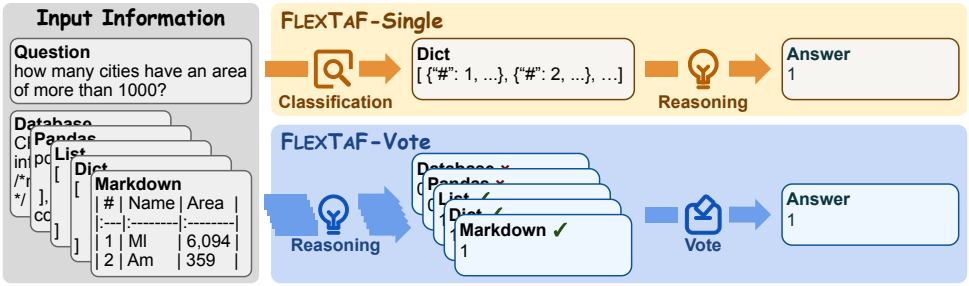

Figure 2: The overview of FLEXTAF. FLEXTAF-Single consists of two steps: (*i*) Classification: A classifier we trained predicts the most suitable tabular format based on the given instance and model. (*ii*) Reasoning: Using the predicted format, the LLM solves the instance by representing the table accordingly. FLEXTAF-Vote consists of two steps: (*i*) Reasoning: Various formats are employed to represent the table and facilitate reasoning with the LLM, resulting in multiple answers. (*ii*) Vote: The final answer is determined using a voting mechanism.

3 Methodology

In this section, we introduce FLEXTAF, which consists of FLEXTAF-Single and FLEXTAF-Vote. The overview of FLEXTAF is shown in Figure 2.

3.1 Task Definition

FLEXTAF focuses on the table reasoning task, which can be formally defined as follows: Given an instance I comprising a natural language question Q and a table T, the table reasoning task aims to derive the corresponding answer $\hat{A} = \mathtt{M}(\mathtt{F}(T), Q)$, where $\mathtt{M}$ represents the model and $\mathtt{F}$ denotes the tabular format. To effectively solve the table reasoning task, the probability that the generated answer $\hat{A}$ matches the gold answer A^* should be maximized.

3.2 FLEXTAF-Single

As discussed in §2, it is essential to determine the most suitable tabular format, so FLEXTAF-Single tries to find the most suitable format by training a classifier given the instance and LLM.

3.2.1 Classification

Classification aims to predict the most suitable tabular format from a set of candidate formats based on the instance and the model. It can be formally expressed as $\hat{\mathtt{F}} = \mathtt{CLS}_{\mathtt{M}}(I), \hat{\mathtt{F}} \in \mathtt{F}^*_{\mathtt{M},I} = \{\mathtt{F} | \mathtt{M}(\mathtt{F}(T), Q) = A^*\}$.

Training Data Collection To train the classifier which is used to predict the most suitable tabular format for the given instance I and the LLM $\mathtt{M}$, we annotate the training data with $\mathtt{M}$. Specifically, for each instance in the training set, we utilize all candidate tabular formats to reason respectively and evaluate the correctness of each answer. We then collect the set of most suitable tabular formats for each instance in the training data, denoted as $\{\mathtt{F}^*_{\mathtt{M},I_t}\}$, for which $\mathtt{M}$ can correctly reason with the format on the instance I_t, and use these as the training data for our classifier. Additionally, we take a data filtering strategy to remove instances from the training set where more than half of the candidate formats are correct or where none are correct, as such instances do not effectively highlight differences between the tabular formats.

Learning Objective Since there could be multiple formats in $\mathtt{F}^*_{\mathtt{M},I}$, we apply a multi-label classification training method (Godbole and Sarawagi, 2004; Tsoumakas and Katakis, 2007; Herrera et al., 2016), where each label denotes a tabular format. Moreover, we utilize a binary relevance method (Godbole and Sarawagi, 2004) for multi-label classification. Specifically, the tabular formats are serialized and transformed into binary vectors for each instance. During training, we adopt Binary Cross-Entropy Loss, normalized by the number of instances N, as follows:

$$\mathtt{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{r=1}^{|\mathtt{F}|}[y_{ir}\mathtt{log}(\hat{y}_{ir}) + (1-y_{ir})\mathtt{log}(1-\hat{y}_{ir})] \tag{3.1}$$

Among them, $|\mathtt{F}|$ refers to the total number of candidate tabular formats, y_{ir} indicates the gold value for the instance i on tabular format r, with possible values of 0 or 1, and $\hat{y}_{ir}$ represents the predicted probability for instance i on the tabular format r.

Predicting Tabular Format After obtaining the classifier, we predict the most suitable tabular format $\hat{\mathtt{F}}$. We regularize the predicted scores of all formats and select one format with the highest probability as our classification result $\hat{\mathtt{F}}$.

3.2.2 Reasoning

After predicting the most suitable table format $\hat{\mathtt{F}}$, we utilize the predicted format for table reasoning. Specifically, we represent the table with the predicted format $\hat{\mathtt{F}}$ in the prompt and employ the LLM to derive the final answer.

3.3 FLEXTAF-Vote

FLEXTAF-Single requires manual data annotation, which could be difficult to obtain, so we propose FLEXTAF-Vote to obtain the final answer by integrating the results from multiple tabular formats, inspired by Qin et al. (2023); Luo et al. (2024).

3.3.1 Reasoning

We first construct multiple table reasoning prompts, representing the table with different formats in each prompt. Then we use the prompts to reason with the LLM, obtaining multiple answers accordingly.

3.3.2 Vote

We retain the most consistent result across multiple results by adopting a voting mechanism, which can be formally expressed as the following equation.

$$\hat{A} = \texttt{argmax}_A \sum_{r=1}^{|\mathtt{F}|} \mathbb{1}(A_r = A) \qquad (3.2)$$

Here, $|\mathtt{F}|$ is the total number of candidate formats, A_r is the answer obtained from the tabular format $\mathtt{F}_r$, and A is each possible answer. The function $\mathbb{1}(f)$ returns 1 if f is true and 0 otherwise. In the event of a tie, we select the answer with the highest logarithmic probability following Luo et al. (2024).

3.4 Comparison

To better employ our two methods, we examine their application scenarios. (*i*) FLEXTAF-Single is ideal for scenarios where high-efficiency online reasoning is required, although it necessitates prior training data annotation and classifier training. (*ii*) FLEXTAF-Vote suits scenarios where annotated data is unavailable while maintaining a certain tolerance for real-time reasoning efficiency.

4 Experiments

4.1 Settings

4.1.1 Datasets

We use WikiTableQuestions (Pasupat and Liang, 2015) and TabFact (Chen et al., 2020) datasets to evaluate FLEXTAF, following previous works (Cheng et al., 2023; Liu et al., 2024a; Wang et al., 2025). WikiTableQuestions is a mainstream dataset for table question answering, containing diverse questions across domains, which requires answering the question based on the table. TabFact is a prominent dataset for the table fact verification task, which needs to determine whether a claim is entailed or refuted by the table. We show the results on TableBench (Wu et al., 2024) in Appendix B.

4.1.2 Metric

We employ accuracy as the evaluation metric for WikiTableQuestions and TabFact, following the previous works (Pasupat and Liang, 2015; Chen et al., 2020), and use accuracy to evaluate the classifier. Accuracy measures the ability to generate the gold answer, which is achieved only when the predicted exactly matches the gold answer. Since FLEXTAF-Single aims to identify the most suitable format, we adopt accuracy to assess whether the top format predicted is in suitable formats.

4.1.3 Models

For Classification, we use ELECTRA-Large (Clark et al., 2020), and for Reasoning, we employ Llama3 (Meta, 2024), DeepSeek-Coder (Guo et al., 2024), and `gpt-4o` (OpenAI et al., 2024). Llama3 and DeepSeek-Coder are popular open-source general and code LLMs, respectively, for their outstanding performance, and `gpt-4o` is considered one of the leading closed-source models. Limited by the computing resources, we evaluate the performance of `gpt-4o` on sampled 128 instances. We do not evaluate the performance of FLEXTAF-Single on `gpt-4o`, since it requires the result of the model on the training set. We choose ELECTRA-Large due to its superior performance in language comprehension and question-answering tasks compared to other pre-trained models of similar size.

