CRAFT: A Unified Counterfactual Reasoning Framework for Tabular Question Answering and Fact Verification

Chenshuo Pan[1], Yu Zhao[1], Jie Zhang[1], Changzai Pan[1],
Zhenhe Wu[1], Jiayi Liang[1], Yujie Mao[1],
Shuangyong Song[1], Yongxiang Li[1]*, Zhongjiang He[1]*
[1] China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd

Abstract

Table reasoning remains challenging for large language models (LLMs), particularly in tasks that require multi-step inference over long and structured tables. Existing approaches predominantly rely on single-direction reasoning, which limits their ability to explore alternative hypotheses across tasks. In this work, we propose **CRAFT**, a unified **C**ounterfactual **Rea**soning **F**ramework that reformulates **T**abular question answering and fact verification into a general bidirectional verification process. Our method explicitly constructs both declarative statements and their counterfactual variants. Evidence is then extracted from reasoning along both the original and counterfactual paths, and integrated via a weighted mechanism to arrive at the final answer. Experimental results show that our approach consistently surpasses representative baselines on table reasoning datasets such as WikiTQ and TabFact, achieving especially large improvements on complex question answering. Our framework also significantly mitigates performance gaps between different backbone LLMs. This indicates that counterfactual reasoning effectively overcomes the limitations of single-direction inference, guiding LLMs toward more discerning reasoning and establishing a more principled paradigm for structured reasoning tasks. Our code is available at https://github.com/cp2ooo/CRAFT.

1 Introduction

Tables are a representative form of structured data and are widely found in financial reports, statistical yearbooks, and various domain-specific knowledge bases (Liu et al., 2023; Chen et al., 2021b). Compared to natural language text, tables organize information explicitly through row–column structures, which poses challenges for data localization, symbolic computation, and entity linking(Fang et al., 2024; Lu et al., 2025). Understanding and reasoning over tables has become an important research topic in natural language processing, with table question answering (QA) and table fact verification (FV) as two representative tasks. To reason over tabular information, Large Language Models(LLMs) must interpret not only natural language questions but also the structural relations across rows and columns(Liu et al., 2023; Ruan et al., 2024). This requires the ability to perform operations such as cross-cell comparison and numerical reasoning to produce reliable inference results (Badaro et al., 2023).

Recent work addresses tabular reasoning challenges through two main strategies. One line of research improves table understanding at the parameter level by incorporating table-specific inductive bias through pre-training or fine-tuning LLMs (Herzig et al., 2020; Su et al., 2024; Zhang et al., 2025b), which is typically computationally expensive. Other research promote explicit reasoning and prompt engineering for sub-table extraction (Ye et al., 2023; Sui et al., 2024; Nahid and Rafiei, 2024; Wang et al., 2024) to break down questions into verifiable statements without updating model parameters (Wei et al., 2022; Chen et al., 2023; Xiong et al., 2025).

Critique-based methods such as posterior self-critique (Yu et al., 2025) and consistency-based aggregation (Ji et al., 2024; Wang et al., 2023; Liu et al., 2024b) have recently proven effective for TableQA by detecting and correcting errors. Nevertheless, the performance gain of repeated single-direction reasoning often reduce rapidly as the number of iterations increases. This suggests that reasoning in a single direction is insufficient for robust table understanding. To effectively challenge initial premises, it is crucial to explicitly integrate counterfactual analysis into the reasoning process, which received little attention in previous studies.

To bridge this gap, we propose **CRAFT**, a novel

*Corresponding authors.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 3934–3959
November 6-10, 2026 ©2026 Association for Computational Linguistics

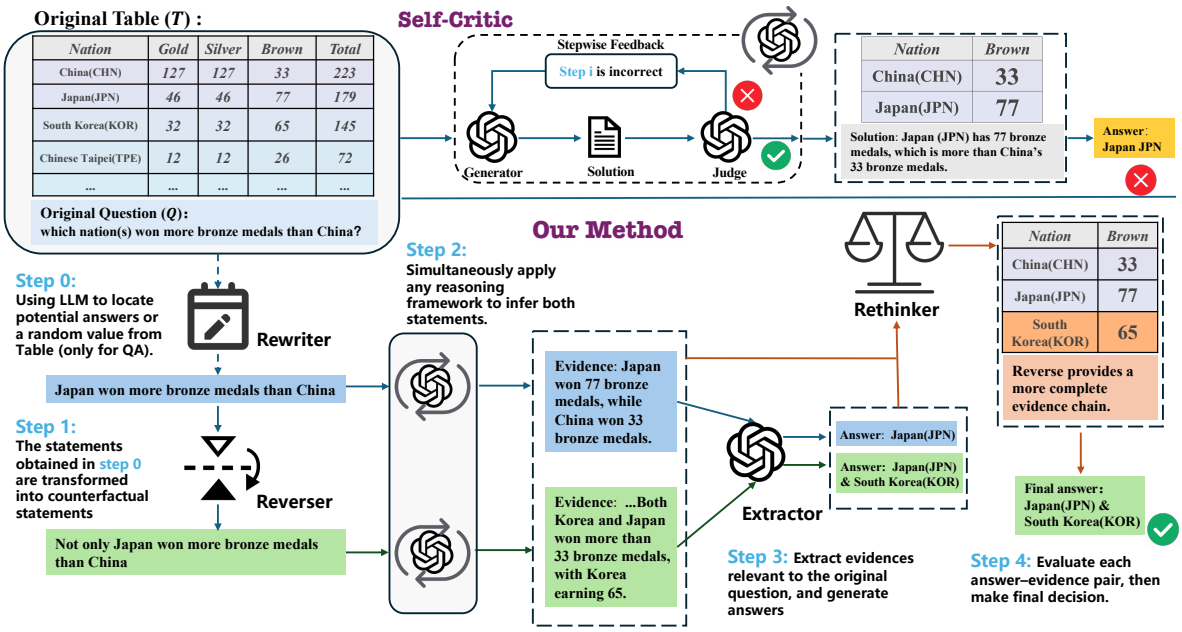


Figure 1: Overview of CRAFT, a multi-agent framework for table reasoning, Rewriter forms an initial statement from question, Reverser derives a counterfactual-inspired reverse statement to create a complementary reasoning path, Extractor gathers evidence and predictions from both paths, and Rethink jointly evaluates and generates the final answer.

counterfactual reasoning framework for tabular data. CRAFT jointly constructs both supporting and counterfactual evidence chains to systematically explore alternative scenarios. Under each assumption, distinct reasoning chains are executed by LLMs, with the resulting evidence then aggregated to synthesize a final answer. This eliminates the need for extensive, task-specific pipeline engineering, offering a versatile solution for both table-based question answering and fact verification. By addressing the brittleness of unidirectional reasoning, CRAFT offers a principled approach advancing toward more structured and reliable table comprehension.

Specifically, we incorporate four cooperative modules: 1) **Rewriter** reformulates the original question into a declarative hypothesis and transforms the problem into a verifiable claim. 2) **Reverser** applies targeted transformation rules to this hypothesis to generate an informative counterfactual statement, creating a complementary scenario for reasoning. 3) **Extractor** distills essential supporting evidence from the intermediate reasoning steps of LLMs, proposing candidate answers under both factual and counterfactual assumptions. 4) **Rethinker** aggregates the extracted evidence and candidate answers to deliberate and arrive at the final, verified decision. The overall framework is illustrated in Figure 1.

In summary, our contributions are three folds:

• We propose CRAFT, a counterfactual reasoning framework agnostic to specific task formulations and applies uniformly for reasoning over structured data.

• Extensive experiments and analyses are conducted to validate the effectiveness and robustness of CRAFT, which significantly outperforms existing methods and consistency-based strategies in both accuracy and stability.

• Our results indicate that counterfactual reasoning helps guide LLMs away from fixed patterns and toward more heuristic inference, providing a new perspective for the development of trustworthy reasoning systems.

2 Related work

Table Reasoning Table reasoning has gained significant attention, with many studies focusing on modifying model parameters to better adapt to the structural characteristics of tabular data. Among early attempts, TAPAS (Herzig et al., 2020) models row–column relations through structured embeddings. A parallel line of research trains models on large collections of table-related corpora

through pre-training, instruction tuning, or structured fine-tuning, exemplified by RePanda(Chegini et al., 2025), enabling them to acquire tabular semantic patterns, alignment behavior, and execution abilities (Liu et al., 2022; Zhang et al., 2025b; Su et al., 2024).Studies also incorporate reinforcement learning, using feedback signals to further enhance model performance (Aly et al., 2023; Wu et al., 2025; Pan et al., 2026).

On the other hand, large language models exhibit strong inherent reasoning abilities, enabling competitive performance on table tasks without modifying parameters (Brown et al., 2020). Prompting-based strategies have therefore been widely explored as a way to elicit this latent reasoning capability(Zhou et al., 2023a; Chen et al., 2023; Brown et al., 2020). Chain-of-Thought (Wei et al., 2022) explicitly encourages models to articulate intermediate reasoning steps. Subsequent work extends this paradigm to tables by exploiting their structural properties. Chain-of-Table (Wang et al., 2024) achieves this by breaking reasoning into executable sub-steps, where each step builds on the sub-table produced by the last. Other methods condense or reorganize table inputs prior to reasoning to ensure that models operate on compact, task-relevant data (Sui et al., 2024; Nahid and Rafiei, 2024), or incorporate external symbolic execution frameworks that offload computation and validation to SQL or Python engines (Cheng et al., 2023; Zhang et al., 2024; Ni et al., 2023; Mao et al., 2025), allowing LLMs to concentrate on planning rather than calculation.

Building further, other studies emphasize multi-round and structural reasoning mechanisms for robustness and corrective capability. Examples include self-consistency (Wang et al., 2023; Li et al., 2025), tree-structured (Ji et al., 2024) and graph-structured reasoning (Li et al., 2025) to enhance stability through multi-branch logical inference. MIX-SC (Liu et al., 2024b) combines textual and symbolic reasoning, while Table-Critic (Yu et al., 2025) introduces critical assessment to allow self-correction of intermediate steps. However, existing table-reasoning methods generally reinforce inference around the original question and do not explore the use of counterfactual reasoning paths for explicit evidence contrast.

