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. While prior works typically rely on a fixed tabular format within prompts, we argue that this overlooks the critical interaction between the format, the specific instance, and the model's capabilities. We demonstrate that the optimal tabular format is not static but functions as a structural prior that modulates the accessibility of reasoning paths depending on the query and the LLM. To address this, we propose two methods to incorporate format flexibility: FLEXTAF-Single, which trains a discriminator to predict the most suitable format for a given instance, and FLEXTAF-Vote, a training-free method that aggregates results from multiple formats via voting. Extensive experiments on WikiTableQuestions, TabFact, and TableBench validate our analysis and demonstrate the effectiveness of our methods. Compared to strong baselines using fixed formats, our methods achieve average improvements of 2.7% and 4.0%, with comparable inference costs[1].

1 Introduction

Table reasoning is a fundamental 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. Leveraging the superior commonsense and logical reasoning capabilities of Large Language Models (LLMs), recent research has adopted them as the dominant approach for table reasoning (Chen, 2023; Zhang et al., 2025a). Consequently, this paper focuses on enhancing LLM-based table reasoning.

Prior prompting methods (Lee et al., 2024; Yang et al., 2025; Zhang et al., 2025b) typically rely on fixed tabular formats. While Sui et al. (2024) note that optimal formats vary by task, existing work overlooks critical variations at the *instance and model levels*. For example, as shown in Figure 1, solving instance (a) requires sequential indexing, which is facilitated by the `List` format, while instance (b) requires identifying columns under specific conditions, making the `Database` format more suitable. Furthermore, since models exhibit distinct reasoning capabilities (Bhandari et al., 2024; Murahari et al., 2024; Tian et al., 2025), the most suitable format is also model-dependent.

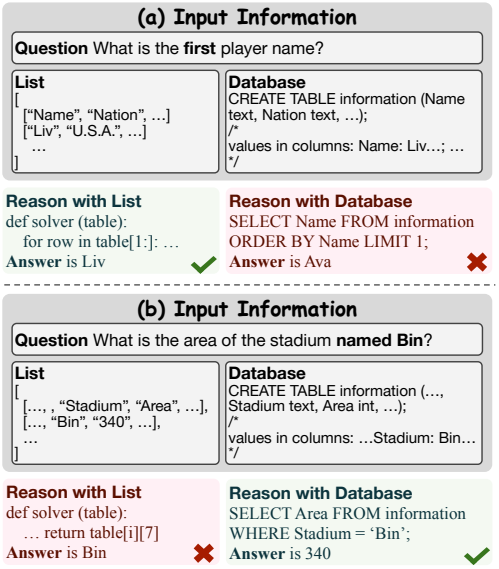

Figure 1: The table reasoning performance varies with tabular formats. The `List` format is convenient for sequential indexing in (a), while the `Database` format facilitates the search for columns that meet specific conditions in (b).

Based on these observations, we investigate the impact of tabular formats on LLM performance: (*i*) We analyze **why optimal formats vary across instances and models**; (*ii*) We enhance table reasoning by **predicting the optimal format**[2] **or aggregating multi-format results**.

[1] Our code is in the supplementary material.

[2] Defined as the format enabling the model to yield the correct solution.

First, we analyze the dependency of optimal formats on the instance and the LLM. We posit that tabular formats function as structural priors that modulate the accessibility of reasoning paths required by specific queries. Furthermore, they act as capability triggers that align input representations with the model's training corpus, thereby steering the model toward its most proficient reasoning modes. Building on this analysis, we propose FLEXTAF-Single and FLEXTAF-Vote to improve performance through flexible formatting. FLEXTAF-Single identifies the optimal format for a given instance and LLM via a trained discriminator, utilizing that format for inference. FLEXTAF-Vote determines the final answer by voting on results generated from multiple formats. Comparatively, FLEXTAF-Single requires only single-pass inference, whereas FLEXTAF-Vote is training-free.

To validate our claims and methods, we conduct experiments on three mainstream table reasoning datasets: WikiTableQuestions (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), and TableBench (Wu et al., 2024). Our analysis reveals that performance varies significantly across LLMs depending on the format. Moreover, certain instances are solvable only within specific formats, confirming that format suitability is instance-specific. Empirically, our results corroborate that the effectiveness of a format hinges on the alignment between its structural priors and the reasoning capabilities required by the instance. Regarding our methods, compared to the best results using a fixed format with greedy decoding and self-consistency (Wang et al., 2023), FLEXTAF-Single and FLEXTAF-Vote achieve average improvements of 2.7% and 4.0% with comparable inference costs, confirming the effectiveness. Notably, the benefits of FLEXTAF-Single and FLEXTAF-Vote are more pronounced on the more challenging WikiTableQuestions and TableBench datasets.

Our contributions are as follows:

1. We demonstrate and analyze why the most suitable tabular formats depend on the specific instance and LLM.
2. We propose FLEXTAF-Single and FLEXTAF-Vote to enhance table reasoning by leveraging flexible tabular formats.
3. Experiments validate our analysis and show that FLEXTAF-Single and FLEXTAF-Vote achieve gains of 2.7% and 4.0% over fixed-format baselines, demonstrating the effectiveness.

2 Preliminaries

We analyze why tabular formats affect the table reasoning performance of LLMs from the aspects of the instance and the model. Empirical results in §4 confirm that the optimal format varies across specific instances and models.

2.1 Different Instances Suit Different Tabular Formats

Tabular formats function as structural priors that govern the topological representation of data $\mathcal{T}$ and determine the accessibility of reasoning paths. Let $\mathcal{F}$ denote the set of serialization functions where each format f imposes specific inductive biases. For example, $f_{Markdown}$ preserves the global two-dimensional manifold, facilitating spatial-semantic reasoning, and f_{Dict} emphasizes explicit key-value mappings. Similarly, f_{List} maintains sequentiality for coordinate-based indexing, and f_{Pandas} and $f_{Database}$ optimize for programmatic aggregation and relational schema dependencies, respectively.

We posit that reasoning efficiency depends on the topological compatibility between the representation $f(\mathcal{T})$ and the target path p required by query q. An optimal format minimizes the computational complexity of traversing p (Zhao et al., 2023; Yang et al., 2025). Consequently, sequential queries favor f_{List} for its linear arrangement, whereas $f_{Database}$ treats data as unordered sets, discarding the positional information essential for such tasks, as shown in Figure 1.

