

Every Answer Has Its Own Path: Agentic Routing for Retrieval-Augmented Table Question Answering

A Young Kim Jisu Shin Donghee Han Mun Yong Yi[†]
 KAIST, Republic of Korea
 {ayoung, jisu3389, handonghee, munyi}@kaist.ac.kr
[†]Corresponding author

Abstract

Documents that users query mix tables with prose, but the QA systems built for them do not: retrieve-and-read pipelines flatten tables, while text-to-SQL pipelines ignore passages. Corporate filings, scientific reports, and Wikipedia articles all sit on this fault line, and a single user question routinely crosses it. We present ATR (AgenticTableRAG), an agentic framework for hybrid table-text question answering. ATR decomposes each question into ordered sub-queries and dispatches each to one of four execution primitives: text reading, cell retrieval, SQL execution, or a hybrid of cell grounding and SQL. When the retrieved evidence does not support the predicted answer, the router is re-invoked with the failed-route history and dispatches a different primitive. Value grounding composed with constrained SQL execution rules out column and value hallucinations by construction. The routing policy is distilled into a lightweight student model that transfers across benchmarks without additional inference-time LLM calls. A single ATR configuration leads on token F1 across three benchmarks without per-dataset tuning.

1 Introduction

In practice, document QA spans both modalities at once. A single user question often combines a table-row lookup with a passage read drawn from the same document, two sub-queries that call for different reasoning primitives.

Because each sub-part of such a question calls for a different primitive, *agentic QA* is a natural framework for this setting: an agentic-QA system decomposes the input and selects among several execution primitives (retrieval, SQL, Python sandboxing) instead of committing to a fixed backbone. Existing agentic-QA systems for hybrid table-text QA, however, still pre-commit to one primitive at the question level, whether sandboxed Python over

Question

What historic place is in the city where
U.S. Highways 2 and 169 are the main routes?

TEXT PASSAGES

The region is known for its mining history.

U.S. Hwys 2 and 169 pass through La Prairie.

The population has grown steadily since 1990.

(1) Multi-hop cross-modal

RELATIONAL TABLE

City	...	Site	Year
Grand Rapids	...	Old Central School	1895
Hwy 2/169 @ La Prairie	...	Frank Gran Farmstead	1920
Coleraine	...	Canisteo District	1907

Answer found via cross-modal bridge

(2) Value alignment:

'Highways' (Q) ~ 'Hwys' (text) ~ 'Hwy 2/169' (cell)

Figure 1: A HybridQA example exhibiting the two failure modes from §1; (1) marks the text→table bridge, (2) marks the surface-form mismatch.

retrieved indices (Chen et al., 2024c), SQL cross-validated with textual retrieval (Yu et al., 2025), or passage retrieval (Yao et al., 2023); none recovers when the chosen tool fails on a sub-part that another primitive would have handled. The bottleneck is not the choice of primitive but the *unit of decision*: decomposition, routing, and re-routing must all operate at the sub-query level.

This question-level commitment shows up as two recurring failure modes (Figure 1). (1) **Multi-hop cross-modal trajectory**. Retrieval-only agents stop at the text passage and pure NL2SQL searches the table without the text clue, so neither carries intermediate clues across modalities. (2) **Lexical mismatch between surface forms**. Embedding-based retrieval can rank the correct cell as a candidate, but the SQL executor demands exact surface-form matching that retrieval similarity does not guarantee.

To address this question-level commitment, we propose ATR (AgenticTableRAG), in which rout-

ing happens at the sub-query level (Figure 2). ATR decomposes each input question into ordered sub-queries; for each sub-query, a router selects one of four execution primitives (text reading, cell retrieval, SQL, or a hybrid of cell grounding and constrained SQL); the selected primitive returns a candidate sub-answer; and a verifier validates the sub-answer, re-routing it to a different primitive when the evidence is insufficient. To make sub-query-level routing efficient at inference time, the router itself is distilled from an LLM-prompted teacher into a lightweight student that incurs no per-sub-query LLM call.

Empirically, a single ATR configuration improves over the strongest agentic prior on HybridQA by +37.2 F1 and +15.2 points on an LLM-judged correctness metric, and matches or exceeds the strongest prior system on TAT-QA. We additionally evaluate on WTQ, MultiHiertt, and SPARTA, none of which the router was trained on, as transfer tests of the routing policy rather than as further hybrid benchmarks.

Our contributions can be summarized as follows:

- **Identification of a Key Challenge.** We identify the *soft-retrieval / hard-execution gap* in retrieval-augmented table-text QA: embedding retrieval ranks the correct cell as a candidate, yet SQL execution still misses the row because the question’s surface form never matches the stored value (Figure 1, failure (2)). Prior retrieval-augmented table QA systems hand retrieved surface forms to the executor verbatim and inherit this gap.
- **Sub-query-level Routing with Retrieval-grounded Constrained Execution.** ATR moves the routing decision from the question to the sub-query, with the route set closed under failure, so re-routing is conditioned on which primitives have already been rejected (Figure 1, failure (1)). ATR issues no SQL before a value-linking step grounds each question entity to a value that exists in the cell index, and constrained execution samples only from retrieved columns and linked values, so column and value hallucination are excluded by construction rather than repaired after failure. The linking mechanism is established work (§2); what is new is its placement as a precondition of execution inside routing, where a failed grounding exits to a different

primitive instead of triggering an in-SQL repair.

- **Empirical Validation.** A single ATR configuration leads on token F1 across three benchmarks; the routing policy distills into a lightweight student that recovers 98.6% of the teacher’s decisions at zero per-sub-query LLM cost and transfers to benchmarks it never saw; fixing the first-choice primitive to HYBRID while keeping everything else costs 8.26 pp F1 (the *No Router* ablation); and replacing the linker with verbatim NL2SQL under the full system costs 2.7 pp, of which 1.2 needs the cross-primitive exit that an in-SQL repair cannot provide (§6).

2 Related Work

Agentic and iterative retrieval. Recent agentic-QA systems extend retrieval-augmented generation (Lewis et al., 2020) with iterative tool selection but typically commit to a single execution strategy at the question level. General-purpose agents such as ReAct (Yao et al., 2023), Self-Ask prompting (Press et al., 2023), and Reflexion (Shinn et al., 2023) explore tool decomposition and self-critique; Self-RAG (Asai et al., 2024) adds reflection-token gating to retrieval; Program-of-Thoughts (Chen et al., 2023) executes reasoning as Python code rather than natural language. Within tables, TableRAG-R (Chen et al., 2024c) indexes schema, rows, and cells and executes inside a Python sandbox; TableRAG-S (Yu et al., 2025) cross-validates SQL with textual retrieval. ODYSSEY (Agarwal et al., 2025) traverses a bidirectional cell/passage graph specific to Wikipedia cell-link tables, and HProPro (Shi et al., 2024) reads a table supplied by question id; both are given their evidence rather than retrieving it (§5). ATR differs in (i) sub-query-level rather than question-level routing, (ii) a fixed four-primitive vocabulary closed under failure-conditioned re-routing, and (iii) distillation of the routing policy into a lightweight classifier.

