

Escaping the Ambiguity Trap: Content-Level Evidence Tracing for Open-Domain Table QA

Yibin Liu¹, Yewang Chen^{1,*}, Zhe Li^{1,*}

¹ College of Computer Science and Technology, Huaqiao University, Xiamen, China

23014083040@stu.hqu.edu.cn **Correspondence:** *ywchen@hqu.edu.cn, 24013083016@stu.hqu.edu.cn

Abstract

Open-domain table QA requires reliable evidence localization beyond candidate retrieval. Compact table fingerprints reduce irrelevant signals during representation learning, but retrieval based on learned representations may still have limited ability to distinguish evidence segments among retrieved candidates. This issue is especially pronounced when tables are partitioned into segments that share similar metadata and schema patterns. We propose **Probe-Grounded Reasoning (PGR)**, a coarse-to-fine framework for content-level evidence localization. PGR generates query-specific probes with metadata cues and cell-level evidence, then reranks candidate segments to improve fine-grained evidence localization. It then derives answers via a decoupled Semantic-Plan-Execute workflow with GRPO-optimized planning and execution. On Open-WikiTable, under a controlled evaluation setting with fixed BM25-retrieved candidates, PGR achieves 85.96% segment-level evidence accuracy at $k = 5$ and reaches 69.0% end-to-end QA accuracy, suggesting that content-level verification supports more reliable evidence grounding for downstream reasoning. Code is available at <https://github.com/XFastDataLab/PGR/>.

1 Introduction

Open-domain Table Question Answering (Table QA) requires effective evidence retrieval from large-scale table corpora to support grounded reasoning. While Large Language Models (LLMs) have significantly advanced the reasoning stage within *retrieve-then-reason* pipelines, overall end-to-end performance remains fundamentally constrained by retrieval granularity.

In open-domain table question answering, large-scale tables are often partitioned into multiple segments for processing. Effective table retrieval requires balancing semantic coverage and representation efficiency, as overly detailed representations

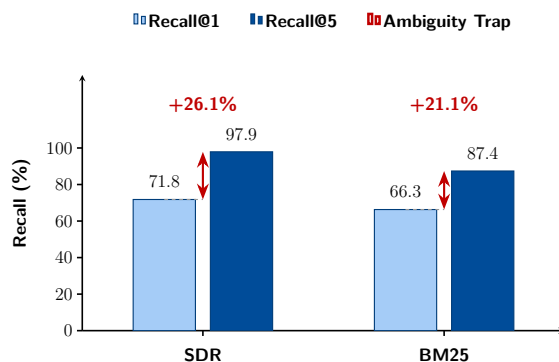


Figure 1: The **ambiguity trap** in partitioned table retrieval under single-ground-truth scenarios. The large Recall@1 vs. Recall@5 gap on Open-WikiTable indicates that high-recall retrieval may still leave fine-grained segment localization challenging.

may introduce redundant information and increase complexity. This motivates compact table fingerprints for retriever training, reducing representation complexity while retaining essential table signals.

However, high-recall retrieval alone does not guarantee fine-grained evidence localization. This limitation is particularly evident in Open-WikiTable, where approximately 52% of tables are partitioned into multiple segments. Under our SDR retrieval setting, single-ground-truth scenarios show a substantial top-1/top-5 recall gap (Figure 1): top-1 recall is 71.8%, compared with 97.9% at top-5, indicating that the relevant evidence segment is often included in the retrieved candidate set but not ranked first. When large tables are split row-wise into segments, sibling segments may share similar metadata and schema patterns, making fine-grained evidence localization challenging. Although retrievers trained with compact fingerprints achieve high recall, their learned representations may have limited ability to distinguish closely related candidates. This motivates content-level verification beyond retrieval-based ranking.

We propose a **coarse-to-fine verification strategy** to improve evidence localization under segment-level ambiguity. We employ a standard dense retriever optimized with compact table-level fingerprints for coarse global retrieval, and delegate fine-grained evidence localization to our **Probe-Grounded Reasoning (PGR)**. PGR implements a *verify-then-reason* workflow that first mitigates segment-level ambiguity and then performs grounded answer generation through two integrated stages: (1) Evidence Localization: PGR synthesizes query-specific probes containing metadata cues and cell-level evidence, using evidence-value matching to score and rank candidate segments. (2) Grounded Reasoning: Once the top-ranked segment is selected, PGR recovers the corresponding full table context and executes a *Semantic-Plan-Execute* (SPE) process to produce the final answer.

Direct code generation from raw tables can lead to compounded errors, as models must simultaneously handle semantic disambiguation, logical planning, and syntactic correctness. To address this, we design *SPE* as a decoupled workflow consisting of semantic enrichment, strategic planning, and program execution. This design improves reasoning controllability by separating intermediate stages and grounding execution on verified evidence. Finally, we optimize the planning and execution stages with Group Relative Policy Optimization to improve reasoning consistency.

Our contributions are as follows:

- **Coarse-to-Fine Strategy:** We propose a coarse-to-fine framework that mitigates the *Ambiguity Trap* in partitioned table QA settings by decoupling global retrieval from fine-grained content verification.
- **Probe-Grounded Reasoning (PGR):** We introduce **PGR**, a verify-then-reason framework that combines probe-based evidence localization with grounded answer generation through a decoupled Semantic-Plan-Execute process.
- **Empirical Validation:** Experiments on the **Open-WikiTable** benchmark show consistent improvements over existing baselines.

2 Related Work

Table Retrieval. Traditional sparse retrieval methods, such as BM25 (Robertson and Zaragoza, 2009), excel in lexical matching but falter in semantic scenarios due to vocabulary mismatch. Dense retrievers (Karpukhin et al., 2020) address this by

embedding queries and tables in a shared space. In open-domain Table QA, a key trade-off emerges between coverage and efficiency: encoding full tables is costly (Herzig et al., 2021; Lin et al., 2023), motivating compact abstractions such as metadata or sampled rows (Wang et al., 2022). While effective for global localization through compact table-level representations, such abstractions may provide insufficient granularity for distinguishing candidate segments that share similar metadata and schema patterns. Standard contrastive objectives with in-batch negatives are commonly used for dense retrieval training (Zhan et al., 2021), but they are not designed to explicitly address fine-grained ambiguity among retrieved candidates, motivating our content-level verification in PGR.

Program-Aided Open-Domain Table QA. In closed-domain settings, where the evidence table is given, many studies enhance LLM reasoning by generating executable programs such as SQL or Python for precise computation (Cheng et al., 2023; Chen et al., 2023). In open-domain settings, the evidence must first be retrieved from a large corpus of tables, often partitioned into segments, producing top- k candidates that may contain structurally similar distractors with similar schemas. This retrieval ambiguity motivates an additional reranking stage to improve evidence segment localization. OpenTab (Kong et al., 2024) extends the program-aided paradigm to open-domain settings by generating SQL queries for each retrieved segment and using execution results for reranking and reasoning. While effective in disambiguating retrieved candidates, this approach requires multiple LLM calls and program executions for each candidate. In contrast, our approach performs probe-based content verification with a single LLM call for probe generation, improving evidence localization among candidate segments.