4.1.4 Implementation Details

We adopt Markdown, Dict, List, Pandas, and Database as tabular formats (introduced in Appendix A), which are commonly used in previous works (Singha et al., 2023; Cheng et al., 2023; Ye et al., 2023; Wang et al., 2024) and generally have high performance across datasets and models. To enhance the table reasoning performance, we utilize the Chain-of-Thought (CoT) prompt (Wei et al., 2022) for Markdown and the Program-of-Thought (PoT) prompt (Chen et al., 2023; Gao et al., 2023) for other formats. In addition, for

Tabular Format	WikiTQ					TabFact				
	Llama3		DeepSeek-Coder		gpt-4o†	Llama3		DeepSeek-Coder		gpt-4o†
	8B	70B	6.7B	33B	-	8B	70B	6.7B	33B	-
Markdown	47.7	63.3	32.2	31.2	71.9	75.2	86.4	60.4	63.6	77.3
Dict	43.0	56.4	25.6	53.6	68.0	65.5	80.0	63.9	78.0	75.0
List	30.5	56.3	19.2	50.8	57.8	57.4	77.5	63.7	75.4	75.0
Pandas	39.2	52.3	39.7	48.8	55.5	47.8	73.6	62.5	75.9	82.0
Database	31.0	48.2	41.8	45.5	51.6	65.0	75.0	70.7	76.3	75.8
FLEXTAF-Single	**50.5**	**69.1**	**46.3**	**54.5**	-	**77.0**	**87.1**	**70.9**	**78.3**	-
Δ	+2.8	+5.8	+4.5	+0.9	-	+1.8	+0.7	+0.2	+2.0	-

Table 1: The accuracy of reasoning using a fixed tabular format with **greedy decoding** and FLEXTAF-Single, across four LLMs on WikiTableQuestions (WikiTQ) and TabFact. The best performance for each LLM and dataset is marked in **bold**. † represents the result of `gpt-4o` on sampled 128 instances. Δ denotes the improvement of FLEXTAF-Single relative to the best performance of using a fixed format with greedy decoding for each LLM and dataset.

Tabular Format	WikiTQ					TabFact				
	Llama3		DeepSeek-Coder		gpt-4o†	Llama3		DeepSeek-Coder		gpt-4o†
	8B	70B	6.7B	33B	-	8B	70B	6.7B	33B	-
Markdown	49.1	62.8	32.1	29.7	72.7	75.2	86.7	60.9	63.6	78.9
Dict	44.8	58.1	28.7	59.7	71.0	67.9	80.9	71.9	82.4	91.4
List	35.4	59.7	27.5	55.6	61.7	59.8	78.0	66.9	78.0	89.8
Pandas	41.4	56.2	43.9	54.4	60.9	56.0	74.7	67.2	79.2	84.4
Database	35.2	48.6	42.5	46.4	54.7	69.5	76.0	70.7	77.2	83.6
FLEXTAF-Vote	**55.7**	**69.9**	**51.4**	**60.9**	**76.6**	**80.3**	**88.5**	**77.9**	**84.4**	**93.8**
Δ	+6.6	+7.1	+7.5	+1.2	+3.9	+5.1	+1.8	+6.0	+2.0	+2.4

Table 2: The accuracy of reasoning using a fixed tabular format with **self-consistency decoding** (Wang et al., 2023) and FLEXTAF-Vote, across four LLMs on WikiTQ and TabFact. The best performance for each LLM and dataset is marked in **bold**. † represents the result of `gpt-4o` on sampled 128 instances. Δ denotes the improvement of FLEXTAF-Vote relative to the best performance of using a fixed format with self-consistency for each LLM and dataset.

the open-source models, we apply 4-shot and 2-shot prompts respectively on WikiTableQuestions and TabFact, since the questions in WikiTableQuestions are more challenging. Due to the superior performance of `gpt-4o`, we adopt the zero-shot prompts. To further improve the performance on WikiTableQuestions, we use different demonstrations for each tabular format, and we explore the results with unified demonstrations in §4.3.1. Detailed prompts are provided in Appendix C. We train the tabular format classifier for each LLM on each dataset, with detailed training information in Appendix E. For self-consistency (Wang et al., 2023), we set temperature to 0.1 which can bring optimal performance across most formats within temperature ≤ 0.8. To ensure a fair comparison with FLEXTAF-Vote, we set sampling_n to 5.

4.2 Main Experiment

Table 1 and Table 2 compare FLEXTAF-Single with using a fixed format with greedy decoding, and FLEXTAF-Vote with using a fixed format with self-consistency (Wang et al., 2023), respectively. We observe that: (*i*) FLEXTAF-Single and FLEXTAF-Vote surpass the best results achieved by the fixed format with greedy decoding and self-consistency, by an average of 2.3% and 4.4% respectively, with comparable computational costs, proving our claims in §2. (*ii*) Compared to flexible tabular formats, the fixed format restricts the performance even with self-consistency. Moreover, self-consistency with Markdown improves slightly or even decreases compared to greedy decoding because self-consistency instability in CoT (Chen et al., 2024a; Renze and Guven, 2024), and diminished instruction following ability at higher temperatures (Zeng et al., 2024; Peeperkorn et al., 2024). Additionally, the tables reveal that:

The improvement of FLEXTAF on the more challenging dataset is more significant. Both FLEXTAF-Single and FLEXTAF-Vote achieve performance improvement across datasets, underscor-

MD	Dict	List	PD	DB	$Acc_{max} - Acc_{min}$
47.7	43.0	30.5	39.2	16.1	31.6

Table 3: The accuracy on WikiTQ using Llama3-8B with greedy decoding, employing **unified demonstrations** in the prompts, which are different from the prompts in the main experiments. MD denotes Markdown, PD denotes Pandas, and DB denotes Database.

ing their efficacy. Specifically, the performance on WikiTQ is significantly better compared to the simpler TabFact. Despite higher classification accuracy on TabFact (Table 5), FLEXTAF-Single shows limited improvement due to the already high-performance baseline. Moreover, FLEXTAF-Vote exhibits less improvement on TabFact due to the greater overlap between simper instances correctly solved by different formats (see Appendix F).

FLEXTAF-Single shows superior performance on the general model compared to the code model. The tabular formats suitable for the code model are closely related to the code with smaller differences, resulting in limited classifier performance, so FLEXTAF-Single exhibits less improvement on the code models.

FLEXTAF-Vote performs better than FLEXTAF-Single consistently. Although the performance of FLEXTAF-Single is constrained by classification accuracy (see Table 5), it demonstrates high reasoning efficiency with only one inference. In contrast, FLEXTAF-Vote achieves superior performance because instances could be resolved correctly by multiple formats, and errors produced by different formats exhibit diversity. However, FLEXTAF-Vote is less efficient in reasoning.

4.3 Analysis

We select Llama3-8B for subsequent experiments due to its high reasoning efficiency. Moreover, we conduct more experiments on WikiTQ, because it encompasses more diverse questions (Pasupat and Liang, 2015; Chen et al., 2020; Dong et al., 2022; Shi et al., 2020). To better illustrate that different instances are suitable for different tabular formats, we present detailed instances in Appendix H.

4.3.1 Is the impact of tabular formats due to the different demonstrations?

We conduct experiments with unified demonstrations in the prompts for each format and present the results in Table 3. Specifically, we adopt the

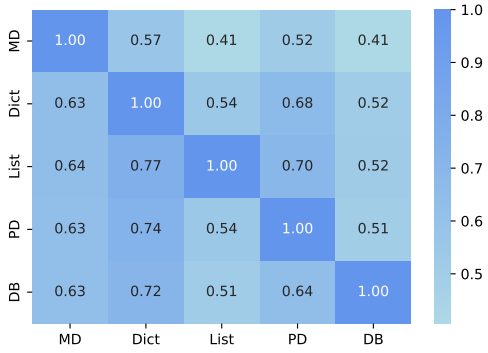

Figure 3: The overlap between instances solved by tabular formats achieved by Llama3-8B on WikiTQ. The values represent the proportion of instances that can be solved by the tabular format corresponding to the vertical axis, within the instances solvable by the format on the horizontal axis.

Model	Scale	MD	Dict	List	PD	DB
Llama3	8B	24.6	7.1	4.6	5.9	8.3
	70B	16.0	1.7	2.2	1.8	4.3
Deep.C.	6.7B	24.7	6.7	3.7	11.0	13.6
	33B	15.6	3.9	3.3	4.4	7.5

Table 4: The percentage of instances that can only be correctly solved by one tabular format, in all instances that the tabular format can correctly solve in WikiTQ. Deep.C. denotes DeepSeek-Coder.

demonstrations of the same questions with different formats, manually annotating the rationale and programs. Detailed prompts are shown in Appendix C. Table 3 indicates that the tabular formats continue to significantly affect performance, with the performance gap widening with unified demonstrations.