NL2SQL and constrained execution. Modern NL2SQL maps natural language to executable SQL through pretrained models such as SQL-PaLM (Sun et al., 2024), execution-guided methods like Binder (Cheng et al., 2023) and Codex SQL (Rajkumar et al., 2022), and decomposition approaches like DIN-SQL (Pourreza and Rafiei,

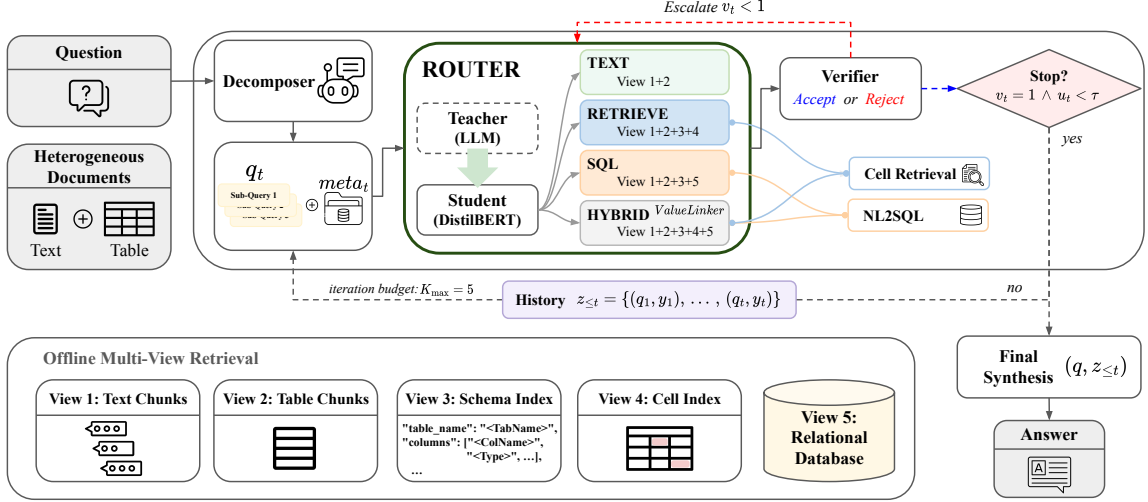


Figure 2: ATR online inference loop (Algorithm 1): Decomposer, distilled Router, four execution primitives over a five-view index, with re-routing on failure.

2023) that break SQL generation into sub-tasks with self-correction. These approaches improve the SQL syntax itself but inherit the lexical-mismatch failure of pure NL2SQL: a syntactically perfect query that emits `'tv'` still misses a cell stored as `'television'`. ATR uses NL2SQL as one of four routes and pairs it with the HybridValueLinker, which grounds every value referenced in `WHERE` clauses against the cell index, addressing the data-grounding bottleneck rather than the syntax bottleneck. Grounding question entities against stored cell values is established practice in the schema- and value-linking literature (Li et al., 2024; Liu et al., 2025); Tool-SQL (Wang et al., 2024) in particular retrieves embedding-similar cells for unmatched conditional values and lets an agent select the replacement, closely paralleling our linker’s retrieve-then-select mechanism, and we claim no novelty for the mechanism itself. The difference is structural: in Tool-SQL a failed grounding is a *repair* signal inside a SQL-only loop, whereas in ATR it is a *routing* signal that dispatches the sub-query to a different primitive, the exit an in-SQL repair loop lacks on narrative-attribute sub-queries with no DB column to ground.

Lightweight distilled controllers. Distillation into smaller students is well studied for classification (Sanh et al., 2019; Jiao et al., 2020), and LLM-routing work such as RouteLLM (Ong et al., 2025) and FrugalGPT (Chen et al., 2024b) learns to dispatch between strong and weak LLMs to reduce inference cost. ATR’s router instead dispatches

between execution primitives (text reading, cell retrieval, SQL, hybrid) for a fixed LLM backbone, a different unit of routing decision. Distilling such an LLM-prompted router for an agentic pipeline, where the teacher emits a discrete route label rather than a sequence, is underexplored. ATR’s student classifier recovers 98.6% of the teacher’s routing decisions and transfers across benchmarks without further supervision.

3 Method

3.1 Overview

Problem setup. We target hybrid table-text QA in the table-grounded setting of our benchmarks: each question q is paired with a source table T and the passages P linked from its cells ($P=\emptyset$ for standalone-table WTQ). ATR indexes these tables and passages into the offline views of §3.2 (text and table chunks, a schema and cell index, and a relational database), and for each sub-query it retrieves the relevant cells, rows, and linked passages rather than locating T in an open-domain store; the system returns a short answer a .

ATR answers a question q through an iterative loop of decomposition, routing, execution, and verification (Figure 2; Algorithm 1). At each iteration $t \in \{1, \dots, K_{\max}\}$ the decomposer emits a sub-query q_t conditioned on previous sub-answers $z_{<t}$, the router dispatches it to one of four primitives $r \in \mathcal{R} = \{\text{TEXT}, \text{RETRIEVE}, \text{SQL}, \text{HYBRID}\}$, the primitive produces an intermediate answer y_t , and a verifier returns $v_t \in \{0, 0.5, 1\}$. When $v_t < 1$,

the router re-selects given the updated failed-route history $H_{<t}$ and the sub-query is re-executed. The loop exits when $\nu_t = 1 \wedge u_t < \tau$ (u_t is the decomposer’s residual uncertainty) or after $K_{\max}$ iterations; an optional final synthesis step fuses the sub-answers into the final answer a . Here $K_{\max}$ bounds the loop from above rather than fixing the decomposition depth; the realised number of steps is set per question by the TERMINATE sentinel and the uncertainty-gated stop controller, and adapts to question difficulty (§6.2).

Two design choices distinguish ATR from prior agentic table-QA systems. Routing happens at the sub-query level, so a single question can mix primitives. And when the verifier rejects an answer, the router (rather than a free-form LLM patch) picks the next primitive conditioned on which routes have already failed, making the rescue step principled rather than open-ended. Full pseudocode: Algorithm 1 (Appendix B).

3.2 Offline Indices

ATR shares a single set of offline indices across all four execution primitives. **View 1** (text chunks) splits each passage with a recursive character splitter; **View 2** (table chunks) serializes each table into row-aligned chunks together with a mapping f from each chunk back to its source table and schema; **View 3** (schema index) embeds each column header to support schema retrieval; **View 4** (cell index) embeds each `(column, value)` pair up to a per-table budget of B cells; and **View 5** is a relational database served by an external SQL endpoint. Each route consumes a different subset of these views (§3.5); the indices themselves are route-agnostic.