Decoupled Reasoning Frameworks. Direct program generation couples semantic disambiguation, planning, and execution, which can lead to compounded errors in complex tables. Recent works advocate decoupling to address this. ReWOO (Xu et al., 2023) separates reasoning from tool observations to decrease latency. Program of Thoughts (Chen et al., 2023) disentangles computation from reasoning. In deep search settings, HiRA (Jin et al., 2025a) explores a hierarchical decoupling of planning and execution to improve efficiency. Related seek-and-solve paradigms have advanced iterative

reasoning over tables (Jiang et al., 2025), while decomposed text-to-SQL approaches separate intermediate reasoning steps from final query generation to improve complex query solving (Pourreza and Rafiei, 2023). Building on these ideas, our SPE framework decouples reasoning into semantic enrichment (normalizing headers), strategic planning (isolating logic), and program execution over the original table grounded in verified evidence. This decoupled strategy improves reasoning controllability and grounds execution on verified evidence, enhancing end-to-end performance in open-domain tabular QA scenarios.

Reinforcement Learning for Reasoning. Aligning LLMs with tasks often uses RL like PPO (Schulman et al., 2017), but instability from critic networks poses challenges. Outcome-based methods like GRPO (Shao et al., 2024) normalize rewards group-wise, proving effective in verifiable tasks like math and code. Recent studies have further extended this paradigm to grounded generation, utilizing verifiable rewards to enhance factual consistency and grounding (Sim et al., 2025). In Table QA, RL-based optimization is increasingly employed to refine reasoning trajectories. Various frameworks have explored diverse directions, including inference-time scaling for complex reasoning (Yang et al., 2025), region-based RL for fine-grained table understanding (Wu et al., 2026), and program-based reasoning tailored for small language models (Jin et al., 2025b). Our SPE workflow leverages this RL-based refinement to improve factual and logical consistency.

3 Methodology

3.1 Overall Framework

We approach open-domain Table QA using a coarse-to-fine strategy that deliberately balances denoising with fine-grained evidence resolution, as illustrated in Figure 2. The pipeline consists of two tightly integrated stages.

Stage 1: Global Localization. Given a question Q and the full corpus $\mathcal{T}$, a standard dense retriever identifies a ranked candidate set $\mathcal{T}_{\text{cand}}$ consisting of the top- K table segments. The retriever is trained with compact table-level fingerprints containing hierarchical metadata, column headers, and the top-2 sample rows, reducing irrelevant signals during representation learning while preserving coarse-grained table semantics. At inference, the learned

retriever performs high-recall candidate retrieval over table segments.

Stage 2: Probe-Grounded Reasoning (PGR). To mitigate segment-level ambiguity among retrieved candidates with similar structural signals, PGR adopts a verify-then-reason workflow:

- **Evidence Localization** ($g_{\text{probe}}, f_{\text{rerank}}$): PGR generates a query-specific probe P_Q in a single LLM call conditioned on Q and the representations of all candidates in $\mathcal{T}_{\text{cand}}$. Candidates sharing identical metadata are grouped within a unified prompt context, with their schema information merged. It then deterministically matches P_Q against candidate metadata and cell values, producing the top-ranked segment s^* through content-level verification.
- **Grounded Reasoning** (f_{exec}): The selected segment s^* is expanded to its parent source table T^* , which provides the complete context for downstream reasoning. PGR then derives the final answer A via a Semantic-Plan-Execute (SPE) workflow. To improve answer reliability over direct generation, SPE adopts a decoupled pipeline consisting of semantic enrichment, strategic planning, and execution grounded in verified evidence. This structured approach enables reliable reasoning while preserving access to the complete table context.

Formally, let $\mathcal{M}$ and $\mathcal{S}$ denote the metadata and schemas of candidate segments in $\mathcal{T}_{\text{cand}}$, respectively. PGR first selects the top-ranked candidate segment s^* , maps it back to its parent source table T^* , and then produces the final answer A as:

$$P_Q = g_{\text{probe}}(Q, \mathcal{M}, \mathcal{S}) \quad (1)$$

$$s^* = f_{\text{rerank}}(P_Q, \mathcal{T}_{\text{cand}}) \quad (2)$$

$$T^* = \text{parent}(s^*) \quad (3)$$

$$A = f_{\text{exec}}(Q, T^*) \quad (4)$$

This coarse-to-fine design separates broad candidate retrieval from fine-grained evidence verification: Stage 1 performs high-recall segment retrieval with a retriever optimized using compact table-level representations, while Stage 2 refines candidate evidence through content-level verification within a decoupled reasoning workflow.

3.2 Candidate Retrieval

In our framework, the retrieval stage narrows down the full corpus to a manageable set of candidate

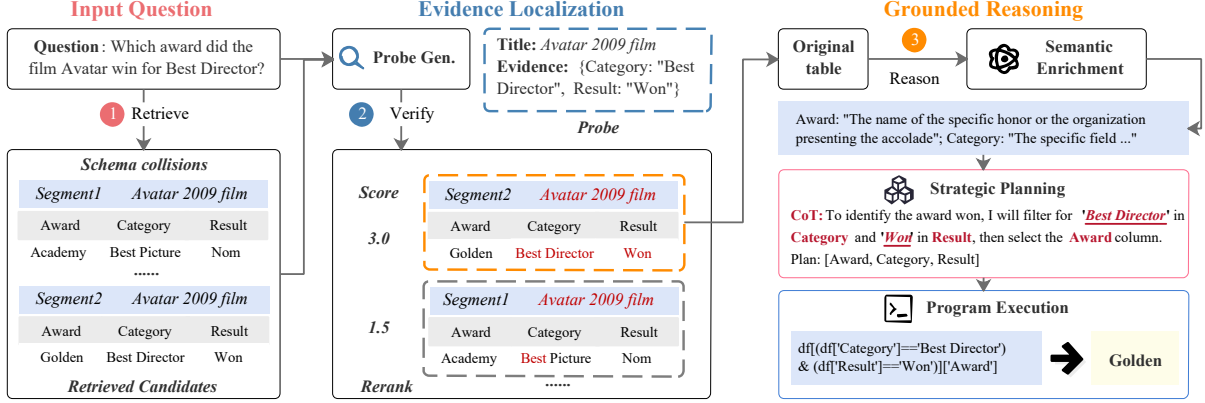


Figure 2: **Overview of the end-to-end QA pipeline.** The framework operates in two integrated stages. In **Stage 1 (Global Localization)**, candidate retrieval obtains top- K segments, where closely related candidates may share similar metadata and schema patterns. In **Stage 2 (Probe-Grounded Reasoning)**, probe verification performs evidence localization, followed by answer generation via the Semantic-Plan-Execute (SPE) workflow.

segments while maintaining high recall. We train the retriever with a table-level contrastive learning objective, where training instances are constructed at the table level to simplify optimization and focus on coarse-grained matching.

We use a compact “fingerprint” serialization for retriever training: each table is represented by its hierarchical metadata (Page Title, Section Title, and Caption), column headers, and only the top-2 sample rows. This compact representation reduces irrelevant content during representation learning and focuses optimization on query-relevant signals.

