STAR-Table: Spatial Textual Anchoring and Reconstruction for Hierarchical Table Reasoning via OCR

Zhoujian Wang[12*] , **Han Zhang**[12*] , **Hanfang Yang**[12†]
[1]Center for Applied Statistics, Renmin University of China
[2]School of Statistics, Renmin University of China
{wangzhoujian, hanzhang0816, hyang}@ruc.edu.cn

Abstract

Tables are widespread in real-world scenarios. In particular, hierarchical tables encode much richer structural semantics; however, their complex layouts pose severe challenges to Large Language Models (LLMs). The core difficulty lies in precisely aligning complex topological structures with multi-level semantic dependencies, where alignment failures often lead to cross-level aggregation errors and hallucinations. To address this challenge, we propose **STAR-Table**, a framework that synergizes visual perception with symbolic reasoning. Unlike traditional text-only approaches, STAR-Table leverages **OCR** models to extract pixel-level spatial coordinates, thereby establishing explicit Spatial-Textual AnchoRs. These anchored spatial priors are formulated as explicit structural hints to guide LLMs in reconstructing the hierarchical topology of tables. Based on the reconstructed topology, complex visual layouts are decomposed into one or more distinct sub-tables, each of which is further mapped into a deterministic flattened database. Finally, we design an adaptive QA module that fuses Vision-Language Model (VLM) semantic summaries with multi-step SQL execution to support robust table-based reasoning. Experimental results across multiple benchmarks demonstrate that **STAR-Table** achieves superior performance, significantly outperforming specialized fine-tuned models and advanced in-context learning paradigms.

1 Introduction

As carriers of structured data, tables play an indispensable role in financial statements, statistical reports and scholarly documents (Cafarella et al., 2008). A table is regarded as hierarchical if its header exhibits a multi-level structure (Wang et al., 2021). In real-world scenarios, flat tables have a limited capacity to store multi-dimensional data; consequently, hierarchical tables are significantly more prevalent (Cheng et al., 2022). Unlike flat ones, these tables implement multi-dimensional organization through nested headers and complex cell merging. While this compact structure is highly intuitive for human readers, its non-linear, two-dimensional nature fundamentally disrupts the sequential processing paradigm of standard LLMs.

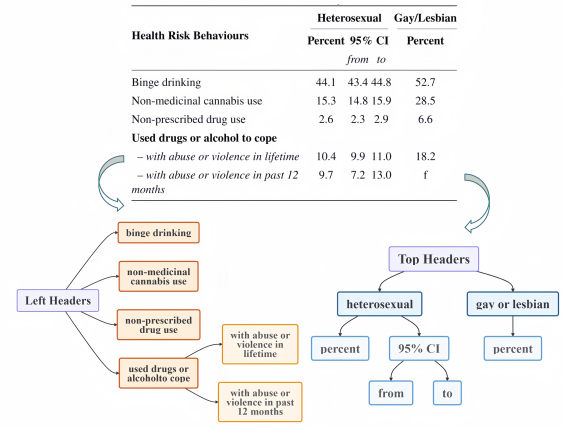


Figure 1: Structural extraction of a hierarchical table. The visual layout is parsed into topological trees, representing the top ($\mathcal{H}_{top}$) and left ($\mathcal{H}_{left}$) header hierarchies.

Despite the remarkable success of recent advancements on flat tables for reasoning tasks like Table Question Answering (Pasupat and Liang, 2015) and Fact Verification (Chen et al., 2020), extending these capabilities to hierarchical tables presents non-trivial bottlenecks. To successfully perform deeper logical processing on such complex layouts, models must systematically reconstruct these hierarchical headers to bridge the gap between visual arrangement and symbolic reasoning. As illustrated in Figure 1, the logical identity of each data cell is determined by its specific path within the two hierarchical trees, requiring the model to align visual spans with their corresponding semantic ancestors. **The difficulty of**

[*] Equal contribution.
[†] Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 34529–34558
October 24-29, 2026 ©2026 Association for Computational Linguistics

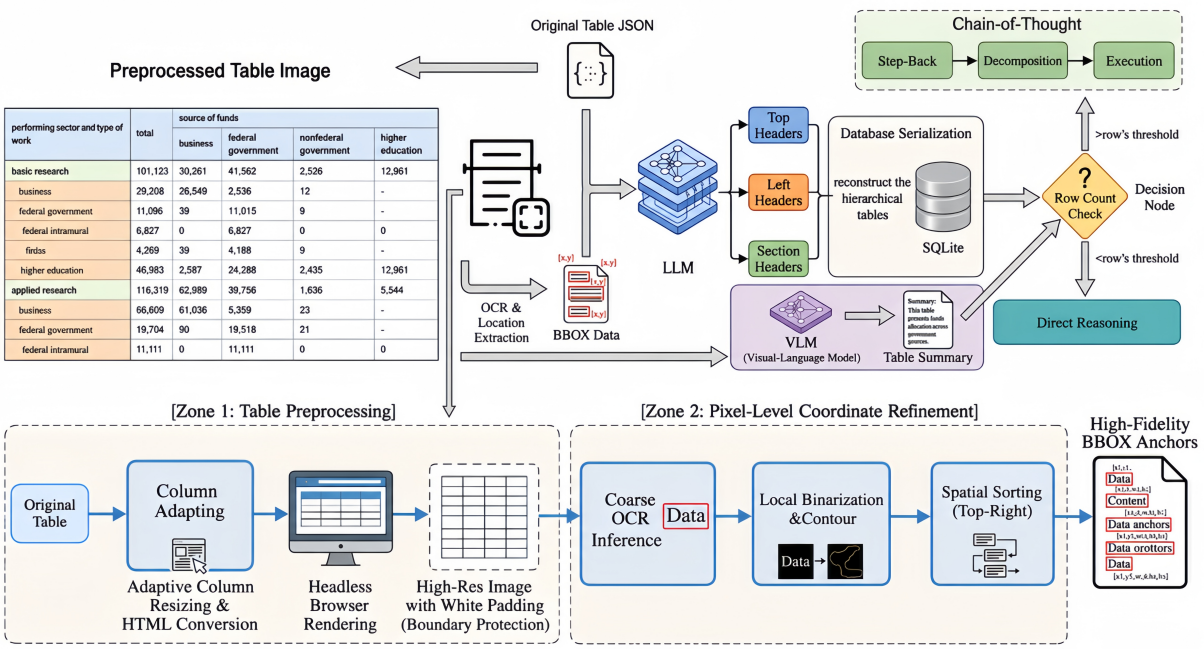


Figure 2: **The overall architecture of our proposed framework.** The framework comprises the following components: 1. Visual Extraction in the bottom deriving high-fidelity bounding boxes via HTML rendering and pixel-level refinement; 2. Structure Reconstruction in the center serializing layouts into a SQLite database via LLMs, alongside VLM summaries; 3. Adaptive Reasoning in the right dynamically routing queries to a CoT pipeline or direct reasoning based on row count.

this task is twofold. On one hand, the inherent two-dimensional spatial characteristics of tables establish complex topological connections that are exceptionally difficult to capture through standard text. On the other hand, unlike flat key-value pairs, the semantics of data cells in hierarchical tables are governed by multi-level header paths. Consequently, complex queries often necessitate cross-level reasoning, where models must traverse the header tree to perform aggregation or comparison across distinct hierarchical levels. Failure to rigorously comprehend these structural constraints and cross-cell dependencies directly leads to reasoning fallacies, such as aggregation errors or semantic hallucinations.

To address these challenges, early paradigms flattened 2D layouts into 1D sequences (Liu et al., 2022). Although subsequent methods introduced hierarchical positional encodings (Eisenschlos et al., 2021; Li et al., 2021) to mitigate structural loss, they still inherently fail to capture the exact visual topology and cell-merging dependencies of complex tables. Consequently, such implicit structural encoding inevitably fails to preserve the intricate multi-dimensional connections, leaving models unable to reliably reconstruct the recursive parent-child pathways essential for defining a cell's semantic scope. Crucially, while models struggle with flattened inputs, humans grasp tabular hierarchies intuitively via vision, suggesting that spatial layouts offer an indispensable global perspective for structural comprehension. While recent studies such as MatCha (Liu et al., 2023) have incorporated Vision-Language Models (VLMs) to simulate this process, these general-purpose models still lack the granularity required to capture fine-grained spatial details. This limitation motivates our core intuition: leveraging advanced OCR models (Cui et al., 2025; Wei et al., 2025; Wang et al., 2024a) can provide superior, fine-grained spatial perception. By leveraging OCR for precise spatial perception and VLMs for robust semantic reasoning, we can decouple table modeling into two complementary dimensions: geometry and semantics. This synergy enables a robust reconstruction of hierarchical structures, providing a solid foundation for downstream question answering (QA) tasks.

Building on this multi-dimensional perspective, we propose **STAR-Table**, a geometry-aware pipeline that integrates fine-grained visual perception with deterministic symbolic execution. Designed for hierarchical table-based reasoning, STAR-Table integrates visual perception with tex-

tual processing, as depicted in Figure 2. Specifically, our framework extracts dense spatial coordinates from the OCR engine to serve as explicit structural priors. By mapping these physical coordinates to textual elements, the LLM can precisely resolve spanning cells and nested hierarchies, translating 2D visual layouts into an executable symbolic schema. In practice, we first render tables from various formats into a unified visual representation, subsequently employing an OCR module to capture fine-grained visual cues. Rather than merely extracting text, we utilize these cues to uncover the table's layout topology, explicitly modeling the spatial coordinates and geometric dependencies between adjacent elements. Subsequently, we leverage an LLM, conditioned on the raw visual information, to comprehend the layout and reconstruct the hierarchical structure of the table. Based on the parsed structure, the raw table content is transformed into a structured database. Finally, we design a tailored LLM in-context learning strategy to execute the table QA task.

Our main contributions are summarized as follows:

- **Spatial-Aware Structural Formulation**: We formulate complex hierarchical layouts as directed trees. By explicitly decoupling spatial coordinates and semantic headers, we establish a robust mathematical foundation for resolving multi-level tabular dependencies.
- **Fine-Grained Spatial Perception**: Unlike general-purpose VLMs that overlook geometric details, our approach leverages a tailored OCR module with pixel-level coordinate refinement to capture the precise topology of nested headers and spanned cells.
- **Deterministic Structure Reconstruction**: We develop a geometry-guided in-context learning strategy to map these visual anchors into executable databases, mitigating the information loss inherent in traditional flattening methods through dynamic sub-table instantiations.
- **Robust Reasoning and Performance**: Extensive evaluations demonstrate that our framework significantly outperforms existing baselines, achieving remarkable robustness against highly complex tabular topologies and adversarial perturbations.

2 Related Work

2.1 Hierarchical Tables

Prior table question answering (Table QA) datasets primarily featured simple table structures. Prominent examples include WikiTableQuestions (Pasupat and Liang, 2015) and Spider (Yu et al., 2018), which predominantly consist of flat tables. To bridge this gap, Parikh introduced ToTTo (Parikh et al., 2020), a dataset containing a subset of complex tables with hierarchical content. Subsequently, in the financial domain, FinQA (Suster et al., 2021) and TAT-QA (Zhu et al., 2021) introduced hybrid data QA tasks, requiring models to possess numerical reasoning capabilities across both textual and tabular data. HiTab (Cheng et al., 2022) dedicates to hierarchical table QA and generation; and it explicitly highlights the challenges posed by non-standard structures through the introduction of Hierarchical Indexing. Furthermore, MultiHiertt (Zhao et al., 2022) proposed a task involving Question Answering over multiple hierarchical tables, demanding that models handle cross-table entity linking and numerical aggregation. The goal of our work is to transform these hierarchical tables into a format that facilitates subsequent reasoning by LLMs while preserving the complete information of the original tables to the greatest extent possible.