2.2 Different Models Suit Different Tabular Formats

The efficacy of these paths is further constrained by the alignment between the input representation and the generative distribution of the training corpus $\mathcal{D}_{train}$ (Xie et al., 2022; Min et al., 2022; Zhou et al., 2024). Formats act as domain indicators that project inputs into latent spaces proximal to specific sub-domains (Garcia et al., 2025). $f_{Markdown}$ mirrors documentation $\mathcal{D}_{doc}$ to induce semantic reasoning, whereas f_{Dict} aligns with code repositories $\mathcal{D}_{code}$ to trigger algorithmic execution. Given that a query q often admits a set of valid reasoning paths $\mathcal{P}_q$ (Yao et al., 2023; Liu et al., 2024a), the format serves as a capability trigger that steers reasoning toward the area where the model competence $C(\mathcal{M}, p)$ is maximized (Tam et al., 2024; Liu et al., 2025; Ye et al., 2025). The optimal format

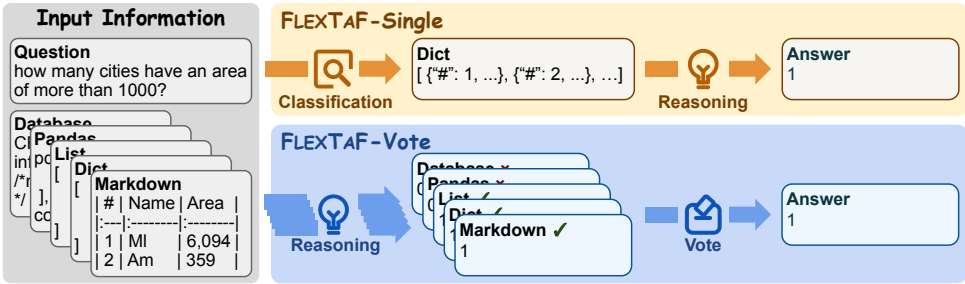

Figure 2: The overview of FLEXTAF. FLEXTAF-Single consists of two steps: (*i*) Classification: A classifier we trained predicts the most suitable tabular format based on the given instance and model. (*ii*) Reasoning: Using the predicted format, the LLM solves the instance by representing the table accordingly. 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.

f^* is defined as:

$$f^* = \arg\max_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_q} P(p|f(\mathcal{T}), \mathcal{M}) \cdot C(\mathcal{M}, p) \tag{2.1}$$

Consider a code-optimized model $\mathcal{M}_{code}$ tasked with conditional retrieval. Since $\mathcal{M}_{code}$ possesses superior competence in algorithmic execution rather than semantic scanning, f_{Dict} is preferable to $f_{Markdown}$ as it maximizes the probability of activating the proficient algorithmic path p_{algo} instead of the weaker spatial path.

3 Methodology

We introduce two methods to optimize the alignment between tabular formats and instance requirements with model capabilities. FLEXTAF-Single identifies the optimal structural prior via a trained discriminator and FLEXTAF-Vote aggregates reasoning paths across multiple formats. Figure 2 illustrates the overall framework.

3.1 Task Definition

The table reasoning task requires deriving a scalar or textual answer $\hat{a}$ given a table $\mathcal{T}$ and a natural language query q. Let $\mathcal{M}$ denote the LLM and $f \in \mathcal{F}$ represent a serialization function that maps the tabular data into a sequence of tokens. The inference process is formalized as $\hat{a} = \mathcal{M}(f(\mathcal{T}), q)$. Our objective is to determine the format f or the aggregation strategy that maximizes the probability of generating the correct answer a^*.

3.2 FLEXTAF-Single

As detailed in §2, the optimal format f^* varies across instances and models due to distinct inductive biases. FLEXTAF-Single approximates the selection of f^* by training a discriminator to predict the format that minimizes the reasoning complexity for the specific instance and model.

3.2.1 Classification

The classification module predicts the most compatible format $\hat{f}$ from the candidate set $\mathcal{F}$ based on the input instance $\mathcal{I} = (\mathcal{T}, q)$. We formulate this as a mapping $\hat{f} = \text{CLS}_{\mathcal{M}}(\mathcal{I})$. A predicted format is valid if it belongs to the set of oracle formats $\mathcal{F}^*_{\mathcal{M},\mathcal{I}} = \{f \in \mathcal{F} \mid \mathcal{M}(f(\mathcal{T}), q) = a^*\}$ that yield the correct answer.

Label Construction We construct a model-specific supervision signal by evaluating the performance of $\mathcal{M}$ across all formats in $\mathcal{F}$. For each training instance we execute the reasoning task using every format and verify the correctness of the generated answers. The resulting set $\mathcal{F}^*_{\mathcal{M},\mathcal{I}}$ serves as the ground truth for the discriminator. We enforce a filtering protocol to discard instances where either all formats fail or more than half succeed as such samples provide negligible discriminative signal regarding topological compatibility.

Learning Objective Given that multiple formats may activate valid reasoning paths we frame this as a multi-label classification problem. We employ a binary relevance strategy where each format corresponds to an independent binary target. The training objective minimizes the normalized Binary Cross-Entropy loss defined as

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|\mathcal{F}|} [y_{ij} \log(\hat{y}_{ij}) + (1 - y_{ij}) \log(1 - \hat{y}_{ij})] \tag{3.1}$$

N is the batch size and $|\mathcal{F}|$ denotes the cardinality of the candidate format set. The binary scalar $y_{ij} \in$

$\{0, 1\}$ indicates whether the j-th format is valid for the i-th instance and $\hat{y}_{ij}$ is the predicted probability.

Format Selection During inference the discriminator outputs a probability distribution over $\mathcal{F}$. We select the single format $\hat{f}$ associated with the maximum likelihood to guide the generation process. This strategy ensures computational efficiency by restricting the expensive reasoning stage to a single optimal representation.

3.2.2 Reasoning

Upon determining the optimal format $\hat{f}$ we serialize the table $\mathcal{T}$ using $\hat{f}$ and prompt the LLM to generate the answer. This process effectively steers the model toward the reasoning path with the highest expected competence.

3.3 FLEXTAF-Vote

FLEXTAF-Vote enhances performance by aggregating reasoning paths from diverse structural priors without requiring additional training.