4.3.2 Are the tabular formats suitable for different instances different?

We analyze the overlap between instances correctly solved by each format and the proportion of instances that can only be solved by a specific format, as shown in Figure 3 and Table 4. Figure 3 illustrates that: (*i*) The overlaps between instances correctly solved by each format are all $\leq 80\%$, indicating the distinctions among formats. (*ii*) The Dict and Pandas formats exhibit higher overlap due to their superior performance (see Table 2), while the DB and List formats have lower overlap. The overlaps between instances solved by different formats on TabFact are provided in Appendix F.

Table 4 presents that: (*i*) Certain instances can only be accurately solved using a specific format, highlighting the differences between formats.

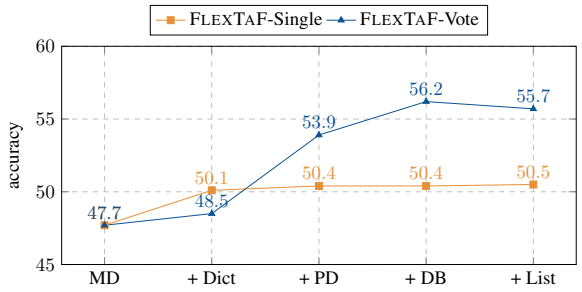

Figure 4: The accuracy of FLEXTAF-Single and FLEXTAF-Vote using Llama3-8B on WikiTQ with different numbers of candidate tabular formats, as candidate tabular formats are added from left to right.

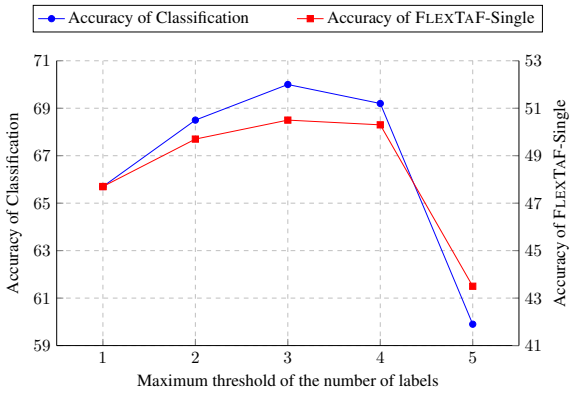

Figure 5: The accuracy of classification and FLEXTAF-Single, with the change of the maximum threshold of the number of labels in the training data.

(*ii*) Markdown and Database formats resolve a greater proportion of instances due to their distinct structures compared to the programming formats. Figure 3 and Table 4 claim that different instances suit different tabular formats.

4.3.3 How does the number of candidate tabular formats affect FLEXTAF?

We perform experiments using varying numbers of candidate formats and present results in Figure 4, where formats are added in descending order of performance. We observe: (*i*) The performance of FLEXTAF-Single gradually stabilizes as the number of candidate formats increases. The classification performance does not always improve due to the increased difficulty with a higher number of labels (Wang et al., 2021; Audibert et al., 2024). (*ii*) FLEXTAF-Vote varies greatly with the increase in the number of formats due to its reliance on the performance of each format. FLEXTAF-Vote does not exceed FLEXTAF-Single with two candidates, as it selects from two different answers only by comparing probabilities, which do not accurately indicate correctness (Wang et al., 2023; Portillo Wightman et al., 2023; Quevedo et al., 2024). Therefore, we select 5 formats as candidates, considering the performance of our two methods.

4.3.4 How to further improve the classification performance?

We analyze both the overall accuracy and the accuracy of instances that can be correctly solved by only one format, as these instances most distinctly highlight the unique features of each format. The classification results are presented in Table 5. We observe that: (*i*) The classification performance of predicting larger-scale LLMs is better than that of smaller-scale LLMs. This is attributed to the greater robustness of larger-scale LLMs (Howe et al., 2024), which results in more consistent features in instances correctly solved with the same format. (*ii*) The classification performance of each format is positively correlated with the size of its training data (see Appendix E). Therefore, future improvements in classification performance can be achieved by reducing noise in the data and increasing the scale of the training data.

4.3.5 How does the data filtering strategy affect FLEXTAF-Single?

We compare various strategies by adjusting the maximum threshold of the number of labels in each training data instance. The experimental results are shown in Figure 5. We find that: (*i*) As the maximum label count rises, overall performance improves, peaking at 3 labels, which demonstrates that training with $\leq$ 3 labels per instance works best, because the increased amount of training data aids the model training. (*ii*) When the label count goes above 3, performance drops. This shows why filtering training data is important, as instances easily resolved by most formats are often simpler and fail to effectively capture the unique features of each label, thus negatively impacting training.

4.4 Case Study

To better illustrate that different instances suit different tabular formats, we present an instance from WikiTQ. As illustrated in Figure 6, when using the Dict format, Llama3-70B generates an incorrect program that processes all table rows without excluding a special "-" row. Conversely, with Markdown, the model ignores the "-" line and correctly identifies the country that won the most gold

Format	WikiTQ				TabFact			
	Llama3		DeepSeek-Coder		Llama3		DeepSeek-Coder	
	8B	70B	6.7B	33B	8B	70B	6.7B	33B
Markdown	**78.7**	**73.8**	47.1	20.3	**64.3**	**67.1**	25.3	**36.0**
Dict	27.9	23.8	35.1	**46.7**	25.0	14.3	**37.5**	**36.0**
List	3.4	29.6	9.7	19.4	20.0	25.0	0.0	0.0
Pandas	2.1	10.0	28.6	30.1	0.0	11.1	22.7	7.7
Database	14.3	15.6	**50.6**	29.7	18.5	26.7	31.3	29.6
Overall	69.6	82.2	68.5	73.1	79.6	88.8	74.3	80.6

Table 5: The overall classification accuracy of FLEXTAF-Single using four LLMs, and the classification accuracy for instances that can only be correctly solved by a single format, with the best performance marked in **bold**.

Figure 6: An instance from the WikiTQ test set using Llama3-70B with Markdown and Dict tabular formats.

medals. Therefore, for this instance, Markdown proves more suitable than Dict with Llama3-70B. Additional instances are provided in Appendix H.

5 Related Works

The table reasoning task aims to answer the natural language question based on the table (Dong et al., 2022; Zhang et al., 2024d; Dong and Wang, 2024). LLM-based methods have become predominant due to their superior performance of table reasoning (Wei et al., 2022; Chen, 2023; Zhao et al., 2023; Chen et al., 2023). Some works focus on enhancing table reasoning ability by fine-tuning LLMs (Zhang et al., 2024b; Bian et al., 2024; Zhang et al., 2024c; Patnaik et al., 2024; Wu and Feng, 2024; Gardner et al., 2024; Li et al., 2024). Also, some works improve table reasoning performance by designing prompts (Zhao et al., 2023; Ye et al., 2023; Nahid and Rafiei, 2024a,b; Lee et al., 2024; Wang et al., 2024; Zhao et al., 2024) or aggregating diverse results (Ni et al., 2023; Liu et al., 2024a).

Previous works generally employ a fixed tabular format across varying instances, regardless of the specific LLM used, which could limit the performance. Since Sui et al. (2024) demonstrate that the performance of table understanding tasks, such as Cell Lookup and Size Detection, varies with different tabular formats, and propose self-augmented prompting, which employs LLMs to extract critical table contents and summarize them into natural language descriptions. Building on this, Singha et al. (2023) evaluate the performance of additional tabular formats and investigate the effects of noise operations like Shuffle Columns and Transpose Table on table understanding. They analyze that the LLM adopts different tabular formats, causing varying robustness on table understanding. Similarly, Deng et al. (2024) examine the different performance of text-based and image-based formats.

However, existing studies primarily discuss the impact of different tabular formats from the perspective of tasks, ignoring the impact of the instances and LLMs. Therefore, we claim that different instances and models suit different tabular formats, based on which, we propose FLEXTAF to improve the table reasoning performance by flexibly employing tabular formats.

6 Conclusion

In this paper, we explore the impact of tabular format on table reasoning performance from two aspects. (*i*) We claim from the perspective of experimental results that different instances and models have different most suitable tabular formats. (*ii*) We propose our methods. FLEXTAF-Single use the classifier to predict the most suitable tabular format according to the instance and the LLM. FLEXTAF-Vote obtains the results from multiple formats and employs the voting mechanism to get the final result. We build experiments on WikiTableQuestions and TabFact. Compared with the best performance of using a fixed format with greedy decoding and self-consistency, FLEXTAF-Single and FLEXTAF-Vote increase by 2.3% and 4.4% on average respectively, demonstrating their effectiveness.

Limitations

(i) The variety of table formats used in our experiments is somewhat limited, where in future work, we will explore allowing LLMs to autonomously determine the appropriate table format for each question, aiming to further enhance table-based reasoning performance; *(ii)* Our experiments are conducted only in English, where in future work, we plan to evaluate our methods across a broader range of languages to further validate their effectiveness.