2.2 In-Context Learning Strategies and Question Decomposition for LLM-based Table QA

Leveraging Chain-of-Thought (CoT) to elicit reasoning in LLMs has become one of the dominant methods for table-based reasoning. For instance, Chain-of-Table (Wang et al., 2024b) proposes simulating the reasoning process by dynamically generating intermediate tables, while Dater (Ye et al., 2023) employs row and column decomposition methods to address the challenges of reasoning over large-scale tables. Furthermore, Least-to-Most (Zhou et al., 2023) and DecomP (Khot et al., 2023) introduced mechanisms that encourage LLMs to generate and solve subquestions during the reasoning process. These approaches have demonstrated significant improvements in handling questions that require multi-step reasoning and complex calculations.

Researchers have also explored incorporating external tools to address tasks requiring high precision, such as numerical computation. Program of

Thoughts (PoT) (Chen et al., 2023) and PAL (Gao et al., 2023) proposed generating Python programs and executing them via an interpreter to solve specific reasoning problems. Similarly, Rajkumar (Rajkumar et al., 2022) introduced LLM-based Text-to-SQL methods and applied them to Table QA tasks. To further extend capabilities, Binder (Cheng et al., 2023) generates SQL or Python programs and extends their capabilities by calling LLMs as APIs in the programs. Beyond such explicit tool-augmented approaches, recent vision-based methods have explored table and chart reasoning from visual inputs. MatCha (Liu et al., 2023) incorporates vision-language modeling by pre-training on chart derendering and mathematical reasoning objectives, while TableFormer (Nassar et al., 2022) applies transformers to table structure understanding from document images. Overall, these methods improve table reasoning via tool execution or visual modeling, while leaving hierarchical schema extraction underexplored.

In this work, we maximize information preservation by converting raw table files into database files through a series of operations. Simultaneously, we introduce a comprehensive QA workflow incorporating question step-back and sub-question decomposition to enhance the LLM's performance in solving complex problems.

3 Preliminaries

In this section, we formally define the hierarchical table structure, followed by the formulation of the table-based reasoning task.

Definition of Table Structure. We formally define a geometry-aware atomic table unit as a tuple $\mathcal{T} = (\mathcal{C}, \mathcal{H}, t)$, where t denotes the table title. The Content set $\mathcal{C}$ comprises data entries, where each cell $c \in \mathcal{C}$ is encapsulated as a dual-modal entity comprising textual content v and a spatial coordinate b. Specifically, we define b as a 4-dimensional vector $b \in \mathbb{R}^4$ ($b = [x_{min}, y_{min}, x_{max}, y_{max}]$), representing the geometric span of the cell on the 2D image plane, where coordinates correspond to the normalized horizontal and vertical boundaries. The Header set $\mathcal{H}$ is the disjoint union of top ($\mathcal{H}_{top}$), left ($\mathcal{H}_{left}$), and section ($\mathcal{H}_{sec}$) headers; distinct from $\mathcal{C}$, headers semantically constrain data cells while sharing the same spatial attributes defined by b.

In hierarchical layouts, we model dependencies as directed trees: a relation $h_i \rightarrow h_j$ denotes that h_i is the parent (refinement) of h_j. To facilitate structural representation, we introduce virtual nodes denoted as $root_{top}$ and $root_{left}$, to serve as the logical roots for the top header hierarchy $\mathcal{H}_{top}$ and the left header hierarchy $\mathcal{H}_{left}$ respectively. A complex composite table is thus formalized as a collection $\mathbf{T} = \{T_1, \ldots, T_n\}$.

Definition of Task. Given a target table T and a natural language question sequence $Q = \{w_1, w_2, \ldots, w_L\}$, the goal of Table QA is to generate the correct answer A. Our proposed framework can be formalized as a mapping function:

$$\mathcal{A} = F(T, Q, I)$$

where I represents the instructional context within the workflow, encompassing all prompt inputs used to guide the models.

4 Methodology

4.1 Overview

We propose an end-to-end framework designed to address hierarchical table-based reasoning tasks. As illustrated in Figure 2, first, we standardize diverse raw table formats into images and employ an optimized OCR module to extract textual primitives accompanied by precise physical coordinates. Second, guided by geometric spatial cues, we utilize LLMs to reconstruct hierarchical structuresincluding nested headers and table sectionsand transform them into an executable relational database. Finally, we design a retrieval-augmented pipeline that synergizes query decomposition strategies with deterministic SQL execution, supplemented by VLM-generated table summaries, to achieve robust reasoning for complex multi-hop queries.

4.2 OCR-Driven Visual Perception

Considering that tabular hierarchies are inherently encoded in 2D layouts, we leverage OCR to extract precise bounding boxes. This yields explicit geometric constraints for robust tree reconstruction via coordinate overlap, bypassing fragile text serialization. However, this spatial reliance is highly vulnerable to upstream visual degradation, where noise or inconsistent formats can trigger cascading topological errors. To neutralize these risks, we introduce a unified preprocessing pipeline. By rendering diverse source formats into high-fidelity images, we bypass deceptive DOM

noise, establishing a unified, format-agnostic visual bottleneck for accurate downstream OCR.

While modern OCR engines excel at general text recognition, they often struggle with the extreme density of hierarchical tables, frequently yielding marginal omissions and excessively loose, overlapping bounding boxes. To provide high-fidelity geometric anchors for downstream reasoning, we tailor an OCR pipeline to specifically address these vulnerabilities. At its core, we deploy a dual-engine architecture to balance high detection recall with structural safety. Furthermore, we apply a dynamic padding strategy to prevent recognition degradation near image boundaries. Crucially, to resolve the overlapping of coarse boxes in densely packed cells, we introduce a pixel-level coordinate refinement mechanism (Bradski, 2000) to tightly bound the text. Ultimately, these spatially sorted, refined coordinates effectively bridge the gap between raw visual layouts and precise symbolic execution.

4.3 Structure Extraction and Database Instantiation

Geometry-Guided Structure Extraction. Although raw OCR provides text and coordinates, it cannot reliably recover complex hierarchical topologies. To address this, we introduce a geometry-guided in-context learning strategy.

We first estimate the column count through vertical projection and use it as a hard constraint for reconstructing the top header $\mathcal{H}_{top}$ as a directed tree. This prior enforces leaf-node alignment with data dimensionality, thereby reducing semantic hallucinations. The maximum vertical boundary (y_{max}) of $\mathcal{H}_{top}$ then separates global header definitions from the numerical grid. Below this boundary, a symmetric procedure extracts the left header $\mathcal{H}_{left}$ by using estimated row counts and horizontal spans to identify parent-child relations. Finally, section headers $\mathcal{H}_{sec}$ are detected via dual-modal cues, including grid discontinuities and semantic signals, to identify macroscopic dividers while filtering ambiguous statistical terms. These components jointly form the complete header set $\mathcal{H}$, which defines the table's topological skeleton.

Context-Aware Database Instantiation. After structure extraction, we transform the topology $\mathcal{H}$ into a relational database for deterministic SQL execution (Cheng et al., 2023).

To mitigate semantic loss during flattening, we adopt a recursive schema flattening strategy. Specifically, labels along each path from $root_{top}$ or $root_{left}$ to leaf nodes are concatenated to form globally unique composite keys (Wang et al., 2021), explicitly preserving ancestry context. For abstract intermediate nodes in $\mathcal{C}$ that contain no direct data, context-aware row alignment anchors left indices to raw data, ensuring that only valid leaf nodes are mapped to executable records. For complex layouts, section headers dynamically instantiate isolated sub-tables for distinct semantic blocks. Finally, all records are populated into a SQLite database, with spatial proximity used to recover rows lacking explicit headers. This process preserves structural fidelity and provides a robust symbolic foundation for SQL-based reasoning.

Algorithm 1: Adaptive Multi-stage Table QA Pipeline

Input: Database $\mathcal{D}$, VLM Summary S_{vlm}, User Query $\mathcal{Q}$
Output: Final Answer $\mathcal{A}$
$C_{schema} \leftarrow$ GetTableSchema($\mathcal{D}$) ; `// Retrieve column names and data types (e.g., REAL, TEXT)`
$C_{values} \leftarrow$ GetUniqueValues($\mathcal{D}, col =$ 'row_name') ; `// Ground valid entities to prevent exact-match hallucination`
$Ctx_{light} \leftarrow \{C_{schema}, C_{values}, S_{vlm}\}$;
$Ctx_{full} \leftarrow Ctx_{light} \cup$ TopRowsHTML($\mathcal{D}, 5$);
$N_{row} \leftarrow$ CountAllRows($\mathcal{D}$);
if $N_{row} < \tau_{row}$ **then**
  $\mathcal{D}_{html} \leftarrow$ SerializeFullTableToHTML($\mathcal{D}$);
  $\mathcal{A} \leftarrow \text{LLM}_{\text{direct}}(\mathcal{Q}, \mathcal{D}_{html}, S_{vlm})$;
else
  $\mathcal{Q}_{sb} \leftarrow$ StepBackPrompt($\mathcal{Q}, Ctx_{light}$);
  $\{q_1, \ldots, q_k\} \leftarrow$ Decompose($\mathcal{Q}, \mathcal{Q}_{sb}, Ctx_{light}$);
  $F_{extra} \leftarrow \emptyset$;
  for $i \leftarrow 1$ **to** k **do**
    $sql_i \leftarrow$ Text2SQL(q_i, Ctx_{full})
    $res_i \leftarrow \mathcal{D}$.execute($sql_i$);
    $ans_i \leftarrow \text{LLM}_{\text{sub_ans}}(q_i, res_i, sql_i)$;
    F_{extra}.add($q_i \rightarrow ans_i$);
  `// Execute Main Query and Synthesize Final Answer`
  $sql_{main} \leftarrow$ Text2SQL($\mathcal{Q}, Ctx_{full}$);
  $res_{main} \leftarrow \mathcal{D}$.execute($sql_{main}$);
  $\mathcal{A} \leftarrow \text{LLM}_{\text{final_cot}}(\mathcal{Q}, res_{main}, F_{extra}, S_{vlm})$;
return $\mathcal{A}$;

4.4 QA Pipeline

To address Table QA tasks requiring both numerical calculation and semantic inference, we propose a pipeline rooted in question decomposition and Retrieval-Augmented Generation (RAG) (Lewis et al., 2020). Because dynamically partitioning complex tables into section-specific databases risks losing macroscopic context, and 1D SQL serialization inherently discards visual cues (e.g.,

spatial grouping), we leverage a VLM to generate a global descriptive summary from the raw image. This summary serves as a semantic bridge for the structurally blind LLM. Furthermore, pre-QA data profilingextracting the schema and top-five rowsprovides grounded context that significantly mitigates hallucinations during Text-to-SQL generation (Rajkumar et al., 2022).

To balance efficiency and accuracy, we implement a scale-adaptive routing mechanism. For small tables, we deploy a Direct QA strategy, feeding the fully serialized database directly to the LLM. This bypasses SQL generation overhead, mitigating over-reasoning and preventing hallucinations induced by applying unnecessarily complex intermediate steps to simple data.

For complex multi-hop queries, we leverage a three-step Chain-of-Thought (CoT) strategy (Wei et al., 2022). To circumvent the error-proneness of direct complex SQL generation, we apply Step-back in-context learning. The system decomposes the query into atomic sub-questions, executes corresponding SQL for each, and synthesizes the main SQL query from intermediate results. Since SQL inherently struggles to capture unstructured semantics like footnotes, the final answer generation integrates symbolic execution results with the VLM-generated summary. Ultimately, this synergy between deterministic symbolic computation and LLM generalization ensures robust performance across diverse tabular layouts and complex reasoning tasks.

5 Experiments

5.1 Experimental Setup

We evaluate on the HiTab (Cheng et al., 2022) test set to assess complex hierarchical parsing and numerical aggregation. Our pipeline integrates InternVL-8B (Chen et al., 2024) (VLM), PaddleOCR (Cui et al., 2025), and GPT-4.1 (OpenAI, 2023). We compare against diverse baselines (details in Appendix C). To demonstrate generalization, we further evaluate our framework on MultiHiertt (Zhao et al., 2022), with full details provided in Appendix D.