Counterfactual Reasoning Many studies construct counterfactual variants of inputs to enhance causal stability (Feng et al., 2021), strengthen out-of-distribution generalization through relation-driven counterfactual contrasts (Yang et al., 2023), and test inference discrimination under atypical reasoning conditions (Webb et al., 2025). It has also been employed for semantic delimitation and self-calibration, such as distinguishing hypothetical from factual knowledge (Li et al., 2023) and improving contextual faithfulness through counterfactual exemplars in prompts (Zhou et al., 2023b). The Direct–Indirect Reasoning (Zhang et al., 2025c) framework provides a more general paradigm for solving mathematical problems by jointly incorporating forward and reverse verification. Kim introduces Counterfactual-Consistency Prompting (Kim and Hwang, 2025), which constructs temporal counterfactual questions to enforce consistency, showing strong performance in relative temporal reasoning. However, this approach relies exclusively on counterfactual and mainly to be effective for time-related verification tasks. To our knowledge, counterfactual reasoning has not been explored in table reasoning, even though the structured nature of tabular data makes it naturally compatible to counterfactual validation.

3 Method

Notation and Definitions. The overall framework of CRAFT consists of four modules—REWRITER, REVERSER, EXTRACTOR, and RETHINKER. To formalize the problem setting, we denote the table as T, ground-truth answer as A, the natural-language question as Q, and the declarative statement derived from Q as S. In table fact verification (FV), the input is already a declarative claim, and we have $S = Q$.

3.1 Rewriter

Rewriter module is designed to transform an open-end table QA task into an informative and verifiable declarative statement that facilitates the subsequent generation of counterfactual statement. Formally, the Rewriter process can be written as:

$$a \sim \Omega_{T,Q} \;:=\; \{\, v \mid \tau(v) = \tau_{A_Q} \,\}. \tag{1a}$$

$$S = \mathcal{M}(Q, a). \tag{1b}$$

We first infer the expected answer type τ_{A_Q} from the semantics of Q alone, without refer to any ground-truth answer at inference time. Here, τ_{A_Q} denotes the semantic type expected by Q (e.g., time, number, or boolean), and $\tau(v)$ denotes the semantic type of a candidate value v. Accordingly,

$\Omega_{T,Q}$ denotes the type-consistent candidate value space induced by T and Q. An atomic value a is sampled from this space and used to instantiate the semantic slot in Q, yielding a declarative statement S. In practice, the LLM is prompted to infer a from cells in T. When no explicit cell is available, it may randomly instantiate an a with a type-consistent value grounded in T and Q.

Unlike prior work that primarily uses QA pairs to improve fact verification (Aly et al., 2023), our REWRITER explicitly bridges QA and FV by transforming a question into a declarative statement S via type-consistent instantiation. In this way, QA is recast into the same statement-based form as FV, allowing the downstream modules to function uniformly across both tasks.

3.2 Reverser

The Reverser module takes a declarative statement S as input and aims to construct an optimal counterfactual statement R^*. Counterfactual reasoning is not a simple binary flip; instead, a statement S may admit multiple counterfactual directions. In particular, when the Q is available, S may carry latent semantics beyond its surface form, so the admissible counterfactual directions should be defined with respect to the joint semantics of (Q, S). For clarity, we denote counterfactuals instantiated along different directions by $R_i = \phi^{\delta_i}(S)$, where δ_i simply indexes distinct admissible directions under the semantic context. Prior work has shown that models may underutilize information depending on input position or presentation (Liu et al., 2024a; Wan et al., 2025).Therefore, we introduce the notion of a latent reasoning space $\Psi(\cdot)$ to denote the latent reasoning space induced by a declarative statement during inference. Under this notation, counterfactuals constructed along different directions are characterized by inducing different reasoning subspaces:

$$\delta_i \neq \delta_j \;\Rightarrow\; \Psi(R_i) \neq \Psi(R_j). \tag{2}$$

The purpose of introducing counterfactuals is to broaden the reasoning space beyond that induced by S alone. We characterize the optimal counterfactual R^* as the one whose induced reasoning, when considered together with that of S, provides the greatest overall informational contribution:

$$R^* \;=\; \arg\max_i \; \mathcal{G}(\Psi(S) \;\cup\; \Psi(R_i)) \,. \tag{3}$$

To construct an informative reverse statement R^*, we first generate a small set of candidate counterfactuals from S using a rule-based template set derived from statement semantics (and the original question Q when available), ensuring that the candidates remain plausible and semantically aligned with the original statement. We then prompt the model to generate a SQL-style verification program for S and each reverse candidate, conditioned on the table's column names and the entities mentioned in the statement, and use the resulting program structure as a proxy for the induced reasoning space. The final reverse statement R^* is selected as the candidate whose induced program yields the largest structural expansion beyond that of S. While more advanced SQL generation techniques may further improve performance, we intentionally adopt a lightweight generation strategy in this work. More implementation details are provided in Appendix D.1.

3.3 Extractor

Extractor aims to produce both supporting evidence E and candidate answers A for the original question Q. Given the rewrite statement S and the counterfactual statement R^*, the model is prompted to perform step-by-step reasoning grounded in the table T[1], resulting in two reasoning traces, $\mathrm{Trace}(S)$ and $\mathrm{Trace}(R^*)$. Each reasoning trace is treated as an instantiation of the corresponding latent reasoning space $\Psi(X)$.

We collect evidence from each trace as follows:

$$\mathbb{I}_{T,Q}(e) = \begin{cases} 1, & \text{if } e \text{ is consistent w.r.t. } (T, Q), \\ 0, & \text{otherwise,} \end{cases}$$

$$E(X) = \sum_{e \in \mathrm{Trace}(X)} \mathbb{I}_{T,Q}(e), \qquad X \in \{S, R^*\}. \tag{4}$$

Here, $\mathbb{I}_{T,Q}(e)$ using an LLM to evaluate whether a candidate evidence item e appearing in $\mathrm{Trace}(X)$ is logically consistent under the table–question context (T, Q) and informative for answering the question. Items that fail this check are discarded, while those with $\mathbb{I}_{T,Q}(e) = 1$ are accumulated along the trace to form the evidence associated with X, denoted $E(X)$.

Applying this procedure to the two traces yields two evidence sets, $E_1 = E(S)$ and $E_2 = E(R^*)$, corresponding to the Rewriter and Reverser reasoning processes, respectively. Each evidence

[1]We implement our method within the Table-Critic framework for execution convenience, but our approach can be deployed within other table reasoning systems as well.

Method	Deepseek-R1-14B		TeleChat3-36B		Qwen2.5-72B		Llama3.3-70B		GPT-5-mini		Average	
	WikiTQ	TabFact	WikiTQ	TabFact	WikiTQ	TabFact	WikiTQ	TabFact	WikiTQ	TabFact	WikiTQ	TabFact
End-to-End	54.3	91.8	58.4	86.7	57.3	85.6	51.1	83.4	85.2	94.7	61.3	88.4
Few-Shot	70.0	91.3	64.1	86.2	60.6	86.0	62.0	83.1	85.7	95.1	68.5	88.3
Binder	68.5	89.5	62.8	85.3	65.5	84.0	57.6	79.9	84.2	94.5	67.7	86.6
Dater	71.6	92.1	67.2	90.1	63.8	89.7	63.3	89.9	84.9	95.9	70.2	91.5
Chain-of-Table	75.1	92.7	74.6	91.6	68.3	89.7	62.1	89.9	86.2	96.0	73.3	92.0
Critic-CoT	75.2	92.5	71.3	90.4	69.0	89.8	66.8	88.0	86.1	94.9	73.7	91.1
Alter	67.4	90.5	69.5	89.2	70.9	87.4	67.8	89.8	–	94.6	68.9	90.3
Table-Critic	76.3	93.3	76.8	93.1	77.7	92.9	70.1	92.1	86.7	95.5	77.5	93.4
CRAFT (CoT)	77.2	93.6	78.4	**94.5**	78.7	92.2	79.9	92.0	87.0	**96.7**	80.2	93.8
CRAFT	**79.9**	**94.2**	**79.1**	94.0	**80.4**	**93.9**	**81.4**	**94.1**	**87.8**	96.2	**81.7**	**94.5**

Table 1: Performance comparison between CRAFT and representative table-reasoning baselines on WikiTQ and TabFact across five backbone LLMs. CRAFT (CoT) uses Chain-of-Table as its extractor. Bold and underlined values indicate the best and second-best performance in each column, respectively. Averages are computed over available results.

set is subsequently used together with the original question and table to produce a candidate answer via the same large language model, i.e., $A_1 = \mathcal{M}(Q, T, E_1)$ and $A_2 = \mathcal{M}(Q, T, E_2)$.

3.4 Rethinker

We design a set of weighted decision rules for the Rethinker module. When A_1 and A_2 coincide, the module directly returns the shared answer. When the two candidate answers differ, we compute self-consistency scores for each (answer, evidence) pair produced by Rewriter and Reverser, mapping each score into the range $[-1, 1]$. The module then compares the score difference and determines the answer according to a predefined set of rules. For cases in which the two candidates cannot be reliably distinguished, the model re-evaluates each candidate under the opposing evidence to test whether its reasoning remains consistent under counter perspectives. The detailed decision algorithm is provided in Appendix D.2.

4 Experiments

4.1 Experimental Setup

Datasets We primarily evaluate the proposed CRAFT on two widely used table-understanding datasets: TabFact (Chen et al., 2020) and WikiTQ (Pasupat and Liang, 2015).

TabFact is a table-based binary fact verification (FV) dataset in which the task is to determine whether a given statement is supported by the table. Following prior work, we report binary classification accuracy as the evaluation metric. In contrast, WikiTQ is a table-based question answering (QA) dataset with short, structured answers. We evaluate performance using the official denotation accuracy.

For all datasets, we conduct systematic evaluations using several strong base models, including Deepseek-R1-distilled-14B(Guo et al., 2025), LLaMA 3.3–70B (Grattafiori et al., 2024), Telechat3-36B (Wang et al., 2025) Qwen 2.5–Instruct (Qwen et al., 2025), and GPT-5-mini (OpenAI, 2025).[2] The full details of all tested model settings, prompt formats, hyperparameters, and additional experiments settings are provided in Appendix A.

Baselines We compare CRAFT with three representative categories of baseline methods, with a detailed comparison against Fobar (Jiang et al., 2024) provided in Appendix B.3:

(1) Standard reasoning methods typically rely on direct prompting. End-to-End QA feeds the table and the question directly into the model, without any explicit intermediate reasoning. Few-Shot QA extends this setting by conditioning the model on a small set of in-context table–question–answer examples, from which the model implicitly learns the desired input–output patterns and reasoning regularities.