3.3.1 Reasoning

We instantiate the input table $\mathcal{T}$ using every format $f \in \mathcal{F}$ to construct a set of parallel prompts. The LLM processes these prompts independently to produce a candidate answer set $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{F}|}\}$.

3.3.2 Vote

We apply a majority voting mechanism to identify the most consistent answer across varying formats. The final prediction $\hat{a}$ is determined by

$$\hat{a} = \arg\max_{a \in \mathcal{V}} \sum_{i=1}^{|\mathcal{F}|} \mathbb{1}(a_i = a) \tag{3.2}$$

where $\mathcal{V}$ is the vocabulary of generated answers and $\mathbb{1}(\cdot)$ is the indicator function. In cases of uniform distribution we resolve ties by selecting the answer with the highest cumulative log-probability, following the previous work (Luo et al., 2024).

3.4 Comparison

To better employ our two methods, we delineate the distinct application scenarios for the two methods. FLEXTAF-Single prioritizes online inference latency and is suitable for resource-constrained environments that permit offline discriminator training. Conversely FLEXTAF-Vote circumvents the need for annotated supervision and serves scenarios where computational overhead is subordinate.

4 Experiments

4.1 Settings

Datasets. We use WikiTableQuestions (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), TableBench (Wu et al., 2024) to evaluate FLEXTAF, following previous works (Liu et al., 2024a; Wang et al., 2025b; Zhang et al., 2025b). We introduce these datasets in detail in Appendix C.1. We also show the results on **HiTab** (Cheng et al., 2022) and **DataBench** (Osés Grijalba et al., 2024) in Appendix E, where our methods also get consistent improvements compared with using fixed formats.

Metric. (*i*) For reasoning, we employ accuracy as the evaluation metric for WikiTableQuestions and TabFact and use Rouge-L (Lin, 2004) for TableBench, following previous works (Pasupat and Liang, 2015; Chen et al., 2020; Wu et al., 2024). We introduce the metrics in Appendix C.2. (*ii*) For Classification, we use accuracy to evaluate the discriminator, which is employed to assess whether the top format predicted is in suitable formats.

Models. (*i*) For Reasoning, we employ Llama3 (Meta, 2024), DeepSeek-Coder (Guo et al., 2024), DeepSeek-R1-Distill-Llama-8B (R1-Llama-8B) (Guo et al., 2025), and `gpt-4o` (OpenAI, 2024). Llama3 and DeepSeek-R1 are popular open-source LLMs and Reasoning LLMs, DeepSeek-Coder are prominent code LLMs, and `gpt-4o` is mainstream closed-source models. (*ii*) For Classification, we use ELECTRA-Large (Clark et al., 2020), with training details in Appendix D.

Baselines. We compare two types of baselines: (*i*) Tabular formats: `Markdown`, `Dict`, `List`, `Pandas`, and `Database` (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. (*ii*) Previous mainstream methods: Binder (Cheng et al., 2023), Dater (Ye et al., 2023), Chain-of-Table (Wang et al., 2024), TabSQLify (Nahid and Rafiei, 2024), and PoTable (Mao et al., 2025).

Implementation Details. (*i*) For prompts, we utilize Chain-of-Thought (CoT) prompt (Wei et al., 2022) for `Markdown` and Program-of-Thought (PoT) prompt (Chen et al., 2023; Gao et al., 2023) for other formats to enhance the table reasoning performance. Since the questions in WikiTableQuestions and TableBench are more challenging,

Format	WikiTQ						TabFact						TBench
	Llama3		Deep.C		gpt-4o†	R1-Llama	Llama3		Deep.C		gpt-4o†	R1-Llama	Llama3
	8B	70B	6.7B	33B	-	8B	8B	70B	6.7B	33B	-	8B	8B
Markdown	47.7	63.3	32.2	31.2	71.9	62.7	75.2	86.4	60.4	63.6	77.3	84.6	23.9
Dict	43.0	56.4	25.6	53.6	68.0	42.2	65.5	80.0	63.9	78.0	75.0	41.9	21.6
List	30.5	56.3	19.2	50.8	57.8	23.1	57.4	77.5	63.7	75.4	75.0	25.9	6.9
Pandas	39.2	52.3	39.7	48.8	55.5	15.1	47.8	73.6	62.5	75.9	82.0	11.9	6.1
Database	31.0	48.2	41.8	45.5	51.6	37.4	65.0	75.0	70.7	76.3	75.8	60.9	22.0
Ours (Single)	**50.5**	**69.1**	**46.3**	**54.5**	-	**67.9**	**77.0**	**87.1**	**70.9**	**78.3**	-	**87.9**	**26.0**
Δ	+2.8	+5.8	+4.5	+0.9	-	+5.3	+1.8	+0.7	+0.2	+2.0	-	+3.3	+2.1

Table 1: The performance of reasoning using a fixed tabular format with **greedy decoding** and FLEXTAF-Single. WikiTQ denotes WikiTableQuestions, Deep.C. denotes DeepSeek-Coder, and TBench denotes TableBench. 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.

we apply 4-shot prompts on them, and employ 2-shot prompts on TabFact. 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 F. (*ii*) For FLEXTAF-Single, we train the discriminator for each LLM on each dataset separately, detailed in Appendix D.1. Considering the **low labeling scenario**, we also show the experiments of training one discriminator on one dataset and using it for other datasets in Appendix D.3.

4.2 Main Experiment

4.2.1 Comparison with Fixed Tabular Formats

To illustrate the advantages of our methods over using a fixed table format, we first compare FLEXTAF-Single and FLEXTAF-Vote with the fixed-format methods. 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 2.7% and 4.0% 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., 2024; Renze and Guven, 2024), and diminished instruction following ability at higher temperatures (Zeng et al., 2024; Peeperkorn et al., 2024). Additionally, the results reveal that:

Different models exhibit significant performance variations across tabular formats. To statistically validate that different models suit different tabular formats, we conduct a chi-squared test. The extremely low p-value ($p \approx 4.7 \times 10^{-219}, p \ll 0.05$) indicates a highly statistically significant difference in the distribution of the most suitable tabular formats across models, which is further discussed in Appendix B.

The improvement of FLEXTAF on the more challenging dataset is more significant. The performance on WikiTQ and TableBench is better compared to the simpler TabFact. Despite higher classification accuracy on TabFact (Table 23), 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 E.4).