5.2 Evaluation Metric

Following standard protocols (Cheng et al., 2022), we adopt execution accuracy as our primary metric, evaluating whether the final predicted result strictly equates to the ground truth across diverse data types.

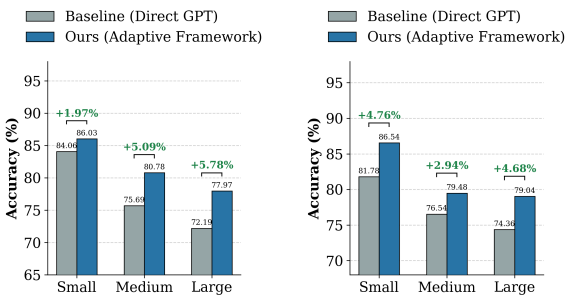
Table 1: Comparison of our method with other baselines on the HiTab dataset. The results of the baselines are mainly taken from their original papers.

Category	Model	Accuracy (%)
Traditional Systems	MAPO (Liang et al., 2018)	44.80
	NSM (Liang et al., 2017)	43.50
Pre-trained Models	TaPEx (Liu et al., 2022)	52.70
	OmniTab (Jiang et al., 2022)	62.80
Specialist LLMs	TableLlama-7B (Zhang et al., 2024)	64.71
	TableLoRA (He et al., 2025)	58.56
	TableGPT2-7B (Su et al., 2024)	70.27
	TableGPT2-72B (Su et al., 2024)	75.57
General LLMs	Qwen3-VL-32B (Bai et al., 2025)	60.49
	GPT-4 (OpenAI, 2023)	71.92
	GPT-4o (Lei et al., 2026)	73.92
	GPT-4.1 (OpenAI, 2023)	78.32
In-context Learning	CoT (Wei et al., 2022)	73.61
	Chain-of-Table (Wang et al., 2024c)	56.94
	ROT (Zhang et al., 2025)	76.70
	ST-Raptor (Tang et al., 2025)	35.13
	TAMO (Li et al., 2024)	66.39
STAR-Table	**GPT-4.1 (Ours)**	**81.47**

5.3 Main Results

Table 1 presents our results on the HiTab benchmark, where the proposed framework achieves an accuracy of 81.47%. These results suggest that integrating explicit visual-spatial anchoring serves as a highly effective structural prior for resolving complex hierarchical tables. We also compare our framework against GPT-4o, a leading multimodal LLM. As reported by Lei et al. (Lei et al., 2026), GPT-4o achieves 73.92% on HiTab when directly processing table images, whereas our pipeline achieves 81.47%. Notably, employing standard GPT-4.1, our method exceeds specialized fine-tuned models like TableGPT2-72B (75.57%) and TableLlama (64.71%), demonstrating that geometry-guided extraction effectively bridges the domain gap without computationally expensive training. Furthermore, translating visual layouts into deterministic SQLite databases alleviates the hallucinations and attention decay common in direct GPT-4 QA. Finally, compared to traditional baselines such as TaPEx (52.70%) and OmniTab (62.80%), our OCR-driven approach mitigates hierarchical information loss by preserving tree integrity which conventional sequence linearization often obscures, thereby facilitating accurate cross-level reasoning.

Robustness to Table Scale. To evaluate our framework's scalability, we partitionize the test



(a) By Row Count (b) By Total Cell Count

Figure 3: Performance comparison between the direct GPT baseline and our adaptive framework across different table size partitions.

set into Small, Medium, and Large subsets using quantile thresholds. Figure 3 illustrates that while the direct GPT baseline's accuracy drops drastically from 84.06% to 72.19% due to attention dilution in lengthy linearized contexts, our framework exhibits exceptional resilience. The performance gap consistently widens as table size increases. By utilizing Adaptive Multi-stage Routing and deterministic SQL execution, our system retrieves only relevant sub-tables, eliminating massive irrelevant data to preserve focused LLM attention on extensive hierarchical structures.

5.4 Ablation Study

To isolate the impacts of visual-spatial perception and global semantics, we independently ablated the OCR and VLM modules. To evaluate robustness against structural complexity, we stratified the HiTab test set by Header Hierarchy Level (L): the maximum number of nested header tiers. Table 2 details performance across flat, two-tier, and deeply nested topologies, highlighting each module's efficacy in resolving escalating structural challenges.

OCR model ablation. To evaluate the necessity of explicit spatial perception, we replaced our tailored OCR module with a text-only baseline and an off-the-shelf OCR model (MinerU). Eliminating spatial coordinates severely degraded overall accuracy to 59.71%, plummeting to 47.21% on deeply nested tables. Error analysis reveals that lacking geometric constraints causes irreversible hierarchical flattening and severe sub-table ambiguity. Furthermore, while MinerU yielded marginal improvements, it remained significantly inferior to our full model. This demonstrates that flattened text sequences inherently fail to resolve complex 2D topologies, confirming that explicit spatial anchoring is indispensable for accurate structural reconstruction.

Table 2: Ablation study on Visual-Spatial Perception. Substituting our tailored OCR pipeline with an off-the-shelf parser (MinerU) or entirely removing spatial anchors leads to severe performance degradation, particularly in deeply nested tables (L $\geq$ 3).

Model Variant	Overall	Header Hierarchy Level		
		L = 1	L = 2	L $\geq$ 3
Ours (Full Model)	**81.47**	**84.74**	**82.93**	**75.93**
w/ MinerU Parser	60.66	65.96	61.16	52.88
Drop (Δ)	(-20.81)	(-18.78)	(-21.77)	(-23.05)
w/o OCR Anchoring	59.71	65.11	56.78	47.21
Drop (Δ)	(-21.76)	(-19.63)	(-26.15)	(-28.72)

OCR enhancements ablation. Having established the necessity of spatial grounding, we further quantified the contributions of our internal OCR enhancements (Table 3). Ablating any component within our customized pipeline noticeably degrades performance, underscoring their collective necessity. Notably, omitting dynamic padding causes a severe 11.70% drop, highlighting its critical role in preventing boundary truncation during sub-table cropping. Similarly, the dual-engine strategy and pixel-level refinement each yield approximately 3% gains, effectively preventing cascaded alignment errors from propagating into the downstream LLM reasoning phase.

Table 3: Ablation study of individual OCR enhancements. Accuracy drops are computed against the full OCR pipeline.

Model Configuration	Accuracy (%)	Δ
Full Pipeline (Ours)	**81.47**	–
w/o Dual-engine strategy	78.47	-3.00
w/o Pixel-level refinement	77.97	-3.50
w/o Padding	69.76	-11.71

VLM semantic ablation. In this experiment, we removed the global table summary generated by the VLM, relying exclusively on local cell content for reasoning (Table 4). Removing the VLM-generated global summary degrades overall QA accuracy to 62.65%, with a precipitous drop to 50.31% on deeply nested tables ($L \geq 3$), despite maintaining structural parsing integrity. This underscores the necessity of high-level semantic context for holistic reasoning. Case analyses reveal

two primary causes for this decline. First, without the VLM to visually resolve ambiguous abbreviations and implicit temporal data, the LLM struggles to accurately map user queries to database fields during SQL generation. Second, the absence of global prior knowledge exacerbates hallucinations during multi-hop reasoning, depriving the LLM of a crucial auxiliary verification mechanism.

Table 4: Ablation study on Global Semantic Understanding. The absence of the VLM summary impairs the model's ability to align non-standard semantics, with errors amplifying as hierarchy depth increases.

Model Variant	Overall	Header Hierarchy Level		
		L = 1	L = 2	L $\geq$ 3
Ours (Full Model)	**81.47**	**84.74**	**82.93**	**75.93**
w/o VLM Summary	62.65	67.37	60.52	50.31
Performance Drop (Δ)	(-18.82)	(-17.37)	(-22.41)	(-25.62)

5.5 Robustness Analysis

To validate that our proposed framework possesses a deep comprehension of table semantics and structure rather than merely overfitting to the dataset distribution, we designed two sets of adversarial perturbation experiments. Based on the original test set, we constructed two variant datasets to stress-test the model's stability across two dimensions: internal semantic noise and external structural variation. As illustrated in Figure 4, our framework consistently outperforms competitive baselines across all perturbed scenarios.

Semantic Perturbation Test. In real-world tabular data, non-standard header naming and irrelevant noise columns are common disturbances. To evaluate robustness against real-world tabular noise, we constructed a semantic perturbation dataset, $\mathcal{D}_{sem}$. Specifically, we injected one to three distractor columns with random values to test schema linking, and applied synonym replacement or case inversion to original headers to assess semantic alignment. Under these perturbations, our framework exhibits exceptional resilience, with accuracy degrading by only 2.53%. In contrast, Chain of Thought and Chain of Table experienced sharper declines of 7.78% and 5.73%, respectively, while specialist models like ST-Raptor deteriorated severely to 28.47%. This robustness is primarily attributed to our VLM-generated global summaries and schema flattening strategy, which synergistically filter incoherent noise and align non-standard header variants.

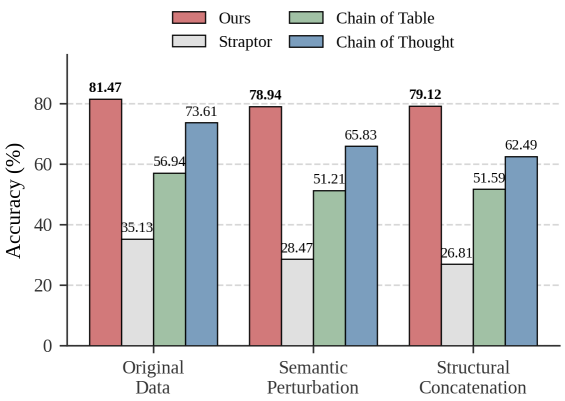


Figure 4: Robustness evaluation under semantic perturbation and structural concatenation.

Structural Concatenation Test. When processing long documents or complex reports, models often encounter multiple independent tables stacked vertically. To evaluate resistance to cross-table aggregation errors common in long documents, we constructed a structural perturbation dataset, $\mathcal{D}_{struct}$. We concatenated a base table with an unrelated, title-less noise table of matching dimensions to create a deceptively continuous data block. Under this perturbation, our framework maintained 79.12% accuracy, demonstrating remarkable resilience against the 11.12% drop seen in Chain of Thought. This stability stems from our Section Header Detection module, which identifies semantic boundaries and isolates noise into separate SQLite sub-tables. Unlike traditional sequence linearization that conflates cross-table values, our SQL-based decoupling ensures strict data integrity, outperforming CoT and Chain of Table by over 16.6% and 27.5%, respectively.

6 Conclusion

We present a geometry-aware framework that synergizes visual perception with symbolic reasoning for complex hierarchical table QA. By explicitly anchoring header topologies via OCR coordinates and transforming unstructured layouts into deterministic SQLite databases, our approach resolves the structural information loss inherent in traditional linearization. Achieving 81.47% accuracy on HiTab, our framework exhibits exceptional robustness against semantic and structural noise, establishing a highly interpretable and reliable parsing-to-execution paradigm for reasoning.

Limitations

While our framework demonstrates exceptional reasoning capabilities on complex hierarchical tables, it presents two primary limitations. First, as a pipeline-based approach, the topological reconstruction inherently relies on the precision of the upstream OCR module. For heavily degraded documents or low-resolution image renderings, inaccurate bounding box extractions may induce cascading alignment errors during spatial anchoring. Second, the multi-stage parsing-to-execution paradigminvolving VLM summarization, iterative LLM decomposition, and dynamic SQL executionincurs higher computational overhead and inference latency compared to traditional end-to-end linearized models. Future work will explore unifying these perception and reasoning modules into a more lightweight, end-to-end multimodal architecture while preserving strict geometric rigor.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments, which led to a significant improvement in this work. The research is supported by the Special Funds of the National Natural Science Foundation of China (Grant No. 72342010).

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Gary Bradski. 2000. The opencv library. *Dr. Dobb's Journal of Software Tools*.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

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

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: Alarge-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24185–24198.