(2) Task decomposition methods decompose table reasoning into structured subtasks. Binder (Cheng et al., 2023) translates questions into executable SQL or Python programs, enabling interpretable symbolic reasoning. Dater (Ye et al., 2023) follows a parsing–execution–filling process and performs question decomposition at the sub-

[2]For each model family, we report the best-performing version observed within the models we evaluated.

table level. Chain-of-Table (Wang et al., 2024) incrementally constructs sub-tables, with each step building on the previous one. ALTER (Zhang et al., 2025a) retrieves a relevant subset of data and augments it with schema and semantic information.

(3) Critic-based methods refine reasoning through self-evaluation or multi-agent critique. Critic-CoT (Zheng et al., 2025) applies a self-critique stage that reviews and refines generated CoTs to revise erroneous inference. Table-Critic (Yu et al., 2025) adopts an agentic multi-stage workflow that iteratively evaluates, critiques, and revises intermediate reasoning steps to enhance logical consistency and reliability.

4.2 Main Results

Table 1 reports the performance of our method and representative baselines under different backbone LLMs.

Across the four backbone LLMs, our method achieves an average of 81.7% on WikiTQ and 94.5% on TabFact, exceeding the strongest baseline by 4.2 and 1.1 points, respectively. This improvement is observed for all four backbones, with statistical significance tests reported in the appendix B.1. Notably, the largest absolute gain occurs with Llama3.3–70B, where WikiTQ accuracy increases by 11.3 points over the strongest baseline. In standard baseline settings and at comparable model scales, Qwen2.5-72B consistently outperforms Llama3.3-70B. With our framework in place, however, this ordering is substantially altered: Llama3.3-70B surpasses Qwen2.5-72B on WikiTQ and TabFact. This reversal indicates that our method significantly reduces cross-model performance discrepancies.

Beyond absolute performance, the results reveal two systematic patterns. First, although open-ended TableQA remains more challenging than fact verification, our framework yields larger relative gains on WikiTQ, providing direct evidence that casting QA into a fact-verification–style reasoning paradigm is an effective modeling choice. Second, our method exhibits uniform applicability across a wide range of model regimes, spanning different model families, different parameter scales, and both open-source and closed-source LLMs. This robustness further extends to different reasoning instantiations: even when executed by a simpler table-reasoning framework such as Chain-of-Table, our method still outperforms other baseline methods, suggesting that it serves as a general reasoning scaffold rather than backbone-dependent optimization.

Method	WikiTQ (%)	TabFact (%)
Qwen2.5-72B		
Table-Critic	77.7	92.9
Rewriter only†	79.1	92.9
Reverser only	78.8	91.3
CRAFT	**80.4**	**93.9**
LLaMA3.3-70B		
Table-Critic	70.1	92.1
Rewriter only†	79.9	92.1
Reverser only	79.7	91.9
CRAFT	**81.4**	**94.1**

Table 2: Accuracy of Rewriter-only, Reverser-only, and the CRAFT based on both Rewriter and Reverser across different backbone models. † Rewriter differs from the Baseline for WikiTQ but is equivalent for TabFact.

Taken together, these results indicate that counterfactual reasoning functions as a general reasoning heuristic rather than a task-specific solution. By guiding how models explore alternatives, it provides a more effective way of thinking about table reasoning, leading to consistent performance gains across settings.

4.3 Ablation Study on Rewriter and Reverser

As shown in Table 2, we conduct ablation experiments to verify the effectiveness of Rewriter, Reverser and their combination. For TabFact, where inputs are already in declarative form. Taking LLaMA3.3–70B as an example, Rewriter-only matches the baseline performance of 92.1%, Reverser-only follows the counterfactual path alone and drops slightly to 91.9%. This difference is as anticipated because counterfactual reasoning deliberately explores alternative and potentially incorrect hypotheses. Notably, the comparable accuracy between the two paths indicates that solving counterfactual statements can still recover sufficient evidence to answer the original question.

In TableQA, however, Rewriter-only leads to a substantial improvement, highlighting the benefit of converting questions into declarative statements. The additional gain from combining Rewriter and Reverser suggests a complementary interaction, where counterfactual reasoning broadens the evidence considered for the original question.

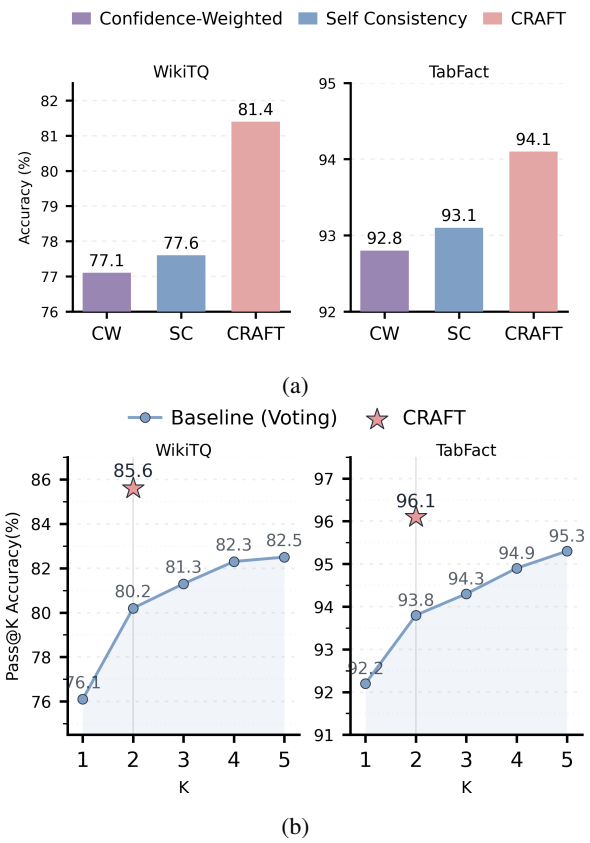


Figure 2: **Repeated-Sampling analyses on WikiTQ/TabFact. (a)** Accuracy comparison between voting methods and CRAFT. **(b)** Ideal(Pass@K) accuracy upper bounds compared with our method.

4.4 Effectiveness Beyond Repeated-Sampling

Given that our framework involves one more reasoning path in structure, we perform controlled comparisons to eliminate the concern that the observed gains are merely due to repeated sampling. Specifically, we compare the voting method based on Table-Critic with (1) **Self-Consistency (SC)**, which samples N answers for each question and selects the final prediction by majority voting over their frequencies; and (2) **Confidence-Weighted (CW)**, which also samples N answers but aggregates them by summing $\exp(\text{score})$ for each unique candidate and choosing the answer with the highest total weight. We choose $N = 3$ to allow a fair comparison within a reasonable computational cost.[3]

As shown in Fig. 2(a), on both table question answering (QA) and table fact verification (FV) tasks, our framework, respectively, achieves improvements of 3.8 and 1.0 percentage points over the best of two voting methods. This indicates that the improvements do not simply stem from ensembling a diverse set of answers.

To further quantify the potential benefit of our method, we adopt the *Pass@K* metric (Chen et al., 2021a) to measure whether the correct answer appears among a set of K generated predictions. Given the set of predictions $\{\hat{y}_1, \ldots, \hat{y}_K\}$ and the ground-truth label y, Pass@K is defined as:

$$\text{Pass@}K = \mathbb{I}(\exists\, i \in \{1, \ldots, K\} \text{ s.t. } \hat{y}_i = y)\,, \quad (5)$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

Pass@K measures whether the candidate answer set contains at least one correct prediction, thereby reflecting how many truly solvable cases are covered by the model's output distribution and providing an upper bound on the achievable accuracy under K samples(Yue et al., 2025).

As reported in Fig. 2(b), we compare our method with repeated sampling of the strongest baseline under different values of K. Our method effectively corresponds to $K = 2$, since it explicitly constructs two distinct reasoning paths. Under the same setting, our method achieves a higher Pass@K than the repeated-sampling baseline. Even when the baselines are allowed to increase K, they do not match the performance of CRAFT. This suggests that the direction of sampling—i.e., encouraging diverse and structured exploration of the reasoning space—is more important than the sheer number of samples. By guiding the model to explore different regions of table semantics rather than repeatedly sampling from the same distribution, our method achieves higher coverage of correct reasoning trajectories with only a small number of candidates.

4.5 Impact of Self-Critique Iterations

The relationship between our method and self-critique is examined by analyzing the effect of varying the number of self-critic iterations. In our method, self-critique is used in the Extractor. Figure 3 shows the effect of varying the number of self-critic iterations on task performance. Across both WikiTQ and TabFact, increasing the number of self-critic iterations consistently improves accuracy compared to the no-self-critic setting, while the performance gain of single-direction reasoning quickly saturate.

Notably, even with an increase in the maximum number of self-critic iterations ($r_{\max} \in [1, 10]$) ,

[3]From this section onward, all experiments use Llama3–70B unless stated otherwise. Outputs are normalized and format mismatches are not counted as errors, unlike in the main table.

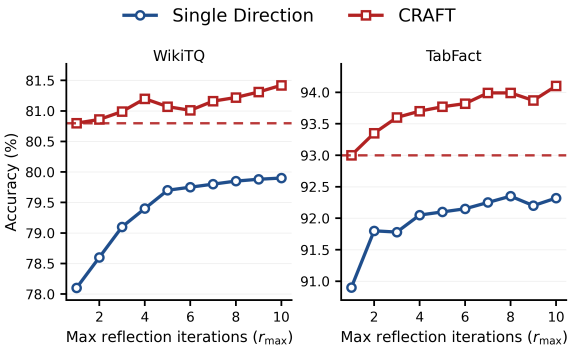


Figure 3: WikiTQ and TabFact accuracy as the number of self-critic iterations increases, comparing CRAFT with single-direction reasoning.

the performance of single-direction reasoning remains below that of our method, which integrates bidirectional evidences from Rewriter and Reverser. At the same time, the performance of CRAFT also improves as $r_{\max}$ increases, indicating that it is compatible with self-critique. These trends suggest that self-critic effectively strengthens individual candidates, while bidirectional reasoning provides additional, complementary gains that cannot be recovered by increasing self-critic iterations alone.

4.6 Effect of Multiple Counterfactual Statements

To examine whether multiple counterfactual directions can further improve reasoning coverage beyond the single-counterfactual setting, we generate K counterfactual statements ($K \in 1, 2, 3$) for each original statement, run the same reasoning pipeline independently on each of them, and evaluate the resulting predictions with Pass@K. As shown in Table 3, increasing K yields consistent but modest gains, with diminishing returns as more counterfactual statements are added. Notably, compared with the repeated-sampling results in Fig. 2(b), introducing additional counterfactual-driven reasoning paths leads to larger Pass@K improvements than simply repeating the reasoning process. However, this comes at higher computational cost, and although higher Pass@K suggests greater potential benefit from multiple paths, reliably selecting the best one remains non-trivial.