FLEXTAF-Single shows superior performance on the general model compared to the code model. Code models demonstrate superior performance in code understanding and generation (Guo et al., 2024). This predisposes them to favor tabular formats that are themselves code-related. However, the minimal variance among these code-related formats, particularly when contrasted with the distinct structure of Markdown, leads to a degradation in discriminator performance and the smaller performance enhancements of FLEXTAF-Single on such models during inference.

FLEXTAF-Vote performs better than FLEXTAF-Single consistently. Although the performance of FLEXTAF-Single is constrained by classification accuracy (see Table 23), it demonstrates high reasoning efficiency. In contrast, FLEXTAF-Vote

Format	WikiTQ						TabFact						TableB
	Llama3		Deep.C		gpt-4o†	R1-Llama	Llama3		Deep.C		gpt-4o†	R1-Llama	Llama3
	8B	70B	6.7B	33B	-	8B	8B	70B	6.7B	33B	-	8B	8B
Markdown	49.1	62.8	32.1	29.7	72.7	63.7	75.2	86.7	60.9	63.6	78.9	86.6	26.5
Dict	44.8	58.1	28.7	59.7	71.0	45.0	67.9	80.9	71.9	82.4	91.4	50.0	26.1
List	35.4	59.7	27.5	55.6	61.7	23.5	59.8	78.0	66.9	78.0	89.8	29.9	17.3
Pandas	41.4	56.2	43.9	54.4	60.9	15.2	56.0	74.7	67.2	79.2	84.4	13.3	10.4
Database	35.2	48.6	42.5	46.4	54.7	44.1	69.5	76.0	70.7	77.2	83.6	68.7	22.7
Ours (Vote)	**55.7**	**69.9**	**51.4**	**60.9**	**76.6**	**65.2**	**80.3**	**88.5**	**77.9**	**84.4**	**93.8**	**88.1**	**31.2**
Δ	+6.6	+7.1	+7.5	+1.2	+3.9	+1.5	+5.1	+1.8	+6.0	+2.0	+2.4	+1.5	+4.7

Table 2: The performance of reasoning using a fixed tabular format with self-consistency decoding and FLEXTAF-Vote. † represents the result on sampled 128 instances. Δ denotes the improvement of FLEXTAF-Vote relative to the best performance of using a fixed format. The best performance under each setting is marked in **bold**.

Method	# Generation	WikiTQ	TabFact
Binder	50	50.5	78.2
Dater	100	42.8	81.6
Chain-of-Table	≤ 22	62.2	85.6
TabSQLify	2	55.8	70.7
PoTable	≤ 10	65.6	82.7
FLEXTAF-Single	1	69.1	87.1
FLEXTAF-Vote	5	**69.9**	**88.5**

Table 3: Comparison of performance and efficiency between our methods and mainstream methods on WikiTQ and TabFact. The best performance is shown in **bold**, and the second-best is underlined. # Generation denotes the average number of samples generated by LLM for a single question on TabFact.

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

Table 4: 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.

achieves superior performance because instances could be resolved correctly by multiple formats, and errors produced by different formats exhibit diversity (Wang et al., 2025a).

4.2.2 Comparison with Previous Methods

To further demonstrate the superiority of our methods, we compare them with mainstream prior methods in Table 3. The performance and efficiency of previous works are reported by Mao et al. (2025) with Llama3.1-70B, while our methods are implemented with Llama3-70B. Our methods consistently and significantly outperform the baselines.

These methods introduce complex designs: Binder (Cheng et al., 2023), TabSQLify (Nahid and Rafiei, 2024), and PoTable (Mao et al., 2025) combine code and natural language to process tables; Dater (Ye et al., 2023) decomposes questions and tables; Chain-of-Table (Wang et al., 2024) decouples the reasoning process and introduces table-processing tools. However, regardless of instances or models, they adopt fixed formats (Markdown or Pandas). In contrast, our methods improve performance solely by modifying the input tabular format, with higher efficiency achieving the average gains of 3.9% and 5.1% on WikiTQ and TabFact compared with the highest baseline. This highlights the necessity of adapting formats flexibly.

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 (Shi et al., 2020; Dong et al., 2022). To illustrate that different instances are suitable for different formats, we present case study in Appendix G. We also analyze the behavior of the discriminator in Appendix E.3.

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 4. Specifically, we adopt the same demonstrations with different formats, manually annotating the rationale and programs. Detailed prompts are shown in Appendix F. Table 4 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,

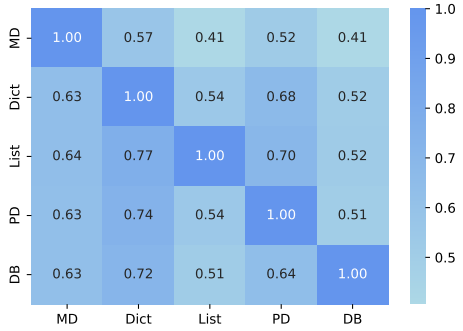

Figure 3: The overlap of instances solved between different tabular formats achieved by Llama3-8B on WikiTQ. The values represent the proportion of instances that can be solved by the tabular format corresponding to the vertical axis, within the instances solvable by the format on the horizontal axis.

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 5: 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.

as shown in Figure 3 and Table 5. 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.

Table 5 presents that: (*i*) Certain instances can only be accurately solved using a specific format, highlighting the differences between formats. (*ii*) `MD` and `DB` formats resolve a greater proportion of instances due to their distinct structures compared to the programming formats. Figure 3 and Table 5 claim that instances suit different formats.

4.3.3 What instances are suitable for different formats?

To empirically validate our analysis regarding the topological compatibility between tabular formats and instance requirements in §2, we sample 50 instances from WikiTQ for manual inspection. We analyze the capabilities required to resolve the queries and the structural characteristics of the tables. Figure 4 visualizes the performance variance across different formats, confirming our hypothesis that the efficiency of reasoning paths is highly dependent on the alignment between the format's

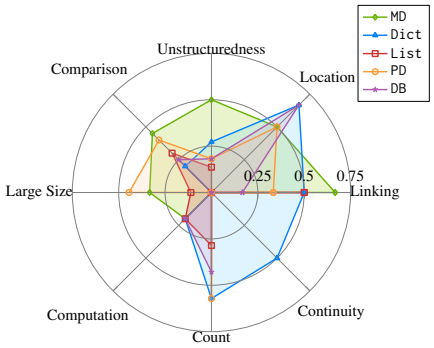

Figure 4: The performance of Llama3.1-8B across tabular formats on instances with different characteristics.

inductive biases and the instance's characteristics.