3.3 Decomposition and Sub-Query Schema

At each iteration t , the decomposer samples the next sub-query conditioned on the original question and the accumulated sub-answer history,

$$q_t \sim p_{\text{DEC}}(\cdot \mid q, z_{<t}), \quad z_{<t} = \{(q_s, y_s)\}_{s=1}^{t-1},$$

where the decomposer is a prompted LLM that emits a structured SUBQUERY object rather than a free-form question; its six fields, types, and downstream consumers are listed in Appendix Table 3, and the full decomposer prompt is given in Appendix Figure 10a. The decomposer also emits the sentinel TERMINATE when it judges q already answered, which ends the loop early. Multi-hop

dependencies, such as a sub-query that references the result of a prior step, are handled implicitly because the previous sub-answers $z_{<t}$ are part of the decomposer’s prompt context. The same JSON output additionally carries a scalar `uncertainty` $u_t \in [0, 1]$: the prompted LLM is asked to estimate, on a fixed rubric, how likely the original question remains under-answered after the sub-queries so far (Appendix Figure 10a). This u_t gates the stop controller in §3.6: when the verifier supports the current sub-answer ($\nu_t=1$) and the decomposer also reports low residual uncertainty ($u_t < \tau$), the outer loop exits.

3.4 Routing

The router maps the tuple $(q_t, meta_t, schema_t, H_{<t})$ to a route $r \in \mathcal{R}$, where $meta_t$ is the SUBQUERY metadata of §3.3, $schema_t$ is the schema restored from the retrieved table chunks by View 2’s mapping f , and $H_{<t} \subseteq \mathcal{R}$ is the set of routes already attempted and verifier-rejected anywhere in the question so far; when every route is in $H_{<t}$ the router falls back to TEXT (cf. §3.6). We train the router by teacher/student distillation: a zero-shot LLM router (the *teacher*) labels routes on training questions, and a compact four-class classifier (Sanh et al., 2019) (the *student*) is distilled to match the teacher’s distribution. Only the student runs at inference, a lightweight classifier that runs at zero LLM cost per sub-query.

3.5 Execution Primitives

Each route in $\mathcal{R}$ is backed by a different execution primitive that shares the offline indices of §3.2 but exposes a different reasoning shape.

TEXT. Retrieved chunks from Views 1 and 2 feed an answer-synthesis prompt that grounds the answer in passages; no schema or SQL is involved.

RETRIEVE. For each entity $e \in E_t$, View 4 returns top- K_v `(column, value)` candidates via embedding similarity, which the HYBRID-VALUELINKER (shared with HYBRID) grounds to canonical cell values, resolving lexical variants like *tv* vs. *television*. Because matched cells alone localize the answer row but not the answer column, we augment View 4 with a per-row index that returns the top- K_r rows R whose concatenated cell text is most similar to q_t . The synthesis prompt receives the grounded cells and the rows R , so the LLM locates the answer row from the grounded

match and reads the target column from the row context, without committing to SQL.

SQL. The candidate columns C retrieved from View 3 feed an NL2SQL prompt that produces a query against View 5.

HYBRID. The HYBRIDVALUELINKER bridges retrieval and SQL: for each entity $e \in E_t$, View 4 returns the top- K_v candidate cells $V_{\text{raw}}^{(e)}$ by embedding similarity, and an LLM verifier LLM_{link} selects a single grounded value $v_e^* = \text{LLM}_{\text{link}}(e, V_{\text{raw}}^{(e)})$, yielding the value constraint set $V^* = \{(e, c_e, v_e^*)\}$, where $c_e \in C$ is the column matched to entity e during cell retrieval. When LLM_{link} rejects all candidates, the linker falls back through three weakening levels: *unconstrained* (drop e 's constraint), *LIKEfuzzy* (substring match), and *reroute* (emit TEXT); routes already in $H_{<t}$ raise the minimum fallback level (e.g., if TEXT is already in $H_{<t}$ the *reroute* fallback is skipped, and the linker proceeds with *LIKE fuzzy* or *unconstrained*). *CONSTRAINEDSQL* then samples a query restricted to the columns C retrieved from View 3 and seeded with V^* . Because both C and V^* are derived from the index, the executor cannot emit a column or cell value absent from the table, eliminating column hallucination (forbidden by C) and lexical mismatch (handled by V^*).

All four primitives share a single set of offline indices, and the router's choice picks which API consumes them. The four views are quantitatively complementary on HybridQA, with no single view dominating: row retrieval covers 69% of gold tables at $K=10$, schema 42%, document chunks 21%, and cell values 6%; a t-SNE projection of the view embeddings (Figure 6) confirms the views encode complementary semantic signals rather than redundant copies.

3.6 Validation, Escalation, and Stop Controller

The intermediate answer y_t is validated by a prompted LLM that consumes $(q_t, y_t, \text{evidence}_t)$ and returns a three-way verdict $\nu_t \in \{0, 0.5, 1\}$ corresponding to *reject*, *insufficient*, and *support*; the rubric and full prompt are given in Appendix Figure 10c. When $\nu_t < 1$, the chosen route is added to $H_{<t}$ and the router is invoked again on the same sub-query (up to three further attempts); because $H_{<t}$ is part of the router's input, each retry picks a primitive that consumes a strictly differ-

ent combination of indices than the failed attempt. The stop controller exits the outer loop as soon as $\nu_t = 1$ and $u_t < \tau$, cutting the mean iteration count without losing accuracy (§6).

If the SQL primitive returns a single scalar (one row and one numeric or short-string column) that value bypasses the synthesis prompt and is recorded verbatim as y_t . This prevents the LLM from paraphrasing a precise value such as 9,461,105 into "about 9.5 million."

3.7 Final Synthesis

The optional synthesis step rereads q together with all sub-answers $\{(q_t, y_t)\}_{t=1}^T$ in a single LLM call. The prompt enforces a minimal output format and arithmetic rules for unit preservation and percentage formulas. On HybridQA, it improves strict Exact EM by +3.25 and Normalized EM by +4.11 points by canonicalizing verbose sub-answers, with a smaller +2.90-point gain on Relaxed EM (§6).

4 Experimental Setup

Datasets. We evaluate on two hybrid table-text benchmarks and a table-only transfer test, the three benchmarks of Table 1, and on two further transfer benchmarks reported in Appendix C. **HybridQA** (Chen et al., 2020) pairs Wikipedia tables with passages linked from each cell, so most questions join cell content with free text; we use the dev split of Yu et al. (2025), the only one with published numbers for the strongest agentic prior. **TAT-QA** (Zhu et al., 2021) draws from financial filings and demands scale-aware arithmetic over tables beside explanatory paragraphs; we use the official dev set. **WTQ** (Pasupat and Liang, 2015) provides compositional queries over *standalone* tables; we use the pristine-unseen split, whose dev tables are disjoint from training, and treat it as a table-only transfer test of the router distilled on HybridQA. **MultiHiertt** (Zhao et al., 2022), numerical reasoning over multi-hierarchical tables and text, and **SPARTA** (Park et al., 2026), whose tree-structured questions run to four hops over corpora too large to prompt, are further transfer tests under the unchanged configuration; SPARTA is also where we compare against the two systems it adapts itself (§5).

