HiR-MKR: Hierarchical Reconstruction and Multi-Source Knowledge Retrieval for Semi-Structured Table Question Answering

Anonymous authors
Paper under double-blind review

Abstract

Semi-structured tables are widely used in real-world applications, but complex layouts such as multi-level headers, merged cells, and nested subtables pose significant challenges to table question answering. ST-Raptor provides structured representations through Hierarchical Orthogonal Trees (HO-Trees), but their limited explicit hierarchical semantics and lack of complementary knowledge representations constrain effective retrieval and reasoning. We propose HiR-MKR, a semi-structured table question answering framework that integrates hierarchical structure reconstruction with multi-source knowledge retrieval. Table Adaptive Hierarchical Reconstruction Algorithm (TAHRA) reconstructs explicit hierarchical structures from HO-Trees and provides a unified basis for a knowledge graph, a Record vector store, and a Path vector store. HiR-MKR further performs query decomposition and type-aware, dependency-driven retrieval and reasoning, with structure reliability assessment to control reliance on reconstructed knowledge. Experiments on WikiTQ-ST, TempTabQA-ST, and SSTQA show that HiR-MKR consistently outperforms existing methods, achieving 82.72% accuracy on SSTQA and outperforming ST-Raptor by 10.33 percentage points. It also reduces average response time and total token consumption by 59.29% and 73.00%, respectively.

1 Introduction

Semi-structured tables are widely used in real-world applications such as finance, government, healthcare, and enterprise management. Their complex layouts, including multi-level headers, merged cells, and nested subtables, pose significant challenges to Table Question Answering (TableQA), where effective answering requires not only understanding table content but also exploiting structural information for retrieval and multi-step reasoning (Jin et al., 2022).

Early TableQA methods mainly followed semantic parsing paradigms, mapping natural language questions to executable logical forms (Pasupat & Liang, 2015), while related Text-to-SQL methods translated questions into executable SQL queries(Guo et al., 2019). Pretrained models such as TaBERT (Yin et al., 2020) and TAPAS (Herzig et al., 2020) further incorporated table structure into language models. However, these approaches generally rely on predefined schemas or linearized representations, making it difficult to capture the hierarchical and spatial semantics of complex semi-structured tables. More recently, ST-Raptor (Tang et al., 2025) introduced HO-Tree and atomic operations for structured TableQA, providing an effective representation for complex table layouts. Nevertheless, its knowledge organization and retrieval mechanism still have several limitations. First, hierarchical semantics are primarily embedded in HO-Tree nodes and paths rather than explicitly represented, limiting their use in semantic retrieval and reasoning. Second, its knowledge representation mainly focuses on tree structure and table content, providing limited explicit modeling of entity-level relations and cross-level multi-hop associations. Third, retrieval remains primarily centered on HO-Tree operations, without explicitly scheduling complementary knowledge sources according to subquestion types and knowledge requirements.

To address these limitations, we propose HiR-MKR (Hierarchical Reconstruction and Multi-Source Knowledge Retrieval), a semi-structured TableQA framework that integrates hierarchical structure

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reconstruction with multi-source knowledge collaboration. The core idea is to reconstruct explicit hierarchical semantics from HO-Tree and use the resulting structure as a unified basis for complementary knowledge representations and query-adaptive retrieval. Specifically, we first propose TAHRA to identify implicit hierarchical fields and aggregate corresponding nodes into explicit parent-child structures. Based on the reconstructed hierarchy, we construct three complementary knowledge stores: a knowledge graph, a Record vector store, and a Path vector store, which capture relational, content-level, and hierarchical-path information, respectively. For question answering, HiR-MKR decomposes complex questions into subquestions, selects knowledge sources according to subquestion types, assesses the reliability of the reconstructed structure, and organizes multi-step reasoning through subquestion dependencies. Our main contributions are summarized as follows:

- We propose **HiR-MKR**, a unified semi-structured TableQA framework that integrates hierarchical structure reconstruction, multi-source knowledge representation, and collaborative retrieval and reasoning.
- We propose **TAHRA**, which identifies implicit hierarchical fields and aggregates HO-Tree nodes to reconstruct explicit parent-child relationships, enhancing the hierarchical semantics of complex tables.
- We develop a complementary knowledge and retrieval mechanism that integrates a knowledge graph, a Record vector store, and a Path vector store, together with type-aware retrieval, structure reliability assessment, and dependency-driven reasoning for adaptive multi-step question answering.
- We conduct comprehensive experiments on WikiTQ-ST, TempTabQA-ST, and SSTQA. HiR-MKR consistently outperforms existing methods and demonstrates substantial improvements in both question answering accuracy and inference efficiency.

2 RELATED WORK

This section briefly introduces semi-structured tables, HO-Tree and the semi-structured table question answering task, providing the necessary background for the proposed method.

2.1 SEMI-STRUCTURED TABLES

Semi-structured tables extend conventional two-dimensional tables with complex structural patterns, including multi-level headers, merged cells, nested subtables, and orthogonal layouts. Their semantics arise jointly from cell contents, spatial layouts, and hierarchical relationships. ST-Raptor(Tang et al., 2025) categorizes semi-structured tables into four basic layout types, L1–L4, as illustrated in Figure 1. L1 and L2 represent basic header-to-data relationships, whereas L3 and L4 describe more complex structures, such as parallel subtables and shared headers. In practice, a table may combine multiple layout types. We follow this layout taxonomy throughout this work.