4.7 Performance Across Different Table Sizes

Large tables pose substantial challenges for large language models (LLMs), as they often struggle to effectively track, integrate, and reason over long input contexts (Liu et al., 2023; Ye et al., 2023). To

# of Reverse	Pass@K (%)	
	WikiTQ	TabFact
0	79.9	92.1
1	85.6	96.1
2	87.7	96.5
3	88.8	96.7

Table 3: Pass@K results when using Rewriter with K Reverse (counterfactual) reasoning paths.

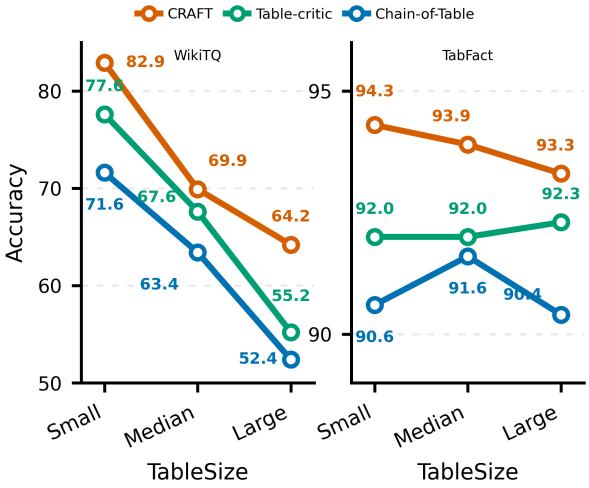


Figure 4: **Performance across different table sizes.** We partition tables into three size groups with thresholds: For WikiTQ, *small* (<2000 tokens), *medium* (2000–4000), and *large* (>4000); For TabFact, *small* (<500 tokens), *medium* (500–800), and *large* (>800).

evaluate the impact of table size on performance, we compare our method against two representative baselines, as shown in Figure 4.

For WikiTQ, all methods exhibit degraded performance as table size increases, a trend that reflects the growing complexity of reasoning over larger contexts. In contrast, evaluation results of TabFact task show no significant correlation with table size. A plausible explanation is that it primarily requires pinpointing and validating specific claims against relevant table cells and is less sensitive to overall table scale. Notably, our approach demonstrates superior robustness, consistently outperforming baselines across all table sizes. It allows the counterfactual framework to maintain computational efficiency as context length increases.

4.8 Generalization to Hybrid Table–Text QA

We further evaluate CRAFT on TAT-QA (Zhu et al., 2021), a financial-domain benchmark requiring joint reasoning over tables and text, using LLaMA3-70B and Qwen2.5-72B. We follow the

Method	LLaMA3-70B	Qwen2.5-72B
Zero-shot	51.1	57.6
Few-shot	55.9	59.2
Chain-of-Table	56.7	55.1
Table-Critic	61.6	65.7
CRAFT (CoT)	68.2	72.2
CRAFT	**71.8**	**73.9**

Table 4: Performance (%) on TAT-QA under the same adapted text-table input setting. CRAFT (CoT) uses Chain-of-Table as the extractor.

Metric	Definition	Value
Step-0 Acc.	$\Pr(\hat{y}_0 = y)$	61.5%
Final Acc.	$\Pr(\hat{y} = y)$	81.4%
Correction Rate	$\Pr(\hat{y} = y \mid \hat{y}_0 \neq y)$	59.3%
Recovered Share	$\Pr(\hat{y}_0 \neq y \mid \hat{y} = y)$	28.1%

Table 5: Step-0 candidate accuracy and error-correction analysis on WikiTQ with LLaMA3

format alignment and evaluation protocol described in Section A.2. As shown in Table 4, CRAFT outperforms all baselines with both backbones, including when Chain-of-Table is used as its extractor, while the full framework achieves the best performance overall. This trend is consistent with the main results, indicating that the conclusions remain unchanged in the text–table hybrid setting.

4.9 Step-0 and Error-Correction Analysis

To examine whether the performance gain mainly comes from the initial candidate already being correct, we conduct an additional WikiTQ analysis with LLaMA3 over three independent runs. Let y denote the gold answer, $\hat{y}_0$ denote the Step-0 candidate, and $\hat{y}$ denote the final prediction. Table 5 reports the mean Step-0 candidate accuracy and the corresponding error-correction results.

The Step-0 potential-answer accuracy is only 61.5% on average, substantially lower than the final-answer accuracy of 81.4%. This indicates that the final performance gain cannot be explained by simply inheriting a correct initial candidate. In particular, CRAFT corrects 59.3% of initially wrong Step-0 predictions, and 28.1% of final correct answers come from cases where the Step-0 prediction was wrong. These results show that CRAFT has a strong error-correction ability: even when the initial candidate is incorrect, the reverse path and the Rethinker can recover the correct answer by integrating complementary table-grounded evidence.

5 Conclusion

We propose CRAFT, an explicit counterfactual framework for table reasoning. The method jointly guides an LLM through both forward and counterfactual reasoning paths, which are integrated via a dedicated rethinker module to arrive at the final decision. The framework not only delivers substantial performance gains but also unifies table-based question answering and fact verification under a shared counterfactual reasoning paradigm. Furthermore, the counterfactual perspective provides new insight into how LLMs can be more effectively steered toward structured table understanding.

Limitations

While our framework achieves competitive and encouraging results, we acknowledge several limitations that call for continued exploration. First, We did not conduct experiments with smaller base models, because tabular inputs typically require long contexts. Second, our method is tailored to table reasoning tasks. Although the underlying ideas may generalize to other forms of reasoning—such as text-based QA—ensuring semantic consistency and generating atomic facts that align closely with ground-truth answers remains highly challenging. Finally, the performance of our RETHINKER module still leaves room for improvement. While the current results are encouraging, the module has not yet fully realized the potential of counterfactual reasoning. Further refinement is required to maximize its corrective and reasoning capabilities, providing an important direction for future work.

Ethics Statement

All experiments in this paper are conducted on publicly available benchmark datasets, which do not contain personal or sensitive information. Parts of the implementation and baseline systems are adapted from prior work and publicly released codebases, with appropriate attribution and in compliance with their respective licenses. We view this framework as a research contribution and encourage its use in settings consistent with its intended scope. We therefore believe that this study complies with the ARR Ethics Policy.

Acknowledgements

This work was supported by the Innovation Studio for Beijing Outstanding Young Engineers.

References

Rami Aly, Marek Strong, and Andreas Vlachos. 2023. QA-NatVer: Question answering for natural logic-based fact verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8376–8391, Singapore. Association for Computational Linguistics.

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*, 11:227–249.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, Andre Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. 2025. LLMs instead of human judges? a large scale empirical study across 20 NLP evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–255, Vienna, Austria. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Atoosa Chegini, Keivan Rezaei, Hamid Eghbalzadeh, and Soheil Feizi. 2025. RePanda: Pandas-powered tabular verification and reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32200–32212, Vienna, Austria. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021a. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

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. *Preprint*, arXiv:1909.02164.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021b. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. 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. *Preprint*, arXiv:2210.02875.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding — a survey. *Transactions on Machine Learning Research*.

Fuli Feng, Jizhi Zhang, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Empowering language understanding with counterfactual reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2226–2236, Online. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Rajarshi Haldar and Julia Hockenmaier. 2025. Rating roulette: Self-inconsistency in LLM-as-a-judge frameworks. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24986–25004, Suzhou, China. Association for Computational Linguistics.

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

Deyi Ji, Lanyun Zhu, Siqi Gao, Peng Xu, Hongtao Lu, Jieping Ye, and Feng Zhao. 2024. Tree-of-table: Unleashing the power of llms for enhanced large-scale table understanding. *Preprint*, arXiv:2411.08516.

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James Kwok. 2024. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6647–6661, Bangkok, Thailand. Association for Computational Linguistics.

Jongho Kim and Seung-won Hwang. 2025. Counterfactual-consistency prompting for relative temporal understanding in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1210–1225, Vienna, Austria. Association for Computational Linguistics.

Jiaxuan Li, Lang Yu, and Allyson Ettinger. 2023. Counterfactual reasoning: Testing language models' understanding of hypothetical scenarios. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 804–815, Toronto, Canada. Association for Computational Linguistics.

Qianlong Li, Chen Huang, Shuai Li, Yuanxin Xiang, Deng Xiong, and Wenqiang Lei. 2025. GraphOTTER: Evolving LLM-based graph reasoning for complex table question answering. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5486–5506, Abu Dhabi, UAE. Association for Computational Linguistics.

Jixiong Liu, Yoan Chabot, Raphaël Troncy, Viet-Phi Huynh, Thomas Labbé, and Pierre Monnin. 2023. From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. *Journal of Web Semantics*, 76:100761.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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. *Preprint*, arXiv:2107.07653.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024b. 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.

Weizheng Lu, Jing Zhang, Ju Fan, Zihao Fu, Yueguo Chen, and Xiaoyong Du. 2025. Large language model for table processing: a survey. *Frontiers of Computer Science*, 19:192350–.

Qingyang Mao, Qi Liu, Zhi Li, Mingyue Cheng, Zheng Zhang, and Rui Li. 2025. Potable: Towards systematic thinking via stage-oriented plan-then-execute reasoning on tables. *Preprint*, arXiv:2412.04272.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. TabSQLify: Enhancing reasoning capabilities of LLMs through table decomposition. 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 5725–5737, Mexico City, Mexico. Association for Computational Linguistics.

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. *Preprint*, arXiv:2302.08468.

OpenAI. 2025. Gpt-5 system card. `https://openai.com/index/gpt-5-system-card/`.

Changzai Pan, Jie Zhang, Kaiwen Wei, Chenshuo Pan, Yu Zhao, Jingwang Huang, Jian Yang, Zhenhe Wu, Haoyang Zeng, Xiaoyan Gu, Weichao Sun, Yanbo Zhai, Yujie Mao, Zhuoru Jiang, Jiang Zhong, Shuangyong Song, Yongxiang Li, and Zhongjiang He. 2026. Reasontabqa: A comprehensive benchmark for table question answering from real world industrial scenarios. *Preprint*, arXiv:2601.07280.

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

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is LLM-as-a-judge robust? investigating universal adversarial attacks on zero-shot LLM assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7499–7517, Miami, Florida, USA. Association for Computational Linguistics.

Yucheng Ruan, Xiang Lan, Jingying Ma, Yizhi Dong, Kai He, and Mengling Feng. 2024. Language modeling on tabular data: A survey of foundations, techniques and evolution. *Preprint*, arXiv:2408.10548.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long,

and 14 others. 2024. Tablegpt2: A large multimodal model with tabular data integration. *Preprint*, arXiv:2411.02059.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2024. TAP4LLM: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10306–10323, Miami, Florida, USA. Association for Computational Linguistics.