To inspire the real application, we detail four representative characteristics below, with additional analyses provided in Appendix G.2. (*i*) **Unstructuredness** involves tables with complex or mixed expressions. The `MD` format performs best here because it preserves the natural 2D layout of the table. Unlike code-based formats that add extra symbols, `MD` allows the model to read the content directly. This makes the reasoning process faster and less complex. (*ii*) **Location** tasks require finding specific information coordinates in a table. The `DB` and `Dict` formats excel at this by using clear key-value mappings. These formats create direct paths between headers and cell values. This structure reduces the search space and is more efficient than formats relying on visual layout. (*iii*) **Linking** maps entities from natural language queries to table headers or cells. The `MD` format works best for this task as it aligns with general text documentation. This format triggers semantic reasoning paths effectively. It utilizes the model's language understanding better than code-based formats. (*iv*) **Continuity** involves analyzing logical sequences such as consecutive rows. The `Dict` format achieved the best results as its structure allows the model to perform algorithmic steps to retrieve sequential items accurately. This confirms that the best format must support the specific retrieval needs of the query.

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

We perform experiments using varying numbers of candidate formats and present results in Figure 5, 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

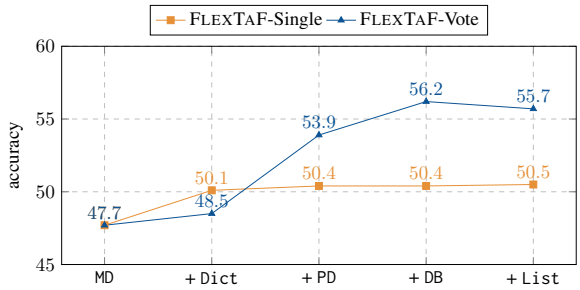

Figure 5: The accuracy of FLEXTAF-Single and FLEXTAF-Vote using Llama3-8B on WikiTQ with different numbers of candidate tabular formats, as candidate tabular formats are added from left to right.

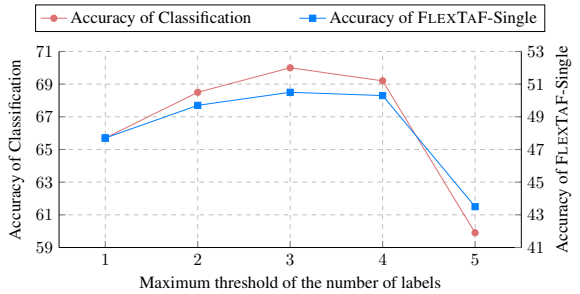

Figure 6: The accuracy of classification and FLEXTAF-Single, with the change of the maximum threshold of the number of labels in the training data. We discuss how to improve classification accuracy further in Appendix H.

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.

4.3.5 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 6. We find that: (*i*) As the maximum label count rises, overall performance improves, peaking at 3 labels 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 characteristics of each label, thus negatively impacting training.

5 Related Works

The table reasoning task aims to answer the natural language question based on the given table (Dong et al., 2022; Dong and Wang, 2024). LLM-based methods have become predominant due to their superior performance of table reasoning (Chen, 2023; Zhang et al., 2025a). Previous works generally employ a fixed tabular format, regardless of the varied LLMs and instances, which could limit the performance. Since Sui et al. (2024) demonstrate that the performance of table understanding tasks 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 of the table reasoning task.

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-Single and FLEXTAF-Vote to improve the table reasoning performance by flexibly employing tabular formats during inference.

6 Conclusion

In this paper, we explore the impact of tabular formats on table reasoning performance from two aspects. (*i*) We discuss why different instances and models have most suitable formats. (*ii*) Accordingly, We propose FLEXTAF-Single and FLEXTAF-Vote. FLEXTAF-Single uses the discriminator to predict the most suitable 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. Experimental results validate our analysis and demonstrate that FLEXTAF-Single and FLEXTAF-Vote increase by 2.7% and 4.0% on average compared with the best performance of using a fixed format with greedy decoding and self-consistency, 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. We employ AI tools for the writing polishing.

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.

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

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. 2024. 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*.

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

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.

Mirian Del Carmen Hipolito Garcia, Camille Couturier, Daniel Madrigal, Ankur Mallick, Robert Sim, Anastasios Kyrillidis, Victor Rühle, and Saravan Rajmohan. 2025. Exploring how LLMs capture and represent domain-specific knowledge.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

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*.

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.

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*.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2025. Regmix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*.

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.

Qingyang Mao, Qi Liu, Zhi Li, Mingyue Cheng, Zheng Zhang, and Rui Li. 2025. Potable: Programming standardly on table-based reasoning like a human analyst.

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

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vishvak Murahari, Ameet Deshpande, Peter Clark, Tanmay Rajpurohit, Ashish Sabharwal, Karthik Narasimhan, and Ashwin Kalyan. 2024. QualEval: Qualitative evaluation for model improvement. 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 2093–2111, Mexico City, Mexico. Association for Computational Linguistics.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. 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. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13471–13488, Torino, Italia. ELRA and ICCL.

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.

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.

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.

Peng Shi, Patrick Ng, Feng Nan, Henghui Zhu, Jun Wang, Jiarong Jiang, Alexander Hanbo Li, Rishav Chakravarti, Donald Weidner, Bing Xiang, and Zhiguo Wang. 2022. Generation-focused table-based intermediate pre-training for free-form question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11312–11320.

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.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236, Miami, Florida, US. Association for Computational Linguistics.

Yufei Tian, Jiao Sun, Nanyun Peng, and Zizhao Zhang. 2025. Skillverse : Assessing and enhancing llms with tree evaluation. *Preprint*, arXiv:2506.00319.

Dingzirui Wang, Xuanliang Zhang, Rongyu Cao, Longxu Dou, Xianzhen Luo, Yingwei Ma, Qingfu Zhu, Wanxiang Che, Binhua Li, Fei Huang, and Yongbin Li. 2025a. Format-adapter: Improving reasoning capability of llms by adapting suitable format. *Preprint*, arXiv:2506.23133.

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. 2025b. 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.

Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. Table retrieval may not necessitate table-specific model design. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 36–46, Seattle, USA. 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.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Zhen Yang, Ziwei Du, Minghan Zhang, Wei Du, Jie Chen, Zhen Duan, and Shu Zhao. 2025. Triples as the key: Structuring makes decomposition and verification easier in LLM-based tableQA. In *The Thirteenth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. 2025. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *The Thirteenth International Conference on Learning Representations*.

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*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025a. A survey of table reasoning with large language models. *Frontiers of Computer Science*, 19(9):199348.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. 2025b. Rot: Enhancing table reasoning with iterative row-wise traversals. *Preprint*, arXiv:2505.15110.

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*.

Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024. The mystery of in-context learning: A comprehensive survey on interpretation and analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14365–14378, Miami, Florida, USA. 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 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 B.1.

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

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.

C Datasets and Metrics

C.1 Experimental Datasets

WikiTableQuestions WikiTableQuestions is a prominent and challenging dataset designed for complex question answering over semi-structured tables. Sourced from Wikipedia, it covers a wide variety of open-domain topics, ensuring high diversity in both data schema and content. Unlike simple retrieval tasks, this dataset requires models to perform multi-step reasoning—such as aggregation, comparison, and arithmetic operations—to derive the correct answer. It serves as a standard benchmark for evaluating the ability of semantic parsers and Large Language Models to understand natural language queries and execute compositional logic against tabular data.

TabFact TabFact is a large-scale dataset specifically constructed for the task of table-based fact verification. It requires models to determine whether a given natural language hypothesis is entailed or refuted by the information presented in a table. The dataset is notable for incorporating both linguistic reasoning and symbolic reasoning, requiring systems to handle complex operations like counting, superlatives, and aggregation. By moving beyond simple semantic matching, TabFact challenges models to perform rigorous logical inference to verify the truthfulness of statements grounded in semi-structured evidence.

TableBench TableBench presents a comprehensive and robust benchmark designed to evaluate the table reasoning capabilities of Large Language Models (LLMs) across a broad spectrum of tasks. Recognizing that previous datasets often focus on narrow skills, TableBench covers diverse question types and varying levels of complexity to better mimic realistic user needs. It tests not only basic data retrieval but also advanced instruction following and complex reasoning. This benchmark provides a holistic framework for assessing how well models generalize to unseen table structures and diverse reasoning scenarios.

C.2 Evaluation Metrics

For WikiTableQuestions and TabFact, we use accuracy as metrics. Accuracy measures the ability to generate the gold answer, which is achieved only when the predicted exactly matches the gold answer. For TableBench, we use Rouge-L, which compares generated answers with gold answer based on the longest common subsequence, considering precision and recall.

D Details of the Discriminator in FLEXTAF-Single

D.1 Training the Discriminator

In this subsection, we introduce the details of training the discriminator 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 21, and we summarize the training data size employed for training the classifier to predict each LLM across various datasets in Table 22. We count the proportion of each tabular format in the training data for different LLMs and 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.

D.2 Prompt Construction for the Discriminator

The discriminator input comprises the question and table, structured as follows:

- The question is prefixed with a <Q> tag.
- The table is prefixed with <T>, with each row marked by <R> and columns separated by |, following established Pre-trained Language Model (PLM) practices for tabular tasks (Liu et al., 2022; Shi et al., 2022).

We deliberately employ these specialized tokens for the PLM-based classifier, rather than the standard formats used for LLMs, as prior research suggests that PLMs achieve better performance with explicit structural identifiers (Wang et al., 2022). Conversely, since the LLMs are used without fine-tuning, we utilize formats more commonly encountered during their pre-training.

D.3 Discriminator on Out-of-Distribution Datasets

Notably, we employ the respective training sets for each dataset to facilitate future work and mitigate data leakage. To access the generalization of FLEXTAF-Single, we also evaluate the performance using the cross dataset. Specifically, FLEXTAF-Single adopts the classifier, which is trained on WikiTQ training data, and is applied on TabFact, getting the accuracy of **76.0** (outperforming the best format performance of 75.2, but lower than the classifier trained on TabFact, whose performance is 77.0).

Method	Markdown	Dict	List
Greedy decoding	37.1	15.5	27.1
FLEXTAF-Single		**40.4**	
Self-consistency	41.0	15.7	29.3
FLEXTAF-Vote		**44.4**	

Table 6: The accuracy of using a fixed tabular format and FLEXTAF, using Llama3-8B on HiTab.

E Additional Experiments

E.1 Experiments on HiTab

HiTab (Cheng et al., 2022) is a table-based question answering dataset that focuses on hierarchical tables, challenging models to comprehend complex structures. We show the performance of FLEXTAF-Single and FLEXTAF-Vote using Llama3-8B-Instruct (Llama3-8B) on HiTab in Table 7. We adopt accuracy as the evaluation metric following HiTab. Since the hierarchical tables in HiTab are incompatible with the formats used by `Pandas` and `Database`, we use three alternative tabular formats. We can observe that FLEXTAF outperforms the best performance of the single format.

E.2 Experiments on DataBench

DataBench (Osés Grijalba et al., 2024) is a comprehensive benchmark designed to evaluate LLMs as tabular reasoners, comprising 65 real-world datasets across diverse domains and 1,300 human-generated questions. We show the performance of FLEXTAF-Single and FLEXTAF-Vote using Llama3-8B-Instruct (Llama3-8B) on DataBench in Table 7. We adopt accuracy as the evaluation metric following DataBench. FLEXTAF-Single and FLEXTAF-Vote consistently and significantly outperform the best fixed-format baseline, with average improvement of 11.7 points. This validates our analysis that the performance gain of our methods is more pronounced on challenging datasets. These results further validate the effectiveness of FLEXTAF and highlight the importance of adaptively selecting formats based on the specific instance and model.