During training, we optimize the standard InfoNCE loss with in-batch negatives:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{e^{(e_Q \cdot e_T)/\tau}}{\sum_{T' \in \text{batch}} e^{(e_Q \cdot e_{T'})/\tau}}, \quad (5)$$

where e_Q and e_T are the embeddings of the question and the table fingerprint, respectively, and τ serves as the temperature parameter that regulates the softness of the resulting probability distribution.

This training objective learns coarse-grained table matching. At inference, applying the encoder to segment serializations yields a high-recall candidate set $\mathcal{T}_{\text{cand}}$ consisting of top- K retrieved segments. PGR further ranks candidate segments through content-level verification.

3.3 Probe-Grounded Reasoning

To refine candidate selection and improve evidence localization under segment-level ambiguity among retrieved candidates, we introduce **Probe-Grounded Reasoning (PGR)**. PGR complements

structural cues with probe-based content verification, adopting a *verify-then-reason* workflow for evidence localization and grounded reasoning.

3.3.1 Evidence Localization

The core of evidence localization is a query-specific probe P_Q that translates the question intent into verifiable constraints. Formally, $P_Q = (c_{\text{meta}}, C_{\text{content}})$, where c_{meta} encodes expected metadata fields (page title, section, caption) and $C_{\text{content}} = \{(h_i, v_i)\}$ associates column headers h_i with expected cell values v_i .

P_Q is synthesized by a generator g_{probe} within a single LLM call, conditioned on Q and candidate segment representations. To reduce prompt redundancy, candidates sharing identical metadata are grouped into a unified input, with their schema information merged, allowing g_{probe} to generate a shared probe set for multiple segments. The generated probes are then applied to individual candidates for content-level verification and reranking. During training, g_{probe} is supervised using automatically constructed evidence paths derived from available SQL annotations via rule-based parsing.

Once generated, PGR executes a deterministic tracing procedure over $\mathcal{T}_{\text{cand}}$:

- Compute metadata alignment score s_{meta} . We enforce a **subset constraint**: $s_{\text{meta}} = 1.0$ if c_{meta} is non-empty and all its tokens appear in the candidate’s metadata context, otherwise 0.0.
- For candidates passing this filter ($s_{\text{meta}} > 0$), we proceed to evaluate every individual row r . For each constraint (h_i, v_i) , we calculate

the **token recall score** of v_i against the corresponding cell value. These scores are summed to yield the row confidence s_r .

- We define the final aggregated table score as:

$$s_{\text{total}} = s_{\text{meta}} + \max_r s_r, \quad (6)$$

where s_{meta} represents the metadata alignment score and $\max_r s_r$ is the highest confidence found among the rows.

The segment s^* with highest s_{total} is selected, with retriever rank used to break ties. Max pooling aggregates row-level evidence into a segment-level ranking score. The full algorithm is in Appendix B.

3.3.2 Grounded Reasoning via SPE

Once the top-ranked candidate segment s^* is selected, PGR expands it to its parent table T^* and derives the final answer A . Direct program generation from raw tables can suffer from compounded errors, as the model must simultaneously handle semantic disambiguation of cryptic headers, logical planning across operations, and syntactic correctness. To improve reasoning controllability and ground the execution process on verified evidence, we propose the **Semantic-Plan-Execute (SPE)** framework, which systematically decouples the reasoning process into three modular steps:

Step 1: Semantic Enrichment. Cryptic or domain-specific column headers are normalized by synthesizing a functional definition d_j for each header h_j using a strong teacher model. This produces a semantic map $M = \{(h_j, d_j)\}$, integrated via SFT to aid subsequent interpretation.

Step 2: Strategic Planning. The strategic planner utilizes the question Q and the semantic map M to synthesize the following outputs:

- **Working Out:** A Chain-of-Thought reasoning path that analyzes the underlying logical dependencies inherent in the query.
- **Structured Plan:** A minimal column subset $\{h'_1, \dots, h'_m\}$ selected specifically to guide the program execution.

This decoupling effectively separates logical reasoning from implementation details and produces a minimal relevant column subset, enabling the subsequent code generation to focus on syntactic correctness using a compact and noise-reduced table representation.

Step 3: Program Generation and Execution.

An executable Pandas script is generated using an **Adaptive Contextual Sampling** of T^* as the generation context, while the generated program is executed over the complete source table T^* :

- **Column Pruning:** Retain only planned columns and those in P_Q .
- **Row Rerank:** Rows are reranked according to their alignment with the verified cell values $\{v_i\}$ from the probe P_Q , promoting evidence-containing rows to higher positions.

By prioritizing evidence-containing rows and relevant columns in the generation context, this adaptive sampling reduces irrelevant signals during program generation.

Outcome-Oriented Optimization. We train our framework in two phases: (1) all components undergo SFT using SQL-derived probes/plans and teacher-distilled semantic maps/code. Specifically, we parse the SQL annotations provided in OpenWikiTable to construct strategic plans and content verification probes for the corresponding QA pairs. (2) the planner and executor are further aligned via Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with group-wise rewards derived from Exact Match outcomes (reward details in Appendix C).

By first verifying evidence deterministically and then reasoning in a decoupled, evidence-aligned manner, PGR mitigates the ambiguity trap and improves reasoning reliability.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our framework primarily on the **Open-WikiTable** benchmark (Kweon et al., 2023), which integrates decontextualized and paraphrased questions from WikiSQL (Zhong et al., 2017) and WikiTableQuestions (WikiTQ) (Pasupat and Liang, 2015). Large tables are split row-wise into overlapping segments, resulting in approximately 52% of tables being distributed across multiple distinct segments. This setting naturally creates cases where candidate segments from the same source table inherit the same metadata and schema, leading to segment-level ambiguity and motivating content-level evidence disambiguation.

Training Set Construction. We construct a curated training corpus from the Open-WikiTable

training split, comprising WikiSQL and WikiTQ. For the WikiSQL subset, we apply a difficulty-aware stratified sampling strategy, while retaining all WikiTQ samples. The resulting corpus is further partitioned into Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) subsets for subsequent training. Detailed statistics and sampling rates are provided in Appendix D.

Evaluation Settings. We evaluate our framework under two open-domain settings: controlled reasoning evaluation and full-pipeline evaluation.

- **Controlled Reasoning Evaluation:** We follow the evaluation protocol established by OpenTab (Kong et al., 2024) and adopted by CoQ (Sui et al., 2025). BM25 provides the same top- K candidates for all methods as input to our PGR. The evaluated baselines include **Few-Shot QA** (Min et al., 2024), **Chain-of-Table** (Wang et al., 2024), **Basic Text-to-SQL** (Rajkumar et al., 2022), **OpenTab** (Kong et al., 2024), **MAG-SQL** (Xie et al., 2024), and **CoQ**.
- **Full-Pipeline Evaluation:** We evaluate our complete pipeline with varying numbers of retrieved candidates. For different K values, the pipeline uses the corresponding top- K retrieved segments and their parent tables as the evidence context.

Experimental Protocol. The controlled evaluation uses fixed BM25-retrieved candidates shared across methods to ensure a consistent retrieval condition. The full-pipeline evaluation uses SDR retrieval followed by the complete PGR workflow for end-to-end QA evaluation.