David Wan, Jesse Vig, Mohit Bansal, and Shafiq Joty. 2025. On positional bias of faithfulness for long-form summarization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8791–8810, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Camera-ready version at ICLR 2023.

Zihan Wang, Xinzhang Liu, Yitong Yao, Chao Wang, Yu Zhao, Zhihao Yang, Wenmin Deng, Kaipeng Jia, Jiaxin Peng, Yuyao Huang, Sishi Xiong, Zhuo Jiang, Kaidong Yu, Xiaohui Hu, Fubei Yao, Ruiyu Fang, Zhuoru Jiang, Ruiting Song, Qiyi Xie, and 19 others. 2025. Technical report of telechat2, telechat2.5 and t1. *Preprint*, arXiv:2507.18013.

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Published as a conference paper at ICLR 2024.

Taylor W. Webb, Keith J. Holyoak, and Hongjing Lu. 2025. Evidence from counterfactual tasks supports emergent analogical reasoning in large language models. *PNAS Nexus*, 4(5):pgaf135.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. 2025. Table-r1: Region-based reinforcement learning for table understanding. *Preprint*, arXiv:2505.12415.

Sishi Xiong, Ziyang He, Zhongjiang He, Yu Zhao, Changzai Pan, Jie Zhang, Shuangyong Song, and Yongxiang Li. 2025. Tablezoomer: a collaborative agent framework for large-scale table question answering. *Vicinagearth*, 2(1):11.

Heerin Yang, Seung-won Hwang, and Jungmin So. 2023. Relation-based counterfactual data augmentation and contrastive learning for robustifying natural language inference models. In *Proceedings of INTERSPEECH 2023*. ISCA. Accepted at INTERSPEECH 2023.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Peiying Yu, Guoxin Chen, and Jingjing Wang. 2025. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17432–17451, Vienna, Austria. Association for Computational Linguistics.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

Han Zhang, Yuheng Ma, and Hanfang Yang. 2025a. ALTER: Augmentation for large-table-based reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 179–198, Albuquerque, New Mexico. Association for Computational Linguistics.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao, Juanzi Li, and Jie Tang. 2025b. TableLLM: Enabling tabular data manipulation by LLMs in real office usage scenarios. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10315–10344, Vienna, Austria. Association for Computational Linguistics.

Yanfang Zhang, Yiliu Sun, Yibing Zhan, Dapeng Tao, Dacheng Tao, and Chen Gong. 2025c. Large language models as an indirect reasoner: Contrapositive and contradiction for automated reasoning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5040–5057, Abu Dhabi, UAE. Association for Computational Linguistics.

Yuchen Zhang, Yuze Gao, Bin Chen, Wenfeng Li, Shuo Sun, and Jian Su. 2025d. High-quality complex text-to-SQL data generation through chain-of-verification. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2368–2379, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

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

Xin Zheng, Jie Lou, Boxi Cao, Xueru Wen, Yuqiu Ji, Hongyu Lin, Yaojie Lu, Xianpei Han, Debing Zhang, and Le Sun. 2025. Critic-CoT: Boosting the reasoning abilities of large language model via chain-of-thought critic. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1768–1806, Vienna, Austria. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Published at ICLR 2023.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023b. Context-faithful prompting for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14544–14556, Singapore. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

Appendix

A Implementation Details

All experiments were conducted on a dedicated server equipped with NVIDIA A100 40GB GPUs (×8) with CUDA 12.2. We evaluated our framework using multiple backbone large language models. Open-source models including Llama-3.3-70B and DeepSeek-R1-14B were deployed locally using the vLLM inference engine, while GPT-5-mini was accessed via the Microsoft Azure OpenAI API.

Unless otherwise specified, we used a temperature of 0 and top-p of 1.0. The maximum generation length was set to at most 2048 tokens. In the main experiments, we selected the number of self-critique iterations r from $1, \ldots, 10$ based on validation performance, and used the best-performing value. For API-based models[4], due to endpoint constraints, we set the temperature to 1. In addition, when inference timeouts occurred, we set the reasoning level to `minimum` to ensure stable execution, while keeping all other settings unchanged.

A.1 Output Normalization and Evaluation Metrics

In our evaluation, post-processing is applied at two different levels, corresponding to the main results and the supplementary analysis, respectively. For the main results reported in the primary tables, we apply a strictly minimal normalization only to DeepSeek-R1 distilled models (e.g., DeepSeek-Distilled-14B) by removing explicit reasoning traces or special markers (e.g., `think` blocks). No matching or answer extraction is performed, and all other models are evaluated on their raw outputs without any post-processing. This setting is intentionally conservative, ensuring that the main comparisons reflect strict adherence to the canonical answer formats of WikiTQ and TabFact.

In the supplementary analysis reported after Section 4.3, we additionally enable a lightweight matching-based extraction on top of the same normalization to better reflect practical usage scenarios in which users accept semantically correct answers even when they are embedded in natural language statements. For example, if a model outputs "The answer is X" or similar declarative forms, we extract `X` and treat it as the predicted answer. Unlike the main results, this matching-based extraction is not restricted to DeepSeek distilled models, and is applied whenever a model's output is non-canonical but contains an unambiguous answer span. Both stages are conservative and purely surface-level: they neither modify model predictions nor introduce task-specific heuristics, but only standardize syntactic variants for consistent evaluation.

[4]We do not report WikiTQ results for `gpt5-mini` under ALTER because a subset of batches returned empty responses during inference, preventing reliable aggregation of results.

A.2 TAT-QA Format Alignment and Evaluation Protocol

For the additional TAT-QA experiments, we do not modify the reasoning procedure or the prompts of any compared baseline. Since TAT-QA is a text-table hybrid dataset, while the evaluated methods are originally designed around table-centric inputs, we only perform input-format alignment to make the dataset compatible with the same evaluation interface. Specifically, we prepend the associated textual context before the table caption in the original table serialization, while keeping the remaining table content and question format unchanged. The same serialized input format is used for all compared methods. To enable a consistent accuracy-style comparison with WikiTQ, we also canonicalize TAT-QA gold answers into WikiTQ-style answer strings and evaluate predictions with the same WikiTQ evaluator. This conversion only standardizes the surface format of the reference answers, such as answer strings or lists of answer strings, and does not change the semantic content of the gold labels. Therefore, the TAT-QA results reported in the supplementary analysis should be interpreted under this adapted accuracy-style protocol.

B Additional Analysis

B.1 Statistical Analysis

We compare CRAFT with the strongest baseline over five runs on WikiTQ, TAT-QA, and TabFact using Qwen2.5 and Llama3. We apply two-sided Welch's t-tests to run-level accuracy on the fixed test sets, with Holm correction across the six comparisons. As shown in Figure 5, CRAFT achieves significantly higher accuracy in all six settings (adjusted $p < 0.01$). Its sample standard deviation ranges from 0.21 to 0.80 percentage points across these settings, indicating limited variability across runs. These results further substantiate CRAFT's advantage over the strongest baseline.

B.2 Additional Results with Qwen3.5

We further conduct supplementary experiments with Qwen3.5 on WikiTQ and TabFact. As shown in Table 6, CRAFT achieves the best performance on both datasets, with 86.1% on WikiTQ and 95.3% on TabFact. This trend is consistent with the main results, indicating that the conclusions remain unchanged when using Qwen3.5-27B as an additional backbone.

B.3 Additional Baseline Comparison

We adapt FOBAR (Jiang et al., 2024) to WikiTQ using Llama3 and Qwen2.5, with 40 forward samples and 8 backward verification samples per candidate. The inference budget is approximately comparable to that of CRAFT (CoT) under the same backbone. Following FOBAR, forward-only selection (self-consistency) chooses the most frequent sampled answer, backward-only selection uses backward verification scores, and FOBAR combines the forward and backward probabilities.

Table 7 shows that FOBAR improves over forward-only when the sampling budget is small, supporting the value of backward verification. However, this advantage diminishes as the number of forward samples increases, and its accuracy remains below that of CRAFT (CoT). This gap may reflect two limitations. In terms of the candidate space, FOBAR scores existing forward candidate answers without introducing answers beyond the sampled set. In terms of answer types, its original mathematical verification procedure focuses on masking and recovering numerical values, whereas table QA also involves lists, dates, and other cell strings. By contrast, CRAFT uses the reverse path to provide complementary table-grounded evidence and integrates this evidence to support open-ended answer discovery.

B.4 Human evaluation

We recruited three master's-level annotators to assess whether the evidence generated by each reasoning path was both relevant to the question and supported by the original table. All three annotators evaluated each example, and the final label was determined by majority vote, requiring agreement from at least two annotators. For each dataset, we annotated 50 randomly sampled examples and 10 additional examples for which the forward and reverse paths generated different candidate-answer texts. These additional answer-disagreement samples refer to disagreement between the model's two path-specific answers, not between human annotators. The evaluation concerns evidence quality; a positive judgment does not imply that the subsequently generated answer is correct.