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

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, et al. 2025. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595*.

Julian Eisenschlos, Maharshi Gor, Thomas Müller, and William Cohen. 2021. Mate: multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Xinyi He, Yihao Liu, Mengyu Zhou, Yeye He, Haoyu Dong, Shi Han, Zejian Yuan, and Dongmei Zhang. 2025. Tablelora: Low-rank adaptation on table structure understanding for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22376–22391.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. Omnitab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular

approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Shizhu He, Jun Zhao, and Kang Liu. 2026. Tarex: Reinforcement learning for code-driven table reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31501–31509.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Chenliang Li, Bin Bi, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2021. Structurallm: Structural pre-training for form understanding. 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 6309–6318.

Liyao Li, Chao Ye, Wentao Ye, Yifei Sun, Zhe Jiang, Haobo Wang, Jiaming Tian, Yiming Zhang, Ningtao Wang, Xing Fu, et al. 2024. Table as a modality for large language models. *arXiv preprint arXiv:2402.12417*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. *Advances in Neural Information Processing Systems*, 31.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Eisenschlos. 2023. MatCha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12756–12770, Toronto, Canada. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. 2022. Tableformer: Table structure understanding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4614–4623.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. 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.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv e-prints*, pages arXiv–2204.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, et al. 2024. Tablegpt2: A large multimodal model with tabular data integration. *CoRR*.

Simon Suster, Pieter Fivez, Pietro Totis, Angelika Kimmig, Jesse Davis, Luc de Raedt, and Walter Daelemans. 2021. Mapping probability word problems to executable representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3627–3640, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zirui Tang, Boyu Niu, Xuanhe Zhou, Boxiu Li, Wei Zhou, Jiannan Wang, Guoliang Li, Xinyi Zhang, and Fan Wu. 2025. St-raptor: Llm-powered semi-structured table question answering. *Proceedings of the ACM on Management of Data*, 3(6):1–27.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. 2024a. Mineru: An open-source solution for precise document content extraction. *arXiv e-prints*, pages arXiv–2409.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 17801790, New York, NY, USA. Association for Computing Machinery.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu

Lee, et al. 2024b. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024c. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv e-prints*, pages arXiv–2401.

Ziwei Wang, Jiayuan Su, Mengyu Zhou, Huaxing Zeng, Mengni Jia, Xiao Lv, Haoyu Dong, Xiaojun Ma, Shi Han, and Dongmei Zhang. 2026. Sheetbrain: A neuro-symbolic agent for accurate reasoning over complex and large spreadsheets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33800–33808.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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*, pages 174–184.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. Tablellama: Towards open large generalist models for tables. 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 6024–6044.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. 2025. RoT: Enhancing table reasoning with iterative row-wise traversals. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 559–579, Suzhou, China. Association for Computational Linguistics.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

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

Hardware and Environment All local model inferences, including the visual semantic extraction via the Vision-Language Model and the spatial coordinate anchoring via the OCR pipeline, were executed on a single NVIDIA A100 GPU. The overall framework and evaluation scripts were implemented in Python.

Model Configurations For the core reasoning, schema linking, and code generation tasks, we utilized the GPT-4 (OpenAI, 2023) and GPT-4.1 large language models, accessed via the Azure OpenAI API. To balance deterministic SQL syntax generation with natural language fluidity, we employed a dynamic temperature scheduling strategy across different phases of the pipeline:

- **SQL Generation:** The temperature was strictly set to 0.01 to enforce deterministic schema linking, exact value matching, and syntactically rigid SQL formulation.
- **Query Decomposition & Step-Back:** The temperatures were set to 0.2 and 0.3, respectively, to encourage adequate semantic abstraction while maintaining strict logical fidelity to the original user query.
- **Final Answer Synthesis:** The temperature was set to 0.2 to synthesize the deterministic SQL execution results into coherent and concise natural language responses.

Vision and OCR Backends For the global semantic understanding phase, we employed InternVL-8B (Chen et al., 2024) as the base VLM to generate the high-level table summaries. For the fine-grained geometric parsing, PaddleOCR (Cui et al., 2025) served as the primary extraction engine. The specific dual-engine configurations and pixel-level refinement strategies for the OCR module are comprehensively detailed in Appendix H.

B Dataset Details: HiTab and MultiHiertt

To rigorously evaluate our geometry-aware reasoning framework, we utilize two comprehensive benchmarks constructed from authentic, complex statistical and financial reports: the HiTab dataset (Cheng et al., 2022) and the MultiHiertt dataset (Zhao et al., 2022).

Overview and Statistics Both datasets are sourced from highly professional and complex data portals, containing non-standard layouts that require complex semantic parsing over authentic, open-domain analytical reports.

- **HiTab** is collected from Statistics Canada (StatCan) and the National Science Foundation (NSF), comprising a total of 3,597 tables and 10,686 question-answering pairs.
- **MultiHiertt** is extracted from corporate financial annual reports, consisting of 11,269 tables and 10,440 QA pairs. Critically, it introduces multi-table scenarios where hierarchical tables are embedded within long textual documents, significantly escalating the retrieval and reasoning difficulty.

Structural Complexity Unlike standard databases (e.g., WikiTableQuestions or Spider) that feature flat, one-dimensional schemas, tables in both HiTab and MultiHiertt exhibit profound structural and topological complexity. They are predominantly characterized by deeply nested top column headers, indented left row headers (indicating parent-child hierarchical relationships), spanning cells, and implicit cross-dimensional category alignments. This high degree of two-dimensional spatial complexity inherently fails traditional sequence-to-sequence linearization approaches, making them ideal touchstones for evaluating our spatial anchoring and parsing-to-execution paradigm.

Reasoning Challenges Beyond structural comprehension, resolving queries in these benchmarks intrinsically demands advanced multi-hop numerical reasoning. The QA pairs frequently require cross-level entity aggregation, conditional filtering, and arithmetic operations such as summation, division, and percentage change computations. Particularly in MultiHiertt, models must also perform cross-table comparison and textual context alignment. Consequently, models must achieve exact spatial and structural alignment before any symbolic or numerical execution can yield accurate results.

C Baseline Descriptions

To comprehensively evaluate the effectiveness of our proposed framework, we compare our method against a wide range of competitive baselines, categorized as follows:

Traditional Systems We include MAPO (Liang et al., 2018) and NSM (Liang et al., 2017). These semantic parsing systems utilize hierarchy-aware logical forms and weakly supervised sequence-to-action models to execute symbolic operations over tables.

Pre-trained Table Models We select TaPEx (Liu et al., 2022) and OmniTab (Jiang et al., 2022). These are encoder-decoder models pre-trained on large-scale tabular data, establishing strong baselines for table comprehension tasks.

Specialist LLMs We compare against LLMs specifically fine-tuned or adapted for tabular reasoning tasks. This includes TableLlama-7B (Zhang et al., 2024), parameter-efficient adaptation approaches like TableLoRA (He et al., 2025), and the recent state-of-the-art TableGPT2 (Su et al., 2024) series.

General LLMs We evaluate powerful closed-source Large Language Models, specifically GPT-4 and GPT-4.1, relying on their strong zero-shot and few-shot reasoning capabilities across unstructured and structured contexts.

Advanced In-Context Learning We evaluate leading in-context learning paradigms designed to enhance reasoning, including the standard Chain-of-Thought (CoT) (Wei et al., 2022) for step-by-step logic derivation, and Chain-of-Table (Wang et al., 2024c), which dynamically evolves table contents throughout the reasoning chain. Furthermore, we include recent state-of-the-art frameworks such as ROT (Zhang et al., 2025) (implemented with Llama 3.3), ST-Raptor (Tang et al., 2025) (with GPT-4.1 and InternVL-8B), and the structure-aware multi-hop reasoning approach TAMO (Li et al., 2024).

D Experiment on MultiHiertt

To rigorously evaluate whether the structural reasoning capabilities of our framework generalize beyond the HiTab dataset, we conducted supplementary experiments on MultiHiertt (Zhao et al., 2022). MultiHiertt is recognized as a highly challenging benchmark, characterized by exceptionally complex multi-level tabular structures and the necessity for multi-hop numerical reasoning.

Importantly, we deploy **STAR-Table** in a strict zero-shot setting. We maintain the exact same hyperparameters and heuristic thresholds (e.g., the row threshold τ_{row}) utilized for HiTab, without introducing any dataset-specific fine-tuning or prompt engineering.

As presented in Table 5, our framework achieves an execution accuracy of 61.90%. This performance not only demonstrates a massive improvement over traditional fine-tuned specialists (surpassing the original dataset SOTA, MT2Net, by an absolute margin of 24.8%) but also significantly outperforms the recent strong zero-shot baseline of GPT-4.1 (53.50%). These results firmly validate that our geometry-aware schema extraction and symbolic execution paradigm generalise robustly to unseen, highly complex tabular topologies.

Table 5: Zero-shot generalization results on the MultiHiertt dataset. **STAR-Table** significantly outperforms both fine-tuned baselines and advanced LLMs without any dataset-specific adaptation.

Model	Setting	Accuracy (%)
TagOp (Zhu et al., 2021)	Fine-tuned Baseline	17.50
FinQANet (Zhao et al., 2022)	Fine-tuned Baseline	32.40
MT2Net (Zhao et al., 2022)	Fine-tuned	37.10
GPT-4.1 (Wang et al., 2026)	Zero-shot Prompting	53.50
STAR-Table (Ours)	**Zero-shot**	**61.90**

E Generalization Across LLM Architectures: Evaluation with Open-Weight Models

A potential concern regarding multi-stage reasoning frameworks is their inherent dependency on frontier proprietary models (e.g., GPT-4.1). Specifically, there is an assumption that complex, multi-prompt pipelines might experience a performance collapse when deployed with less capable open-weight models, thereby limiting their practical and cost-effective deployment.

To empirically address this model dependency, we evaluated our framework by substituting GPT-4.1 with an open-weight alternative, Qwen2.5-32B (Bai et al., 2023). Remarkably, the pipeline achieves a highly competitive execution accuracy of 79.55%, proving that our architecture does not collapse in the absence of elite proprietary models.

More fundamentally, this result confirms that our framework's success is not derived from relying on parameter-heavy models to brute-force the parsing of unstructured 2D visual layouts. Instead, by preemptively translating complex hierarchical topologies into deterministic SQL schemas, our

pipeline acts as a robust cognitive bridge. This explicit geometric-to-symbolic grounding drastically lowers the spatial perception and logical reasoning burden placed on the downstream LLM, enabling cost-effective, mid-scale models to navigate complex tabular reasoning efficiently.

F Token Consumption Analysis

To rigorously evaluate the computational efficiency, API cost, and scalability of our parsing-to-execution pipeline, we conducted a comprehensive token consumption analysis across both the offline structural parsing phase and the online Question Answering (QA) phase.

F.1 Table Structure Recognition

The statistics for the offline table structure recognition module were collected across 538 complex hierarchical table samples, yielding a total consumption of approximately 8.16 million tokens (averaging 15,177 tokens per table).

Input-Heavy, Output-Light Paradigm A defining characteristic of our framework is the extreme asymmetry between prompt tokens and completion tokens. Our analysis reveals that prompt tokens account for 89.57% (7,313,781 tokens) of the total computational overhead, while completion tokens constitute merely 10.43% (851,481 tokens). This distribution underscores the robustness of our prompt engineering: the LLM is provided with dense, rich visual-spatial context (heavy input), but is strictly constrained to generate concise, deterministic structural formats such as JSON or arrays (light output). This "read-heavy, write-light" paradigm not only minimizes the latency associated with autoregressive generation but also effectively suppresses LLM hallucinations during schema construction.