2.2 HIERARCHICAL ORTHOGONAL TREE

HO-Tree proposed by ST-Raptor(Tang et al., 2025) is a tree-based representation for semi-structured tables that unifies table-header hierarchies, data contents, and their associations. An HO-Tree consists of a Meta Tree (MTree) and a Body Tree (BTree). The MTree represents table headers and their hierarchical semantics, where each root-to-leaf path corresponds to the hierarchical semantics of a field. The BTree organizes table data, where each root-to-leaf path corresponds to a data record, and node mappings establish semantic associations between fields and data. For tables containing orthogonal layouts or nested subtables, an HO-Tree can be further embedded within BTree nodes, enabling recursive representation of complex table structures. HO-Trees therefore provide a unified representation of header hierarchies, data contents, and nested structures, forming the basis for structural understanding and question answering over semi-structured tables. In this work, we further reconstruct the implicit hierarchical information encoded in HO-Trees into explicit parent-child relationships, as detailed in Section3.2.

Layout Type	Description	Example
L1:Herder-Single-Value	The table contains a single header associated with a single value.	Name | Job
L2:Header-Multiple-Vlaues	The table contains a single header associated with multiple values.	Name | Job | Lisa | Lily
L3:Orthogonal Tables	The table consists of multiple independent sub-tables, each containing its own header and corresponding values.	Name | Job Department | Finance Tel | 3599826
L4:Header-Orthogonal-Tables	The table contains a shared header and one or more associated orthogonal sub-tables through key-value relationships.	Info Name | Tel Job | 3599826 Lisa | 3587542 Lily | 3565129

Figure 1: Schematic illustration of semi-structured table layouts.

2.3 Semi-Structured Table Question Answering

Semi-Structured Table Question Answering (STQA) aims to retrieve relevant information from complex semi-structured tables, perform the required reasoning, and generate answers to natural language questions. Given a semi-structured table $\mathcal{T}$, a natural language question q, and its corresponding answer a, the STQA task can be formally defined as: $a = f(\mathcal{T}, q)$, where $f(\cdot)$ represents the question answering process, including information retrieval and reasoning over the table.Based on the primary knowledge requirements of question answering, we categorize STQA into three types:

- **Content**: focuses on entity attributes and factual information, e.g., "What is Zhang San's contact information?"
- **Structure**: focuses on hierarchical or subordinate relationships among entities or fields, e.g., "How many second-level indicators are included in the performance indicators?"
- **Reasoning**: involves operations such as counting, summation, and comparison, and typically requires jointly leveraging content and structural information, e.g., "What is the total monthly revenue in the first quarter?"

This taxonomy provides the basis for the type-aware multi-source collaborative retrieval described in Section3.4.

3 METHODOLOGY

3.1 Overview

Figure 2 illustrates the overall architecture of HiR-MKR, which consists of two main stages: (1) hierarchical structure reconstruction and multi-source knowledge representation, and (2) multi-source collaborative retrieval and reasoning. In the first stage, TAHRA reconstructs the implicit hierarchical structure of the original HO-Tree and builds a multi-source table knowledge base based on the resulting representation. In the second stage, the system analyzes the user's natural language question, performs structure-aware and type-aware retrieval, and conducts dependency-driven reasoning to generate the final answer. The overall pipeline can be summarized as: **hierarchical reconstruction** $\rightarrow$ **knowledge representation** $\rightarrow$ **query analysis** $\rightarrow$ **structure assessment** $\rightarrow$ **collaborative retrieval** $\rightarrow$ **dependency-driven reasoning** $\rightarrow$ **answer generation**.

Formally, let T denote an input semi-structured table and Q denote a natural language question. The table is first parsed into an HO-Tree H. TAHRA reconstructs the implicit hierarchical semantics in H to obtain a hierarchy-enhanced HO-Tree H_h. Based on H_h, we construct a multi-source

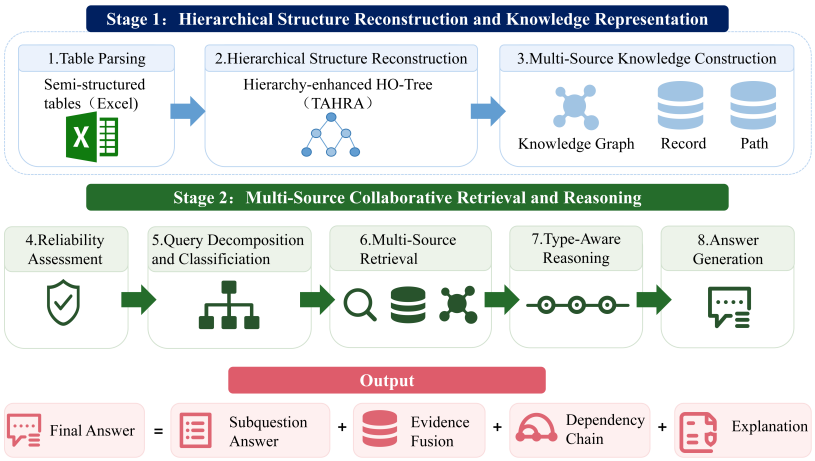

Figure 2: Overview of the HiR-MKR Framework.

knowledge base:

$$\mathcal{K} = \{\mathcal{G}, \mathcal{V}_R, \mathcal{V}_P\}, \tag{1}$$

where $\mathcal{G}$ denotes the table knowledge graph, while $\mathcal{V}_R$ and $\mathcal{V}_P$ denote the Record and Path vector stores, respectively. These three sources provide complementary representations of relational, content, and hierarchical-path knowledge.

Given Q, an LLM first decomposes it into a set of subquestions $Q_s = \{q_1, q_2, \ldots, q_n\}$, assigns a query type t_i to each subquestion, and constructs a dependency graph D describing their information dependencies. The system then retrieves evidence from appropriate knowledge sources according to the requirements of each subquestion and organizes information transfer and reasoning according to D. Let $\mathcal{E}_i$ denote the evidence retrieved for subquestion q_i. The final answer is generated by jointly considering the original question, subquestions, dependencies, evidence, and intermediate answers:

$$a = \mathrm{LLM}(Q, Q_s, D, \mathcal{E}, A_s), \tag{2}$$

where $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ denotes the evidence set and A_s denotes the set of intermediate subquestion answers.