E.3 Behavior of the Discriminator

Table 8 shows the distribution of predicted labels on the evaluation set. We observe that: The most frequently predicted format aligns with the best-performing fixed format for that specific setting. The partial ordering of the predicted distribution

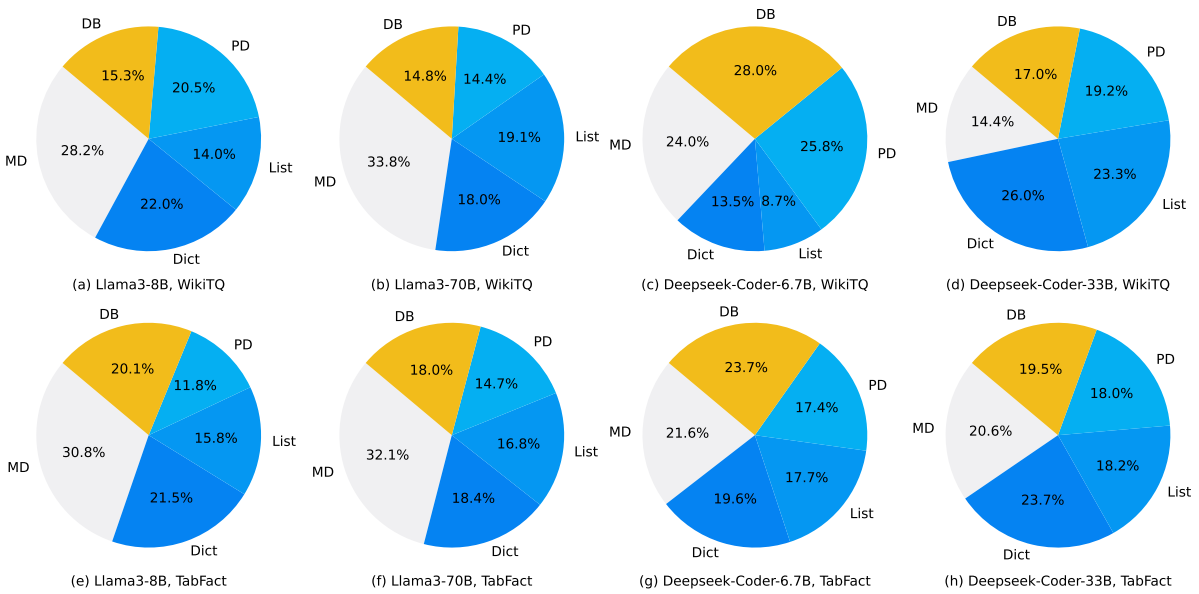

Figure 7: The proportion of each tabular format in the training data.

Method	MD	Dict	List	PD	DB
Greedy decoding	33.0	33.0	30.2	32.4	30.2
FLEXTAF-Single			**43.0**		
Self-consistency	36.3	33.5	33.5	35.2	36.3
FLEXTAF-Vote			**49.7**		

Table 7: The accuracy of using a fixed tabular format and FLEXTAF, using Llama3.1-8B on DataBench.

matches the performance ranking of different formats. This suggests the discriminator successfully learns model-specific format preferences. Furthermore, our manual analysis in §4.3.3 reveals that 80% of the discriminator's predictions match the best-performing format for the corresponding instance category. This indicates the discriminator captures instance-level features. However, the best-performing format for a category does not universally solve every instance within it. Moreover, when an instance exhibits features from multiple categories, the discriminator's performance drops, indicating limited proficiency in balancing conflicting features.

E.4 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.

E.5 Comparison FLEXTAF with Oracle

In this section, we compare the performance of FLEXTAF with that of Oracle, as shown in Table 9. 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

	WikiTQ					TabFact				
	Llama3		Deep.C		R1-Llama	Llama3		Deep.C		R1-Llama
	8B	70B	6.7B	33B	8B	8B	70B	6.7B	33B	8B
Markdown	49.8	51.1	28.8	17.1	41.7	49.6	49.6	17.1	21.2	40.6
Dict	22.7	13.5	13.8	38.0	27.2	21.1	16.2	23.5	31.9	19.9
List	7.6	14.2	4.5	12.8	9.6	9.7	8.4	15.5	10.9	12.6
Pandas	7.8	12.1	17.2	15.5	5.5	8.2	7.6	16.2	13.7	10.1
Database	12.1	9.0	35.7	14.6	16.1	11.4	18.1	27.7	22.3	16.9

Table 8: The distribution of the format predicted by the discriminator in FLEXTAF-Single.

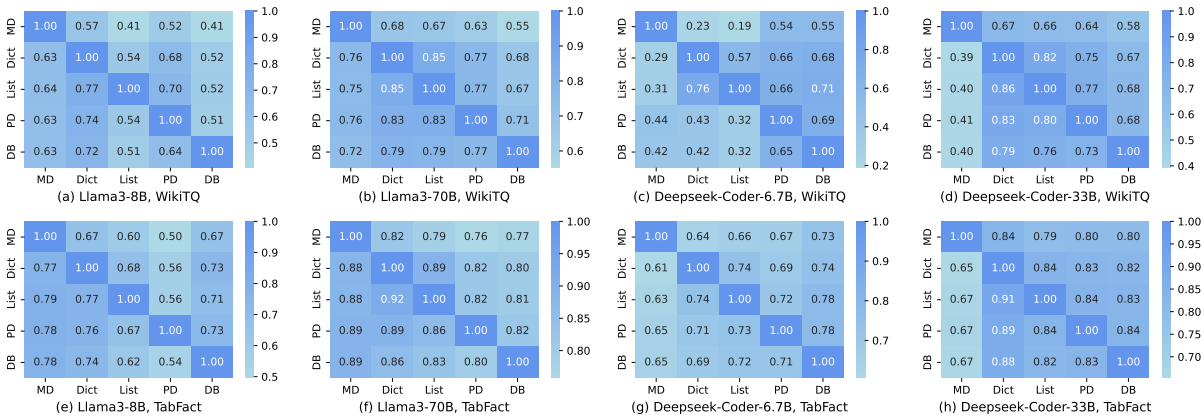

Figure 8: The overlap among instances that are correctly solved with each tabular format using four LLMs on two datasets. The value represents the proportion that can be solved by the tabular format corresponding to the vertical axis in the instances that are solved by the format corresponding to the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

use of the rich semantic information in the LLM solutions, causing the limited improvement (Ni et al., 2023).

F Prompts with Each Tabular Format

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

F.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 10, Table 11, Table 12, Table 13, and Table 14. And the prompts for TabFact (Chen et al., 2020) in main experiments with a single tabular format are shown in Table 15, Table 16, Table 17, Table 18, and Table 19. 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.

F.2 Prompts for Experiments of Unifying Demonstrations

We present the prompt with the unified demonstrations in this subsection. From Table 10, Table 11, Table 12 and Table 13, 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 20.

G Cases Study

G.1 Case Study of Different Instances and Models

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

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.

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 9: 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.

different instances and LLMs have different most suitable tabular formats.