Stage-wise Distribution and Structural Complexity Our structural recognition pipeline operates in three sequential steps, and their token footprints directly reflect the inherent topological complexity of hierarchical tables:

- **Step 1: Top Column Header Recognition (31.55%):** Averaging 4,788 tokens per table, this step efficiently resolves the vertical dimensional hierarchy.
- **Step 2: Left Row Header Recognition (32.21%):** Averaging 4,888 tokens per table, this step identifies row indentation and parent-child categories with comparable computational effort to Step 1.
- **Step 3: Section Header Detection (36.24%):** Consuming the highest average at 5,501 tokens, this step involves differentiating sub-tables and isolating disjoint semantic blocks. The elevated token cost aligns with our theoretical observation that identifying implicit semantic boundaries across large, unstructured spans is cognitively the most demanding task for LLMs, requiring holistic global context.

One-time Preprocessing Efficiency Crucially, the 15,177 tokens expended per table represent a *one-time offline cost*. Once the hierarchical visual layout is successfully parsed and instantiated into a deterministic SQLite database, all subsequent user queries interact solely with the flattened database schema. This architectural decoupling means that our downstream reasoning engine entirely avoids the exponential token explosion associated with repeatedly linearizing complex tables for every new query.

F.2 QA Reasoning and Execution

Beyond the initial offline database construction, we further profiled the token consumption during the online QA phase. Across 1,576 complex user queries, our framework initiated 4,093 API calls, consuming a total of 14.97 million tokens.

High-Precision Context Routing On average, resolving a single hierarchical query necessitates approximately 2.6 multi-stage API interactions (e.g., schema linking, SQL generation, and answer synthesis), consuming an average of 9,498 tokens per query. Remarkably, the average token footprint per individual API call is strictly bounded at merely 3,657 tokens. This compact context window empirically validates the efficacy of our SQL-based decoupling strategy: rather than repeatedly injecting entire linearized tables into the LLM's prompt, our framework selectively retrieves only the abstract schema and the mathematically relevant data rows, circumventing the token exponential growth typical of standard prompting paradigms.

Deterministic Output Consistency Consistent with the structural parsing phase, the QA phase exhibits a profound token asymmetry: prompt tokens constitute 95.87% (14,352,766 tokens) of the

workload, whereas completion tokens account for only 4.13% (617,258 tokens). This highly constrained generation profile indicates that the LLM is focusing its computational cycles strictly on deterministic logic (i.e., generating precise SQLite queries and concise factual answers) rather than generating verbose, hallucination-prone natural language responses.

Long-tail Scalability and Cost Predictability When aggregating token consumption by physical table instances, the distribution demonstrates exceptional scalability. Over 58.2% of the tables consumed less than 10K tokens across all their associated queries, and an additional 29.2% remained under 50K tokens. Only a marginal 2.8% of extreme outlier tables exceeded the 200K token threshold. This long-tail distribution effectively proves that our geometry-aware architecture possesses strong "token-inflation resistance"; the operational cost scales gracefully and remains highly predictable even as the intrinsic topological complexity of the source tables increases.

G Implementation Details of the Unified Preprocessing Pipeline

To ensure structural fidelity, visual-spatial consistency, and geometric precision across diverse hierarchical table layouts, our unified preprocessing pipeline transforms raw hierarchical datasets (e.g., JSON formats from HiTab or MultiHiertt) into high-resolution, standardized image canvases. This section details the complete engineering heuristics, formatting protocols, and hyperparameter configurations deployed across the three major stages of the pipeline.

G.1 Hierarchical Serialization to Grid Structure

Before visual rendering, the multi-level hierarchical topologies are serialized into a standard 2D spreadsheet layout using `openpyxl`. To preserve structural relationships within a flat relational grid, we implement the following specific rules:

- **Hierarchical Indentation:** For nested left headers ($\mathcal{H}_{left}$), each ancestral level induces a deterministic physical offset. We inject exactly 4 spaces per hierarchy depth layer into the text primitive of the primary cell, guaranteeing that logical nesting is explicitly encoded as spatial indents.
- **Adaptive Column Width Allocation:** To prevent text clipping or excessive whitespace, column widths are adaptively calculated based on cell content length. Let $\mathcal{L}$ denote the maximum character length of a specific column (excluding the table title). The allocated Excel column width $\mathcal{W}_{\text{excel}}$ is governed by:

$$\mathcal{W}_{\text{excel}} = \min\left(\max\left(\mathcal{L} \times 1.15 + 2, 10\right), 60\right) \tag{1}$$

  To accommodate the physical offset introduced by hierarchical nesting, the character length $\mathcal{L}$ for the first column (Col_1) is scaled by an expansion factor of 1.3 before applying Eq. (1).
- **Text Wrapping and Cell Constraints:** To decouple table title text from column width calculations, the global table title is anchored to row 1, column 1 with text wrapping activated (`wrap_text=True`) and aligned left-center. Conversely, all text elements within data rows are strictly constrained to single-line rendering (`wrap_text=False`) to prevent unpredictable row-height expansions.

G.2 Layout-Preserving HTML Translation

To bridge the gap between static spreadsheet layouts and modern browser rendering engines, the generated workbooks are compiled into structured HTML5 documents. The alignment and geometry are tightly controlled via inline CSS styles:

- **Rigid Geometry Enforcement:** The overarching table element is bound by `table-layout: fixed` and `border-collapse: collapse`. This forces the rendering engine to strictly adhere to predefined column dimensions rather than dynamically adjusting based on text content.
- **Pixel-Level Dimension Mapping:** Excel width dimensions are mapped into HTML pixels using a constant scaling coefficient derived from standard font metrics. The explicit pixel width $\mathcal{W}_{\text{px}}$ defined in the HTML `<colgroup>` is computed as:

$$\mathcal{W}_{\text{px}} = \lfloor \mathcal{W}_{\text{excel}} \times 8 \rfloor \tag{2}$$

  Columns lacking explicit dimensions in the source workbook are assigned a default fallback width of 100px.

- **Whitespace and Text Control:** To ensure that browser layout optimization does not collapse consecutive spaces (which hold the hierarchical indents), data cells are styled with `white-space: pre`, `vertical-align: top`, and `overflow: hidden`. Title cells are configured with `white-space: normal` and `word-wrap: break-word` to guarantee that long descriptions gracefully wrap within the fixed column boundaries.

G.3 High-Fidelity Visual Rendering and Optimization

The compiled HTML documents are rendered into standardized, high-resolution image primitives using a headless rendering engine (`wkhtmltopdf` via `imgkit`) paired with OpenCV post-processing:

- **Anti-Truncation Viewport Configuration:** To categorically prevent horizontal or vertical truncation when rendering exceptionally wide or deep hierarchical structures, the headless browser viewport is set to an inflated fixed width of 4000px with smart width optimization disabled (`disable-smart-width` activated).

- **Resolution Scaling:** To supply the downstream OCR engine with sharp, fine-grained textual features, the document is rendered with a magnification zoom factor of 1.5 at maximum rendering quality (100).

- **Deterministic Margin Elimination:** To remove uncontrolled blank space outside the active table grid, the raw rendered PNG image undergoes automatic bounding-box cropping. The image is converted to the RGB color space, inverted (such that the white background becomes black (`0,0,0`)), and evaluated via local contour bounding-box extraction (`getbbox`). The image is then tightly cropped to these active visual boundaries.

- **Volume-Constrained Compression Fallback:** To shield downstream Vision-Language Models (VLMs) from out-of-memory (OOM) exceptions triggered by massive images, an adaptive compression protocol is activated for assets exceeding 5MB. Such images undergo proportional scaling using Lanczos resampling (`Image.Resampling.LANCZOS`) and are saved in the JPEG format with a quality factor of 85, striking an optimal Pareto front between character recognition recall and memory footprint.

H OCR Implementation Details

To accurately extract text bounding boxes and content from diverse unstructured tables, we implemented a robust, customized Optical Character Recognition (OCR) pipeline based on PaddleOCR. To mitigate common tabular OCR errors such as cross-column text merging or confusing graphical lines with text, our pipeline employs a dynamic dual-engine strategy alongside tailored pre- and post-processing steps.

Image Pre-processing Tables frequently contain text elements positioned extremely close to the image boundaries, which degrades detection accuracy. To resolve this, we apply a dynamic, proportional padding strategy prior to inference. Specifically, we pad the image with a constant white padding equivalent to 15% of both the original height and width ($pad_h = 0.15 \times H, pad_w = 0.15 \times W$). Coordinates are deterministically restored to the original scale post-extraction.

Dual-Engine OCR Configuration We maintain two distinct OCR engines to balance high recall with structural safety:

- **Aggressive Engine (Primary):** Designed for high recall to capture fragmented or faint text. Configuration: `text_det_limit_side_len = 4000`, `det_db_unclip_ratio = 2.2` (expanded bounding boxes), and `det_db_box_thresh = 0.5`.

- **Conservative Engine (Fallback):** Triggered only when anomalies are detected. Configuration: `text_det_limit_side_len = 3200`, `det_db_unclip_ratio = 1.6` (tighter boxes), `det_db_box_thresh = 0.6`, with `use_textline_orientation = True` activated.

Anomaly Detection and Fallback Logic We heuristically analyze the initial outputs from the Aggressive Engine to detect topological anomalies. The pipeline automatically falls back to the Conservative Engine up to 5 times if any of the following criteria are met:

1. **Row Merging:** A bounding box horizontally spans $> 92\%$ of the image width with a height > 15 pixels.

2. **Column Merging:** A bounding box vertically spans $> 85\%$ of the image height.

3. **Line Misclassification:** A bounding box exhibits an extreme aspect ratio ($width/height > 60$), indicating table borders misidentified as text.

Pixel-Level Bounding Box Refinement While the Aggressive Engine ensures high recall (via `unclip_ratio = 2.2`), it occasionally yields oversized bounding boxes that overlap with adjacent cells. To address this, we apply an OpenCV-based contour tightening algorithm. A binary threshold (Otsu's method) is applied to the localized Region of Interest (ROI) expanded by 5 pixels. We then compute the minimal bounding rectangle enveloping the actual text contours. To prevent overshrinking or collapsing boxes into lines, this refinement is aborted if the tightened width or height is reduced by more than 50% compared to the original detection, or if the absolute height falls below 8 pixels.

I Extended Ablation Analysis and Component Contributions

To clearly demonstrate the contribution of each individual component and confirm our framework's performance using open-weight models, we conducted a progressive ablation study. Across all configurations, the VLM and OCR modules remain identical to our main experiments.

As shown in Table 6, each module provides a steady performance gain, culminating in the highest accuracy when explicit spatial anchoring (OCR) and deterministic reasoning (SQL) are combined.

Table 6: Progressive ablation of framework components on the HiTab dataset.

Experiment	Configuration	Qwen2.5-32B (%)	GPT-4.1 (%)
Direct QA	LLM	66.73	76.24
Summary QA	LLM + VLM	69.35	78.13
OCR-augmented	OCR + LLM + VLM	74.82	80.15
Ours (STAR-Table)	**OCR + LLM + VLM + SQL**	**79.55**	**81.47**

Quantifying the Net Contribution of Spatial Anchoring While STAR-Table achieves an 81.47% accuracy compared to GPT-4.1's direct image processing baseline (78.32%), evaluating this 3.15% overall improvement requires isolating the specific contribution of spatial anchoring from the benefits of our elaborate SQL and Chain-of-Thought (CoT) prompt engineering.

To determine whether the performance gain is merely a result of advanced prompting or if it fundamentally relies on explicit geometric constraints, we designed a controlled-variable baseline: we fed plain OCR text—strictly excluding all coordinate and bounding box information—into the exact same GPT-4.1 + SQL reasoning pipeline.

Without spatial coordinates, the accuracy of the SQL-based pipeline dropped severely to 59.71%.

This substantial 21.76% degradation (from 81.47% to 59.71%) directly isolates and quantifies the net contribution of spatial anchoring. It demonstrates that advanced prompt engineering alone is insufficient to resolve complex hierarchical tables. Without exact geometric constraints securely anchoring the 2D layout, LLMs cannot accurately reconstruct the table schema, leading to severe cascading errors during the SQL transformation phase. Therefore, the explicit spatial information provided by OCR is the critical bottleneck solved by STAR-Table, rather than relying solely on reasoning prompts.