Ethics Statement

Every dataset and model used in the paper is accessible to the public, and our application of them adheres to their respective licenses and conditions.

References

Alexandre Audibert, Aurélien Gauffre, and Massih-Reza Amini. 2024. Exploring contrastive learning for long-tailed multi-label text classification. *Preprint*, arXiv:2404.08720.

Kushal Raj Bhandari, Sixue Xing, Soham Dan, and Jianxi Gao. 2024. On the robustness of language models for tabular question answering. *Preprint*, arXiv:2406.12719.

Junyi Bian, Xiaolei Qin, Wuhe Zou, Mengzuo Huang, Congyi Luo, Ke Zhang, and Weidong Zhang. 2024. Helm: Highlighted evidence augmented language model for enhanced table-to-text generation. *Preprint*, arXiv:2311.08896.

Samuel R. Bowman. 2023. Eight things to know about large language models. *Preprint*, arXiv:2304.00612.

Yihan Cao, Shuyi Chen, Ryan Liu, Zhiruo Wang, and Daniel Fried. 2023. API-assisted code generation for question answering on varied table structures. In *Proc. of EMNLP*.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. 2024a. Two failures of self-consistency in the multi-step reasoning of LLMs. *Transactions on Machine Learning Research*.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *Proc. of ICLR*.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, and Jianye Hao. 2024b. Sheetagent: A generalist agent for spreadsheet reasoning and manipulation via large language models. *Preprint*, arXiv:2403.03636.

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. In *The Eleventh International Conference on Learning Representations*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Naihao Deng, Zhenjie Sun, Ruiqi He, Aman Sikka, Yulong Chen, Lin Ma, Yue Zhang, and Rada Mihalcea. 2024. Tables as texts or images: Evaluating the table reasoning ability of LLMs and MLLMs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 407–426, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5426–5435. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Haoyu Dong and Zhiruo Wang. 2024. Large language models for tabular data: Progresses and future directions. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2997–3000, New York, NY, USA. Association for Computing Machinery.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. 2024. Large scale transfer learning for tabular data via language modeling. *Preprint*, arXiv:2406.12031.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, pages 22–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María J. del Jesus. 2016. *Multilabel Classification*, pages 17–31. Springer International Publishing, Cham.

Nikolaus Howe, Michał Zajac, Ian McKenzie, Oskar Hollinsworth, Tom Tseng, Pierre-Luc Bacon, and Adam Gleave. 2024. Exploring scaling trends in llm robustness. *Preprint*, arXiv:2407.18213.

Younghun Lee, Sungchul Kim, Ryan A. Rossi, Tong Yu, and Xiang Chen. 2024. Learning to reduce: Towards improving performance of large language models on structured data. In *First Workshop on Long-Context Foundation Models @ ICML 2024*.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024. Table-gpt: Table fine-tuned gpt for diverse table tasks. *Proc. ACM Manag. Data*, 2(3).

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024a. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. 2024b. Understanding llms: A comprehensive overview from training to inference. *Preprint*, arXiv:2401.02038.

Xianzhen Luo, Qingfu Zhu, Zhiming Zhang, Libo Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Python is not always the best choice: Embracing multilingual program of thoughts. *Preprint*, arXiv:2402.10691.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. Technical report, Meta.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024a. Normtab: Improving symbolic reasoning in llms through tabular data normalization. *Preprint*, arXiv:2406.17961.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024b. Tabsqlify: Enhancing reasoning capabilities of llms through table decomposition. *Preprint*, arXiv:2404.10150.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. 2023. Lever: Learning to verify language-to-code generation with execution. In *Proc. of ICML*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex

Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proc. of ACL*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. CABINET: Content relevance-based noise reduction for table question answering. In *The Twelfth International Conference on Learning Representations*.

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? *Preprint*, arXiv:2405.00492.

Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. 2023. Strength in numbers: Estimating confidence of large language models by prompt agreement. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada. Association for Computational Linguistics.

Libo Qin, Qiguang Chen, Xiachong Feng, Yang Wu, Yongheng Zhang, Yinghui Li, Min Li, Wanxiang Che, and Philip S. Yu. 2024. Large language models meet nlp: A survey. *Preprint*, arXiv:2405.12819.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2695–2709, Singapore. Association for Computational Linguistics.

Ernesto Quevedo, Jorge Yero, Rachel Koerner, Pablo Rivas, and Tomas Cerny. 2024. Detecting hallucinations in large language model generation: A token probability approach. *Preprint*, arXiv:2405.19648.

Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *Preprint*, arXiv:2402.05201.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in LLMs. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 645–654, New York, NY, USA. Association for Computing Machinery.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Ran Wang, Xi'ao Su, Siyu Long, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2021. Meta-LMTC: Meta-learning for large-scale multi-label text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8633–8646, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves

chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yile Wang, Sijie Cheng, Zixin Sun, Peng Li, and Yang Liu. 2025. Leveraging language-based representations for better solving symbol-related problems with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5544–5557, Abu Dhabi, UAE. Association for Computational Linguistics.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *Preprint*, arXiv:2401.04398.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Guanglin Niu, Tongliang Li, and Zhoujun Li. 2024. Tablebench: A comprehensive and complex benchmark for table question answering. *Preprint*, arXiv:2408.09174.

Zirui Wu and Yansong Feng. 2024. Protrix: Building models for planning and reasoning over tables with sentence context. *Preprint*, arXiv:2403.02177.

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 *Proc. of SIGIR*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024a. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *Preprint*, arXiv:2403.02951.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024b. Tablellama: Towards open large generalist models for tables. *Preprint*, arXiv:2311.09206.

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

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024d. A survey of table reasoning with large language models. *Preprint*, arXiv:2402.08259.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2024e. Reactable: Enhancing react for table question answering. *Proc. VLDB Endow.*, 17(8):1981–1994.

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. In *Proc. of EMNLP*.

Yilun Zhao, Lyuhao Chen, Arman Cohan, and Chen Zhao. 2024. TaPERA: Enhancing faithfulness and interpretability in long-form table QA by content planning and execution-based reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12824–12840, Bangkok, Thailand. Association for Computational Linguistics.

A Introduction to Five Tabular Formats

In this section, we introduce the five tabular formats used in experiments: Markdown, Dict, List, Pandas, and Database. The Markdown format refers to the representing tables in the Markdown language. The Dict format employs `List[Dict: [str, Any]]` to index the table, in which each row is stored in a dictionary, and the column name of each column is indexed as the key of the dictionary. The List format adopts `List[List[Any]]` to index the table, in which each row, including the header, is stored in the list, and each column needs to be indexed with the serial number of the column. The Pandas format is a Python code snippet that uses the Pandas DataFrame API to define the table, which points out each line of the table and the header. Database format refers to the format of representing a table as a database, describing the column name with a CREATE statement, and listing specific values.

B Additional Experiments

TableBench (Wu et al., 2024) is a dataset that is closer to industrial scenarios, containing 18 question classifications. We use Llama3.1-Instruct-8B (Llama3.1-8B) for the experiment, employ the zero-shot prompt, and keep other settings consistent with the main experiment. We adopt Rouge-L (Lin, 2004) as the evaluation metric following TableBench. The results of greedy decoding using the fixed tabular formats and the results of FLEXTAF are shown in Table 6. The results reveal that FLEXTAF-Single and FLEXTAF-Vote consistently surpass the best performance of using the fixed format, proving the effectiveness. Moreover, on the more challenging dataset, the greater performance gap between different tabular formats suggests the use of flexible tabular formats.

C Prompts with Each Tabular Format

In this section, we present the prompts used in experiments.

C.1 Prompts for Main Experiments

The prompts for WikiTQ (Pasupat and Liang, 2015) in main experiments with a single tabular format are shown in Table 7, Table 8, Table 9, Table 10, and Table 11. And the prompts for TabFact (Chen et al., 2020) in main experiments with a single tabular format are shown in Table 12, Table 13, Table 14, Table 15, and Table 16. It can be found that only the demonstrations with the Database format are different among the prompts for WikiTQ. If we use the same demonstrations as other prompts, the SQL given in the prompt is too complicated to reduce the reasoning performance of the model.

C.2 Prompts for Experiments of Unifying Demonstrations

We present the prompt with the unified demonstrations in this subsection. From Table 7, Table 8, Table 9 and Table 10, it can be seen that the demonstrations used in the Markdown, Dict, List, and Pandas are unified, so we only change the prompt with the Database format, which is shown in Table 17.

D Tabular Format Distributions on Models

In this section, we show the specific process of proving that the tabular format is distributed differently on the model, which is discussed in §2.