3.2 Hierarchical Structure Reconstruction

The original HO-Tree preserves the node and path structures of semi-structured tables, but some hierarchical semantics remain implicitly encoded in node paths and tree topology. This limits the direct use of such information for structural retrieval and relational reasoning. To make these hierarchical semantics explicit, we propose the Table Adaptive Hierarchical Reconstruction Algorithm. TAHRA identifies potential hierarchical fields from the statistical characteristics of array nodes and aggregates records according to the identified hierarchy. The resulting hierarchy-enhanced HO-Tree provides an explicit parent–child structure while preserving the remaining record-level attributes, serving as the structural basis for subsequent multi-source knowledge construction.

Hierarchical Field Identification. TAHRA takes each array node A in an HO-Tree as the basic analysis unit. Candidate fields are first obtained by excluding fields unsuitable for hierarchical identification, such as highly missing, predominantly numeric, date/time, and index- or identifier-like fields. Let the resulting candidate field set be $F = \{f_1, f_2, \ldots, f_m\}$.For each candidate field $f \in F$, TAHRA extracts three statistical indicators: unique value count u_f, repetition ratio r_f, and average text length l_f. The repetition ratio is defined as $r_f = 1 - \frac{u_f}{|A|}$, where $|A|$ denotes the number of records in array node A. Since higher-level fields generally contain fewer distinct values and exhibit stronger repetition, the indicators are direction-aligned as

$$\mathbf{x}_f^{(0)} = \left(r_f, \frac{1}{u_f + \epsilon}, \frac{1}{l_f + \epsilon} \right), \tag{3}$$

where ϵ is a small positive constant. After Min–Max normalization, TAHRA applies entropy weighting to adaptively determine the contribution of each indicator and computes the hierarchy score

$$S_f = \sum_{j=1}^{3} w_j x_{fj}, \tag{4}$$

where x_{fj} is the normalized value of the j-th indicator and w_j is its entropy-based weight.

To avoid falsely identifying ordinary two-dimensional tables as hierarchical, TAHRA performs a conservative hierarchy test. Let

$$\Delta_S = \max_{f \in F} S_f - \min_{f \in F} S_f. \tag{5}$$

If all candidate fields have the same unique value count and $\Delta_S < 0.05$, no clear hierarchical distinction is assumed and the original HO-Tree is retained. Otherwise, the candidate fields are ordered primarily by increasing unique value count and secondarily by decreasing hierarchy score:

$$L = \mathrm{Sort}(F;\, u_f \uparrow,\, S_f \downarrow), \tag{6}$$

where $L = (f_{(1)}, f_{(2)}, \ldots, f_{(p)})$ denotes the resulting hierarchy order. The complete field filtering, feature normalization, entropy weighting, and hierarchy identification procedure is provided in Algorithm 1.

Hierarchical Node Aggregation. Given the hierarchy order L, TAHRA organizes the records of each hierarchical array into an explicit parent–child structure. For each record ρ, the hierarchical fields in L are traversed sequentially from the root. At each level, TAHRA reuses an existing node with the same field value under the current parent, or creates a new node otherwise. The selected node then becomes the parent for the next hierarchical level. Formally, the node transition is defined as

$$p_j = \mathrm{FindOrCreateChild}\left(p_{j-1}, f_j, \rho[f_j]\right), \tag{7}$$

where p_0 denotes the root node, f_j is the j-th hierarchical field in L, and $\rho[f_j]$ is its corresponding value in record ρ.

After all hierarchical fields have been processed, the remaining non-hierarchical fields are preserved as attributes of the corresponding leaf record. Records sharing the same hierarchical path therefore reuse the same hierarchy nodes while retaining their individual attribute values. This aggregation process transforms implicit hierarchical information in the original HO-Tree into explicit parent–child relationships while preserving record-level information. The complete aggregation procedure is summarized in Algorithm 2.

To further expose the reconstructed hierarchy to downstream knowledge construction, TAHRA attaches structured annotations to each hierarchy node, including its field name, node value, hierarchy index, parent information, and child field. These annotations explicitly record the position and local relationships of each node and are subsequently used for knowledge graph construction and Path vector store generation.

3.3 Multi-Source Knowledge Representation

Although the hierarchy-enhanced HO-Tree explicitly represents table organization, a single tree structure cannot simultaneously support relational reasoning, semantic content matching, and hierarchical structure retrieval. We therefore construct three complementary knowledge sources—a knowledge graph, a Record vector store, and a Path vector store—using the hierarchy-enhanced HO-Tree as their shared structural basis.During knowledge construction, we traverse H_h to enumerate leaf records and assign unique record identifiers, ensuring that record-level information remains distinguishable when hierarchy nodes are shared.

Table Knowledge Graph. Based on H_h, we use a structure-constrained LLM to extract relational knowledge, including header-to-hierarchy-node, parent-to-child, field-to-value, and hierarchy-node-to-record relations. Record entities are first linked to their corresponding hierarchy nodes, after which their record-level attributes are extracted independently. This separation prevents attribute mixing across records that share hierarchy nodes. The resulting triples are stored in Neo4j and provide explicit relational knowledge for structural queries, entity verification, and multi-step reasoning.

Record and Path Vector Stores. To complement the explicit relations in the knowledge graph with semantic matching, we construct two vector representations at different granularities.

For each record ρ_i, we concatenate its hierarchical path with its field-value information to obtain a Record representation:

$$d_i^R = [\mathrm{Path}(\rho_i)\,;\,\mathrm{Fields}(\rho_i)]. \tag{8}$$

The resulting representations are encoded and stored in the Record vector store $\mathcal{V}_R$, which primarily supports semantic retrieval of factual content.