Metrics. For retrieval, we report **Recall@ K** ($K \in \{1, 5, 10, 20\}$). For probe-based evidence localization, we report **Segment-level Evidence Accuracy** and **Table-level Evidence Accuracy**. A top-ranked candidate is counted as correct if its segment or source table belongs to the corresponding ground-truth set. For end-to-end QA, we use **Execution Accuracy (Acc)**, measured by executing generated programs (SQL or Pandas) and comparing the resulting answers with ground truth. We apply answer normalization before comparison, including mapping None/NaN, empty strings, and empty containers to "none", and converting single-element lists to scalars. We follow OpenTab’s official evaluation script for consistent execution.

Implementation Details. The Standard Dense Retriever (SDR) uses a BERT-base backbone trained with in-batch negatives. The PGR is initialized with Qwen3-8B and trained through a two-stage process consisting of supervised fine-tuning (SFT) followed by outcome-oriented alignment with GRPO. All experiments are conducted on a single NVIDIA RTX 4090 (24GB) GPU using the Unsloth library¹. Additional training details are provided in Appendix E.

4.2 Main Results

Table Retrieval Performance. Table 1 and Table 2 summarize retrieval performance on the OpenWikiTable validation set. We report both *Full Recall@ K* (all ground-truth segments retrieved) and *Hit Recall@ K* (at least one segment retrieved). Results are further decomposed into *Single-GT* and *Multi-GT* subsets.

Our Standard Dense Retriever (SDR) achieves an Overall Full Recall@5 of 96.6%, compared with the reported DPR-BERT baseline performance of 87.0%. Its Overall Hit Recall@5 reaches 98.0%. These results suggest that training with compact table-level fingerprints enables effective global table localization in this retrieval setting.

Analyzing the Ambiguity Trap. Despite achieving high overall recall, a substantial gap exists in *Full Recall*, with R@1 achieving 61.1% and R@5 reaching 96.6%. Decomposing results into Single-GT and Multi-GT subsets provides further insight into this gap: in Single-GT scenarios, this gap reaches 0.261. Since only one segment is required as evidence in these queries, this gap provides an indicator of segment-level ambiguity among retrieved candidates: retrieved candidates may contain multiple segments from the same source table, where the evidence-bearing segment is not always ranked first among structurally similar sibling segments with shared metadata and schema patterns. This observation motivates our PGR, which performs fine-grained content-level verification to improve evidence localization among candidate segments.

End-to-End QA Performance. We evaluate QA performance under open-domain settings. For probe-based evidence localization, we report both segment-level and table-level evidence accuracy.

Open-domain Evaluation. 1) *Controlled Reasoning Evaluation.* As shown in Table 3, PGR out-

¹<http://github.com/unslothai/unsloth>

Table 1: Retrieval performance (Full Recall@K) on Open-WikiTable validation set. Full R@1 for Multi-GT is 0.000 because the metric requires retrieving *all* ground-truth segments. DPR-BERT* results are as reported in Kong et al. (2024).

Category	Method	R@1	R@5	R@10	R@20
Overall	DPR-BERT*	-	0.870	0.933	0.975
	BM25 (repro.)	0.565	0.832	0.893	0.940
	SDR (Ours)	0.611	0.966	0.986	0.993
<i>Single-GT</i>	BM25 (repro.)	0.663	0.874	0.922	0.960
	SDR (Ours)	0.718	0.979	0.989	0.995
<i>Multi-GT</i>	BM25 (repro.)	0.000	0.589	0.722	0.828
	SDR (Ours)	0.000	0.890	0.964	0.987

Table 2: Retrieval performance (Hit Recall@K) on Open-WikiTable validation set.

Category	Method	R@1	R@5	R@10	R@20
Overall	BM25 (repro.)	0.671	0.877	0.922	0.959
	SDR (Ours)	0.740	0.980	0.990	0.994
<i>Single-GT</i>	BM25 (repro.)	0.663	0.874	0.922	0.960
	SDR (Ours)	0.718	0.979	0.989	0.995
<i>Multi-GT</i>	BM25 (repro.)	0.714	0.890	0.919	0.951
	SDR (Ours)	0.867	0.987	0.991	0.994

performs all baselines under various top- K settings. As K increases from 1 to 10, PGR’s execution accuracy improves from 0.579 to 0.716, consistent with the benefit of having a larger candidate pool for evidence localization. Under the same settings, PGR achieves 90.35% segment-level evidence accuracy and 92.88% table-level evidence accuracy at $K = 10$. These results suggest that content-level verification helps identify relevant evidence among retrieved candidates, providing more reliable evidence grounding for downstream reasoning.

2) *Full-Pipeline Evaluation.* Table 4 summarizes the end-to-end performance of our proposed framework under the full-pipeline setting. With SDR retrieval and the complete PGR workflow, the framework achieves an execution accuracy of 74.90% on the Test set with $K = 5$. Increasing K from 1 to 5 improves segment-level evidence accuracy from 76.34% to 93.99% and QA accuracy from 73.92% to 74.90%. These results indicate that PGR improves evidence localization among retrieved candidates. Given the strong table-level localization already achieved by SDR, the corresponding QA improvement is relatively smaller.

Overall, these results show that combining denoising-oriented retrieval, content-level verification, and decoupled reasoning achieves effective evidence localization under segment-level ambigu-

Table 3: Controlled Reasoning Evaluation: End-to-End QA Accuracy on the Open-WikiTable test dataset. All methods follow the identical evaluation protocol (Kong et al., 2024; Sui et al., 2025), using the exact same top- K candidate segments retrieved by BM25. Baseline results are from Sui et al. (2025).

Method	Top-1	Top-2	Top-5	Top-10
<i>Baselines (from Sui et al. (2025))</i>				
Few-Shot QA	0.336	0.357	0.383	0.388
Chain-of-Table	0.404	0.430	0.463	0.467
Basic Text-to-SQL	0.429	0.458	0.490	0.496
OpenTab	0.491	0.523	0.556	0.565
MAG-SQL	0.457	0.476	0.516	0.524
CoQ	0.527	0.552	0.592	0.608
Ours				
Segment-level Evidence Acc	66.92	77.89	85.96	90.35
Table-level Evidence Acc	72.22	81.66	89.02	92.88
PGR	0.579	0.640	0.690	0.716

Table 4: Full-pipeline evaluation of the proposed framework on the Open-WikiTable dataset. K denotes the number of retrieved segments.

Metric	Validation				Test			
	K=1	K=5	K=10	K=20	K=1	K=5	K=10	K=20
<i>Evidence Localization</i>								
Segment-level Evidence Acc.	73.96	92.14	92.67	92.17	76.34	93.99	94.35	94.15
Table-level Evidence Acc.	94.83	97.30	97.24	96.89	96.71	98.17	98.02	97.83
<i>End-to-End QA Accuracy</i>								
SDR+PGR	72.46	73.58	73.80	73.22	73.92	74.90	74.72	74.81

ity and strong end-to-end QA performance. These findings suggest that the proposed framework provides more reliable evidence grounding for downstream reasoning.

4.3 Ablation Studies and Analysis

We conduct ablation studies to analyze the contribution of the optimization strategy, reasoning workflow, and probe-guided table refinement.