Specifically, we take the number correctly represented on each model as the observed frequency O_i, calculate the average correct number of each format under different models as the expected frequency E_i, and calculate the Chi-square statistics as Equation D.1.

$$\mathcal{X}^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{D.1}$$

The corresponding degree of freedom dof is $dof = (N_m - 1) * (N_r - 1)$, where N_m is the number of models, and N_r is the number of tabular formats. According to the calculated Chi-square $\mathcal{X}^2$ and the corresponding degree of freedom dof, we find the corresponding P-value from the square distribution table, which is less than 0.05, proving that there is a distinct difference in the distribution of the correct tabular formats on different models.

E Details of Training the Classifier in FLEXTAF-Single

In this section, we introduce the details of training the classifier in FLEXTAF-Single. Our model is implemented with PyTorch (Paszke et al., 2019) and transformers (Wolf et al., 2020). We present training details in Table 18, and we summarize the training data size employed for training the classifier to predict each LLM across various datasets in Table 19. We count the proportion of each tabular format in the training data for different LLMs and

Method	Markdown	Dict	List	Pandas	Database
Greedy decoding	23.9	21.6	6.9	6.1	22.0
FLEXTAF-Single			**25.0**		
Self-consistency	26.5	26.1	17.3	10.4	22.7
FLEXTAF-Vote			**31.2**		

Table 6: The Rouge-L of using a fixed tabular format and FLEXTAF, using Llama3.1-8B on TableBench.

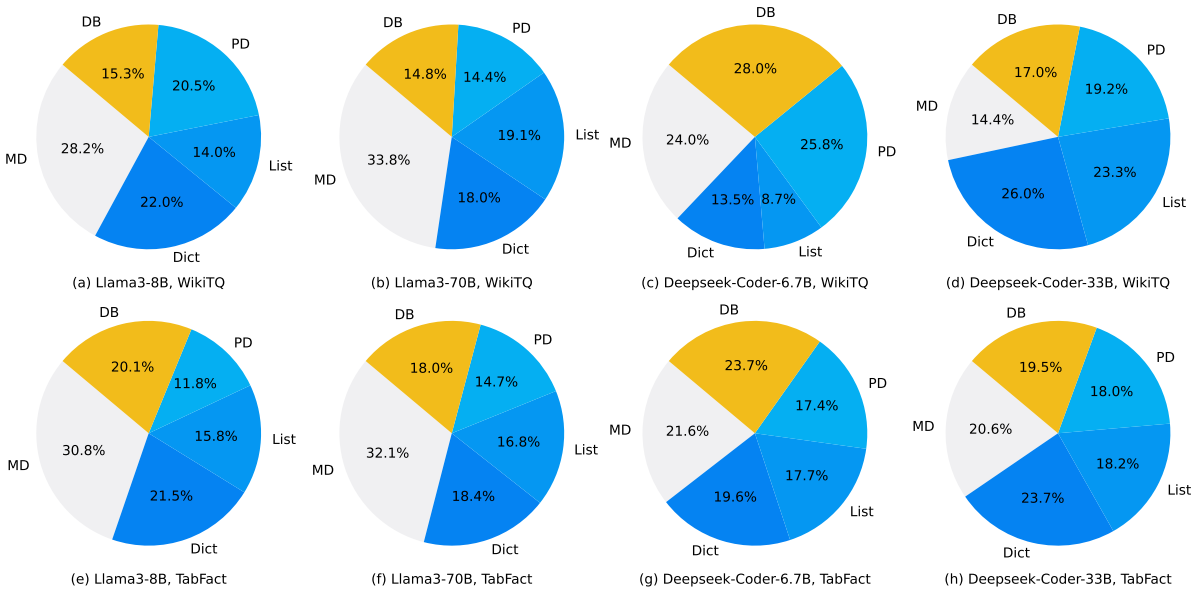

Figure 7: The proportion of each tabular format in the training data.

datasets, as shown in Figure 7. Specifically, to compare the proportions of different tabular formats in the training data, we calculate the proportion of each format to all labels of all instances in the training data.

F Overlap between Tabular Formats

In this section, we analyze the overlap between instances correctly solved by different tabular formats, which is shown in Figure 8. The overlap between instances solved by different tabular formats is all $\leq 100\%$, which proves that different instances are suitable for different tabular formats. We can find that: (*i*) The overlap caused by using DeepSeek-Coder is greater than that caused by Llama3 because DeepSeek-Coder is better at code formats. The difference between code formats such as Dict and List is smaller than that between code formats and natural language formats, such as Markdown. (*ii*) The overlap caused by employing large-scale LLMs is more serious than that of small-scale LLMs because the large-scale LLMs have higher table reasoning performance, leading to more instances correctly solved by tabular formats. (*iii*) The overlap on TabFact is greater than that on WikiTQ because the questions of TabFact are simpler and easier to solve by more tabular formats.

G Comparison FLEXTAF with Oracle

In this section, we compare the performance of FLEXTAF with that of Oracle, as shown in Table 20. We observe that: (*i*) The excellent oracle performance shows that there exist differences between different formats, which further suggests that different instances have their suitable tabular formats. (*ii*) The performance of FLEXTAF-Single is limited by classification and has a gap with the oracle performance, since the Pre-trained Language Model (PLM) cannot predict the behavior of LLMs well (Qin et al., 2024; Liu et al., 2024b; Bowman, 2023). However, we do not conduct experiments to fine-tune LLMs for classification limited by computing resources. (*iii*) The performance of FLEXTAF-Vote also has a gap with the oracle performance, because the voting mechanism we adopt does not make full use of the rich semantic information in the LLM solutions, causing the limited improvement (Ni et al.,

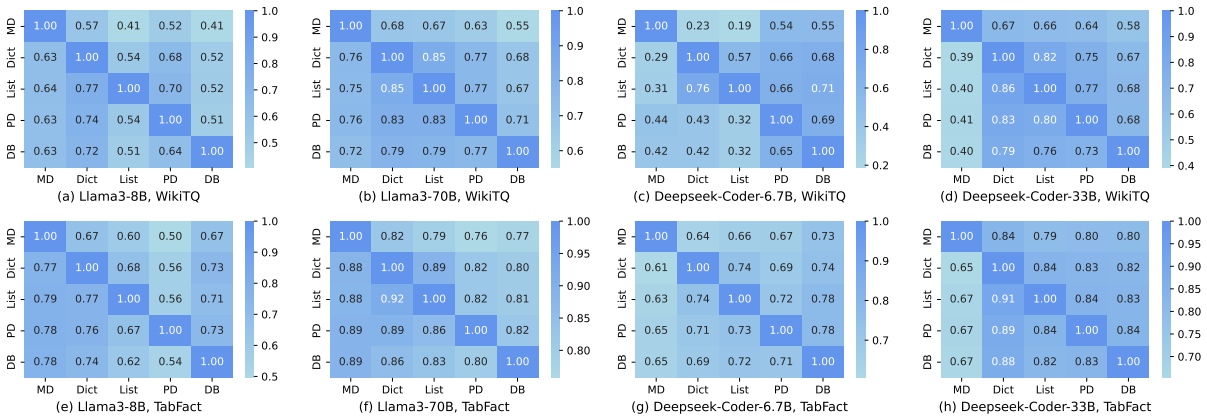

Figure 8: The overlap among instances that are correctly solved with each tabular format using four LLMs on two datasets. The value represents the proportion that can be solved by the tabular format corresponding to the vertical axis in the instances that are solved by the format corresponding to the horizontal axis.

2023).

H Cases Study

To clearly show the impact of the table, we show more instances in WikiTQ in this section, as shown in Figure 9. We observe that: (*i*) For the first instance with the question "*what is the highest points scored by an opponent?*", DeepSeek-Coder-33B correctly solves the instance with the Dict format, while Llama3-70B is suitable for Markdown and Pandas formats. (*ii*) When we use the same model Llama3-70B, the last instance with the question "*in cycle 4, how many contestants are older than 20?*" is solved correctly with Dict, which is different from the previous instance. The cases claim that different instances and LLMs have different most suitable tabular formats.