Baselines. We compare ATR against four shared-protocol agentic baselines together with a no-retrieval floor. **Naive LLM** answers from the question alone, a contamination floor. **NaiveRAG** re-

Table 1: Main results (Gemini 2.5 Flash, BGE-M3, $sc=1$). **Bold** = best per column. *Judge* = Gemini Flash binary rubric (Appendix Figure 10d).

System	HybridQA				TAT-QA				WTQ			
	EM	Relax	F1	Judge	EM	Relax	F1	Judge	Norm	Relax	F1	Judge
<i>Baselines</i>												
Naive LLM	13.11	16.39	17.76	17.05	0.60	1.26	2.05	0.00	6.90	8.30	8.54	8.80
NaiveRAG (Lewis et al., 2020)	16.39	25.57	22.33	23.28	36.75	69.78	44.18	66.73	49.90	54.50	53.14	52.30
ReAct (Yao et al., 2023)	18.03	25.25	25.14	25.57	21.34	47.30	27.78	45.20	45.10	47.60	47.56	47.55
TableRAG-R (Chen et al., 2024c)	11.18	18.75	15.28	18.77	13.61	33.77	16.81	34.02	51.30	56.80	55.22	56.90
TableRAG-S (Yu et al., 2025)	5.90	44.92	16.84	38.82	1.44	38.97	9.11	56.30	16.30	67.80	32.06	66.60
<i>Ours</i>												
ATR (LLM router)	40.98	55.74	51.02	52.13	38.07	69.89	44.91	72.22	55.30	62.30	60.63	61.32
ATR (learned router)	40.83	61.59	54.02	54.02	38.55	68.32	44.73	71.00	57.60	64.00	62.81	63.29

trieves five chunks with BGE-M3 (Chen et al., 2024a) and a cross-encoder reranker, then answers in one call. **ReAct** (Yao et al., 2023) runs a Thought/Action/Observation loop over the same primitives, capped at ten steps. **TableRAG-R** (Chen et al., 2024c) combines BM25 with FAISS indices over schema, cell, and row views plus a Python sandbox. **TableRAG-S** (Yu et al., 2025) adds a relational SQL backend and cross-validates each sub-query between SQL and textual retrieval; it is the strongest agentic prior on the LLM-judged metrics (Relax, Judge). **HProPro** (Shi et al., 2024), a program-based system purpose-built for hybrid QA, is re-run from the authors’ released code on HybridQA and adapted to TAT-QA and WTQ, which its paper does not cover. Its pipeline places the question’s table in full in the prompt and generates a program against it, with no retrieval component of its own, so Appendix Table 16 reports it under both evidence conditions instead. Every row of Table 1 shares the protocol: identical Gemini 2.5 Flash backbone, identical BGE-M3 retrieval, and $sc=1$ (no self-consistency (Wang et al., 2023)), as do the HProPro re-runs. We also report an **oracle-context** control that removes retrieval from the comparison entirely. Dataset-specific supervised models (TagOp) and HybridQA-specialised graph methods (ODYSSEY) are excluded from this head-to-head (§5, Scope of comparison).

Metrics. Every cell in Table 1 uses each dataset’s official evaluator. For HybridQA we report the SQuAD-style `normalize_answer` Exact Match and token F1. For TAT-QA we report the official TagOp scale-aware Exact Match and numerical F1 from the original evaluator (Zhu et al., 2021). For WTQ we report the official Pasupat and Liang Normalized Exact Match together with token

F1. As a complementary semantic signal we also report an **LLM Judge** score on all three datasets, produced by a Gemini 2.5 Flash call with a fixed correctness rubric (Appendix Figure 10d), which credits correct answers that the strict EM evaluators penalise for surface form. We do not headline the substring-tolerant Relaxed Exact Match, which sentence-form predictions systematically inflate.

Implementation details. All systems share Gemini 2.5 Flash through Vertex AI as the backbone LLM for every prompted component (an open-weight Llama-3.3-70B backbone swap controlling for closed-source contamination is reported in §6.2); retrieval uses BGE-M3 (Chen et al., 2024a) with the BGE-reranker-v2-M3 cross-encoder, and the SQL backend reuses the MySQL service of Yu et al. (2025). Default ATR hyperparameters: $K_{\max}=5$, $\tau=0.1$, cell-index budget $B=10,000$ per dataset; only $K_{\max}$ and τ are selected on HybridQA-train and held fixed for TAT-QA and WTQ. The DistilBERT router (66M parameters) is distilled from 9,998 HybridQA-train LLM-router labels with class-weighted cross-entropy and AdamW (Loshchilov and Hutter, 2019); inference is a ~ 1 ms GPU forward pass at zero LLM cost. Full hyperparameter, hardware, and reproducibility details are in Appendix A.

5 Main Results

One pipeline across benchmarks. In Table 1, a single ATR configuration with the learned router and default hyperparameters leads on token F1 across all three datasets. HProPro has no row there: every row answers from cells, rows, and passages retrieved under the shared budget, whereas HProPro reads a whole table from its prompt and retrieves nothing (Appendix Table 16, both

evidence conditions). Under matched evidence, with neither system told which table holds the answer, ATR leads by 13.1 pp EM on HybridQA and 1.3 on TAT-QA, and the two are level on WTQ (−0.6) and on SPARTA, where the pooled difference over 1,685 questions is indistinguishable from zero ($\Delta = -0.68$, $p = 0.57$; Appendix Tables 16 and 18). Three seeds give F1 53.87 ± 0.41 and Relaxed 61.32 ± 0.56 ; the gaps carrying our claims (+37.2 F1 over the strongest agentic prior, −8.26 for *No Router*) exceed that spread by one to two orders of magnitude (see Limitations). The HybridQA EM/Relax gap is a surface-form artifact (Appendix Table 13): the Judge score (54.02) sits between them, ruling out Relax-only inflation. On the two cells we do not lead (WTQ Relaxed and Judge), TableRAG-S emits sentence-form outputs far longer than ATR’s, which substring-tolerant and LLM-judged metrics reward: that is what separates ATR’s 57.60 Normalized EM from TableRAG-S’s 16.30 while the Relaxed gap runs the other way, and why we headline Normalized EM and token F1 (Appendix Table 14). The aggregate TAT-QA F1 gap to NaiveRAG under-reports ATR: many TAT-QA dev questions are single-span extractions answered verbatim in the retrieved passage, and the `per-answer_type` breakdown (Table 6) puts ATR +5.43 pp ahead on *arithmetic* and +6.25 on *count*, where routing pays. The configuration also transfers to MultiHiertt (33.8 EM/39.6 F1 against NaiveRAG’s 28.9/34.2; Appendix Table 15). Top-1 table accuracy (11.8% HybridQA, 17.6% TAT-QA, 37.5% WTQ) is HProPro’s ceiling, not ATR’s: bound to one retrieved table, HProPro scores below it on all three and cannot recover when retrieval misses, while ATR fuses chunks across sources, re-routes on failed verification, and answers past that ceiling on HybridQA (19.7 against 11.8). Appendix D (Figure 9) traces a HybridQA two-hop question end-to-end, where ATR mixes HYBRID and TEXT within one question, the signature of sub-query-level routing.