G.2 Case Study for Different Types of Instances

We first introduce the other characteristics in Figure 4. (*i*) **Count** requires the model to count entities that satisfy a specific condition to find the answer. The `Pandas` format is highly suitable for counting problems because relevant functions can be directly invoked. (*ii*) **Computation** requires the model to perform mathematical calculations. (*iii*) **Large Size** refers to tables with large dimensions, specifically those containing over 200 cells. (*iv*) **Comparison** requires the model to compare numerical data to arrive at the answer.

We show the cases with different characteristics in Figure 10, 11, 12, 13, 14, 15.

H How to Improve Classification Performance Further?

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 characteristics of each format. The classification results are presented in Table 23. We observe that: (*i*) The classification performance of predicting larger-scale LLMs is better than that of smaller-scale LLMs. Larger-scale LLMs have greater robustness (Howe et al., 2024), resulting in more consistent characteristics 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 D.1). Therefore, future improvements in classification performance can be achieved by reducing noise data and increasing the scale of the training data.

Case of Unstructuredness

Question which building held the record for tallest freestanding structure before the burj khalifa?

Table

Name	Country	Town	Height metres / ft	Structural type	Held record	Notes
Great Pyramid of Giza	Egypt	Giza	146 / 480	Mausoleum	2570 BC–1311	Due to erosion today it stands at the height of 138.8 metres (455 ft).
Lincoln Cathedral	England	Lincoln	159.7 / 524	Church	1311–1549	Spire collapsed in 1549; today, stands at a height of 83 metres (272 ft).
				…		
CN Tower	Canada	Toronto	553 / 1,815	Tower	1976–2007	-
Burj Khalifa	United Arab Emirates	Dubai	829.8 / 2,722	Skyscraper	2007–present	Topped-out on 17 January 2009

Answer CN Tower

Figure 10: The case for the type of Unstructuredness, with the feature is highlighted in blue.

Case of Linking

Question how many matches had a tie for the progressive total?

Table

Match no.	Match Type	Team Europe	Score	Team USA	Progressive Total
25	Singles	Mika Koivuniemi	217 - 279	Chris Barnes	14 - 11
26	Singles	Osku Palermaa	217 - 244	Tommy Jones	14 - 12
			…		
30	Baker	Team Europe	202 - 203	Team USA	15 - 15
32	Singles	Osku Palermaa	196 - 235	Tommy Jones	15 - 17

Answer 1

Figure 11: The case for the type of Linking, with the feature is highlighted in blue.

Case of Continuity

Question how many consecutive seasons did the kansas city brass not qualify for the playoffs?

Table

Year	Division	League	Regular Season	Playoffs	Open Cup
1998	4	USISL PDSL	14th, Central	Division Finals	1st Round
1999	4	USL PDL	5th, Heartland	Did not qualify	Did not qualify
			…		
2012	4	USL PDL	5th, Heartland	Did not qualify	Did not qualify
2013	4	USL PDL	4th, Heartland	Did not qualify	Did not qualify

Answer 15

Figure 12: The case for the type of Continuity, with the feature is highlighted in blue.

Case of Count

Question what are the total number of times all appears under the rounds column?

Table

Team	No	Driver	Class	Rounds
Josef Kaufmann Racing	4	Robin Frijns	-	All
Josef Kaufmann Racing	5	Hannes van Asseldonk	R	All
		…		
Fortec Motorsport	25	George Katsinis	-	All
Fortec Motorsport	26	Christof von Grünigen	-	All

Answer 15

Figure 13: The case for the type of Count, with the feature is highlighted in blue.

Case of Computation

Question what is the total amount of lives lost among the argus, james carruthers, and the hydrus

Table

Ship	Type of Vessel	Lake	Location	Lives lost
Argus	Steamer	Lake Huron	25 miles off Kincardine, Ontario	25 lost
James Carruthers	Steamer	Lake Huron	near Kincardine	18 lost
		…		
Wexford	Steamer	Lake Huron	north of Grand Bend, Ontario	all hands
Lightship No. 82	Lightship	Lake Erie	Point Albino (near Buffalo)	6 lost

Answer 71

Figure 14: The case for the type of Computation, with the feature is highlighted in blue.

Case of Comparison

Question what was the first year the vmax reached 100 km/h?

Table

Type	Construction period	Cylinder	Capacity	Power	Vmax
10 PS (7 kW; 10 hp)	1901–1902	straight-2	1.527 cc	18 PS (13,2 kW)	50 km/h (31 mph)
8/14 PS	1902–1905	straight-2	1.527 cc	14 PS (10,3 kW)	50 km/h (31 mph)
			…		
Sedina	1937–1940	straight-4	2.406 cc	55 PS (40 kW)	110 km/h (68 mph)
Arkona	1937–1940	straight-6	3.610 cc	80 PS (59 kW)	120 km/h (75 mph)–140 km/h (87 mph)

Answer 1919-1921

Figure 15: The case for the type of Comparison, with the feature is highlighted in blue.

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:

$Aircraft$	$Description$	$MaxGrossWeight$	$Totaldiskarea$	$MaxdiskLoading$
$RobinsonR-22$	$Lightutilityhelicopter$	$1,370lb(635kg)$	$497ft^2(46.2m^2)$	$2.6lb/ft^2(14kg/m^2)$
$Bell206B3JetRanger$	$Turboshaftutilityhelicopter$	$3,200lb(1,451kg)$	$872ft^2(81.1m^2)$	$3.7lb/ft^2(18kg/m^2)$
$CH-47DChinook$	$Tandemrotorhelicopter$	$50,000lb(22,680kg)$	$5,655ft^2(526m^2)$	$8.8lb/ft^2(43kg/m^2)$
$MilMi-26$	$Heavy-lifthelicopter$	$123,500lb(56,000kg)$	$8,495ft^2(789m^2)$	$14.5lb/ft^2(71kg/m^2)$
$CH-53ESuperStallion$	$Heavy-lifthelicopter$	$73,500lb(33,300kg)$	$4,900ft^2(460m^2)$	$15lb/ft^2(72kg/m^2)$

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 10: 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 11: 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 10. 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",
        ...
    ],
    ...
]
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 12: 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 10. 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 13: 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 10. 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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 20: 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 10.

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

Table 21: 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 22: The training data size employed for training the classifier in FLEXTAF-Single.

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 23: 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**.