Question
what is the highest points scored by an opponent?
Table

Opponent	Score	…
at Toronto Argonauts	13–6	
…		
vs. Montreal Alouettes	14–12	…

DeepSeek-Coder-33b
Dict
```
def solver(table):
    max_points = 0
    for row in table:
        points = int(row["Score"].split("–")[0])
        if points > max_points:
            max_points = points
    return max_points
```
Answer
45

Pandas
```
def solver(table):
    import pandas as pd
    table[['Team_Score', 'Opponent_Score']] =
table['Score'].str.split('–', expand=True)
    table['Team_Score'] = table['Team_Score'].astype(int)
    return table['Opponent_Score'].astype(int).max()
```

Answer
17

Markdown
To find out the highest amount of points scored in a game by an opponent, we need to look at the "Score" column of the table provided. According to the table, the highest amount of points scored in a game by an opponent is 38, which was scored against Hamilton Tiger-Cats, so the answer is: **38**
Answer
38

Question
what is the highest points scored by an opponent?
Table

Opponent	Score	…
at Toronto Argonauts	13–6	
…		
vs. Montreal Alouettes	14–12	…

Llama3-70b
Dict
```
def solver(table):
    max_points = 0
    for row in table:
        score = row["Score"].split("-")
        opponent_points = int(score[1])
        if opponent_points > max_points:
            max_points = opponent_points
    return max_points
```
Answer
17

Pandas
```
def solver(table):
    import pandas as pd
    max_points_scored = table['Score'].apply(lambda x:
int(max(x.split('–')[0], x.split('–')[1]))).max()
    return max_points_scored
```

Answer
45

Markdown
To find out what is the highest amount of points scored in a game by an opponent, we need to look at the "Score" column of the table provided. According to the table, the highest amount of points scored in a game by an opponent is 45, so the answer is: **45**
Answer
45

Question
in cycle 4,how many contestants are older than 20?
Table

Contestant	Age	…
Isabelle Raisa	16	…
…		
Nadine Trinker	21	…

Llama3-70b
Dict
```
def solver(table):
    count = 0
    for row in table:
        if int(row["Age"]) > 20:
            count += 1
    return count
```
Answer
5

Pandas
```
def solver(table):
    import pandas as pd
    contestants_over_20 = table[table["Age"] >
20].shape[0]
    return contestants_over_20",
```

Answer
Error occurred

Markdown
To find out how many contestants were older than 20 in Cycle 4, we need to look at the "Age" column of the table provided. According to the table, the contestants older than 20 are Sabrina Angelika Rauch(21), Katharina Mihalovi(23), Michaela Schopf(21), Christine **Riener(20)**, Bianca Ebelsberger(24), Nadine Trinker(21), and **Dzejlana Lana(20)**, so the answer is: **7**
Answer
7

Figure 9: Two instances of WikiTQ test set using Llama3-70B and DeepSeek-Coder-33B with Markdown and Dict tabular format.

The prompt with the Markdown format for WikiTQ.

Please answer the question with the given table, present the final result in the format "..., so the answer is: (answer)". Please note that utilize the format, do not include periods. Here are some instances you may refer to:

—

table:
$Aircraft$	$Description$	$MaxGrossWeight$	$Totaldiskarea$	$MaxdiskLoading$
: − − −	: − − −	: − − −	: − − −	: − − −
$RobinsonR-22$	$Lightutilityhelicopter$	$1,370lb(635kg)$	$497ft(46.2m)$	$2.6lb/ft(14kg/m)$
$Bell206B3JetRanger$	$Turboshaftutilityhelicopter$	$3,200lb(1,451kg)$	$872ft(81.1m)$	$3.7lb/ft(18kg/m)$
$CH-47DChinook$	$Tandemrotorhelicopter$	$50,000lb(22,680kg)$	$5,655ft(526m)$	$8.8lb/ft(43kg/m)$
$MilMi-26$	$Heavy-lifthelicopter$	$123,500lb(56,000kg)$	$8,495ft(789m)$	$14.5lb/ft(71kg/m)$
$CH-53ESuperStallion$	$Heavy-lifthelicopter$	$73,500lb(33,300kg)$	$4,900ft(460m)$	$15lb/ft(72kg/m)$

utterance:
What is the max gross weight of the Robinson R-22?
answer:
To find out what is the max gross weight of Robinson R-22, we need to look at the "Max Gross Weight" column of the table provided. According to the table, the max gross weight of the Robinson R-22 is 1,370 lb (635 kg), so the answer is: 1,370 lb (635 kg)

—

table:
$Player$	$No.$	$Nationality$	$Position$	$YearsinToronto$	$School/ClubTeam$
: − − −	: − − −	: − − −	: − − −	: − − −	: − − −
$MarkBaker$	3	$UnitedStates$	$Guard$	$1998-99$	$OhioState$
$MarcusBanks$	3	$UnitedStates$	$Guard$	$2009-10$	$UNLV$
$LeandroBarbosa$	20	$Brazil$	$Guard$	$2010-2012$	$Tilibra/Copimax(Brazil)$
$RasualButler$	9	$UnitedStates$	$Guard-Forward$	$2011-12$	$LaSalle$

utterance:
How many players were with the school or club team La Salle?
answer:
To count the number of players with the school or club team La Salle, we need to look at the "School/Club Team" column of the table provided. According to the table, there are 1 player whose School/Club Team is La Salle, so the answer is: 1

—

table:
$Model$	1991	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004
: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −
$SkodaFelicia$	172,000	210,000	−	288,458	261,127	241,256	148,028	44,963	−
$SkodaOctavia$	−	−	−	47,876	102,373	143,251	158,503	164,134	164,017	165,635	181,683
$SkodaFabia$	−	−	−	−	−	823	128,872	250,978	264,641	260,988	247,600
$SkodaSuperb$	−	−	−	−	−	−	−	177	16,867	23,135	22,392

utterance:
is the number on skoda fabia for 1999 more or less than 1000?
answer:
To find out if the number on the Skoda Fabia for 1999 is more or less than 1000, we need to look at the data provided for the "Skoda Fabia" in "1999", which is 823, that is, the number for the Skoda Fabia in 1999 is less than 1000, so the answer is: less

—

table:
$Place$	$Rider$	$Country$	$Team$	$Points$	$Wins$
: − − −	: − − −	: − − −	: − − −	: − − −	: − − −
1	$SylvainGeboers$	$Belgium$	$Suzuki$	3066	3
2	$AdolfWeil$	$Germany$	$Maico$	2331	2
3	$JohnBanks$	$UnitedKingdom$	CZ	971	0
4	$MarkBlackwell$	$UnitedStates$	$Husqvarna$	604	0
5	$BradLackey$	$UnitedStates$	CZ	603	0
6	$GaryJones$	$UnitedStates$	$Yamaha$	439	0
7	$JohnDeSoto$	$UnitedStates$	$Suzuki$	425	0

utterance:
which country had the most riders that placed in the top 20 of the 1971 trans-ama final standings?
answer:
To find out which country had the most riders in the top 20, we need to look at the "Country" column of the table provided and count the number of times each country appears. According to the table, United States had the most riders, so the answer is: United States

—

table:
<table>
utterance:
<utterance>
answer:

Table 7: The prompt with the Markdown format used in the main experiments for WikiTQ.

The prompt with the Dict format for WikiTQ.

Answer the question with the given table using python code.
You should generate a function with the following signature without any other parameters. Here are some instances you may refer to:

—

```
table = [
    {
        "Aircraft": "Robinson R-22",
        "Description": "Light utility helicopter",
        "Max Gross Weight": "1,370 lb (635 kg)",
        ...
    },
    ...
]
utterance: What is the max gross weight of the Robinson R-22?
def solver(table):
    for row in table:
        if row["Aircraft"] == "Robinson R-22":
            return row["Max Gross Weight"]
```

—

```
table = [...]
utterance: How many players were with the school or club team La Salle?
def solver(table):
    players_la_salle = set()
    for row in table:
        if row["School/Club Team"] == "La Salle": players_la_salle.add(row["Player"])
    return len(players_la_salle)
```

—

```
table = [...]
utterance: is the number on skoda fabia for 1999 more or less than 1000?
def solver(table):
    for row in table:
        if row["Model"] == "Skoda Fabia": num_1999 = row["1999"].replace(",", "")
            if int(num_1999) > 1000: return "more"
            else: return "less"
    return "less"
```

—

```
table = [...]
utterance: which country had the most riders that placed in the top 20 of the 1971
trans-ama final standings?
def solver(table):
    country_counts = {}
    for row in table:
        country = row["Country"]
        if country in country_counts: country_counts[country] += 1
        else: country_counts[country] = 1
    max_riders = max(country_counts.values())
    countries_with_max_riders = [country for country, count in country_counts.items()
    if count == max_riders]
    return countries_with_max_riders[0]
```

—

Based on the above demonstrations, answer the following utterance with the following table using Python code.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 8: The prompt with the Dict format used in the main experiments for WikiTQ. Due to the limited length of the paper, we do not list the contents of all the tables in the demonstrations, which are the same tables in Table 7. We will release the full prompt upon the acceptance.

The prompt with the List format for WikiTQ.