Impact of GRPO Optimization. To evaluate the contribution of GRPO, we conduct ablation studies on the Open-WikiTable validation set with identical Top-10 candidate segments retrieved by SDR and the same probe-based evidence localization pipeline. The only difference between variants is the optimization strategy of the reasoning model: supervised fine-tuning (SFT) versus SFT followed by GRPO. We report set exact match (Set-EM), plan-level F1 (P-F1), execution success rate (Exec.), logic error rate (Err.), and final QA accuracy (QA), where Set-EM measures order-invariant exact plan matching and Err. is the wrong-answer rate over successful executions where the selected segment is in the gold segment set.

As shown in Table 5, removing GRPO consis-

Table 5: GRPO ablation. All runs use identical Top-10 retrieved candidates and probe-based evidence localization.

Variant	Set-EM	P-F1	Exec	Err	QA
SFT	0.695	0.891	0.950	0.304	0.633
SFT+GRPO	0.816	0.929	0.994	0.229	0.738
Δ	+0.121	+0.038	+0.044	-0.076	+0.105

tently degrades reasoning and execution performance. Compared with SFT-only training, GRPO improves plan generation quality by 12.1 points in Set-EM and 3.8 points in P-F1, while increasing execution success rate by 4.4 points and reducing logic error rate by 7.6 points. These improvements translate into a 10.5-point absolute gain in end-to-end QA accuracy, from 63.3% to 73.8%, indicating that GRPO improves both intermediate reasoning quality and final execution reliability under the same evidence context.

Ablation of SPE Workflow and Probe-guided Table Refinement on Open-WikiTable. To analyze the contribution of different reasoning stages and probe-guided table refinement, we perform controlled ablations under the closed-domain setting on the Open-WikiTable test set. All variants use the same backbone and evaluation protocol. We compare three reasoning variants: SPE (full), PE (without semantic enrichment), and E (without semantic enrichment and strategic planning). Each variant is evaluated with and without probe-guided table refinement, which consists of probe-based column pruning and row-level reranking.

Table 6 shows that Probe-guided Table Refinement improves QA accuracy across all reasoning variants. Plan metrics are fixed across refinement settings since the same plans are reused; the observed gains mainly reflect differences in execution contexts. SPE obtains slightly higher plan metrics than PE, although their final QA performance remains comparable. Both SPE and PE outperform Executor-only execution in this setting, suggesting that intermediate reasoning stages provide useful context for downstream execution.

5 Conclusion

Partitioned table QA requires fine-grained evidence localization, as high-recall retrieval alone may leave relevant evidence under-ranked. In settings such as Open-WikiTable, sibling segments inherit the same metadata and schema, making

Table 6: Ablation study of the reasoning workflow and Probe-guided Table Refinement on Open-WikiTable test set. Plan metrics are unchanged across refinement settings because generated plans are reused.

Method	Set-EM	P-F1	QA w/ Ref.	QA w/o Ref.
SPE	84.07	94.40	76.07	70.05
PE	81.94	93.72	76.23	69.71
E	—	—	71.24	62.77

evidence localization more challenging. We propose a coarse-to-fine framework that decouples global retrieval from fine-grained content verification. PGR performs probe-based localization and content-level verification to improve fine-grained evidence localization, followed by a decoupled SPE process for grounded answer generation.

Experiments on Open-WikiTable show that fingerprint-based retrieval achieves high recall, while segment-level ambiguity remains among retrieved candidates. Under a controlled evaluation setting with fixed BM25-retrieved candidates, PGR achieves strong evidence localization performance and improves end-to-end QA accuracy over existing reasoning baselines. These results demonstrate that content-level evidence grounding improves evidence localization under segment-level ambiguity in partitioned table QA.

Limitations

Our framework focuses on single-table reasoning, limiting its effectiveness for queries requiring cross-table aggregation; extending PGR to support multi-table assembly is a key direction for future work.

Moreover, the strategic planning and probe generation modules rely on the availability of structured SQL data, which constrains its applicability to domains lacking such supervision. In addition, the generated probes may occasionally contain hallucinations, which introduce noise into evidence localization and affect the accuracy of subsequent reasoning.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 62576143, 62476103, and in part by Xiamen Science and Technology Project under Grant 3502Z20251019.

During the preparation of this work, the authors used Gemini 3 in order to refine the structure of the text and enhance its academic clarity. After

using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

References

- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. [Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks](#). *Transactions on Machine Learning Research*.
- Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. [Binding language models in symbolic languages](#). In *The Eleventh International Conference on Learning Representations*.
- Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. [Open domain question answering over tables via dense retrieval](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 512–519, Online. Association for Computational Linguistics.
- Ruya Jiang, Chun Wang, and Weihong Deng. 2025. [Seek and solve reasoning for table question answering](#). In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.
- Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yang Zhao, Hongjin Qian, and Zhicheng Dou. 2025a. [Hira: A hierarchical reasoning framework for decoupled planning and execution in deep search](#). *Preprint*, arXiv:2507.02652.
- Rihui Jin, Zheyu Xin, Xing Xie, Zuoyi Li, Guilin Qi, Yongrui Chen, Xinbang Dai, Tongtong Wu, and Ghohamreza Haffari. 2025b. [Table-r1: Self-supervised and reinforcement learning for program-based table reasoning in small language models](#). *Preprint*, arXiv:2506.06137.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. [Opentab: Advancing large language models as open-domain table reasoners](#). In *International Conference on Learning Representations*, volume 2024, pages 1301–1317.
- Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. [Open-WikiTable : Dataset for open domain question answering with complex reasoning over table](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8285–8297, Toronto, Canada. Association for Computational Linguistics.
- Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. [An inner table retriever for robust table question answering](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.
- Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, and Qianren Wang. 2024. [Exploring the impact of table-to-text methods on augmenting LLM-based question answering with domain hybrid data](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 464–482, Mexico City, Mexico. Association for Computational Linguistics.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Mohammadreza Pourreza and Davood Rafiei. 2023. [Din-sql: Decomposed in-context learning of text-to-sql with self-correction](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36339–36348. Curran Associates, Inc.
- Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. [Evaluating the text-to-sql capabilities of large language models](#). *Preprint*, arXiv:2204.00498.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Foundations and Trends in Information Retrieval*, 4(1-2):1–174.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *Preprint*, arXiv:1707.06347.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Shang Hong Sim, Tej Deep Pala, Vernon Toh, Hai Leong Chieu, Amir Zadeh, Chuan Li, Navonil

- Majumder, and Soujanya Poria. 2025. [Lessons from training grounded llms with verifiable rewards](#). *Preprint*, arXiv:2506.15522.
- Songyuan Sui, Hongyi Liu, Serena Liu, Li Li, Soo-Hyun Choi, Rui Chen, and Xia Hu. 2025. [Chain-of-query: Unleashing the power of LLMs in SQL-aided table understanding via multi-agent collaboration](#). 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 957–986, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.
- Zhensheng Wang, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. 2025. [Retqa: a large-scale open-domain tabular question answering dataset for real estate sector](#). In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press.
- Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. [Table retrieval may not necessitate table-specific model design](#). In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 36–46, Seattle, USA. Association for Computational Linguistics.
- Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. [Chain-of-table: Evolving tables in the reasoning chain for table understanding](#). In *The Twelfth International Conference on Learning Representations*.
- Zhenhe Wu, Jian Yang, Zhongjiang He, Changzai Pan, Jiaheng Liu, Xianjie Wu, Yu Zhao, Shuangyong Song, Yongxiang Li, Zhoujun Li, and Xuelong Li. 2026. [Table-r1: Region-based reinforcement learning for table understanding](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 27374–27391, San Diego, California, United States. Association for Computational Linguistics.
- Wenxuan Xie, Gaochen Wu, and Bowen Zhou. 2024. [Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql](#). *Preprint*, arXiv:2408.07930.
- Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. [Rewoo: Decoupling reasoning from observations for efficient augmented language models](#). *Preprint*, arXiv:2305.18323.
- Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. 2025. [Table-r1: Inference-time scaling for table reasoning tasks](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20605–20624, Suzhou, China. Association for Computational Linguistics.
- Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. [Optimizing Dense Retrieval Model Training with Hard Negatives](#). In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 1503–1512.
- Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2sql: Generating structured queries from natural language using reinforcement learning](#). *Preprint*, arXiv:1709.00103.