Cross-dataset transfer of the distilled router.

The DistilBERT router never observes WTQ or TAT-QA training data, yet matches the LLM teacher on both unseen benchmarks, leading on WTQ and trailing by at most 1.6 pp on TAT-QA, at zero per-sub-query cost (Table 1); Appendix C traces the transfer to the operator- and modality-conditioned fields the student routes on.

Oracle-context control. Leading across benchmarks does not by itself separate routing from compensation for missed retrieval. We therefore feed each dev question’s gold table and all linked passages to a single Gemini 2.5 Flash call with no retrieval and no decomposition; every HybridQA dev question fits the context window (Appendix C). The result splits along exactly that line: with every gold token in context the baseline still trails ATR on five of the six reported cells, by 4.3 pp EM and 2.2 pp F1 on HybridQA and by 4.4 pp on TAT-QA *arithmetic* and 9.4 pp on *count* (Appendix Table 9), because one unconstrained call mis-executes multi-row computation even with perfect evidence; it leads only on TAT-QA aggregate F1, carried by the *span* questions that need only be read off. What survives once retrieval is fully compensated is what routing and constrained execution add.

Scope of comparison. Systems differ on two axes: protocol and *the evidence they are given*. *Shared-protocol baselines* (NaiveRAG, ReAct, TableRAG-R, TableRAG-S) match ATR’s backbone, retrieval stack, and $sc=1$; these are the only head-to-head we claim. On the second axis, ATR and the retrieval baselines answer from the shared indices under a fixed budget, while the purpose-built hybrid-QA systems consume theirs whole: HProPro the question’s entire table, ODYSSEY the gold tables, the full passage store, and the cell-passage link graph. Neither assumption survives deployment, so Appendix C reports both as references, each under the evidence it assumes and under ATR’s retrieval. *Dataset-specific methods* (TagOp (Zhu et al., 2021), ODYSSEY (Agarwal et al., 2025)) additionally gain from benchmark-tied training or architecture. TagOp is supervised on TAT-QA itself; ODYSSEY’s published numbers, and the three differences that keep it out of Table 1 (evaluation sample, GPT-4 backbone, and a cell-link graph only HybridQA provides), are in Appendix C.

6 Ablations and Analysis

6.1 Component ablation

Table 2 (top block) isolates each component’s contribution by removing it one at a time from Full ATR. On HybridQA token F1, capping the loop at $K_{\max}=1$ collapses the system to one-shot QA at −48.8 pp, and the sweep in Figure 5 shows accuracy recovering gradually with the iteration budget

Table 2: Ablation on HybridQA dev. *Top*: component removal; *bottom*: backbone swap (learned router fixed; full matrix in Table 7). **Bold** = best per column; Δ F1 vs Full ATR.

Variant	Exact	Norm	Relax	F1	Judge	Δ F1
Full ATR (Gemini Flash + learned router)	37.02	40.83	61.59	54.02	54.02	—
<i>Component removal</i>						
– Escalation	12.46	13.77	23.61	18.32	19.08	−35.70
– Iteration ($K_{\max}=1$)	3.28	3.28	5.25	5.26	3.28	−48.76
– Decomposer (raw question)	8.52	9.18	16.07	13.49	14.43	−40.53
– Final Synthesis	33.77	36.72	58.69	48.77	51.64	−5.25
No Router (forced HYBRID)	33.11	36.72	49.51	45.76	44.26	−8.26
<i>Backbone swap (learned router fixed)</i>						
Gemini 2.5 Pro	38.36	42.30	63.28	54.58	54.10	+0.56
Claude Haiku 4.5	35.74	39.67	58.69	50.15	54.28	−3.87
Llama-3.3-70B-AWQ	39.67	44.92	56.39	54.68	55.08	+0.66
Qwen3.5-122B-A10B-GPTQ-Int4	40.33	42.62	57.05	52.76	51.15	−1.26

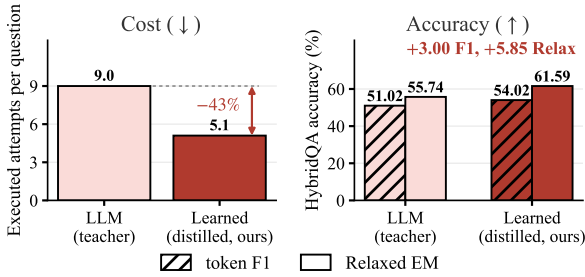


Figure 3: Accuracy and cost trade-off of router variants on HybridQA dev: mean executed attempts per question (left); token F1 and Relaxed EM (right).

rather than switching on at a threshold; removing re-routing on failure costs −35.7 pp; and replacing the decomposer with the raw question costs −40.5 pp. These contributions compose: the iterative loop *generates* sub-queries that the router *dispatches* and the verifier *rescues*. The *No Router* (forced HYBRID) row of the same block shows that removing routing alone costs −8.3 pp of F1.

6.2 $K_{\max}$, router, and backbone ablations

$K_{\max}$ saturation. Figure 5 reports the full $K_{\max}$ sweep on HybridQA dev. Relaxed climbs steeply to $K_{\max}=3$, peaks at 5, and regresses past it, which Appendix C reads as dilution; the strict metrics instead gain at most 3 pp out to $K_{\max}=10$, buying answer-form canonicalization rather than coverage (§3.7). We adopt $K_{\max}=5$, selected on HybridQA-train and not on this curve. The realised loop length is itself adaptive, with 8% of questions terminating after one iteration and 38% after two while 30% reach the cap, where the sweep above places the trade-off.

Teacher vs. student. The distilled student recovers 98.6% of the LLM teacher’s routing decisions

on HybridQA; the resulting accuracy profile is at or above the teacher’s on WTQ, gains 3.00 pp token F1 and 5.85 pp Relaxed on HybridQA, and stays within 1.6 pp on every TAT-QA metric (Table 1, ATR (LLM router) vs. ATR (learned router)). The student exceeds the teacher’s token F1 while cutting mean executed attempts per question from 9.0 to 5.1 (Figure 3). *A counter-intuitive finding*: the student distilled from the LLM teacher’s labels outperforms a student trained on *oracle* routing labels. The oracle-labeled student gives up 2.4 pp F1 and 3.2 pp Relaxed and fires escalation 9.3 pp more often (47.6% against 38.3%), at 91.4% agreement with the teacher (Appendix Table 10). Oracle labels mark which route *can* solve each sub-query in isolation, whereas every teacher label is emitted under a rule that excludes any route already in the failed-route set (Appendix Figure 10b, rule R6), conditioning them on the escalation chain. They diverge most where that conditioning bites: on the SQL $\leftrightarrow$ HYBRID borderline band (0.8% of sub-queries) the oracle student picks SQL in 78% of cases against 31% for the teacher student. R6 as the mechanism is checkable in the prompt itself; the broader reading, a downstream-cost-aware policy rather than an isolated correctness classifier, we offer as a hypothesis.