For each record, we further construct a Path representation by sequentially concatenating the nodes along its root-to-leaf hierarchy:

$$d_i^P = [\mathrm{Node}_1 \rightarrow \mathrm{Node}_2 \rightarrow \cdots \rightarrow \mathrm{Node}_k]. \tag{9}$$

These representations are encoded and stored in the Path vector store $\mathcal{V}_P$, which primarily supports hierarchical region localization and structural semantic retrieval.

Together, the three knowledge sources capture complementary aspects of table knowledge: explicit relations through the knowledge graph, factual content through the Record vector store, and hierarchical context through the Path vector store. They form the shared knowledge basis for subsequent type-aware multi-source collaborative retrieval and reasoning.

3.4 Multi-Source Collaborative Retrieval and Reasoning

Given the multi-source knowledge base, HiR-MKR performs query-adaptive retrieval and dependency-driven reasoning for complex semi-structured table questions. Instead of applying a fixed retrieval strategy to all questions, the framework decomposes a complex question into subquestions, identifies their information requirements and dependencies, and dynamically selects the most appropriate knowledge sources.

3.4.1 Query Decomposition and Structure Reliability Assessment

Given a user question Q, an LLM decomposes it into subquestions $Q_s = \{q_1, q_2, \ldots, q_n\}$. Each subquestion is assigned one of three query types: $\{\textit{Content}, \textit{Structure}, \textit{Reasoning}\}$. Content questions focus on factual information and entity attributes, Structure questions focus on hierarchical and structural relationships, and Reasoning questions involve aggregation, comparison, ranking, or multi-step logical operations.

The LLM further constructs a dependency graph $D = (Q_s, E_D)$, where each edge describes the information dependency between subquestions. We consider three dependency types: *Independent*, *Derive*, and *Condition*. Independent subquestions can be executed directly, Derive subquestions reuse previous answers for logical or numerical reasoning, and Condition subquestions use previous answers as conditions for subsequent retrieval.

Since the multi-source knowledge base is derived from the reconstructed hierarchy, unreliable structural reconstruction may introduce errors into subsequent retrieval. We therefore assess the reliability of H_h before retrieval. Let $\rho(H_h) \in \{\mathrm{True}, \mathrm{False}\}$ denote the reliability judgment. When the reconstructed hierarchy is considered reliable, the complete multi-source retrieval strategy is activated. Otherwise, HiR-MKR reduces its reliance on reconstructed structural knowledge and falls back to conservative question answering over the original table. This mechanism limits the propagation of structural errors to downstream retrieval and reasoning.

3.4.2 Type-Aware Multi-Source Retrieval

Different subquestion types require different forms of evidence. HiR-MKR therefore selects a primary and an auxiliary knowledge source according to the information requirement of each subquestion, as summarized in Table 4.

For *Content* subquestions, the Record vector store is used as the primary source for semantic retrieval of relevant records, while the knowledge graph provides entity verification. For *Structure* subquestions, the Path vector store first localizes the relevant hierarchical region, after which the knowledge graph verifies the corresponding structural relations. For *Reasoning* subquestions, the

Record vector store and knowledge graph jointly provide factual evidence and relational constraints, while the Path vector store is additionally used when the reasoning scope depends on a specific hierarchical region.

For structure-oriented retrieval, the localized hierarchy nodes are further traversed in a constrained top-down manner to obtain their child nodes and associated records. This enables structural questions to retrieve both hierarchical relations and the data associated with the relevant region.

Overall, type-aware retrieval avoids indiscriminate retrieval from all knowledge sources and reduces irrelevant evidence and redundant retrieval operations.

3.4.3 Dependency-Driven Reasoning and Answer Generation

The dependency graph D determines the execution order and information flow among subquestions. For a dependency $(q_j \rightarrow q_i) \in E_D$, HiR-MKR handles the current subquestion according to its dependency type:

$$\mathrm{Exec}(q_i) = \begin{cases} \mathrm{Retrieve}(q_i), & \textit{Independent}, \\ \mathrm{Reason}(q_i, a_j), & \textit{Derive}, \\ \mathrm{Retrieve}(\mathrm{Condition}(q_i, a_j)), & \textit{Condition}. \end{cases} \tag{10}$$

Thus, independent subquestions can be answered directly, Derive dependencies reuse intermediate answers without redundant retrieval, and Condition dependencies use previous answers to progressively narrow the retrieval scope.

After all subquestions are resolved, their intermediate answers and retrieved evidence are combined with the original question and dependency graph for final answer generation:

$$a = \mathrm{LLM}\left(Q, Q_s, D, \mathcal{E}, A_s\right), \tag{11}$$

where $A_s = \{a_1, \ldots, a_n\}$ denotes the intermediate answers and $\mathcal{E}$ denotes the retrieved evidence. The resulting pipeline integrates question understanding, structure assessment, type-aware retrieval, and dependency-driven reasoning into a unified TableQA process.

4 Experiments

4.1 Experimental Setup

Datasets. To ensure comparability with previous work, we follow the basic model configuration of ST-Raptor. We evaluate HiR-MKR on three semi-structured table question answering datasets: WikiTQ-ST, TempTabQA-ST, and SSTQA(Tang et al., 2025), as summarized in Table 3. These datasets cover representative structural patterns, including hierarchical headers, temporal hierarchies, and real-world complex tables, as well as question types such as attribute retrieval, conditional filtering, aggregation, and multi-step reasoning.The details of the datasets are described in AppendixA.3.

Baseline. To comprehensively evaluate HiR-MKR, we compare it with 10 representative methods across six technical categories, including NL2SQL, general-purpose LLMs, fine-tuned models, agent-based methods, vision-language models, and hybrid approaches.The details of the baseline are described in AppendixA.4.

Evaluation Metrics. We use Accuracy (Acc) as the evaluation metric, which measures the consistency between the predicted and ground-truth answers. Higher Acc indicates better overall question answering performance.