Answer the question with the given table using python code.
You should generate a function with the following signature without any other parameters. Here are some instances you may refer to:

—

```
table = [
    [
        "Aircraft",
        "Description",
        "Max Gross Weight",
        ...
    ],
    [
        "Robinson R-22",
        "Light utility helicopter",
        "1,370 lb (635 kg)",
        ...
    ],
    ...
]
utterance: What is the max gross weight of the Robinson R-22?
def solver(table):
    for row in table[1:]:
        if row[0] == "Robinson R-22": return row[2]
```

—

```
table = [...]
utterance: How many players were with the school or club team La Salle?
def solver(table):
    players_la_salle = set()
    for row in table[1:]:
        if row[5] == "La Salle": players_la_salle.add(row[0])
    return len(players_la_salle)
```

—

```
table = [...]
utterance: is the number on skoda fabia for 1999 more or less than 1000?
def solver(table):
    for row in table[1:]:
        if row[0] == 'Skoda Fabia':
            if row[6] == "-" or int(row[6].replace(",", "")) < 1000: return "less"
            else:mreturn "more"
```

—

```
table = [...]
utterance: which country had the most riders that placed in the top 20 of the 1971
trans-ama final standings?
def solver(table):
    country_counts = {}
    for row in table[1:]:
        country = row[2]
        if country in country_counts: country_counts[country] += 1
        else: country_counts[country] = 1
    max_riders = max(country_counts.values())
    countries_with_max_riders = [country for country, count in country_counts.items()
    if count == max_riders]
    return countries_with_max_riders[0]
```

—

Based on the above demonstrations, answer the following utterance with the following table using Python code.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 9: The prompt with the List format used in the main experiments for WikiTQ. Due to the limited length of the paper, we do not list the contents of all the tables in the demonstrations, which are the same tables in Table 7. We will release the full prompt upon the acceptance.

The prompt with the Pandas format for WikiTQ.

Answer the question with the given table using python code.
You should generate a function with the following signature without any other parameters. Here are some instances you may refer to:

—

```
table = pd.DataFrame([
    [
        "Robinson R-22",
        "Light utility helicopter",
        "1,370 lb (635 kg)",
        ...
    ],
    ...
], columns = [
    "Aircraft",
    "Description",
    "Max Gross Weight",
    ...
]
)
utterance: What is the max gross weight of the Robinson R-22?
def solver(table):
    import pandas as pd
    max_gross_weight_r22 = table[table["Aircraft"] == "Robinson R-22"]
    ["Max Gross Weight"].iloc[0]
    return max_gross_weight_r22
```

—

```
table = [...]
utterance: How many players were with the school or club team La Salle?
def solver(table):
    import pandas as pd
    la_salle_count = table[table["School/Club Team"] == "La Salle"].shape[0]
    return la_salle_count
```

—

```
table = [...]
utterance: is the number on skoda fabia for 1999 more or less than 1000?
def solver(table):
    import pandas as pd
    fabia_row = table[table['Model'] == 'Skoda Fabia']
    fabia_1999_sales = fabia_row['1999'].values[0]
    if fabia_1999_sales == '-' or int(fabia_1999_sales.replace(",", "")) < 1000:
        return 'less'
    else:
        return 'more'
```

—

```
table = [...]
utterance: which country had the most riders that placed in the top 20 of the 1971
trans-ama final standings?
def solver(table):
    import pandas as pd
    most_riders_country = table['Country'].value_counts().idxmax()
    return most_riders_country
```

—

Based on the above demonstrations, answer the following utterance with the following table using Python code.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 10: The prompt with the Pandas format used in the main experiments for WikiTQ. Due to the limited length of the paper, we do not list the contents of all the tables in the demonstrations, which are the same tables in Table 7. We will release the full prompt upon the acceptance.

The prompt with the Database format for WikiTQ.

Please complete the sql below to solve the question with the given database.
Here are some instances you may refer to:
—

```
database:
CREATE TABLE information (
year int ,
division int ,
...
);
/*
Columns and instances in each column :
year: 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, ... ;
...
*/
utterance:
when was the first time the kansas city brass qualified for the playoffs?
sql:
SELECT year FROM information WHERE playoffs != 'Did not qualify' ORDER
BY year ASC LIMIT 1;
```

—

```
database:
CREATE TABLE information (...);
/*...*/
utterance:
what was the next episode after \"do-si-do?\"
sql:
SELECT episode FROM information WHERE num = (SELECT num FROM information
WHERE episode = 'Do-Si-Do') + 1;
```

—

```
database:
CREATE TABLE information (...);
/*...*/
utterance:
which dino album yielded the most songs on the billboard hot 100?
sql:
SELECT album FROM information WHERE chart = 'Billboard Hot 100' GROUP BY album
ORDER BY COUNT(*) DESC LIMIT 1;
```

—

```
database:
CREATE TABLE information (...);
/*...*/
utterance:
when was the last year team penske finished first?
sql:
SELECT MAX(year) FROM information WHERE team = 'Team Penske' AND finish = 1;
```

—

Based on the above demonstrations, answer the following utterance with the following database using SQL.
database:
<table>
utterance:
<utterance>
sql:
SELECT

Table 11: The prompt with the Database format used in the main experiments for WikiTQ. Due to the limited length of the paper, we do not list the contents of all the tables in the demonstrations. We will release the full prompt upon the acceptance.

The prompt with the Markdown format on TabFact.

Verify the consistency between the table and the utterance.
Please present the final result in the format "..., so the answer is: (answer)" and the "(answer)" is "True" or "False".
Please note that utilize the format, do not include periods.
Here are some demonstrations you may refer to:
—
table:
$tournament$	$wins$	$top-5$	$top-10$	$top-25$	$events$	$cutsmade$
: − − −	: − − −	: − − −	: − − −	: − − −	: − − −	: − − −
$masterstournament$	0	1	2	4	4	4
$usopen$	0	2	3	4	6	5
$theopenchampionship$	1	2	2	2	3	3
$pgachampionship$	0	0	1	2	5	4
$totals$	1	5	8	12	18	16
utterance:
tony lema be in the top 5 for the master tournament , the us open , and the open championship
answer:
To verify whether tony lema be in the top 5 for the master tournament , the us open , and the open championship, we need to look at the "top - 5" column of the table provided. According to the table, the "top - 5" column of the "masters tournament", "us open", and "the open championship" are all more than zero, so the answer is: True
—
table:
$year$	$competition$	$venue$	$position$	$event$
: − − −	: − − −	: − − −	: − − −	: − − −
2006	$worldcrosscountrychampionships$	$fukuoka, japan$	$10th$	$individualjuniorrace$
2006	$worldcrosscountrychampionships$	$fukuoka, japan$	$3rd$	$teamjuniorrace$
2006	$africanchampionshipsinathletics$	$bambous, mauritius$	$5th$	$10000m$
2006	$worldroadrunningchampionships$	$debrecen, hungary$	$7th$	$individual20km$
2006	$worldroadrunningchampionships$	$debrecen, hungary$	$3rd$	$team20km$
2007	$worldcrosscountrychampionships$	$mombasa, kenya$	$7th$	$individual$
2007	$all-africagames$	$algiers, algeria$	$2nd$	$10000m$
2007	$worldchampionshipsinathletics$	$osaka, japan$	$13th$	$10000m$
2009	$worldcrosscountrychampionships$	$amman, jordan$	$17th$	$individual$
2013	$worldchampionships$	$moscow, russia$	$3rd$	$marathon$
utterance:
japan and hungary host the competition 3 time each
answer:
To verify whether japan and hungary both host the competition 3 time, we need to look at the "venue" column of the table provided. According to the table, "japan" hosts the competition 3 times, but "hungary" hosts the competition 2 times, so the answer is: False
—
Based on the above demonstrations, Verify the consistency between the following table and utterance.
table:
<table>
utterance:
<utterance>
answer:

Table 12: The prompt with the Markdown format used in the main experiments for TabFact.

The prompt with the Dict format for TabFact.

Verify the consistency between the table and the utterance with "True" or "False" using python code.
You should generate a function with the following signature without any other parameters:
Here are some demonstrations you may refer to:
—

```
table = [
    {
        "tournament": "masters tournament",
        "wins": "0",
        "top - 5": "1",
        "top - 10": "2",
        "top - 25": "4",
        "events": "4",
        "cuts made": "4"
    },
    {
        "tournament": "us open",
        "wins": "0",
        "top - 5": "2",
        "top - 10": "3",
        "top - 25": "4",
        "events": "6",
        "cuts made": "5"
    },
    ...
]
utterance: tony lema be in the top 5 for the master tournament , the us open ,
and the open championship
def solver(table):
    top_5_tournament = [row["tournament"] for row in table if int(row["top - 5"]) > 0]
    if "masters tournament" not in top_5_tournament:
        return False
    if "us open" not in top_5_tournament:
        return False
    if "the open championship" not in top_5_tournament:
        return False
    return True
```

—

```
table = [
    {
        "year": "2006",
        "competition": "world cross country championships",
        "venue": "fukuoka , japan",
        "position": "10th",
        "event": "individual junior race"
    },
    ...
]
utterance: japan and hungary host the competition 3 time each
def solver(table):
    japan_host_time = 0
    hungary_host_time = 0
    for row in table:
        if "japan" in row["venue"]:
            japan_host_time += 1
        elif "hungary" in row["venue"]:
            hungary_host_time += 1
    return (japan_host_time == 3 and hungary_host_time == 3)
```

—
Based on the above demonstrations, Verify the consistency between the following table and utterance.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 13: The prompt with the Dict format used in the main experiments for TabFact.