J Hierarchical Direct Evaluation Metrics

To provide a comprehensive assessment of our table structure recognition capabilities, we supplement our main results with hierarchical direct evaluation metrics. This analysis evaluates the accuracy of the parsed HTML/JSON topology against the ground truth, with a specific focus on structural integrity across varying header hierarchy levels ($L = 1$, $L = 2$, and $L \geq 3$).

Experimental Setup and Metrics Evaluations were conducted on a subset of 500 tables from the HiTab dataset. To quantify structural reconstruction performance, we report the following metrics:

- **Tree-Edit-Distance-Based Similarity (TEDS-Struct):** Measures the tree-structure similarity between the parsed output and the ground truth.

- **Grid Table Similarity Topology (GriTS-Top F1):** Evaluates the topological correctness of the predicted table grid.

- **Schema Alignment Node-F1:** Assesses the accuracy of the hierarchical schema representation.

- **Cell Detection:** Evaluated via Containment Recall and Precision to measure the alignment (IoU) between the OCR output and the rendered image.

Results Across the evaluated 500 tables, our approach achieves reliable cell localization, yielding a **Containment Recall of 0.726** and a **Containment Precision of 0.804**.

Table 7 details the structural and topological evaluation metrics, stratified by the depth of the header hierarchy. The results demonstrate that the upstream structural reconstruction remains highly robust, maintaining strong performance even as table complexity increases to three or more hierarchy levels.

Table 7: Hierarchical topology metrics on the HiTab dataset, stratified by header hierarchy level.

Metric	Overall	L=2 (n=73)	L≥3 (n=454)
TEDS-Struct	0.804	0.831	0.800
GriTS-Top (F1)	0.850	0.885	0.844
Schema-Align Node-F1	0.886	0.893	0.885

K Error Analysis

To identify the current boundaries and bottlenecks of the STAR-Table framework, we conducted a rigorous manual error analysis on a random sample of 100 failure cases. By filtering out false negatives caused by formatting mismatches (e.g., exact string matching penalizing "Quebec" vs. "que", or "107" vs. "107.0"), we classified the genuine reasoning failures into three primary stages of our pipeline.

Category 1: OCR & Visual Recognition Failures (∼17%) While our padding and dual-engine strategies mitigate most visual errors, highly degraded document renderings or extremely dense cell layouts occasionally cause the OCR module to miss characters or merge adjacent text blocks. This upstream noise inherently corrupts the raw data before any reasoning can occur.

Category 2: Structural Parsing & Database Conversion Errors (∼37%) Even with accurate OCR coordinates, the LLM occasionally struggles to reconstruct implicit hierarchical relationships (e.g., deeply nested headers or indented sub-categories). This leads to misaligned columns or flattened schemas during the SQLite database instantiation.

- **User Query:** *"What was the percent change of the srr of injury hospitalization for unintentional falls among the aboriginal females?"*

- **Ground Truth:** `-57.9`

- **Model Output:** `-49`

- **Analysis:** The pipeline incorrectly linked the aggregated parent category ("aboriginal population", which has a value of `-49`) to the child node ("females") during the 2D-to-SQL structural parsing. This hierarchical flattening caused the subsequent SQL query to retrieve data from the wrong depth of the tree.

Crucially, our analysis reveals that the LLM's database conversion errors are heavily concentrated in this type of hierarchical loss. This indicates a strong inter-module dependency: enhancing the precision of the upstream OCR module (e.g., capturing micro-level pixel indentations or subtle spatial gaps) could provide stronger geometric constraints, thereby directly reducing the LLM's structural parsing error rate.

Category 3: SQL Generation & QA Reasoning Failures (∼48%) The vast majority of errors occur in the final reasoning phase. In these instances, the database is perfectly constructed, but the LLM generates flawed SQL logic or fails to align with the query's semantic intent. These failures primarily manifest as entity-value confusion or multi-hop mathematical calculation errors.

To better understand the limitations of our proposed framework, we conduct a qualitative error analysis on a sample of incorrect predictions. We categorize the predominant failure modes into the following figure 5.

L Prompt Templates

The specific system instructions and contextual user prompts for each distinct phase in our **STAR-Table** pipeline are detailed in this section. Variables dynamically injected into the prompts during execution are denoted by angle brackets, such as ⟨*Variable*⟩.

To provide a comprehensive overview without disrupting the two-column document flow, we summarize the six core prompts utilized across our framework below, with their complete configurations presented in the corresponding full-width figures:

- **Top Column Header Extraction (Figure 6):** Designed to extract the top hierarchical headers, this prompt explicitly injects the estimated column count as a hard geometric constraint to enforce the alignment of leaf nodes with data columns.
- **Left Row Header Extraction (Figure 7):** Utilizing the lowest y-coordinate of the resolved top headers as a structural boundary, this prompt isolates row headers while rigorously capturing any inherent hierarchical indentation structures.
- **Section Header Extraction (Figure 8):** This step employs a sequential heuristic logic flow, utilizing both visual emptiness (spatial heuristics) and structural repetition logic to isolate macroscopic category dividers within the table body.
- **VLM Semantic Summary (Figure 9):** Passed to the VLM alongside the rendered table image, this prompt extracts a domain-specific global semantic context to aid the downstream SQL execution phase, strictly avoiding the transcription of raw numerical data.
- **Text-to-SQL Generation (Figure 10):** For tables routed to the SQL pipeline, this prompt forces the LLM to ground its generation in explicitly provided SQLite schemas, HTML sample rows, and categorical value distributions, mitigating cross-table hallucinations.
- **Chain-of-Thought Answer Synthesis (Figure 11):** Utilizing a CoT verification protocol, this final prompt prioritizes deterministic SQL outputs while leveraging the VLM global summary and decomposed sub-query results for ambiguity resolution and calculation.

M Case Study

To provide a more transparent view of how STAR-Table operates across different table structures, we present four representative case studies. The first two cases illustrate successful end-to-end reasoning under small-table direct QA and large-table Text-to-SQL routing, while the latter cases analyze both a typical aggregation failure and a relational comparison success. All cases follow a unified template: source visual layout, structured schema extraction, and final reasoning or diagnosis.

Qualitative Error Case Taxonomy

Case 1: Column Semantic Misalignment

Question: "in northern ontario, what proportion of workers who have worked in the restaurant and food services sector was french-language workers?"

Wrong vs. Correct Answer: The model predicted `[48.6]`, while the reference answer is `[55.0]`.

Reason and Analysis: The relevant row is "food service", and the target column should be `northern_ontario_french_language_workers`, whose value is 55.0. The prediction 48.6 suggests that the model drifted to an incorrect demographic column or a derived proportion. This case reflects a column-selection failure in a table where region and worker-type dimensions are densely encoded in column headers.

Case 2: Retention of Hierarchical Provenance

Question: "which sedentary time has the largest beta coefficients?"

Wrong vs. Correct Answer: The model predicted `['sedentary time → weekend']`, while the reference answer is `['weekend']`.

Reason and Analysis: The model correctly identified that the maximum beta coefficient corresponds to the "weekend" node. However, it returned the full hierarchical path rather than the required leaf entity. This is not a reasoning failure over the numeric comparison, but an answer-normalization error where the system preserved excessive schema provenance instead of outputting the minimal semantic unit expected by the benchmark.

Case 3: Numeric Sorting and Top-k Ranking Failure

Question: "what are the top three industries among self-employed refugees?"

Wrong vs. Correct Answer: The model predicted `['transit and ground passenger transportation', 'truck transportation', 'retail trade']`, while the reference answer is `['lessors of real estate', 'services to buildings and dwellings', 'specialty trade contractors']`.

Reason and Analysis: The task requires ranking industries by the self-employed refugee percentage in descending order. The returned industries correspond to lower-value or mismatched rows, indicating a failure in numeric ordering or row-value alignment. This type of error may arise from treating numeric values as strings, generating an incorrect `ORDER BY` clause, or losing alignment across flattened sub-tables during top-k derivation.

Case 4: Upstream Structural Extraction Loss

Question: "how many passing yards did ryan tannehill get in 2018?"

Wrong vs. Correct Answer: The model predicted `[Data not available]`, while the reference answer is `[1979.0]`.

Reason and Analysis: The correct target is Ryan Tannehill's 2018 passing yards, which is 1979.0. The model abstained because the extracted relational schema did not preserve the specific "passing yards" metric clearly enough, instead exposing a more generic or ambiguous passing-related column. This suggests that the failure originates from upstream table-structure extraction rather than purely from downstream reasoning.

Case 5: Spurious Metric Selection

Question: "how many million dollars did funding for research and development to universities and colleges increase in 2017?"

Wrong vs. Correct Answer: The model predicted `[1265]`, while the reference answer is `[29796.0]`.

Reason and Analysis: The question asks for the 2017 funding amount for research and development to universities and colleges, whose correct value is 29796.0 million dollars. The model instead selected 1265, likely from an adjacent metric, a difference column, or another nearby funding category. This illustrates a spurious metric-selection error when multiple semantically similar numeric fields appear in the same table region.

Case 6: Over-Abstention Induced by Ambiguity

Question: "what was the mean absolute revision to the growth in annual real GDP for 1982 to 2014?"

Wrong vs. Correct Answer: The model predicted `[]`, while the reference answer is `[0.11]`.

Reason and Analysis: The correct cell corresponds to "mean absolute revision to the growth in annual real GDP" for the period "1982 to 2014", with value 0.11. The model appeared to locate a plausible candidate but abstained because the schema phrasing did not exactly match the natural-language query. This reflects over-conservative rejection under lexical ambiguity, where strict schema matching overrides reasonable semantic alignment.

Case 7: Surface-Form Normalization Penalty

Question: "what the matches did gremio participate in?"

Wrong vs. Correct Answer: The model predicted `['Campeonato Brasileiro Série A', 'Copa do Brasil', 'Copa Libertadores', 'Primeira Liga/Campeonato Gaúcho']`, while the reference answer is `['campeonato brasileiro serie a', 'copa do brasil', 'copa libertadores', 'primeira liga / campeonato gaucho']`.

Reason and Analysis: The predicted entities are semantically equivalent to the reference tournaments, but differ in casing, accents, and punctuation. The reference uses a lowercase, accent-stripped, normalized format, whereas the model outputs the more natural surface form. This case highlights an evaluation artifact caused by strict exact match, rather than a substantive factual reasoning error.

Case 8: Asymmetric Resolution of Implicit Entities

Question: "what were the percent changes of the ssr of injury hospitalization for all unintentional causes among the aboriginal population and the total population respectively?"

Wrong vs. Correct Answer: The model predicted `[-59.9, No data]`, while the reference answer is `[-59.9, -49.3]`.

Reason and Analysis: The model correctly retrieved the Aboriginal population value, -59.9, but failed to map "total population" to the table-specific row alias "total british columbia", whose corresponding value is -49.3. This demonstrates an asymmetric entity-resolution failure: the explicit group was resolved successfully, but the implicit comparative group required domain-specific alias matching within a hierarchical row structure.

Figure 5: Representative error cases grouped into eight failure archetypes. The full-width two-column layout follows the prompt and case-study template, enabling compact comparison of question intent, model prediction, reference answer, and failure diagnosis.

Prompt 1: Top Column Header Extraction

Role: You are an expert in Table Structure Recognition.
Task: Extract *exclusively* the Top Column Headers.
Input Modalities:

1. **OCR List:** Visual boxes.
   ⟨*Inserted OCR Bounding Box List with IDs and Coordinates*⟩
2. **Reference Text:**
   ⟨*Inserted Ground Truth Raw Header Texts*⟩
3. **Geometric Prior:** The table body appears to have roughly ⟨**Estimated Column Count**⟩ columns of data.