Implementations. To ensure comparability with previous work, we follow the basic model configuration of ST-Raptor. We use InternVL2.5-26B(Chen et al., 2024) as the vision-language model for table parsing and HO-Tree construction, DeepSeek-V3(Liu et al., 2024) as the reasoning model, and Multilingual-E5-Large(Wang et al., 2024) as the text embedding model.

4.2 Main Results

The overall results on the three benchmarks are reported in Table 1. HiR-MKR achieves the best accuracy on all three datasets, reaching 82.14%, 87.43%, and 82.72% on WikiTQ-ST, TempTabQA-ST, and SSTQA, respectively. Compared with ST-Raptor, HiR-MKR improves accuracy by 10.97, 9.84, and 10.33 percentage points, respectively. The consistent improvements across datasets demonstrate the effectiveness of HiR-MKR for semi-structured table question answering.

Compared with general-purpose LLMs and existing structure-aware methods, HiR-MKR provides additional gains through explicit hierarchical structure and complementary knowledge sources. These results suggest that combining the hierarchy-enhanced HO-Tree with the Record vector store, Path vector store, and KG enables more effective use of both structural and semantic information for complex table question answering.

Table 1: Main experimental results on three semi-structured table question answering benchmarks.

Methods	WikiTQ-ST Acc	TempTabQA-ST Acc	SSTQA Acc
OpenSearch-SQL (Xie et al., 2025)	38.89	4.76	24.00
TableLLaMA (Zhang et al., 2024)	35.01	32.70	40.39
TableLLM (Zhang et al., 2025)	62.40	9.13	7.84
ReAcTable (Zhang et al., 2023)	68.00	35.88	37.24
TAT-LLM (Zhu et al., 2024)	23.21	61.86	39.78
TableLLaVA (Zheng et al., 2024)	20.41	6.91	9.52
mPLUG-DocOwl1.5 (Hu et al., 2024)	39.80	39.80	29.65
GPT-4o (Achiam et al., 2023)	60.71	74.83	66.45
DeepSeek-V3 (Liu et al., 2024)	69.64	63.81	63.22
ST-Raptor (Tang et al., 2025)	71.17	77.59	72.39
HiR-MKR	**82.14**	**87.43**	**82.72**

We also analyzed the capability of HiR-MKR to handle questions of varying complexity, as well as its performance across different models, as detailed in Appendix A.5.1 and Appendix A.5.2.

4.3 Ablation Study

We conduct an ablation study on the SSTQA dataset to evaluate the contribution of the key components of HiR-MKR. The results are reported in Table 2. Removing any component leads to a performance degradation, indicating that the components provide complementary benefits.

Removing HO-Tree construction causes the largest performance drop, reducing accuracy from 82.72% to 63.48% (19.24 percentage points). In this setting, the original Excel tables are directly used as the data source, bypassing the Excel-to-HO-Tree conversion process. This result highlights the importance of structured table representation as the foundation for subsequent hierarchical reconstruction and knowledge construction. Removing TAHRA further decreases accuracy by 6.41 percentage points, demonstrating that explicitly reconstructing hierarchical relationships provides additional benefits beyond the original HO-Tree representation. KG retrieval, query decomposition, and structure reliability assessment result in additional decreases of 8.90, 6.28, and 4.97 percentage points, respectively. Overall, the ablation results validate the complementary contributions of structural representation, hierarchical reconstruction, multi-source retrieval, and adaptive question processing.

4.4 Efficiency Analysis

We evaluate the inference efficiency of HiR-MKR on the SSTQA dataset by comparing its response time, number of LLM calls, and token consumption with ST-Raptor. The results are reported in Table 3.

HiR-MKR substantially reduces inference overhead across most efficiency metrics. The average response time decreases from 34.46s to 14.03s, corresponding to a 59.29% reduction. The average

Table 2: Ablation results on the SSTQA dataset.

Model	SSTQA Acc
Full Model	**82.72%**
w/o HO-Tree Construction	63.48% (-19.24 pp)
w/o TAHRA	76.31% (-6.41 pp)
w/o KG Retrieval	73.82% (-8.90 pp)
w/o Query Decomposition	76.44% (-6.28 pp)
w/o Structure Reliability Assessment	77.75% (-4.97 pp)

number of LLM calls and input tokens are reduced by 41.07% and 74.05%, respectively. Although the average output tokens increase by 24.84%, the average total token consumption is reduced by 73.00%, from 15,128 to 4,085 tokens.

These efficiency gains mainly result from offline knowledge construction and type-aware retrieval. By constructing the knowledge graph and vector stores before question answering, HiR-MKR avoids repeated knowledge organization during online inference. Meanwhile, query decomposition and type-aware retrieval enable the system to retrieve only the knowledge sources required by each subquestion, reducing redundant context and LLM calls.

Overall, HiR-MKR achieves substantially lower inference overhead while maintaining higher question answering performance, demonstrating a favorable trade-off between accuracy and efficiency.

Table 3: Performance and cost analysis on the SSTQA dataset.

Metric	ST-Raptor	HiR-MKR	Relative Comparison
Average time consumption (s)	34.46	14.03	↓59.29%
Average LLM calls	11.2	6.6	↓41.07%
Average input tokens	14967	3884	↓74.05%
Average output tokens	161	201	↑24.84%
Average total tokens	15128	4085	↓73.00%

5 CONCLUSION

We present HiR-MKR, a semi-structured table question answering framework that integrates hierarchical structure reconstruction with multi-source knowledge collaboration. TAHRA reconstructs explicit hierarchical semantics from HO-Trees, providing a unified basis for a knowledge graph, Record vector store, and Path vector store. Type-aware retrieval and dependency-driven reasoning then enable adaptive evidence selection for different query requirements. Experiments on WikiTQ-ST, TempTabQA-ST, and SSTQA show that HiR-MKR consistently outperforms existing baselines, achieving 82.72% accuracy on SSTQA and improving over ST-Raptor by 10.33 percentage points. It also substantially reduces inference overhead, with average response time and total token consumption reduced by 59.29% and 73.00%, respectively. These results demonstrate the effectiveness of combining explicit hierarchical structure with multi-source knowledge for complex semi-structured table question answering. Future work will focus on adaptive structure learning for noisy tables, dynamic knowledge updating and cross-table reasoning, and extending the framework to other semi-structured data such as JSON, XML, and Web DOM.