Table 8 shows that both paths generally produce relevant and table-supported evidence on the random samples. The forward path has a modest advantage on WikiTQ, while the two paths receive identical positive-label rates on TabFact. Among

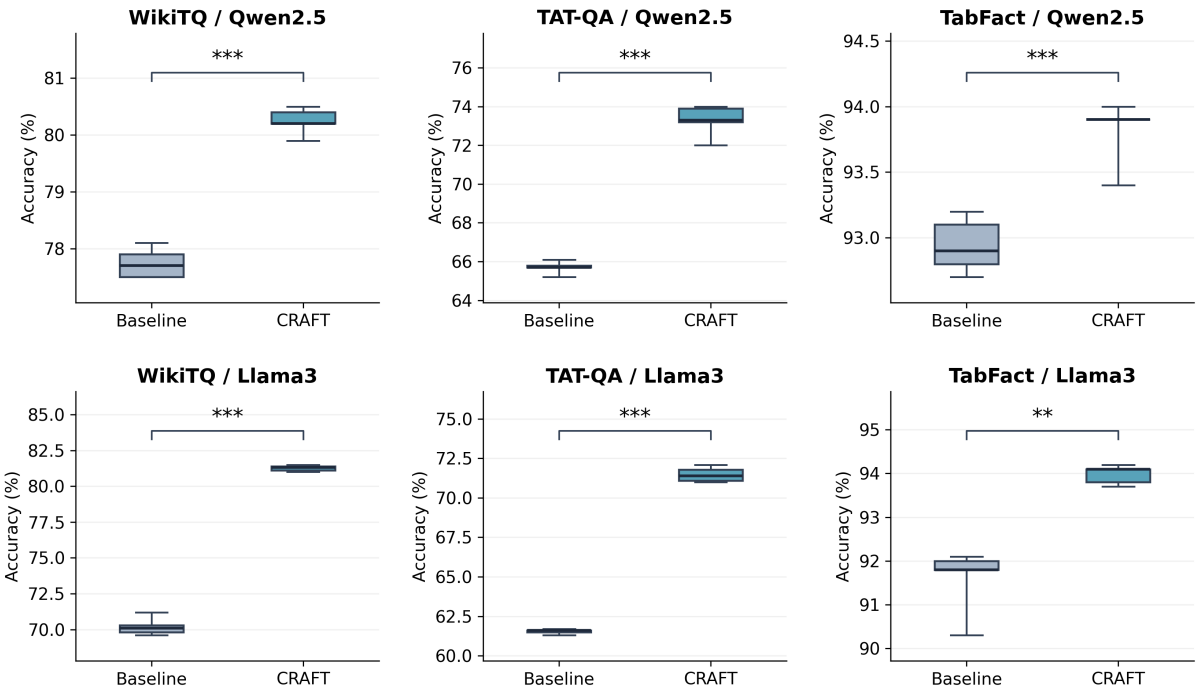


Figure 5: Accuracy across five runs for CRAFT and the strongest baseline. Boxes show the interquartile range, center lines indicate medians, and whiskers span the minimum and maximum. Significance is based on Holm-adjusted two-sided Welch's t-tests: $^{**}p < 0.01$ and $^{***}p < 0.001$. Each panel uses its own vertical scale.

Dataset	Zero-shot	Few-shot	Chain-of-Table	Table-Critic	CRAFT + CoT	CRAFT
WikiTQ	70.0	71.4	80.0	85.4	85.2	**86.1**
TabFact	88.0	86.9	93.7	93.9	94.3	**95.3**

Table 6: Performance (%) of different methods on WikiTQ and TabFact with Qwen3.5 as the backbone.

	Llama3		Qwen2.5	
Method	10+8	40+8	10+8	40+8
Forward-only (SC)	66.76	68.54	74.84	76.46
Backward-only	67.59	67.55	74.66	75.24
FOBAR	67.73	68.41	75.10	76.28
CRAFT (CoT)	79.9		78.7	

Table 7: WikiTQ accuracy (%). SC denotes self-consistency. Sampling counts apply to FOBAR and its variants; forward-only uses no backward samples. CRAFT (CoT) is compared with the 40+8 setting.

Dataset	Forward evidence	Reverse evidence
Random samples ($n = 50$)		
WikiTQ	41/50 (82%)	38/50 (76%)
TabFact	45/50 (90%)	45/50 (90%)
Additional answer-disagreement samples ($n = 10$)		
WikiTQ	5/10 (50%)	7/10 (70%)
TabFact	8/10 (80%)	8/10 (80%)

Table 8: Human verification of evidence relevance and table support.

the additional answer-disagreement samples, the reverse path receives more positive judgments on WikiTQ, while the two paths remain equal on TabFact. These findings provide no evidence of an overwhelming advantage of forward evidence over reverse evidence. Instead, the pattern is consistent with the complementary value of the two paths, suggesting that reverse reasoning can provide useful, table-supported evidence even when the paths yield different answers.

C Evaluations on Token Consumption

In this section, we analyze the token consumption of the full reasoning pipeline under different numbers of critic evaluations. The reported token counts shown in Table 9 cover the entire process, including candidate answer generation, critic-based evaluation, and final decision making.

In our framework, the critic is applied to assess the quality and evidence consistency of candidate answers. We vary the number of critic evaluations

Method	Input Tokens (M)	Output Tokens (M)	Total Weighted Tokens (M)
Chain-of-Table	73.5	1.6	10.6
Table-Critic	135.5	3.8	20.3
CRAFT w/ Chain-of-Table Extractor	171.2	6.2	26.8
CRAFT (c=1)	234.3	8.9	37.1
CRAFT (c=5)	363.2	21.0	63.8

(a) WikiTQ

Method	Input Tokens (M)	Output Tokens (M)	Total Weighted Tokens (M)
Chain-of-Table	29.3	0.6	4.2
Table-Critic	62.1	2.1	9.6
CRAFT w/ Chain-of-Table Extractor	71.1	3.4	11.9
CRAFT (c=1)	104.2	4.1	16.6
CRAFT (c=5)	158.9	6.8	25.8

(b) TabFact

Table 9: Token consumption comparison across different methods on (a) table question answering (WikiTQ) and (b) fact verification (TabFact). Token counts are measured over the full reasoning pipeline, including candidate generation, critic evaluation, and final decision making.

from $c = 1$ to $c = 10$, where each additional critic evaluation independently reassesses the candidate answers using the same criteria. As the number of critic evaluations increases, the overall token consumption grows substantially, since each evaluation requires reprocessing the candidate answers together with their associated evidence.

Beyond raw token counts, we also consider a more realistic cost model that reflects current commercial pricing schemes, in which input and output tokens are priced differently. Following the latest OpenAI pricing, output tokens are substantially more expensive than input tokens; we therefore normalize token usage by weighting output tokens more heavily. Concretely, we adopt a normalized cost defined as:

$$\text{Normalized Cost} \;=\; 0.125\,T_{\text{in}} \;+\; 0.875, T_{\text{out}},$$

where T_{in} and T_{out} denote the numbers of input and output tokens, and P_{in} and P_{out} denote their respective per-token prices.

As shown in the main results, using a larger number of critic evaluations generally leads to stronger performance, as repeated evaluation helps reduce unreliable decisions. Therefore, higher values of c are preferred when computational resources permit. At the same time, even a single critic evaluation ($c = 1$) already consistently outperforms the baseline, without further critic evaluation, while incurring substantially lower normalized cost. In addition, introducing Extractor stage based on Chain-of-Table, CRAFT significantly reduces token consumption and doesn't introduce critic, while its performance remains among the leading methods in the main results.

These results indicate a clear trade-off between performance and computational cost controlled by the number of critic evaluations. While multiple critic evaluations yield stronger results, using $c = 1$ serves as an economical alternative that retains most of the performance gains with substantially reduced token usage under realistic pricing assumptions.

D A Pseudocode Description of Rethinker/Reverser

This section includes pseudocode and brief supplementary notes on the implementation of the Reverser and Rethinker modules.

D.1 The Reverser

The Reverser procedure is summarized in Algorithm 1. To instantiate reverse candidates, we compile a small set of commonly used reverse templates based on empirical observation of statement semantics. These templates are summarized in Appendix Table 10 and are used to prompt candidate generation while keeping the candidates semantically aligned with the original statement.

Given the original statement and each candidate reverse statement, we independently prompt the model to generate a minimal SQL-style query for verification. This design is intended to reduce reward hacking during SQL generation: instead of

Category	Trigger	Reverser Template
Tied extrema	most / least / highest / lowest / biggest / fastest	Prefer multiplicity: "Not only X ..." or "Besides X, other entities also ...".
Uniqueness negation	first / only / sole / unique / earliest / latest / exactly one	Directly negate the named entity / time / place; do not use "not only".
Exclusivity reversal	which + only / solely / uniquely / all	Use multiplicity: "Not only X ...".
Time/place/value contradiction	WHEN / WHERE / HOW MANY / HOW MUCH	Directly contradict the cited time / place / value, preserving units and context.
Comparison reversal	higher/lower; earlier/later; bigger/smaller	Flip to the opposite option; never rewrite as Yes/No.
Lower-bound reversal	at least X	Rewrite as: less than X.
Upper-bound reversal	at most X	Rewrite as: more than X.
Exact-value negation	exactly X or other categories all failed	Rewrite as: not X.

Table 10: Reverser categories, triggers, and templates.

Algorithm 1 Reverser Constructor

Require: Declarative statement S; optional question Q; table column names $\mathcal{C}$; reverse template set $\mathcal{D}$
Ensure: Counter statement R^*

Step 1: Template matching and candidate instantiation ▷ generate multiple reverse candidates

$\{\delta_1, \ldots, \delta_k\} \leftarrow \mathcal{M}_{\text{MATCH}}(S, Q, \mathcal{D})$
for $j = 1$ to k **do**
  $R_j \leftarrow \phi^{\delta_j}(S, Q)$
end for
$\mathcal{R} \leftarrow \{R_1, \ldots, R_k\}$

Step 2: SQL-signature induction ▷ use SQL structure as proxy for induced reasoning space

$P_S \leftarrow \mathcal{M}_{\text{SQL}}(S, \mathcal{C})$
$\Sigma_S \leftarrow \text{EXTRACTSIGNATURE}(P_S)$
for $j = 1$ to k **do**
  $P_j \leftarrow \mathcal{M}_{\text{SQL}}(R_j, \mathcal{C})$
  $\Sigma_j \leftarrow \text{EXTRACTSIGNATURE}(P_j)$
end for

▷ each signature is represented by operation–object pairs extracted from minimal SQL

Step 3: Reverse selection ▷ maximize structural expansion beyond S

$j^* \leftarrow \arg\max_{1 \le j \le k} \left|\Sigma_S \cup \Sigma_j\right|$
$R^* \leftarrow R_{j^*}$
return R^*

allowing candidates to benefit from arbitrarily long or structurally inflated programs, we explicitly require each statement to be expressed by a minimal verification query.

We first elicit the SQL program itself and then apply rule-based extraction to derive normalized structural representations from the generated SQL. Specifically, we represent each SQL query using operation–object signatures, where each signature associates a SQL operation with its main object. In implementation, reverse selection is based primarily on structural expansion at the operation–object level, namely, the amount of new operation–object structure introduced beyond the original statement. If multiple candidates introduce the same amount of operation–object expansion, we further ask the model to decide which candidate is more helpful for solving the original question. Based on the SQL query patterns and functions discussed by (Zhang et al., 2025d), we summarizes the SQL operator vocabulary and the corresponding object-level extraction used in our implementation in Table 11.

Rule coverage and final selection. Table 12 summarizes the Reverser's candidate pool and final selections across categories. All eight template categories generate candidates and contribute to the final selections on both datasets, collectively accounting for 87.27% of final selections on WikiTQ and 86.04% on TAT-QA. The remaining selections, listed as *Additional candidates*, come from our LLM-based fallback mechanism, which is invoked when the selection criteria cannot establish a clear preference among the template-generated candidates.