The prompt with the List format for TabFact.

Verify the consistency between the table and the utterance with "True" or "False" using python code.
You should generate a function with the following signature without any other parameters:
Here are some demonstrations you may refer to:
—

```
table = [
    [
        "tournament",
        "wins",
        "top - 5",
        "top - 10",
        "top - 25",
        "events",
        "cuts made"
    ],
    [
        "masters tournament",
        "0",
        "1",
        "2",
        "4",
        "4",
        "4"
    ],
    ...
]
utterance: tony lema be in the top 5 for the master tournament , the us open ,
and the open championship
def solver(table):
    top_5_tournament = [row[0] for row in table[1:] if int(row[2]) > 0]
    if "masters tournament" not in top_5_tournament:
        return False
    if "us open" not in top_5_tournament:
        return False
    if "the open championship" not in top_5_tournament:
        return False
    return True
```

—

```
table = [
    [
        "year",
        "competition",
        "venue",
        "position",
        "event"
    ],
    ...
]
utterance: japan and hungary host the competition 3 time each
def solver(table):
    japan_host_time = 0
    hungary_host_time = 0
    for row in table[1:]:
        if "japan" in row[2]:
            japan_host_time += 1
        elif "hungary" in row[2]:
            hungary_host_time += 1
    return (japan_host_time == 3 and hungary_host_time == 3)
```

—
Based on the above demonstrations, answer the following utterance with the following table using Python code.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 14: The prompt with the List format used in the main experiments for TabFact.

The prompt with the Pandas format for TabFact.

Verify the consistency between the table and the utterance with "True" or "False" using python code.
You should generate a function with the following signature without any other parameters:
Here are some demonstrations you may refer to, and you don't need to answer the demonstrations:
—

```
table = pd.DataFrame([
    [
        "masters tournament",
        "0",
        "1",
        "2",
        "4",
        "4",
        "4"
    ],
    ...
], columns = [
    "tournament",
    "wins",
    "top - 5",
    "top - 10",
    "top - 25",
    "events",
    "cuts made"
]
)
utterance: tony lema be in the top 5 for the master tournament , the us open ,
and the open championship
def solver(table):
    tournaments = ["masters tournament", "us open", "the open championship"]
    for tournament in tournaments:
        row = table[table['tournament'] == tournament]
        if row.empty or int(row['top - 5'].values[0]) == 0:
            return False
    return True
```

—
Based on the above demonstrations, answer the following utterance with the following table using Python code.
table = <table>
utterance: <utterance>
def solver(table):
# Your code here

Table 15: The prompt with the Pandas format used in the main experiments for TabFact.

The prompt with the Database format for TabFact.

Please complete the sql below to solve the question with the given database.
Here are some instances you may refer to:
—

```
CREATE TABLE information (
tournament text ,
wins int ,
top_5 int ,
top_10 int ,
top_25 int ,
events int ,
cuts_made int
);
/*
Columns and instances in each column :
tournament: masters tournament, us open, the open championship, pga championship, totals ;
wins: 0, 0, 1, 0, 1 ;
top_5: 1, 2, 2, 0, 5 ;
top_10: 2, 3, 2, 1, 8 ;
top_25: 4, 4, 2, 2, 12 ;
events: 4, 6, 3, 5, 18 ;
cuts_made: 4, 5, 3, 4, 16 ;
*/

utterance:
tony lema be in the top 5 for the master tournament , the us open ,
and the open championship

SQL:
SELECT CASE  WHEN (SELECT COUNT(*) FROM information WHERE tournament IN
('masters tournament', 'us open', 'the open championship') AND top_5 > 0) = 3
THEN 'True' ELSE 'False' END AS result;
```

—

```
CREATE TABLE information (
year int ,
competition text ,
venue text ,
position text ,
event text
);
/*
Columns and instances in each column :
year: 2006, 2006, 2006, 2006, 2006, 2007, 2007, 2007, 2009, 2013 ;
competition: world cross country championships, world cross country championships, ... ;
venue: fukuoka , japan, fukuoka , japan, bambous , mauritius, debrecen , hungary, ... ;
position: 10th, 3rd, 5th, 7th, 3rd, 7th, 2nd, 13th, 17th, 3rd ;
event: individual junior race, team junior race, 10000 m, ... ;
*/

utterance:
japan and hungary host the competition 3 time each

SQL:
SELECT CASE WHEN (SELECT COUNT(*) FROM information WHERE venue LIKE '%japan%') = 3 AND
(SELECT COUNT(*) FROM information WHERE venue LIKE '%hungary%') = 3 THEN 'True'
ELSE 'False' END AS result;
```

—
Based on the above demonstrations, answer the following utterance with the following database using SQL.
database:
<table>
utterance:
<utterance>
sql:
SELECT

Table 16: The prompt with the Database format used in the main experiments for TabFact.

The prompt with the Database format using unified demonstrations.

Please complete the sql below to solve the question with the given database.
Here are some instances you may refer to:
—

```
database:
CREATE TABLE information (
aircraft text ,
...
);
/*
Columns and instances in each column :
aircraft: Robinson R-22, Bell 206B3 JetRanger, CH-47D Chinook, ... ;
...
*/
utterance:
What is the max gross weight of the Robinson R-22?
sql:
SELECT max_gross_weight FROM information WHERE aircraft = 'Robinson R-22';
```

—

```
database:
CREATE TABLE information (
player text ,
...
);
/*...*/
utterance:
How many players were with the school or club team La Salle?
sql:
SELECT COUNT(*) FROM information WHERE school_club_team = 'La Salle';
```

—

```
database:
CREATE TABLE information (
model text ,
...
);
/*...*/
utterance:
is the number on skoda fabia for 1999 more or less than 1000?
sql:
SELECT CASE WHEN CAST(column_1999 AS INTEGER) < 1000 THEN 'less' ELSE 'more'
END AS result FROM information WHERE model = 'Skoda Fabia';
```

—

```
CREATE TABLE information (
...
);
/*...*/
utterance:
which country had the most riders that placed in the top 20?
sql:
SELECT country, COUNT(rider) as rider_count FROM information WHERE place <= 20
GROUP BY country ORDER BY rider_count DESC LIMIT 1;
```

—
Based on the above demonstrations, answer the following utterance with the following database using SQL.
database:
<table>
utterance:
<utterance>
sql:
SELECT

Table 17: The prompt with the Database format with the unified demonstrations. Due to the limited length of the paper, we do not list the contents of all the tables in the demonstrations, which are the same tables in Table 7.

	WikiTQ	TabFact
batch size	128	128
epoch	200	400
learning rate	$1e-5$	$5e-6$
max tokens	512	512
training device	2× NVIDIA A100 40G GPU	2× NVIDIA A100 40G GPU
training time	$16h$	$36h$

Table 18: Classification training details in FLEXTAF-Single on WikiTQ and TabFact.

	WikiTQ				TabFact			
	Llama3		DeepSeek-Coder		Llama3		DeepSeek-Coder	
	8B	70B	6.7B	33B	8B	70B	6.7B	33B
Data Size	7,247	5,649	7,848	5,944	48,073	26,050	45,194	29,619

Table 19: The training data size employed for training the classifier in FLEXTAF-Single.

Tabular Format	WikiTQ				TabFact			
	Llama3		DeepSeek-Coder		Llama3		DeepSeek-Coder	
	8B	70B	6.7B	33B	8B	70B	6.7B	33B
FLEXTAF-Single	50.5	69.1	46.3	54.5	77.0	87.1	70.9	78.3
FLEXTAF-Vote	**55.7**	**69.9**	**51.4**	**60.9**	**80.3**	**88.5**	**77.9**	**84.4**
Oracle	71.8	83.5	67.7	74.6	96.9	98.1	95.4	97.1

Table 20: The performance of FLEXTAF-Single, FLEXTAF-Vote, and Oracle, which denotes the result that each instance uses the tabular format that can get the correct answer.