A Table Linearization and Input Representation

Each table is represented together with its hierarchical metadata, including page titles, section titles, and captions. Table 7 shows an example.

Field	Content
Page Title	Avatar (2009 film)
Section Title	Accolades
Caption	List of awards and nominations received by Avatar
Schema	Award, Category, Result
Row 1	Academy Awards, Best Picture, Nominated
Row 2	Golden Globe, Best Director, Won

Table 7: Example representation of a table with hierarchical metadata.

The metadata, schema, and rows are serialized into a structured sequence of the form: {Page Title} {Section Title} {Caption} \n {Header} \n {Rows}. During training, the resulting sequence is used as input to the encoder:

Linearized Input Sequence

Avatar (2009 film) Accolades List of awards and nominations received by Avatar
Award	Category	Result
Academy Awards	Best Picture	Nominated
Golden Globe	Best Director	Won

B Formal Evidence Tracing Procedure

This section details the deterministic procedure for content-level Evidence Tracing and Reranking. This algorithm (Alg. 1) takes the structured query probe generated by the LLM and ranks candidate segments through metadata matching and cell-level verification across retrieved candidates.

C Reward Shaping Details

To guide the Group Relative Policy Optimization (GRPO) process, we design a multi-dimensional tiered reward system. This system provides dense feedback for both the *Strategic Planner* (Step 2) and the *Program Executor* (Step 3), penalizing format violations while incentivizing logical minimality and execution correctness.

C.1 Strategic Planning Rewards

The planning reward function R_{plan} evaluates the generated reasoning path alongside the corresponding column selection plan. In addition to the re-

Algorithm 1 Evidence Tracing (Verification Stage)

Require: Candidate segments $\mathcal{T}_{\text{cand}}$, Probe $P = (c_{\text{meta}}, C_{\text{content}})$, Initial scores S_{init}

Ensure: Target segment s^*

1: **Definitions:**

2: $\text{Tok}(x)$: set of non-empty whitespace tokens in string x

3: $\text{ovl}(v, c) = \frac{|\text{Tok}(v) \cap \text{Tok}(c)|}{|\text{Tok}(v)|}$ if $v, c \neq \emptyset$ else 0

4: **for each** $s \in \mathcal{T}_{\text{cand}}$ **do**

5: $s_{\text{meta}} \leftarrow \mathbb{I}(\text{Tok}(c_{\text{meta}}) \neq \emptyset \wedge \text{Tok}(c_{\text{meta}}) \subseteq \text{Tok}(s.\text{metadata}))$

6: $s_{\text{row}} \leftarrow 0$

7: **if** $s_{\text{meta}} = 1$ **then**

8: $s_{\text{row}} \leftarrow \max_{r \in s.\text{rows}} \sum_{(h,v) \in C_{\text{content}}} \text{ovl}(v, r[h])$

9: **end if**

10: $S_{\text{total}}(s) \leftarrow s_{\text{meta}} + s_{\text{row}}$

11: **end for**

12: **return** $\text{argmax}_{s \in \mathcal{T}_{\text{cand}}} (S_{\text{total}}(s), S_{\text{init}}(s))$

wards in Table 8, an auxiliary tag-counting reward is included, which assigns +0.5 for each required structural tag appearing exactly once, and -1.0 otherwise (missing or duplicated).

Condition	Score
Structural Format	
Exact format match (reasoning + <PLAN>)	+3.0
Format violation or missing tags	-2.0
Grounding Constraints	
Schema hallucination (non-existent columns)	-2.0
Logical Reasoning Quality	
Tier 1: Exact match with gold plan	+5.0
Tier 2: Correct column set, incorrect order	+3.5
Tier 3: Core order correct, with redundancy	$\max(0, 2.0 - 0.5N_{\text{red}})$
Tier 4: Core order wrong, with redundancy	$\max(0, 1.0 - 0.5N_{\text{red}})$
Tier 5: Missing essential evidence columns	-2.0

Table 8: Detailed reward shaping for Strategic Planning (g_{plan}). N_{red} denotes the count of redundant columns.

C.2 Program Execution Rewards

The execution reward function R_{exec} assesses the syntactic correctness and execution results of the

generated Python script.

Condition	Score
Structural Format	
Missing <CODE> block or essential df variable	−2.0
Missing required reasoning steps or duplicate end-tags	−2.0
Execution Policy	
Use of forbidden functions (e.g., print, .iterrows())	−1.0
Execution Outcome	
Success: Executes correctly and matches target answer	+4.0
Partial: Executes successfully but yields wrong answer	+2.0
Failure: Runtime error, syntax error, or missing output	−1.0

Table 9: Reward shaping for Program Execution (f_{exec}).

D Training Data Statistics

Sampling Strategy. We construct the probe training set using difficulty-aware stratified sampling on WikiSQL, retaining 4%, 10%, and 15% of Level 1–3 samples and all Level 4 samples, while fully retaining WikiTQ. After filtering unsuccessful annotations from the collected examples, 7,993 examples remain in the curated probe set. This set is divided into an SFT subset of 3,993 examples and a GRPO subset of 4,000 examples. The SFT subset is further processed to construct planner and executor training data, resulting in 3,839 and 3,756 valid examples, respectively. For semantic enrichment, we additionally sample 8,000 table instances.

Table 10: Statistics of the curated training data. The probe training set is constructed from WikiSQL and WikiTQ after filtering unsuccessful annotations, while semantic enrichment uses an additional randomly sampled table subset.

Complexity Level	Orig. Count	Rate	Final Size
<i>WikiSQL Subset</i>			
Level 1 (Simple)	28,685	4%	1,147
Level 2 (Multi-Cond)	10,995	10%	1,099
Level 3 (Agg/Group)	9,868	15%	1,480
Level 4 (Complex)	10	100%	10
WikiSQL Total	49,558	—	3,736
WikiTQ (Full)	4,261	100%	4,261
Curated Probe Set	—	—	7,993
<i>Training Usage</i>			
Planner SFT	—	—	3,839
Executor SFT	—	—	3,756
GRPO Training	—	—	4,000
Semantic Enrichment Training	—	—	8,000

E Training and Implementation Details

This section provides the comprehensive training configurations for the Standard Dense Retriever (SDR) and the Probe-Grounded Reasoning (PGR).