The contribution of individual categories varies across datasets. Uniqueness negation and time/place/value contradiction contribute the

Operator	Detected from	Object-level signature example
SELECT	SELECT ... FROM ... clause	SELECT::height, SELECT::max(height)
WHERE	WHERE clause	WHERE::country = 'Germany'
GROUP_BY	GROUP BY clause	GROUP_BY::country
HAVING	HAVING clause	HAVING::count(*) > 1
ORDER_BY	ORDER BY clause	ORDER_BY::height DESC
LIMIT	LIMIT clause	LIMIT::1
DISTINCT	DISTINCT in SELECT expression	DISTINCT::distinct country
COUNT	aggregation call COUNT(...)	COUNT::*, COUNT::height
SUM	aggregation call SUM(...)	SUM::sales
AVG	aggregation call AVG(...)	AVG::score
MIN	aggregation call MIN(...)	MIN::year
MAX	aggregation call MAX(...)	MAX::height
CASE	CASE WHEN or IF(...) expression	CASE::case
EXISTS	EXISTS clause	EXISTS::exists
SUBQUERY	nested SELECT query	SUBQUERY::subquery

Table 11: SQL operator vocabulary and object-level signature construction used in reverse selection.

largest numbers of selected candidates on WikiTQ, whereas time/place/value contradiction accounts for more than half of the selections on TAT-QA. Among the eight template categories, tied extrema and uniqueness negation have the highest selection rates on WikiTQ, while time/place/value contradiction and uniqueness negation have the highest rates on TAT-QA. Together, these results demonstrate broad use of the template set across both benchmarks, with different categories contributing to different extents. The reported coverage and selection rates describe the selection process rather than the correctness of individual candidates.

D.2 The Rethinker

Using large language models as critics or judges has been shown to be effective, but it is also challenging in practice, as their feedback can be noisy, inconsistent, overconfident, or sensitive to prompt phrasing and context (Raina et al., 2024; Bavaresco et al., 2025; Haldar and Hockenmaier, 2025).As a result, we do not rely on the model's feedback alone. Instead, based on empirical validation, we adopt a hybrid design that combines model feedback with simple structural constraints. This design allows us to leverage the model's semantic judgments while preventing unstable or unconstrained revisions. Algorithm 2 provides the pseudocode description of this module.

We examine the joint effects of τ and τ_{swap} over $\tau \in \{0, 0.25, 0.5, \ldots, 2.0\}$ and $\tau_{\text{swap}} \in \{0, 0.25, 0.5, 0.75, 1.0\}$. As shown in Figure 6, WikiTQ accuracy generally improves as τ increases toward 1.5, particularly with moderate values of τ_{swap}, before leveling off or declining at larger thresholds. TabFact exhibits much smaller fluctuations, with a broad plateau over $\tau \in [0.25, 1.0]$ and limited sensitivity to τ_{swap}. This pattern is consistent with its binary answer space: the two reasoning branches disagree less frequently than on open-ended QA, so substantially fewer cases enter the Rethinker component.

To assess the contribution of this design, we compare Rethinker with a prompt-based variant that relies solely on model feedback. As shown in Table 13, Rethinker improves accuracy of 1.5 and 2.4 percentage points, respectively. These results support combining model feedback with structural constraints rather than relying on prompted judgments alone.

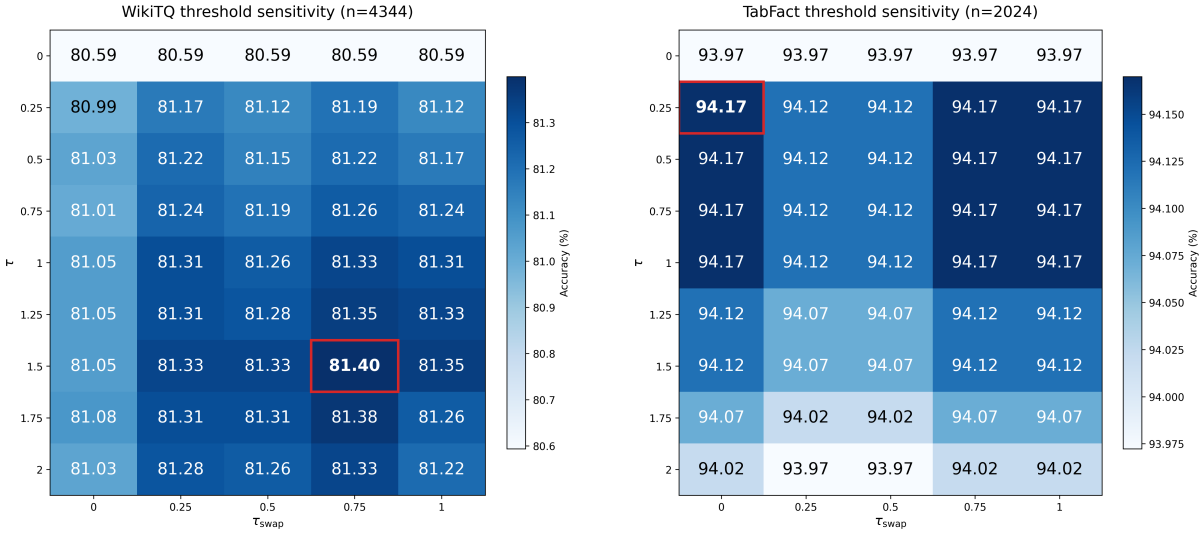
E Case Study and Prompts

We provide qualitative illustrations and implementation details for each step of the CRAFT. It is organized to support a clearer understanding of how the proposed method operates, how its components interact, and how its behavior differs from that of prior reflection-based approaches.

We first present a detailed case study 7 that traces the reasoning process of our method and contrasts it with that of a representative self-reflection–based baseline on the same example. By comparing the key intermediate steps under the two settings, the case study illustrates how counterfactual construction leads the model to explore alternative reasoning paths, and how inappropriate intermediate restrictions in prior approaches can result in incorrect conclusions, whereas the proposed framework reaches a consistent and correct outcome. In a separate example 8 , we further show that CRAFT can still approach the correct solution even when neither the Rewriter nor the Reverser statements

Category	WikiTQ		TAT-QA	
	Candidates → selected	SR	Candidates → selected	SR
Tied extrema	1,046 → 424	40.54%	174 → 54	31.03%
Uniqueness negation	2,644 → 1,063	40.20%	196 → 65	33.16%
Exclusivity reversal	219 → 83	37.90%	68 → 16	23.53%
Time/place/value contradiction	3,079 → 1,050	34.10%	2,013 → 670	33.28%
Comparison reversal	498 → 138	27.71%	237 → 42	17.72%
Lower-bound reversal	393 → 84	21.37%	47 → 10	21.28%
Upper-bound reversal	842 → 179	21.26%	128 → 33	25.78%
Exact-value negation	2,191 → 770	35.14%	925 → 213	23.03%
Additional candidates	1,381 → 553	40.04%	794 → 179	22.54%
Total	12,293 → 4,344	—	4,582 → 1,282	—

Table 12: Reverser candidate counts and final selections by category. SR denotes the selection rate, which is the proportion of candidates selected within each category.



(a) WikiTQ (b) TabFact

Figure 6: Sensitivity to τ and τ_{swap} on WikiTQ and TabFact. Red boxes mark the best result within each displayed grid.

Method	WikiTQ (%)	TabFact (%)
Prompt-based	79.9	91.7
RETHINKER	81.4	94.1
Δ	+1.5	+2.5

Table 13: Ablation study results comparing RETHINKER with a purely model-based prompting baseline on WikiTQ and TabFact tasks, using Llama 3 as the backbone model.

include the gold answer. CRAFT leverages the partial but complementary clues uncovered along the two reasoning paths. This allows CRAFT to filter out less reliable candidates, retain the more informative comparisons, and ultimately recover the correct answer through the final evidence aggregation stage.

Following this, we report the exact prompts used for all modules in CRAFT to ensure transparency and reproducibility. Specifically, we provide the prompts for the Rewriter, Reverser, Extractor, and Rethinker components (see Figure 9 – Figure 12). Since the intermediate reasoning traces produced by baseline methods can be directly used as evidence inputs, we only include the Extractor prompt that maps a given piece of evidence to a final answer. This prompt defines how evidence is interpreted and converted into a prediction.

Algorithm 2 Rethinker Component

Require: Question Q, table T; Rewriter (a_A, e_A); Reverser (a_B, e_B); fallback answer a_{base}; thresholds τ, τ_{swap}.

Ensure: Selected answer $\hat{a} \in \{a_A, a_B\}$.

Function SCORE(Q, T, a, e):
$S \leftarrow$ TOSTATEMENT(Q, a)
$(y, \alpha) \leftarrow$ LLMJUDGE(S, e, T) ▷ $y \in \{\mathsf{Support}, \mathsf{Contradict}\},\ \alpha \in [0, 1]$
if $y = \mathsf{Support}$ **then**
    return $+\alpha$
else
    return $-\alpha$
end if

Main-view margin (self-evidence)
$s_A \leftarrow$ SCORE(Q, T, a_A, e_A), $s_B \leftarrow$ SCORE(Q, T, a_B, e_B) ▷ $s \in [-1, 1]$
$\Delta \leftarrow |s_A - s_B|$
if $\Delta \geq \tau$ **then**
    return $\arg\max_{a \in \{a_A, a_B\}}\ s(a)$
end if
▷ If $\Delta < \tau$, Stage 1 is inconclusive and we enter Stage 2

Difference-aware re-evaluation
$d \leftarrow$ summary of differences between e_A and e_B
$s'_A \leftarrow$ SCORE$(Q, T, a_A, (e_A, d))$, $s'_B \leftarrow$ SCORE$(Q, T, a_B, (e_B, d))$
$\Delta_{\text{swap}} \leftarrow |s'_A - s'_B|$

Decision
if $\Delta_{\text{swap}} \geq \tau_{\text{swap}}$ **then**
    return $a_{\arg\max_{v \in \{A, B\}} s'_v}$
else
    return a_{base}
end if

player	nationality	position	years for jazz	school / club team
rick adelman	united states	guard	1974 - 75	loyola (ca)
john amaechi	england	center / forward	2001 - 03	penn state
louis amundson	united states	forward	2007	unlv
j j anderson	united states	forward	1982 - 85	bradley
shandon anderson	united states	guard / forward	1996 - 99	georgia
rafael araújo	brazil	center	2006 - 2007	byu
carlos arroyo	puerto rico	guard	2002 - 05	florida international
isaac austin	united states	center	1991 - 93	arizona state
anthony avent	united states	forward	1998 - 99	seton hall

Statement: after 2000, only brazil and england contributed one player to the utah jazz

Rewrite Execution Trace

Step 1 — select_row (Critical)

Process: Select rows [2, 6] corresponding to England and Brazil.