AI USE STATEMENT

In this work, we employed large language models (LLMs) as auxiliary tools, primarily for the following purposes: (1) proofreading and polishing the manuscript; (2) automatic code completion during experimental implementation; (3) organizing and summarizing literature content of related studies. The research ideas, methodology, and findings presented in this work are entirely the responsibility of the authors.

ETHICS STATEMENT

The datasets used in this study are all sourced from publicly available datasets on GitHub. They do not contain any personally identifiable information or other sensitive data, and thus our experiments present no apparent ethical concerns regarding data privacy or human subjects.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4524–4535, 2019.

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

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3096–3120, 2024.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pp. 174–186. Springer, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Panupong Pasupat and Percy Liang. 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)*, pp. 1470–1480, 2015.

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

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. Opensearch-sql: Enhancing text-to-sql with dynamic few-shot and consistency alignment. *Proceedings of the ACM on Management of Data*, 3(3):1–24, 2025.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8413–8426, 2020.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 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)*, pp. 6024–6044, 2024.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, et al. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10315–10344, 2025.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*, 2023.

Mingyu Zheng, Xinwei Feng, Qingyi Si, Qiaoqiao She, Zheng Lin, Wenbin Jiang, and Weiping Wang. Multimodal table understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9102–9124, 2024.

Fengbin Zhu, Ziyang Liu, Fuli Feng, Chao Wang, Moxin Li, and Tat Seng Chua. Tat-llm: A specialized language model for discrete reasoning over financial tabular and textual data. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pp. 310–318, 2024.

A Appendix

A.1 Type-Aware Retrieval

HiR-MKR selects knowledge sources according to the information requirements of each subquestion type. The detailed retrieval strategy is summarized in Table 4. For *Content* questions, the Record vector store is used for semantic matching of relevant records, while the KG is used for entity verification. For *Structure* questions, the Path vector store first localizes the relevant hierarchical region, after which the KG is used to verify the corresponding structural relations. For *Reasoning* questions, the Record vector store and KG jointly provide factual evidence and relational constraints, while the Path vector store is used when the reasoning scope depends on hierarchical context.

Table 4: Type-aware retrieval strategies for different subquestion types.

Question Type	Primary Knowledge Source	Auxiliary Knowledge Source	Main Function
Content	Record	KG	Content matching and entity verification
Structure	Path	KG	Hierarchical localization and relation verification
Reasoning	Record + KG	Path	Factual evidence and relational constraints

The three knowledge sources are complementary at different retrieval granularities: the Record vector store provides content-level semantic matching, the Path vector store captures hierarchical context, and the KG provides explicit relational information. Their selective combination allows HiR-MKR to retrieve only the knowledge required by each subquestion, reducing irrelevant evidence and redundant retrieval.

Content Retrieval. For a Content subquestion q_i, HiR-MKR first retrieves the top-$k$records from the Record vector store. The entities contained in the retrievedrecords are then verified against the KG to remove semantically similar but structurally irrelevant candidates.

Structure Retrieval. For a Structure subquestion, HiR-MKR first retrieves relevant hierarchical paths from the Path vector store. The localized hierarchy nodes are then used as entry points for KG queries, where constrained top-down traversal retrieves the required parent–child relations and associated records.

Reasoning Retrieval. For a Reasoning subquestion, HiR-MKR jointly retrieves factual evidence from the Record vector store and relational constraints from the KG. When the reasoning operation is restricted to a specific hierarchical region, Path retrieval is additionally performed to determine the relevant structural scope.

A.2 TAHRA ALGORITHM

TAHRA reconstructs implicit hierarchical relationships in the original HO-Tree through two successive procedures: hierarchical field identification and hierarchical node aggregation. The first procedure identifies potential hierarchical fields and determines their ordering from statistical characteristics. The second procedure uses the resulting hierarchy order to construct explicit parent–child relationships while preserving record-level attributes. The detailed procedures are described below.

A.2.1 HIERARCHICAL FIELD IDENTIFICATION

Given an array node A, TAHRA first filters fields that are unsuitable for hierarchical identification, including highly missing, predominantly numeric, date/time, and index- or identifier-like fields. Let the resulting candidate field set be$F = \{f_1, f_2, \ldots, f_m\}$.

For each candidate field $f \in F$, three statistical indicators are extracted: unique value count u_f, repetition ratio r_f, and average text length l_f. The repetition ratio is defined as $r_f = 1 - \frac{u_f}{|A|}$, where $|A|$ denotes the number of records in array node A.

Since the three indicators have different scales and directions, TAHRA first constructs a direction-aligned feature vector:

$$\mathbf{x}_f^{(0)} = \left(r_f,\ \frac{1}{u_f + \epsilon},\ \frac{1}{l_f + \epsilon} \right), \tag{12}$$

where ϵ is a small positive constant. Larger values of all three components therefore indicate stronger evidence that a field occupies a higher hierarchical level.