E.1 Retriever Training (SDR)

The SDR is built upon a BERT-base-uncased backbone. We fine-tune the model for 10 epochs using a contrastive learning objective with in-batch negatives. We use a learning rate of 1×10^{-5} with a linear warmup ratio of 0.1 and a batch size of 64, employing a temperature parameter $\tau = 0.05$ for the InfoNCE loss. The table fingerprints are serialized as described in Appendix A.

E.2 PGR Training Phases

The PGR components are initialized from Qwen3-8B and optimized with component-specific objectives using the Unsloth library.

Phase I: SFT. We perform supervised fine-tuning for all components for 3 epochs using AdamW with weight decay 0.01 and a warmup ratio of 0.1. The probe module is fine-tuned on the curated probe training set, while the semantic enrichment module, planner, and executor are trained on their respective datasets reported in Table 10. The probe and semantic modules use a linear learning-rate schedule with a peak learning rate of 1×10^{-5} , while the planner and executor use a cosine schedule with a peak learning rate of 2×10^{-5} .

Phase II: GRPO. We perform alignment for 1 epoch using GRPO with a group size $G = 8$, a learning rate of 5×10^{-6} , and a warmup ratio of 0.1. The training is guided by a tiered reward system based on Exact Match outcomes, as detailed in Appendix C.

Inference Configuration. All PGR inference uses deterministic decoding with temperature = 0, top- $p = 1.0$, and top- $k = 1$, with a maximum generation length of 2,048 tokens. LLM inference is served through vLLM. A fixed random seed of 42 is used for reproducibility.

F Additional API-based Model Analysis

We provide an additional analysis comparing our reasoning pipeline with API-based Direct Pandas variants under a controlled table-QA setting. All methods operate on the same gold table and the

same 2,000 validation instances, with a standardized input format consisting of normalized questions, column names, and cell values.

We compare two DeepSeek-V3-based Direct Pandas variants:

- **Direct Pandas (w/o row rerank):** The model generates a single-step Pandas-style program executed on the original table.
- **Direct Pandas (w/ probe-based row rerank):** The same Direct Pandas pipeline is augmented with probe-based row reranking before program generation.

We additionally evaluate our PGR pipeline with the Qwen3-8B backbone using the decoupled SPE workflow, including semantic enrichment, strategic planning, and adaptive contextual sampling.

Table 11 reports results on the sampled validation subset. PGR achieves 72.65% accuracy, compared with 70.50% for Direct Pandas without row reranking and 73.65% for the variant with probe-based row reranking. Probe-based row reranking improves the Direct Pandas variant by 3.15 percentage points, highlighting the contribution of evidence-focused row selection. PGR achieves comparable performance while additionally providing an explicit decoupled reasoning process.

Table 11: Auxiliary API-based analysis on a 2,000-sample subset of the Open-WikiTable validation set. All methods are provided with the same gold table.

Method	Backbone	Accuracy (%)
Direct Pandas (w/o row rerank)	DeepSeek-V3	70.50
Direct Pandas (w/ probe-based row rerank)	DeepSeek-V3	73.65
PGR (Local)	Qwen3-8B	72.65

G Latency Sensitivity Analysis

Experimental Setup. We measure inference latency on the first 100 queries of the Open-WikiTable validation set with batch size 1, using vLLM to serve the LLM inference calls. Latency is defined as the wall-clock time of one complete query, including all PGR reasoning and execution stages after retrieval. Retrieval is reused from pre-computed candidates and excluded from the timing. The results are summarized in Table 12.

Table 12: End-to-End Latency vs. Candidate Size K .

Candidate Size (K)	Avg. End-to-End Latency (s)	LLM Calls per Query
$k = 5$	9.795	4
$k = 10$	9.828	4
$k = 20$	9.857	4

The latency remains relatively stable as the candidate size increases from $K = 5$ to $K = 20$. Since all settings use the same PGR pipeline and number of model invocations, increasing K introduces limited additional overhead in this evaluation. The measured variation mainly reflects runtime differences in the full inference process.

H Diagnostic Analysis of Evidence Localization

The main text reports both Table-level and Segment-level Evidence Accuracy. We further analyze Segment-level Hit Accuracy across query complexities using SDR’s top-5 retrieved candidates on the Open-WikiTable validation set. Table 13 provides a finer-grained breakdown of probe-based evidence localization.

Table 13: Diagnostic analysis of segment-level evidence localization across query complexities using the top-5 candidates retrieved by SDR on Open-WikiTable. The three reranking outcomes are mutually exclusive.

Metric	Overall	Single-GT	Multi-GT
Total Samples	6,602	5,625	977
Segment-level Hit Accuracy	92.14	91.18	97.65
Non-fallback: Top-1 Changed	22.98	21.49	31.53
– Hit	94.00	93.05	97.73
– Miss	6.00	6.95	2.27
Non-fallback: Top-1 Retained	61.31	68.48	20.06
– Hit	99.09	99.09	98.98
– Miss	0.91	0.91	1.02
Fallback	15.71	10.03	48.41
– Hit	62.30	33.16	97.04
– Miss	37.70	66.84	2.96

Based on the deterministic reranking procedure, the probe module produces three mutually exclusive outcomes. In 22.98% of cases, probe scoring changes the retriever’s initial Top-1 without fallback, and 94.00% of these cases achieve a segment-level hit. In 61.31% of cases, probe scoring retains the initial Top-1 with a unique highest score, with a 99.09% hit rate. The remaining 15.71% of cases use the fallback mechanism when probe-based scoring is unavailable or does not yield a unique maximum score. Fallback is more frequent in Multi-GT cases, occurring in 48.41% of instances with a 97.04% hit rate, whereas Single-GT fallback cases are more challenging, with a 33.16% hit rate.

H.1 Error Analysis of Probe-based Evidence Localization

We further analyze probe-based evidence localization failures under the single-ground-truth setting

using the Top-5 candidate sets retrieved by SDR on the validation set.

Table 14: Failure mode analysis of probe-based localization under single-ground-truth cases.

Failure Mode	Count	Ratio
Constraint Value Mismatch	210	42.34%
Gold Segment Not in Top- K	119	23.99%
Metadata Mismatch	86	17.34%
Ranking Competition	46	9.27%
Empty Evidence	25	5.04%
Column Name Mismatch	10	2.02%

Table 14 summarizes the failure modes among the 496 incorrect cases. The dominant failure category is constraint value mismatch (42.34%), followed by missing gold segments from the retrieved candidate set (23.99%) and metadata mismatch (17.34%). The analysis identifies evidence matching and retrieval coverage as the main sources of the remaining localization errors.

Question and Generated Probe

Question: “What was the expenditure per capita if the revenue per capita was \$7,755 in federal taxes and expenditures per state?”

Generated Probe:

```
{  "title": "Federal taxation
and spending by state ...",
  "evidence": [{  "column":
"Revenue_per_capita",
                  "value": "7,755"
}] }
```

Ground Truth Segment (Seg 1294)

State_	Revenue_per_capita	Spending_per_capita
Tennessee	\$7,755	\$8,358
Indiana	\$6,724	\$7,447

Question and Generated Probe

Question: “If in the opinion polls for the 2009 Norwegian general election, the average polling rate per month in July 2009 was 7.2%, what was it in May 2009?”