Backbone. Table 2 (bottom block) reports backbone sensitivity on HybridQA dev with the learned router fixed across one representative from each major family, spanning closed-weight (Gemini, Claude) and open-weight (Llama, Qwen); frontier checkpoints are reported in Appendix C (Table 7). Across this set the architectural contribution in the top block dwarfs the backbone effect by an order of magnitude, and swapping only the decom-

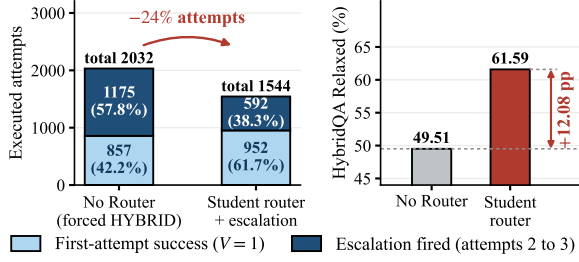


Figure 4: Router value vs. No Router (forced HYBRID): -24% attempts and +12.08 pp Relaxed accuracy on HybridQA.

poser or only the verifier verdict likewise moves F1 by at most 5.9 pp across seven checkpoints per role (Appendix Table 8). Self-hosted Llama-3.3-70B and Qwen3.5-122B via vLLM (Kwon et al., 2023) bracket the Flash headline within ± 1.3 pp F1 at no API cost (Table 7), so ATR’s gains do not require closed-weight access on Wikipedia-style tables. TAT-QA is less symmetric: open-weight backbones still trail on its strict-arithmetic evaluator, a gap the architecture alone does not close (see Limitations).

Where the router’s value comes from. The router’s value shows on both axes at once (*No Router* row of Table 2; Figure 4): against the No Router (forced HYBRID) baseline, the student router raises Relaxed accuracy by +12.08 pp on HybridQA while cutting escalation firings from 57.8% to 38.3% of executed attempts, and total attempts by 24%.

6.3 HybridValueLinker telemetry and value-alignment ablation

Replacing the linker with verbatim NL2SQL costs 3.9 pp F1 inside HYBRID-routed sub-queries, where its output constrains the SQL, and near zero elsewhere; over the whole dev set that dilutes to 2.7 pp on HybridQA and 1.8 on TAT-QA (Appendix Table 11). The effect sits where the mechanism predicts: REWRITE and PARTIAL alignments cover 55.7% of linked entities on HybridQA and 42.9% on TAT-QA (Appendix Figure 8), exactly the cases verbatim SQL misses, and disabling those levels places 2.3 of the 2.7 pp there. Because the linker also gates the fallback to TEXT, disabling it shifts escalation firings from 38.3% to 46.9% and mean attempts from 5.1 to 5.9, so it buys accuracy and compute together. What we measure here is the integration into routing, not the linking mechanism (§2). Reimplementing Tool-SQL’s in-

loop selection without the cross-primitive fallback leaves 1.2 of the 2.7 pp unrecovered: repairing in place reaches 1.5, and the rest needs an exit to a *different* primitive.

6.4 Failure modes

Three regimes dominate the residual error, and two of them reach the output only when ATR’s recovery path fails as well. **Borderline first choices** sit on the SQL $\leftrightarrow$ HYBRID band of §6.2, where a wrong route costs an extra attempt rather than the answer. **Verifier false-support** ($\nu_t=1$ on a wrong sub-answer) contributes the larger share of the verifier-attributable error, but the oracle-verifier bound (Table 8) shows a perfect verdict recovers only +2.2 pp F1, so most residual error is sub-answers ATR never produces, not verdict mistakes. Measuring the verdict directly, two authors annotated 200 dev decisions stratified across verdicts ($\kappa=0.81$): the verifier is right on 89.2% of the sub-answers it accepts and 93.7% of those it rejects, and the residual false-support (10.8%) clusters on rows admitting more than one answer: two equally plausible Kamakura-period authors for one work, several disaster types in one row. **Narrative-attribute sub-queries** have no DB column to ground, so a HYBRID route there relies on the linker’s failed grounding to exit to TEXT (§3.5); the *No Router* ablation prices forcing it everywhere.

7 Conclusion

We presented ATR, an iterative framework whose central choice is the *unit* of routing: each sub-query, rather than each question, is dispatched to one of four execution primitives, with the route set closed under failure, and value linking sits inside that loop as a precondition of constrained execution, not a repair applied after it. A single ATR configuration matches or exceeds the strongest agentic prior on HybridQA and TAT-QA and transfers unchanged to WTQ, MultiHiertt, and SPARTA, which the router never saw.

The distilled student removes the per-sub-query LLM call and outperforms a student trained on oracle routing labels: the teacher’s labels are conditioned on the verifier’s rejections, which isolated correctness labels cannot express. Routing is worth 8.26 pp F1 while moving escalation from 57.8% of executed attempts to 38.3%, and transfers to open-weight backbones. Two directions follow: subgoal planning beyond $K_{\max}=5$ and multimodal tables.

Limitations

Headline numbers are point estimates on each benchmark’s development set: we repeat the HybridQA headline over three seeds (§5) but report single runs on TAT-QA, WTQ, and MultiHiertt, run a paired bootstrap only for the SPARTA comparison of Appendix Table 18, and read ablation deltas below ~ 1 pp as within noise; *Judge* adds a further ± 0.5 pp run-to-run variance. The verifier is itself a prompted LLM; its measured error is small (§6.4) but comes from one annotated sample on one benchmark. The open-weight Llama backbone matches Flash on Wikipedia-style tables but trails on TAT-QA’s strict-arithmetic evaluator, a backbone gap the architecture alone does not close. ATR caps the outer loop at $K_{\max}=5$, so genuinely deeper compositional chains (5+ hops) likely need explicit subgoal planning; our main hybrid benchmarks reach at most two hops, and on the deeper tree-structured SPARTA every system we measure drops steeply beyond single-hop questions, so we claim robustness of that comparison, not coverage of deep composition. The scalar fast path records single-scalar SQL outputs verbatim, preventing paraphrase of precise values, but the short outputs it emits lose on substring-tolerant metrics (§5). All benchmarks are English and cover Wikipedia, financial filings, and crowd-written tables only; cross-lingual transfer is untested, though the BGE-M3 retriever is itself multilingual. The distilled router pays no LLM cost at inference, but training requires labels from the LLM teacher, and reproduction requires the MySQL-backed SQL service of Yu et al. (2025) per dataset.

Ethics Statement

Use of Scientific Artifacts. We use PyTorch (Paszke et al., 2019), HuggingFace (Wolf et al., 2020), and vLLM (Kwon et al., 2023); open-weight backbones and BGE retrieval checkpoints are obtained through HuggingFace, and closed-source LLMs through Google Vertex AI and OpenAI under each provider’s terms for academic research. HybridQA, TAT-QA, WTQ, MultiHiertt, and SPARTA are used as publicly released.