Entropy-weighted hierarchy scoring. The direction-aligned features are normalized using Min–Max normalization:

$$x_{fj} = \frac{x_{fj}^{(0)} - \min_{f' \in F} x_{f'j}^{(0)}}{\max_{f' \in F} x_{f'j}^{(0)} - \min_{f' \in F} x_{f'j}^{(0)} + \epsilon}. \tag{13}$$

The normalized values are converted into proportions:

$$p_{fj} = \frac{x_{fj}}{\sum_{f' \in F} x_{f'j} + \epsilon}. \tag{14}$$

The entropy of indicator j is calculated as

$$e_j = -\frac{1}{\ln(|F| + \epsilon)} \sum_{f \in F} p_{fj} \ln(p_{fj} + \epsilon). \tag{15}$$

The corresponding information divergence is$g_j = 1 - e_j$,and the entropy-based weight is given by

$$w_j = \frac{g_j}{\sum_{k=1}^{3} g_k + \epsilon}. \tag{16}$$

The hierarchy score of field f is then computed as

$$S_f = \sum_{j=1}^{3} w_j x_{fj}. \tag{17}$$

Algorithm 1 Hierarchical Field Identification

Input: Array node A
Output: Field scores S, hierarchy order L
$F \leftarrow \text{FilterCandidateFields}(A)$
if $|F| = 0$ **then**
  return $\emptyset, \emptyset$
end if
for each field $f \in F$ **do**
  $u_f \leftarrow \text{UniqueCount}(f)$
  $r_f \leftarrow 1 - \frac{u_f}{|A|}$
  $l_f \leftarrow \text{AvgTextLength}(f)$
end for
$X \leftarrow \{(r_f, \frac{1}{u_f+\epsilon}, \frac{1}{l_f+\epsilon})\}_{f \in F}$
$X \leftarrow \text{MinMaxNormalize}(X)$
$w \leftarrow \text{EntropyWeights}(X)$
$S_f \leftarrow \sum_j w_j X_{fj}, \quad \forall f \in F$
if $|F| = 1$ **then**
  return S, F
end if
$\Delta_S \leftarrow \max_{f \in F} S_f - \min_{f \in F} S_f$
if $\max_{f \in F} u_f = \min_{f \in F} u_f \ \wedge \ \Delta_S < 0.05$ **then**
  return S, F
end if
$L \leftarrow \text{SortFields}(F;\ u_f \uparrow,\ S_f \downarrow)$
return S, L =0

Hierarchy decision and field ordering. To distinguish hierarchical arrays from ordinary two-dimensional tables, TAHRA further examines the variation of the hierarchy scores and unique value counts across candidate fields. Let

$$\Delta_S = \max_{f \in F} S_f - \min_{f \in F} S_f. \tag{18}$$

If all candidate fields have the same unique value count and $\Delta_S < 0.05$, the candidate fields are considered insufficient to establish a hierarchical distinction, and the original HO-Tree is retained. Otherwise, the fields are ordered primarily by increasing unique value count and secondarily by decreasing hierarchy score:

$$L = \text{Sort}(F;\ u_f \uparrow,\ S_f \downarrow), \tag{19}$$

where $L = (f_{(1)}, f_{(2)}, \ldots, f_{(p)})$ denotes the resulting hierarchy order.

A.2.2 Hierarchical Node Aggregation

Let A denote an array node containing a set of records $\mathcal{R} = \{\rho_1, \rho_2, \ldots, \rho_n\}$, and let $L = (f_{(1)}, f_{(2)}, \ldots, f_{(p)})$ denote the hierarchy order identified by the field identification procedure. We define the corresponding set of hierarchical fields as $F_H = \{f_{(1)}, f_{(2)}, \ldots, f_{(p)}\}$.

For each record $\rho \in \mathcal{R}$, TAHRA constructs a hierarchical path by traversing the fields in L from the root. At hierarchy level j, the node is determined by the field–value pair$(f_{(j)}, \rho[f_{(j)}])$. Let p_{j-1} denote the parent node at the previous level. The corresponding child node is obtained by $p_j = \text{FindOrCreateChild}\left(p_{j-1}, f_{(j)}, \rho[f_{(j)}]\right), j = 1, \ldots, p$, where p_0 denotes the root node.

The resulting sequence $P(\rho) = (p_1, p_2, \ldots, p_p)$ represents the explicit hierarchical path associated with record ρ.When multiple records share the same hierarchical field values, the corresponding nodes are reused under the same parent, resulting in a shared hierarchical path. If records differ in their hierarchical field values, they are organized into different branches. Therefore, the aggregation process shares common hierarchical structure without duplicating hierarchy nodes across records.After all hierarchy fields have been processed, the remaining non-hierarchical fields are preserved as record-level attributes:

$$A_\rho = \{(f, \rho[f]) \mid f \in F \setminus F_H\}, \tag{20}$$

Algorithm 2 Hierarchical Node Aggregation

Input: Array node A, hierarchy order L, field set F
Output: Hierarchy-enhanced tree H_h
$T_h \leftarrow$ InitializeTree()
$F_H \leftarrow$ Fields(L)
for each record $\rho \in A$ **do**
  $p \leftarrow$ Root(H_h)
  for each field $f \in L$ **do**
    $v \leftarrow \rho[f]$
    $n \leftarrow$ FindChild(p, f, v)
    if $n = \emptyset$ **then**
      $n \leftarrow$ CreateNode(f, v)
      AddChild(p, n)
    end if
    $p \leftarrow n$
  end for
  AttachAttributes($p, \rho[F \setminus F_H]$)
end for
return H_h =0

Table 5: Structured annotation fields for the reconstructed hierarchy.

Annotation Field	Corresponding Information
`level_field`	Field name of the current hierarchy node
`level_value`	Value of the current hierarchy node
`level_index`	Index of the current hierarchy level
`parent_field`	Field name of the parent node
`parent_value`	Value of the parent node
`child_field`	Field name of the child hierarchy

where F denotes the complete field set of the array.