Generated Probe:

```
{  "title": "Opinion polling for
the Norwegian parliamentary
election, 2009 ...",
  "evidence": [{  "column":
"Jul_2009",
                  "value": "7.2"
}] }
```

Ground Truth Segment (Seg 3849)

Party	May_2009	Jul_2009
Labour	32.9%	33.1%
Socialist Left	7.3%	7.2%

These examples illustrate constraint value mismatch failures. In both cases, the generated probes identify the relevant columns and value patterns, but fail to match the ground-truth cells due to differences between probe constraints and table cell representations. Specifically, the generated probes capture the underlying values but do not fully match the cell representations in the table.

I Additional Generalization and Ablation Analysis

We provide additional analyses beyond the main Open-WikiTable evaluation on RETQA (Wang et al., 2025), a real-estate table QA dataset. We use a filtered RETQA test subset containing 2,976 single-table questions and perform zero-shot evaluation with models trained only on Open-WikiTable. Based on this setting, we first examine probe-based evidence localization under different retrieval configurations and then analyze the contribution of different components in the Semantic-Plan-Execute (SPE) workflow through closed-domain ablations.

I.1 Probe-based Evidence Localization under Different Retrieval Configurations

We further evaluate segment-level probe localization under two retrieval configurations. The first configuration uses the table-level `table_caption_label` field, where BM25 retrieves candidates without table partitioning. The second configuration adopts a segment-based setting, where tables are split into fixed-length 500-token chunks and retrieved using the full segment serialization. We follow the same segment-level hit criterion defined in the main text.

Table 15: Retrieval hit accuracy and segment-level probe localization accuracy under different retrieval configurations.

Setting	Retrieval Recall				Probe Evidence Acc.			
	Top-1	Top-2	Top-5	Top-10	Top-1	Top-2	Top-5	Top-10
Metadata	0.663	0.678	0.733	0.809	0.663	0.662	0.687	0.753
Segmented	0.463	0.561	0.703	0.798	0.463	0.528	0.605	0.652

Table 16: Ablation study of the reasoning workflow and Probe-guided Table Refinement on RETQA under the closed-domain setting.

Method	Set-EM	P-F1	QA w/ Ref.	QA w/o Ref.
SPE	70.8	91.1	71.8	68.0
PE	37.6	84.0	71.4	68.1
E	–	–	75.1	62.4

Table 15 reports retrieval hit accuracy and segment-level localization accuracy under the two configurations. At Top-10, conditioned on the ground-truth segment being included in the retrieved candidate set, probe localization achieves 93.0% conditional accuracy in the table-level retrieval setting and 81.7% in the segmented retrieval setting. These results suggest that probe-based localization remains applicable across different retrieval configurations and enables fine-grained evidence localization within retrieved candidates.

I.2 Closed-domain Reasoning Ablation

We further analyze the interaction between the SPE reasoning workflow and Probe-guided Table Refinement on RETQA under the closed-domain setting. The results are shown in Table 16.

On RETQA, removing semantic enrichment substantially degrades plan-related metrics, reducing Set-EM by 33.2 points and P-F1 by 7.1 points. However, this improvement in intermediate plans brings limited gains in final QA accuracy, as SPE and PE achieve similar QA performance under the same refinement setting. In contrast, the executor-only variant benefits substantially from refinement, improving from 62.4% to 75.1%. Further analysis shows that among the correct executor-only cases with explicit SQL conditions, 15.49% have generated filters that deviate from the annotated constraints while still producing correct answers, suggesting that refinement can preserve sufficient evidence through alternative filtering conditions. Overall, reasoning workflow and evidence refinement contribute through complementary pathways.

Refinement-SQL Mismatch Example

Question: Among Yue Long Garden and Jian Fa Central Mansion in Baiyun District, Guangzhou, which one has a lower floor area ratio?

SQL: WHERE "Project Name" IN ('Yue Long Garden', 'Jian Fa Central Mansion')

Pandas: df['Land Parcel Name'].isin(['AB0405037', ...])

J Prompt Templates

J.1 Probe Generation Prompt

System Prompt for Probe Generation

You are an expert analyst extracting evidence from Question into a JSON probe to check table relevance. Provide a step-by-step reasoning in <THINK></THINK> block. Then, output the JSON probe in <PROBE></PROBE> block.

Reasoning steps:

1. Summarize the user’s intent.
2. First, identify the target table by selecting the most relevant 'title' from the provided schema based on key phrases in the 'Question'.
3. Second, based exclusively on the selected 'title', examine its available columns. Map specific values from the 'Question' to the correct 'column'. For comparisons (>, <, >=, <=), treat the boundary value as evidence. Leave 'evidence' empty **ONLY** for aggregation or sorting without specific filters.

The probe block must be valid JSON with 'title' and 'evidence' (list of {'column', 'value'}).

J.2 Semantic Enrichment Prompt

Semantic Enrichment Prompt

Role: You are an expert data analyst. Your task is to analyze a raw data table and generate a concise, semantic description for each of its columns.

Instructions:

1. **Understand the Table Context:** Examine the table’s title, column and sample cell values together to infer what the entire table represents. Based on this combined information, write a concise sentence that summarizes the table’s domain, subject, and overall purpose. This judgment should capture the background or type of information recorded in the table.
2. **Generate Column-Level Descriptions:** After forming the overall understanding, write a clear and specific sentence for each column, describing what the column represents within the context of the table title and overall table meaning. Explicitly include the table title or its key context in each column description to ensure the semantic link.

Format the Output:

- Provide your final answer as a single JSON object.
- Include one reserved key called 'table_summary' for the overall table-level background sentence.
- Each other key should be an exact column name from the input table, and its value should be your generated semantic description.
- Do not include any other text, explanations, or markdown formatting outside of this JSON object.

J.3 Strategic Planning Prompt

Strategic Planning Prompt

You are an expert query planner for a database table. Your goal is to create a logical, step-by-step plan to answer a user's question by selecting the necessary column descriptions in the correct order.

Think step by step to formulate a plan. Explain how you will use the columns to find the answer. Place your reasoning between `<start_working_out>` and `<end_working_out>`.

Provide the final list of required column names in their logical order. Then, provide your solution between `<PLAN>` and `</PLAN>`.

J.4 Program Execution Prompt

Program Execution Prompt

You are an expert Python data analyst. Write Pandas code to answer the user's question. Think step-by-step before coding and put your reasoning between `<start_working_out>` and `<end_working_out>`.

Reasoning must include:

- 1. Condition Setup:** Determine the filtering and selection conditions needed to answer the question.
- 2. Data Preparation (if needed):** Describe preprocessing like handling missing values, type conversion, or excluding summary rows.
- 3. Implementation:** Follow Execution Plan columns in order to write the pandas code.

Code requirements:

- Use the variable 'df' to access the DataFrame.
- Use exact string matching (case-insensitive) for string filters.
- Use `pd.to_numeric(df['col'], errors='coerce')` before finding maximums, minimums, or performing arithmetic.
- Store the final result in `final_answer`.

Output only the reasoning and code. Put code between `<CODE>` and `</CODE>`.