✗ This step prematurely restricts the table to mentioned entities, removing potential counterexamples.

Step 2–4 — select_column → group_column → sort_column

Process: Filter relevant columns, group players by nationality, and order rows by years for jazz to enforce the temporal constraint (after 2000).

Step 5 — simple_query

Output: YES

Explanation (retained, compressed)

The subtable contains only two players after 2000—one from England (John Amaechi) and one from Brazil (Rafael Araújo). Since no other nationalities appear in the subtable, the model concludes that only Brazil and England contributed one player each.

Judge

Decision: Correct

Reason (explicit): The judgment is made with respect to the restricted subtable, where the exclusivity condition imposed by "only" is satisfied.
However, this conclusion does not hold for the full table, which contains additional players from other nationalities after 2000

Reverse Execution Trace

Statement: after 2000, Not only brazil and england contributed one player to the utah jazz

Step 1 — select_row

Process: Select rows [2, 6, 7] corresponding to England, Brazil, and Puerto Rico.

✓Unlike the only case, row selection remains open, allowing additional nationalities beyond those explicitly mentioned.

Step 2–4 — select_column → group_column → sort_column

Step 5 — simple_query

Output: YES

Explanation (retained, compressed)

After 2000, the subtable includes players from England (John Amaechi), Brazil (Rafael Araújo), and Puerto Rico (Carlos Arroyo).
The presence of a third nationality satisfies the condition "more than Brazil and England".

Judge

Decision: Correct

Reason (explicit): The counterfactual statement requires only existence of additional nationalities, not exhaustive coverage
Since the subtable includes a valid counterexample (Puerto Rico) the judgment aligns with the semantics of "more than" and remains correct even under partial table execution.

Decision: Reverse

Reason:

The Rewrite trace makes an incorrect subtable restriction at row selection, filtering to only mentioned nationalities and injecting an implicit exclusivity. The subsequent self-reflection fails to reliably identify and undo this upstream filtering error, so the final judgment is based on a biased subtable.

By contrast, the Reverse trace keeps row selection open, so the execution remains semantically valid and robust.

Final Answer: No

Figure 7: A Case Study Comparing Self-Critique and Counterfactual Reasoning Paths

Name	Country	Town	height (metres / ft)	held record
great pyramid of giza	egypt	giza	146 / 480	2570 bc–1311
st. mary's church	germany	stralsund	151 / 500	1549–1647
strasbourg cathedral	germany and/or france (today france)	strasbourg	142 / 470	1647–1874
st nikolai	germany	hamburg	147.3 / 483	1874–1876
notre-dame cathedral	france	rouen	151 / 500	1876–1880
cologne cathedral	germany	cologne	157.4 / 516	1880–1884
washington monument	united states	Washington, d.c	169.3 / 555	1884–1889

Question: what building had the least height in germany?

Statement: St Mary's Church was the least height building in germany

Rewrite Execution Trace

Step 1 — select_row (Critical)

Process: Select rows [2, 4, 6] corresponding to Country = germany.

✗ This step prematurely restricts the table to only rows explicitly marked as Germany, excluding historically relevant or cross-border cases

Step 2–3 — select_column → sort_column

Process: Keep building names and height values, then sort the selected German buildings by height in ascending order.

Step 4 — simple_query

Output: No

Explanation (retained, compressed)

The filtered subtable contains only buildings whose country field is explicitly labeled as Germany. Within this restricted subtable, St Nikolai (147.3 metres) is lower than St Mary's Church (151 metres).So the model concludes that St Mary's Church is not the least-height building.

Judge

Decision: Correct

Reason (explicit): The judgment is locally correct under the restricted subtable, because St Nikolai is indeed lower than St Mary's Church among the rows retained after filtering.

Reverse Execution Trace

Statement: St Mary's Church was not the least height building in germany

Step 1 — select_row

Process: Select rows [2, 3, 4, 6] relevant to buildings in Germany, while leaving room for historically or semantically related cases.

✓Compare the candidate buildings and their heights, and look for any building lower than St Mary's Church.

Step 2–3 — select_column → sort_column

Step 4 — simple_query

Output: YES

Explanation (retained, compressed)

Considering the broader historical/geographical semantics, the relevant buildings include St Mary's Church (151 metres), St Nikolai (147.3 metres), and Strasbourg Cathedral (142 metres).

Judge

Decision: Correct

Reason (explicit): The counterfactual statement only requires a valid counterexample showing that St Mary's Church is not the minimum.

Since Strasbourg Cathedral provides such a counterexample, the reverse reasoning remains valid and robust even when the rewrite path is trapped by an early filtering error.

Decision: Reverse

Reason:

Because the question asks about buildings **in Germany**, we should consider a broader Germany-related scope. The Reverse trace preserves this better than the Rewrite trace.

Final Answer: Strasbourg Cathedral

Figure 8: A Case Study Showing how CRAFT get correct answer when Both Reasoning Paths start with a wrong answer

You are a table-aware rewriting assistant.

Input:
- A natural-language question Q.
- A table T given as rows (the first row is the header).

Task:
1) Compute answer A strictly from T using only the column(s) implied by Q , If you can't get answer, randomly select answer A following semantic. A must be either:
  (i) a value copied verbatim from a single table cell, or
  (ii) a minimal result of simple reasoning over the target column(s) (e.g., sum, difference, count, date difference, comparison).
2) Rewrite Q into one declarative sentence S that explicitly contains A.

Rules:
- Use only T; do not introduce external information.
- Entity answers: copy exactly one table cell (preserve casing, punctuation, and spacing).
- Numeric answers:
  * If copied, preserve the unit exactly as in T.
  * If computed, output digits only in A, and express the unit in S (from T or Q).
- If Q specifies a unit (e.g., "in kg"), do not repeat it in A, but keep it in S.
- Counting: A is a single number; do not enumerate items.
- OR-questions ("A or B"): return the selected option/relation as A (never Yes/No) and preserve the OR semantics in S.
- Binary questions ("Did X …?"): A may be "Yes" or "No"; reflect the polarity in S.
- Multiple values (if required): join with "|", each value must be a table cell.
- Time/date: return the exact token from T (no merging across columns).

Output:
Return JSON only: {"answer": "<string>", "statement": "<string>"}.

Question: {question}
Table: {table_text}

Figure 9: Rewriter Prompt Template.

You are a logically precise language model.

Given an Original Question and a Statement, write three one-sentence Counter that plausibly contradicts the Statement while being maximally useful for answering the Original Question.

Rules:
(1) Target the most misleading or critical claim in the Statement.
(2) Use the Question's intent to guide the contradiction; don't negate randomly.
(3) Reuse numbers / entities / places / times from the Question / Statement; do not invent specifics.
(4) Output exactly one concise grammatical sentence: no hedging, no explanations.

A reverse Template is already selected and provided.
The placeholders in the Template are the only slots you may fill.
You must strictly follow the given Template and the examples provided with that Template.

Now, consider the following:
Original Question: "{original_question}"
Statement: "{statement}"
Template: "{template}"
Counter:

You are an expert SQL planner for SINGLE-TABLE fact verification.

Given statement S and available column names, write the MINIMAL SQL query
that is sufficient to verify whether S is true on table " {table_name} ".
- Do not use columns outside the provided header list.
- Prefer the shortest logically complete query.

Allowed operations vocabulary (closed set):
{sorted(ALLOWED_SQL_OPS)}

Output JSON only:

{{
"sql": "...",
"why_minimal": "one short sentence"
}}

Table name: {table_name}
Columns: {table_column}
Statement S: "{statement}"

Figure 10: Reverser Prompt Template.

You are a precise answering assistant.

Task
Extract the answer to the user's question, strictly following the rules below.

Assumption
The table is guaranteed to contain sufficient information to answer the question.
You must always produce an answer and must not refuse, abstain, or claim missing information.

Output Format
- Output ONLY the final answer string.
- If multiple values are required, join them with `|` (no spaces).
- Never output calculations, reasoning, or explanations.

Source Priority
1) Use the table if provided (exact cell tokens or computations over explicit table numbers only).
2) Use textual evidence only if the table does not contain the answer.

Question-Type Rules
- OR questions ("A or B"): output the chosen option text exactly (never Yes/No).
- Binary questions: output Yes/No only if the source explicitly contains Yes/No.
- Time/date questions: output the exact time token only.
- Location/event questions: output the location token only.
- Entity questions ("which/what"): output the exact entity string only.
- Duration questions ("how many days"): compute the inclusive count.
- Range count questions ("rows X to Y"): output the inclusive count.

Copying and Computation
- Copy strings verbatim (case, punctuation, symbols, spacing).
- If an entity contains parentheses, drop the parenthesized part.
- Remove non-semantic prefixes (row/index/reference markers).
- Use abbreviations only if shown in the source.
- If copied: keep units exactly as shown.
- If computed: output digits only, without units.
- Use only numbers explicitly present in the source.

Final Check Before Output
- The output type matches the question (time vs. location vs. entity vs. number).
- No extra tokens are included (no units on computed numbers, no time in locations, etc.).

Figure 11: Extractor Prompt Template.

You are a precise answering assistant.

You are an Evidence-Checker Agent for table-based fact verification.

Inputs ({side_tag}): STATEMENT S={question}; EVIDENCE={evidence}; TABLE={table_str}. The TABLE is ground truth; the EVIDENCE is auxiliary and may be incomplete or incorrect.

Task:
- Determine what the TABLE implies about the STATEMENT and how well the EVIDENCE reflects that implication.
- If the STATEMENT contains "|", evaluate each value separately.
- Set statement_truth as "true" (TABLE supports S), "false" (TABLE contradicts S), or "unclear" (TABLE insufficient).
- Set coverage as "full" (EVIDENCE reflects most key facts), "partial" (some but misses important), or "minimal" (barely overlaps).
- Set strength (0.0–1.0) as your confidence in statement_truth; conflicts between EVIDENCE and TABLE may reduce it.

Rules: always trust TABLE over EVIDENCE; treat conflicting EVIDENCE parts as unreliable; do not use external knowledge beyond S, EVIDENCE, and TABLE; do not answer the original question.

Output (exactly one JSON line):
{"statement_truth":"true|false|unclear","strength":0.0-1.0,"coverage":"full|partial|minimal","explanation":"brief justification based on TABLE and EVIDENCE"}

Figure 12: Rethinker Prompt Template.