Thus, hierarchical fields determine the structural organization of the tree, whereas non-hierarchical fields remain associated with the corresponding leaf record. If multiple records terminate at the same hierarchical leaf, their attribute sets are retained as separate records rather than being merged. This ensures that node aggregation shares only the hierarchical structure while preserving the semantic boundaries of individual records.

A.2.3 Hierarchical Structure Annotations

To explicitly preserve the structural information introduced during hierarchical reconstruction, TAHRA attaches structured annotations to each hierarchy node in the hierarchy-enhanced HO-Tree. These annotations record the field and value of the current node, its position in the hierarchy, and its local parent–child relationships.

The main annotation fields are summarized in Table 5. These annotations make the reconstructed hierarchy explicitly accessible to downstream knowledge construction. In particular, they provide structural cues for hierarchy traversal and relation extraction during knowledge graph construction, and support the generation of hierarchical paths for the Path vector store.

A.3 Datasets

We evaluate HiR-MKR on three semi-structured table question answering benchmarks: WikiTQ-ST, TempTabQA-ST, and SSTQA. WikiTQ-ST and TempTabQA-ST are subsets of WikiTQ and TempTabQA selected by ST-Raptor (Tang et al., 2025) to retain tables with semi- structured characteristics, such as hierarchical headers and merged cells. SSTQA is introduced by ST-Raptor (Tang et al., 2025) and contains real-world semi-structured tables with more complex layouts, including

nested cells, multi-row/column headers, and irregular structures. Detailed dataset statistics are summarized in Table 6.

Table 6: Datasets used in our experiments.

Dataset	Tables	QA Pairs	Characteristics
WikiTQ-ST	21	112	Hierarchical headers
TempTabQA-ST	97	183	Temporal hierarchy
SSTQA	102	764	Real-world semi-structured tables

A.4 Baseline Models

To comprehensively evaluate the effectiveness of HiR-MKR, we compare it with representative methods from six technical categories: NL2SQL, general-purpose LLMs, fine-tuned table models, agent-based methods, vision-language models (VLMs), and hybrid table question answering frameworks. These baselines cover different paradigms for table understanding and question answering, ranging from structured query generation and general-purpose reasoning to specialized table modeling and multi-component systems. The complete list of baseline models is presented in Table 7.

Table 7: Baseline models used in the experiments.

Category	Methods
NL2SQL	OpenSearch-SQL
LLM	GPT-4o, DeepSeek-V3
Fine-tuned	TableLLaMA, TableLLM
Agent	ReAcTable, TAT-LLM
VLM	TableLLaVA, mPLUG-DocOwl1.5
Hybrid	ST-Raptor

NL2SQL. OpenSearch-SQL represents the conventional approach of translating natural language questions into executable structured queries. It provides a comparison with methods that rely on explicit query construction for table question answering.

General-purpose LLMs. GPT-4o and DeepSeek-V3 are included to evaluate the question answering capabilities of general-purpose large language models without introducing task-specific table architectures. They provide reference points for assessing the benefits of explicit structural and knowledge representations.

Fine-tuned Table Models. TableLLaMA and TableLLM represent models specialized or fine-tuned for table question answering. These methods provide a comparison with approaches that enhance table understanding through task-specific model adaptation.

Agent-based Methods. ReAcTable and TAT-LLM represent agent-based approaches that incorporate iterative reasoning, tool use, or task decomposition for table question answering. They provide a comparison with methods that explicitly organize multi-step question solving.

Vision-Language Models. TableLLaVA and mPLUG-DocOwl1.5 represent vision-language approaches that process tables through visual and multimodal representations. They are included to evaluate whether direct visual table understanding can effectively handle the structural characteristics of semi-structured tables.

Hybrid Frameworks. ST-Raptor is the most closely related baseline to HiR-MKR. It introduces the HO-Tree representation for structurally modeling semi-structured tables and supports question answering through structured table operations. We therefore use ST-Raptor as the primary reference when evaluating the effectiveness of hierarchical reconstruction and multi-source knowledge collaboration in HiR-MKR.

A.5 Additional Experiments

A.5.1 Performance across Query Complexities

Table 8: Performance analysis on SSTQA for questions of different difficulties

Methods	Simple	Medium	Hard
DeepSeekV3	92.94%	61.83%	47.19%
GPT-4o	86.64%	59.75%	43.26%
ST-Raptor	93.97%	62.66%	58.43%
Ours	87.68%	**86.12%**	**68.54%**

To further evaluate the robustness of HiR-MKR across different levels of question complexity, we follow the complexity categories defined by ST-Raptor and report accuracy on Simple, Medium, and Hard questions in SSTQA. The results are shown in Table 8. HiR-MKR shows substantial improvements on more complex questions, achieving 86.12% on Medium and 68.54% on Hard questions, which correspond to gains of 23.46 and 10.11 percentage points over ST-Raptor, respectively. This suggests that hierarchical reconstruction and query-adaptive multi-source retrieval are particularly beneficial for questions requiring structural understanding and multi-step reasoning. In contrast, HiR-MKR achieves 87.68% on Simple questions, below ST-Raptor's 93.97%, indicating that the additional reasoning and retrieval stages may introduce unnecessary processing for questions that can be answered through direct retrieval.

A.5.2 Adaptability to Different Reasoning Models

Table 9: SSTQA performance with different reasoning models.

Model	Acc
DeepSeek-V3 (Liu et al., 2024)	82.72%
GPT-4o (Achiam et al., 2023)	80.24%
GLM-5 (Zeng et al., 2026)	81.28%

To assess whether the performance of HiR-MKR depends on a specific reasoning backbone, we evaluate the framework with three different reasoning models on SSTQA. The results are shown in Table 9. HiR-MKR consistently achieves over 80% accuracy across DeepSeek-V3, GPT-4o, and GLM-5, with a maximum difference of only 2.48 percentage points. This indicates that the effectiveness of HiR-MKR is relatively robust to the choice of reasoning model.