Crucial Constraints:

1. **Spatial Localization:** Top Headers are strictly located at the uppermost region (minimal y-coordinates).
2. **Column Exclusion:** Elements residing in the leftmost column (minimal x-coordinates) serve as Row Headers and must be explicitly *ignored*.
3. **Leaf-Node Alignment Protocol:**
   - Within hierarchical headers, the terminal sub-headers (leaf nodes) must maintain a one-to-one correspondence with the underlying numerical data columns.
   - *Validation Mechanism:* The total count of extracted leaf-node items must align with the geometric prior ($\approx$ ⟨*Estimated Column Count*⟩).
   - *Example:* If 5 columns are estimated and only the parent node "Year" is extracted (1 item), its corresponding children (e.g., "Q1", "Q2") must also be explicitly delineated.

Output Specification: JSON list of objects strictly adhering to the schema: `{"text", "ids", "parent_text"}`.

Figure 6: The system prompt for Top Column Header Extraction. It incorporates spatial geometric priors (e.g., coordinate localization) and logical constraints (e.g., leaf-node alignment) to enforce strict topological extraction of the top hierarchical elements.

Prompt 2: Left Row Header Extraction

Role: You are an expert in Table Structure Recognition.
Task: Extract the Left Row Headers (stub column), rigorously capturing any inherent hierarchical structures.
Input Modalities:

1. **OCR List:** Visual boxes.
   ⟨*Inserted OCR Bounding Box List with IDs and Coordinates*⟩
2. **Contextual Anchors:**
   - Top Header IDs: ⟨*List of Identified Top Header IDs*⟩
   - **Structural Boundary:** Top Headers terminate at $y = \langle$*Top Header Bottom Y-Coordinate*$\rangle$.
   - **Geometric Prior:** The table body appears to have roughly ⟨**Estimated Row Count**⟩ rows of data.

Crucial Constraints:

1. **Spatial Constraint:** Selections are strictly limited to elements positioned below the structural boundary ($y >$ ⟨*Top Header Bottom Y-Coordinate*⟩).
2. **Horizontal Positioning:** Target the leftmost textual elements within each corresponding row.
3. **Hierarchical Indentation:**
   - Row headers frequently exhibit hierarchy (e.g., "North America" → "USA").
   - Sub-headers may present horizontal indentation shifts (shifted right), but functionally remain Row Headers.
4. **Leaf-Node Alignment Protocol:**
   - The total count of terminal row headers (leaf nodes) must mathematically correspond to the geometric prior ($\approx \langle$*Estimated Row Count*$\rangle$).
   - Indented child nodes must not be omitted.

Output Specification: JSON list of objects strictly adhering to the schema: `{"text", "ids", "parent_text"}`.

Figure 7: The system prompt for Left Row Header Extraction. This prompt utilizes the computed bounding y-coordinate of the top headers to constraint the spatial search region for left headers, addressing internal hierarchical structures.

Prompt 3: Section Header Extraction

Role: You are an expert in Table Structure Recognition.
Task: Identify intra-table Section Headers (macroscopic category dividers).
Contextual Anchors:

- Top Headers: ⟨*List of Identified Top Header IDs*⟩
- Left Headers: ⟨*List of Identified Left Header IDs*⟩ (Validated Row Headers)

Input Modality:
⟨*Inserted OCR Bounding Box List with IDs and Coordinates*⟩
Heuristic Decision Flow: Evaluate candidates in strict systematic order.

Phase A: Visual Isolation (High-Confidence Candidates)
Identify boundaries utilizing spatial emptiness.

1. **Right-Side Sparsity:** The candidate exhibits substantial empty space unequivocally to its right, lacking data entries.
2. **Structural Lone Nodes:** The candidate occupies an entire row independently.
3. **Exclusion Criterion:** Strictly exclude the global table title (typically localized at the minimal y-coordinate). Section Headers are confined to the table body.

Phase B: Structural Periodicity (Ambiguity Resolution)
Apply cautiously for candidates missed by Phase A (e.g., headers with adjacent aggregations).

1. **Semantic Uniqueness (Critical Filter):**
   - Candidates accompanied by adjacent data must be strictly **UNIQUE** within the local scope.
   - **Discard globally repetitive terms:** Statistical terms (e.g., "Total", "Subtotal", "Average") or repeating items (e.g., "Revenue" appearing 5 times) are NOT Section Headers. They are Data Rows.
   - **Accept unique categorical blocks:** Terms like "North America", "2024", "Segment A" (acting as block titles) can be Section Headers despite adjacent data.
2. **Vertical Sub-group Periodicity:**
   - **Strict Vertical Alignment:** Examine the vertical axis within the identical x-coordinate span.
   - **Pattern Definition:** A valid Section Header serves as the parent to a specific sequence of sub-items that semantically repeats beneath parallel headers.
3. **Exclusion Criterion:** A putative header must span and group **multiple underlying records**; singleton wrappers corresponding to a single row are invalid.

Output Specification: Standard JSON list encompassing string IDs.
Note: If no section headers satisfy these geometric and semantic criteria, explicitly return an empty list `[]`. Do not force a selection, as non-sectioned tables are prevalent.

Figure 8: The system prompt for Section Header Extraction. A sequential heuristic logic flow is adopted to systematically isolate macroscopic category dividers via spatial heuristics and structural periodicity checks.

Prompt 4: VLM Semantic Summary

Role: You are an expert Vision-Language Model.
Task: Analyze the overarching semantic context of the provided table image to facilitate downstream Text-to-SQL querying on a flattened relational schema.
Deliverable: Synthesize a **concise domain-specific formulation summary** (100–150 words).
Strict Guidelines:

1. **Data Exclusion:** Strictly refrain from transcribing specific numerical cell values or enumerating row instances.
2. **Layout Abstraction:** Since the target schema is topologically flattened, avoid describing physical spatial arrangements (e.g., "merged cells", "nested rows").
3. **Semantic Focus:** Concentrate purely on business logic, categorical dimensions, and core concepts.

Required Analytical Dimensions:

1. **Global Context:** Formulate the primary subject matter and the spatio-temporal scope (e.g., "Life expectancy demographics in Canada, 2011").
2. **Column Concept Dimensions:** Abstract the visual headers into categorical attributes (e.g., "Variables are segmented across Population Groups and Biological Sexes").
3. **Row Granularity:** Define the semantic identity of an individual logical record (e.g., "Each row quantifies metrics for a specific Age Group cohort").
4. **Metric Definition:** Specify the unit of measurement governing the continuous data distributions (e.g., "Values assert chronological life expectancy in years").

Objective: This summary aims to ground the SQL generator in real-world nomenclature, explicitly resolving abbreviations or implicit domain variables (e.g., extrapolating "inuit_le" or "male_le").

Figure 9: The system prompt for VLM Semantic Summary generation. This isolates descriptive visual layout characteristics and enforces purely abstract semantic extraction across global context, entity definitions, and metrics, essential to anchor symbolic multi-step reasoning.

Prompt 5: Schema-Grounded Text-to-SQL Generation

Role: You are a precise Data Scientist.
Task: Write a SQLite SQL query to answer the user question. The database contains multiple sub-tables. You must identify the relevant table(s) and columns to execute the query.
Input Modalities (For each sub-table):

1. **Schema Definition:** DDL statements detailing column types (e.g., `REAL` for numeric calculation, `TEXT` for strings).
2. **Categorical Values:** ⟨*Exhaustive List of Unique Values in the Row Index Column*⟩
3. **Sample Rows:** A serialized HTML table providing a visual and structural preview of the data format.

Contextual Metadata Hints:

- The explicitly defined Row Name column typically acts as the semantic index.
- Indentation markers (e.g., "→") in the Row Name column strictly imply hierarchical parent-child relationships.

Crucial Constraints:

1. **Schema Linking Rule:** First, explicitly identify and declare which specific table and columns map to the semantic concepts within the user question.
2. **Exact Value Matching:** When filtering strings via the `WHERE` clause, you MUST utilize the exact string formats provided in the categorical value lists. (e.g., If the list contains 'females → 1', do not write '1 year old female' in the SQL).

Output Specification:

- **Constraints:** ⟨*Explicitly state the Table, Column, and Value constraints identified*⟩
- **SQL:** ⟨*The executable SQLite query*⟩

Figure 10: The system prompt for Text-to-SQL Generation. It integrates deterministic schema definitions, HTML samples, and strict value-matching constraints to mitigate hallucinations during cross-table SQL formulation.

Prompt 6: Chain-of-Thought Final Answer Synthesis

Role: You are an expert Data Analyst.
Task: Synthesize the provided deterministic query results and contextual metadata to formulate a precise final answer to the User Query.
Input Modalities:

1. **Table Background Summary:** ⟨*Global semantic context generated by the VLM*⟩
2. **Main Executed Logic:**
   - SQL Executed: ⟨*The primary SQL query*⟩
   - Resulting Sub-table: ⟨*Markdown representation of the SQL execution output*⟩
3. **Extra Information (Decomposed Sub-queries):** ⟨*Results from prior multi-hop reasoning steps, if applicable*⟩

Chain of Thought (CoT) Verification Protocol:

1. **Analyze Result:** Evaluate the Main Sub-table Result. Does it contain the target answer or is it empty?
2. **Contextual Fallback:** If the primary result yields "No Data", consult the Extra Information or the Table Background Summary for implicit answers.
3. **Mathematical Verification:** If the query necessitates arithmetic (e.g., sum, difference, percentage) and the sub-table only yields raw numerical components, perform the explicit calculation.
4. **Trust Decision Rule:** If the Main Sub-table Result contains a valid value, **TRUST IT** over any contradictory "No data" claims in the Extra Information.

Output Specification:

- **Thought:** ⟨*Step-by-step reasoning explicitly following the verification protocol*⟩
- **Answer:** ⟨*A natural language sentence encapsulating the finding*⟩
- **[Direct Answer]:** ⟨*A strict list format, e.g., [15.5] or ['canada']*⟩

Figure 11: The system prompt for Final Answer Synthesis. It enforces a strict logical hierarchy, mandating the LLM to prioritize the deterministic output of the main SQL execution while employing contextual metadata for ambiguity resolution and multi-hop mathematical calculations.

End-to-End Case Study: Resolving Deep Hierarchical Nesting (Table 112)

[Phase 1: Source Visual Table (2D Layout)]
The raw table exhibits a complex 2D structure with spanning column headers (e.g., Woodlands/Wetlands split into hectares and percent) and implied row hierarchies.

Province	Area of Agri. Land	Cropland		Natural Land for Pasture		Woodlands and Wetlands	
	hectares	*hectares*	*percent*	*hectares*	*percent*	*hectares*	**percent**
Canada	64,812,723	35,350,270	54.5	14,703,330	22.7	4,897,367	7.6
Atlantic provinces	1,063,343	430,363	40.5	65,711	6.2	450,031	42.3
Quebec	3,341,333	1,874,760	56.1	134,147	4.0	1,057,417	31.6
Ontario	5,126,653	3,613,821	70.5	398,538	7.8	646,578	**12.6**
Manitoba	7,293,839	4,348,869	59.6	1,466,968	20.1	549,444	7.5

[Phase 2: Physical Parsing to Semantic Schema (Log Extraction)]
Our extraction module autonomously parses the visual coordinates and flattens the hierarchy into strict semantic paths via concatenation, constructing the schema.

```
[System Log] Processing File ID: 112 | Structure Extraction Module

[Top Headers]
area of agricultural land -> hectares
cropland -> hectares
...
woodlands and wetlands -> hectares
woodlands and wetlands -> percent [Target Column]

[Left Headers]
canada
atlantic provinces
quebec
ontario [Target Row]
manitoba

[Flattened Target Field]
woodlands_and_wetlands_percent = 12.6
```

[Phase 3: Autonomous Reasoning & Question Answering (Authentic Log)]
*The framework features a dynamic routing mechanism. For small-scale tables, it bypasses SQL generation and utilizes a **Direct QA mode**, leveraging the explicitly flattened schema to perform zero-hallucination text reasoning.*

[System Routing Strategy]: `Small Table (Total Rows: 7). Using Direct QA (No SQL, No Decompose).`

User Query: *"what was the percentage of agricultural land in the ontario provinces reported as woodlands and wetlands in 2011?"*

LLM Thought Process (Extracted directly from logs):
`Thought: The user is asking for the percentage of agricultural land in Ontario that was reported as woodlands and wetlands in 2011. From the provided table, each row represents a province, and Ontario is one of them. The column "woodlands_and_wetlands_percent" gives the percentage of agricultural land in each province that is classified as woodlands and wetlands. For Ontario, this value is found directly in the "woodlands_and_wetlands_percent" column.`

Final Output: `[12.6]` **Reference Ground Truth:** `[12.6]` **(✓Correct)**

Figure 12: End-to-end trace of the STAR-Table pipeline on Table 112. The log demonstrates how our framework dynamically routes small tables to a Direct QA strategy. By decoupling the 2D layout into explicit semantic paths (Phase 2), the LLM can precisely anchor its reasoning (Phase 3) onto the correct synthesized column name (`woodlands_and_wetlands_percent`), completely neutralizing the visual distance between nested headers.