Use of AI Assistants. We only used Google Gemini to provide a better expression and to refine the wording. Some of the code used in the experiment was written with the assistance of Anthropic

Claude. All AI-assisted outputs (writing and code) were reviewed and verified by the authors, who take full responsibility for the final content of this paper.

Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (Grant No. RS-2022-NR068758, Grant No. RS-2025-16071337).

References

- Ankush Agarwal, Chaitanya Devaguptapu, and Ganesh S. 2025. Hybrid Graphs for Table-and-Text based Question Answering using LLMs. In *Proceedings of NAACL-HLT 2025*, pages 858–875.
- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *ICLR 2024*.
- Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In *Findings of EMNLP 2020*.
- 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*.
- Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of ACL 2024*.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2024b. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*.
- Si-An Chen, Lesly Miculicich, Julian Martin Eisen-schlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024c. TableRAG: Million-Token Table Understanding with Language Models. In *NeurIPS 2024*.
- Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding Language Models in Symbolic Languages. In *ICLR 2023*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

- Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*, pages 4171–4186.
- Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of EMNLP 2020*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *SOSP 2023*.
- 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 *NeurIPS 2020*.
- Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-Source Language Models for Text-to-SQL. *Proceedings of the ACM on Management of Data*, 2(3), Article 127.
- Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going? *IEEE Transactions on Knowledge and Data Engineering*, 37:5735–5754.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR 2019*.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to Route LLMs from Preference Data. In *ICLR 2025*.
- Sungho Park, Jueun Kim, and Wook-Shin Han. 2026. SPARTA: Scalable and Principled Benchmark of Tree-Structured Multi-hop QA over Text and Tables. In *ICLR 2026*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS 2019*.
- Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *ACL-IJCNLP 2015*.
- Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *NeurIPS 2023*.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of EMNLP 2023*, pages 5687–5711.
- Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the Text-to-SQL Capabilities of Large Language Models. *arXiv:2204.00498*.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. In *NeurIPS EMC² Workshop*.
- Qi Shi, Han Cui, Haofeng Wang, Qingfu Zhu, Wanxiang Che, and Ting Liu. 2024. Exploring Hybrid Question Answering via Program-based Prompting. In *ACL 2024*, pages 11035–11046.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *NeurIPS 2023*.
- Ruoxi Sun, Sercan O. Arik, Alexandre Muzio, Lesly Miculicich, Satya Kesav Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, and Tomas Pfister. 2024. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. *Transactions on Machine Learning Research*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *ICLR 2023*.
- Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Jaemin Kim. 2024. Tool-Assisted Agent on SQL Inspection and Refinement in Real-World Scenarios. *arXiv:2408.16991*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP 2020: System Demonstrations*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR 2023*.
- Xiaohan Yu, Pu Jian, and Chong Chen. 2025. TableRAG: A Retrieval Augmented Generation Framework for Heterogeneous Document Reasoning. In *EMNLP 2025*, pages 14063–14082.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data. In *ACL 2022*, pages 6588–6600.

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 *ACL-IJCNLP 2021*.

A Implementation Details

Backbone LLM. Gemini 2.5 Flash is served through Vertex AI (us-central1) for every prompted component (decomposer, LLM-router teacher, executors, verifier, value linker, final synthesis), with default temperature and top- p . Open-weight variants in Table 2 are served via self-hosted vLLM 0.6 with AWQ-INT4 quantisation on $2 \times$ A6000 (tensor-parallel size 2), exposed through an OpenAI-compatible endpoint.

Retrieval stack. BGE-M3 (Chen et al., 2024a) provides the dense 1024-dim embedding for Views 1 through 4; the BGE-reranker-v2-M3 cross-encoder reranks the top candidates per query. Document chunks are produced with a recursive character splitter at 512-token chunk size and 64-token overlap. The MySQL SQL service of Yu et al. (2025) is reused without modification.

ATR hyperparameters. Default values: $K_{\max}=5$, $\tau=0.1$, cell-index budget $B=10,000$ per dataset, schema top- $K=5$, cell top- $K_v=15$, row top- $K_r=10$, escalation cap of three further attempts per sub-query, worker-thread count = two. No per-dataset tuning beyond $K_{\max}$ and τ on HybridQA-train.

Router training. The DistilBERT router (66M parameters) is fine-tuned for five epochs with class-weighted cross-entropy on 9,998 HybridQA-train sub-query labels distilled from the LLM-router teacher. We use AdamW ($\text{lr } 2 \times 10^{-5}$, batch size 32); class weights are inversely proportional to route frequencies in the training set. The teacher emits hard route labels (argmax), so the student is supervised on auto-labelled data rather than soft teacher logits. Training completes in under one hour on a single NVIDIA RTX 3080.

Reproducibility. Inference runs are deterministic (seed 42, sampling temperature 0 except where noted) and single-seed apart from the HybridQA headline, which we repeat over three seeds (see

Limitations). Random sub-sets (e.g., the TAT-QA dev sample used in the backbone-swap analysis) use the same seed. Code, training scripts, and prompt templates are available at <https://github.com/ayoung206/ATR>.

Hardware and wallclock. Inference runs are I/O-bound against the Vertex AI / OpenAI / vLLM endpoints rather than compute-bound on the local machine; the client runs on a single CPU node with eight worker threads and two parallel sub-query workers. End-to-end wallclock per HybridQA dev pass is ~ 30 minutes for Gemini 2.5 Flash, ~ 2.5 hours for Claude Opus 4.7, and ~ 2.5 hours for the self-hosted Qwen3.5-122B-A10B-GPTQ-Int4 / Llama-4-Scout-17B-16E-AWQ runs on $2 \times$ A6000. The DistilBERT router adds ~ 1 ms per sub-query on a single GPU and is therefore a negligible contribution to total wallclock. The MySQL SQL service of Yu et al. (2025) runs on the same node as the client; for self-hosted backbones the vLLM endpoint lives on a separate $2 \times$ A6000 box reachable over a 10 Gb/s LAN link, and the per-call round-trip latency is dominated by LLM generation rather than network transport.

Data preprocessing. We follow the preprocessing of Yu et al. (2025) on every benchmark: HybridQA uses their canonical dev split (linked passages held fixed across all backbones; cell-grounded answer spans are not used at training or inference); TAT-QA uses the official dev split with the per-question scale annotations from the TagOp (Zhu et al., 2021) evaluator; WTQ uses the Pasupat and Liang pristine-unseen split with the standard `normalize_answer` function. Excel tables are persisted to MySQL by the offline pipeline of Yu et al. (2025), with the table name normalisation rules (`transfer_name` offline / `_normalize_table_key` online) kept in sync between the offline ingestor and the online SQL router.