End-to-End Case Study: Large-Scale Table with Sub-Table Schema (Table 1397)

[Phase 1: Source Visual Table (2D Layout Fragment)]
The raw table exhibits a complex 2D structure with multiple sub-sections and dozens of rows. Standard LLMs inevitably fail due to context dilution and token limits.

Incident Characteristics	Unwanted Sexualized Behaviours			Discriminatory Behaviours		
	Percent	95% Confidence Interval		Percent	95% Confidence Interval	
		from	*to*		*from*	*to*
Number of perpetrators						
Always one	24	19	30	37	28	46
Always two or more	24	19	30	22	15	32
Bystanders present						
Yes, in at least one instance	**88**	84	92	75	66	82
No, in at least one instance	**64**	58	70	45	36	55
Don't know	7	4	11	f	..	..
Bystanders took action						
Yes, in at least one instance	40	34	47	34	24	45
No, in at least one instance	87	82	91	83	73	89

[Phase 2: Semantic Chunking & Relational Database Construction]
Instead of feeding a massive table directly to the LLM, the extraction module autonomously segments the rows into distinct database sub-tables based on the automatically detected ***Section Headers*** *[IDs: 14, 43, 72...].*

```
[System Log] Processing File ID: 1397 | Structure Extraction Module
[Top Headers] unwanted sexualized behaviours -> percent (IDs: [7]) [Target DB Column] unwanted sexualized behaviours
(IDs: [4]) 95unwanted sexualized behaviours -> 95...
[Left Headers] number of perpetrators (IDs: [14]) number of perpetrators -> always one (IDs: [15]) ... bystanders
present (IDs: [72]) bystanders present -> yes, in at least one instance (IDs: [73]) [Target DB Row A] bystanders
present -> no, in at least one instance (IDs: [80]) [Target DB Row B] ... relationship of perpetrator (IDs: [170])
...
[Section Headers] IDs: [14, 43, 72, 94, 114, 170] [Sub-table Splitting Anchors]
```

[Phase 3: Autonomous Multi-Step Reasoning (Chain-of-Thought Workflow)]
The agent employs a rigorous cognitive framework (Step-Back → Decompose → Link → SQL) to prevent hallucination during complex cross-row arithmetic operations.

[System Routing Strategy]: `Large Table (Total Rows: 38). Using Text-to-SQL.`

User Query: *"What is the absolute difference in percentage between students who reported bystanders were present (in at least one instance) and those who reported bystanders were NOT present during unwanted sexualized behaviours?"*

Agent Internal Execution Log:

`[Step 0: Step-Back Abstraction]`
» Paraphrased Goal: What are the respective percentages of bystander presence ('yes' vs. 'no') during unwanted sexualized behaviours?

`[Step 1: Logical Decomposition]`
» Sub-task 1: Retrieve percent value for 'bystanders present -> yes, in at least one instance' (Target A)
» Sub-task 2: Retrieve percent value for 'bystanders present -> no, in at least one instance' (Target B)
» Sub-task 3: Compute ABS(Target A - Target B)

`[Step 2 & 3: Table Routing & Main SQL Generation]`
» Schema Linked: Target Sub-table → "table_bystanders_present"
» Generated Executable SQL:

```
SELECT ABS(
MAX(CASE WHEN row_name = 'bystanders present -> yes, in at least one instance' THEN
unwanted_sexualized_behaviours_percent END) -
MAX(CASE WHEN row_name = 'bystanders present -> no, in at least one instance' THEN
unwanted_sexualized_behaviours_percent END)
) FROM table_bystanders_present;
```

`[Step 4: Final Output Validation]`
» SQL Engine Result: | ABS_Diff | → | 24 |

Final Answer: `[24]` **Reference Ground Truth:** `[24.0]` **(✓Correct)**

Figure 13: End-to-end trace of the STAR-Table pipeline on Table 1397. For large-scale tables, the framework dynamically switches to a **Text-to-SQL strategy**. Phase 2 detects `Section Headers` to chunk the massive 2D layout into highly cohesive relational sub-tables (e.g., `table_bystanders_present` via ID 72). This schema isolation enables the LLM (Phase 3) to execute precise arithmetic queries (calculating the 24% absolute difference between the 88% and 64% target coordinates) without context hallucination.

Error Analysis Case Study: Aggregation Misalignment (Table 2610)

[Phase 1: Source Visual Table (2D Layout Fragment)]
This table displays employment statistics distributed across three distinct terciles. The semantic challenge lies in aggregating horizontal data points (a SUM operation) rather than pinpointing a single coordinate.

Profile of Economic Regions (2003-2013)	Tercile		
	Lower	*Middle*	*Upper*
Number			
Economic regions	**23**	**23**	**23**
Thousands			
Average number of employees aged 18 to 64	360	171	59
Percent			
Layoff rate	4.7	6.8	12.6
Hiring rate	19.9	21.1	23.8

[Phase 2: Semantic Structure Extraction (Authentic Log)]
The extraction module successfully parses the complex 2D hierarchy. It perfectly isolates the target entities, proving that the upstream visual parsing and relational mapping functioned flawlessly.

```
[System Log] Processing File ID: 2610 | Structure Extraction Module
[Top Headers] economic regions (IDs: [7]) [Target Entity] tercile -> lower (IDs: [3]) [Target Vector Space] tercile
-> middle (IDs: [4]) [Target Vector Space] tercile -> upper (IDs: [5]) [Target Vector Space] ...
[Left Headers] average number of employees aged 18 to 64 (IDs: [12]) layoff rate (IDs: [17]) hiring rate (IDs: [21])
percentage of employees -> with 24 months of tenure or less (IDs: [26]) percentage of employees -> in manufacturing
(IDs: [50]) ...
```

[Phase 3: Failure Diagnosis - The Aggregation Gap]
*Despite flawless structural retrieval, the final Agent failed at the semantic aggregation boundary. The query asks for the **total count** of regions, which requires an implicit SUM operation over the extracted vector.*

User Query: *"How many ERs were sorted on the basis of the average annual layoff rate over the 2003-to-2013 period and grouped into those in the upper-, middle- and bottom-third of the layoff rate distribution?"*

SQL Execution (Flawless Retrieval):
`SELECT "tercile_lower", "tercile_middle", "tercile_upper" FROM "main_table" WHERE "row_name" = 'economic regions';`
`Result Vector: | 23 | 23 | 23 |`

Agent's Natural Language CoT (Implicit Calculation Successful):
`Thought: The question asks how many ERs were sorted... The Main Sub-table Result directly answers this, showing 23 ERs in each tercile group. A total of 69 economic regions were sorted... with 23 regions grouped into each...`

Final Metric Extraction Mismatch (Format Degradation):
`[Direct Answer Output]:` **`[23, 23, 23]`** **(✗ Evaluation Metric Failure)**
`[Reference Ground Truth]:` **`[69]`**

Discussion: The framework exhibits an "Aggregation Format Mismatch". As shown in Phase 2 and the SQL log, the system correctly localized and retrieved the required vector. Furthermore, the LLM successfully performed the implicit addition (23+23+23=69) in its natural language Chain-of-Thought. However, constrained by rigid formatting protocols (`[Direct Answer]`), the agent conservatively reverted to outputting the raw, unaggregated array `[23, 23, 23]`, triggering an exact-match penalty. This exposes a fundamental limitation in mapping derived mathematical concepts into strict evaluation templates.

Figure 14: Negative Case Analysis on Table 2610. This example highlights a critical evaluation boundary. STAR-Table perfectly extracted the structural coordinates (Phase 2) and formulated the correct retrieval SQL (Phase 3). Paradoxically, while the LLM successfully reasoned the sum (69) in free text, it output the raw array `[23, 23, 23]` in the final structured block. This underscores the necessity for more robust post-processing heuristics when handling scalar aggregation tasks.

End-to-End Case Study: Hierarchical Row Schema with Relational Comparison (Table 2404)

[Phase 1: Source Visual Table (2D Layout Fragment)]
The raw table contains dozens of hierarchical row categories describing demographic and administrative attributes that affect the likelihood of being missed in the National Population Registry. The target answer requires identifying the correct age-group rows and comparing their odds ratios.

Characteristics	Univariate Models Odds Ratios	Multivariate Model
Age group		
Under 20 years	1.91	1.77
20 to 24 years	2.14	1.95
25 to 29 years	2.08	**1.88**
30 to 34 years	1.00	**1.00**
35 years and over	1.30	1.35

```
[System Log] Processing File ID: 2404 | Structure Extraction Module

[Relational Schema]
CREATE TABLE “main_table” (
id INTEGER PRIMARY KEY,
row_name TEXT,
univariate_models_odds_ratios TEXT,
multivariate_model TEXT
);

[Hierarchical Row Values]
age group
age group -> under 20 years
age group -> 20 to 24 years
age group -> 25 to 29 years [Target Row A]
age group -> 30 to 34 years [Target Row B]
age group -> 35 years and over
sex -> male
sex -> female
region of birth -> europe
...
```

[Phase 3: Autonomous Multi-Step Reasoning (Chain-of-Thought Workflow)]
The agent switches to Text-to-SQL because the table has 47 rows. The task is not arithmetic subtraction, but a relational comparison: retrieve the two candidate odds ratios and select the row with the larger value.

[System Routing Strategy]: `Large Table (Total Rows: 47). Using Text-to-SQL.`

User Query: *“Who were significantly more likely to be missed, respondents aged 25 to 29 years or those aged 30 to 34?”*

Agent Internal Execution Log:

```
[Step 0: Step-Back Abstraction]
» Paraphrased Goal: What are the odds ratios for being missed in the NPR for each candidate age group?
[Step 1: Logical Decomposition]
» Sub-task 1: Retrieve the odds ratio for respondents aged 25 to 29 years.
» Sub-task 2: Retrieve the odds ratio for respondents aged 30 to 34 years.
» Sub-task 3: Compare the two odds ratios and select the larger one.
[Step 2 & 3: Schema Linking & Main SQL Generation]
» Schema Linked: Target Table → "main_table"
» Target Column → "multivariate_model"
» Target Rows → "age group -> 25 to 29 years", "age group -> 30 to 34 years"
» Generated Executable SQL:
SELECT "row_name", "multivariate_model"
FROM "main_table"
WHERE "row_name" IN ('age group -> 25 to 29 years',
'age group -> 30 to 34 years');
[Step 4: Final Output Validation]
» SQL Engine Result:
| age group -> 25 to 29 years | 1.88 |
| age group -> 30 to 34 years | 1.00 |
» Since 1.88 > 1.00, the selected answer is the 25 to 29 years group.
```

Final Answer: `['25 to 29 years']` **Reference Ground Truth:** `['25 to 29 years']` **(✓Correct)**

Figure 15: End-to-end trace of the STAR-Table pipeline on Table 2404. The table encodes hierarchical demographic attributes as flattened row paths, enabling precise linking from the query phrase “respondents aged 25 to 29 years” to `age group -> 25 to 29 years`. The Text-to-SQL agent retrieves the multivariate-model odds ratios for the two candidate rows and performs a pairwise comparison, selecting the age group with the larger odds ratio (1.88 vs. 1.00).