SubQuery schema. Table 3 enumerates the six fields of the structured SUBQUERY object emitted by the Decomposer at each iteration t (§3.3). Each field has a designated downstream consumer: the Router reads `expected_operator`, `required_modalities`, and `entity_mentions`; `need_global_table_view` enters the distilled router’s meta-features; the HYBRID-VALUELINKER grounds `entity_mentions`;

the stop controller reads `uncertainty`. The schema is therefore the load-bearing interface between the LLM-prompted decomposer and the rest of the pipeline, and constrains the decomposer to a small set of emit-time decisions that the next-stage modules can act on without re-prompting. A typical emitted object on HybridQA looks like `{sub_query: "What is the population of the city where the 2009 final was held?", expected_operator: lookup, required_modalities: both, need_global_table_view: false, entity_mentions: {2009 final}, uncertainty: 0.6}`, which the router reads as “cell-grounded look-up over a table with a linked passage” and dispatches to RETRIEVE.

B Online Inference Loop

Algorithm 1 gives the line-by-line procedure of the online loop described in §3.1. The pseudocode is a page-wide float because its lines do not fit a single column without breaks; section references in the algorithm point back to the corresponding method subsections.

The loop has two nested iterations. The *outer* loop iterates over decomposed sub-queries (bounded by $K_{\max}=5$); the *inner* loop iterates over routing attempts for a single sub-query (the first-choice route plus up to $M=3$ escalation attempts). Two early-exit paths short-circuit the outer loop: (i) the decomposer emits the sentinel TERMINATE when it judges the question already answered, and (ii) the stop controller exits when the verifier returns $\nu_t=1$ and the residual uncertainty u_t falls below $\tau=0.1$. Together these two exits save roughly a third of the maximum iteration budget on HybridQA dev: the mean realised iteration count is 3.1 (vs. 5 cap), and 30% of questions reach the cap.

Inside the inner loop, the failed-route history $H_{<t}$ enters the router’s input, so each escalation is conditioned on which primitives have already been rejected, a discrete form of counterfactual routing rather than a free-form LLM patch. The teacher’s escalation preferences differ by sub-query type, operator-heavy sub-queries and lookup/filter sub-queries falling back through different primitives, and the student internalises both. If Final Synthesis is disabled (toggled off in the – Final Synthesis ablation), the loop returns the most recent verifier-accepted answer y_t directly, which works

for single-hop questions but loses the multi-hop aggregation that drives the -5.3 pp F1 in Table 2.

C Additional Ablations

This section reports the ablations referenced from §6, organised around four questions: how routing trades latency for accuracy, where the router’s value actually comes from, how iteration interacts with student/teacher agreement, and how the four offline views overlap.

Iteration and retrieval views. Figure 5 reports the $K_{\max}$ sweep on HybridQA dev: Relaxed EM climbs from 16.1 at $K_{\max}=1$ to 61.6 at $K_{\max}=5$ and slightly regresses at $K_{\max}=7$ (58.7). Its $K_{\max}=1$ point is the no-decomposition run, which answers each question in one shot; the – Iteration row of Table 2 keeps the decomposer and executes only its first sub-query, so the two single-iteration configurations do not coincide. The early steep slope ($K_{\max}=1 \rightarrow 3$: +28 pp) reflects multi-hop questions that need at least two to three sub-queries to resolve; the diminishing return between 3 and 5 tracks questions whose remaining hops are answered by Final Synthesis rather than further decomposition. The regression at $K_{\max}=7$ is a dilution effect: the decomposer keeps emitting sub-queries past the natural compositional depth of the question, and their loosely related sub-answers push Final Synthesis toward off-topic paraphrases. We therefore use $K_{\max}=5$ as the default. Coupling this with the stop controller (§3.6) keeps the mean realised iteration count to 3.1 even at the 5-iteration cap.

Figure 6 projects the four offline view embeddings (§3.2) into a shared BGE-M3 space via t-SNE; clusters by view are largely disjoint, confirming that the views encode complementary semantic signals rather than redundant copies. Gold-table coverage at $K=10$ is uneven across views (row 69%, schema 42%, chunks 21%, cells 6%): row retrieval is the highest-recall single view, but no view covers everything, and the cell view is the only one that surfaces the precise (column, value) target needed for value linking, even though its standalone gold-table recall is the lowest. This is exactly the asymmetric-utility pattern that a route-conditional consumer can exploit: HYBRID needs cells for grounding, SQL needs schema, TEXT needs chunks, and RETRIEVE needs rows. A single fused index would either over-weight the high-recall but low-precision row view (collapsing

Table 3: The SUBQUERY object emitted by the DECOMPOSER.

Field	Type	Range / Domain	Role
sub_query	string	free text	natural-language sub-question
expected_operator	enum	lookup, filter, count, compare, aggregate, sort, arithmetic	coarse reasoning primitive; routing feature
required_modalities	enum	table, text, both	evidence modality; routing feature
need_global_table_view	bool	true, false	full-table aggregation vs. local lookup
entity_mentions E_t	set	$\{e_1, \dots, e_n\}$	spans grounded by the value linker
uncertainty u_t	real	$[0, 1]$	residual fraction of q unanswered after q_t ; stop-controller signal

schema and cell signal) or pay the cost of maintaining four indices anyway; we therefore keep them separate and let the router pick which to consult.

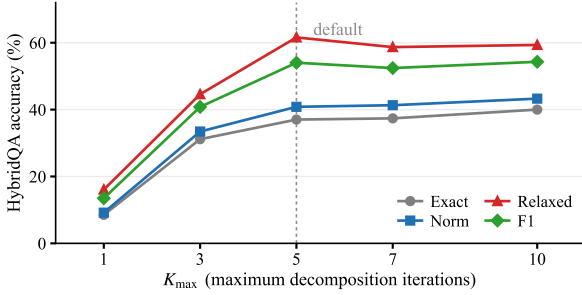


Figure 5: $K_{\max}$ sweep on HybridQA dev. Relaxed peaks at the default $K_{\max}=5$; the strict metrics creep up to $K_{\max}=10$.

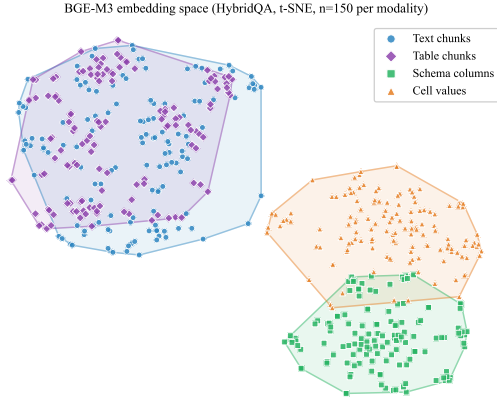


Figure 6: t-SNE projection of the four offline retrieval views (text chunks, table chunks, schema columns, cell values) on a shared BGE-M3 space (HybridQA).

Design-space comparison. Table 4 maps each system against seven load-bearing axes (cell index, NL2SQL, multi-route, verifier + escalation, iterative decomposition, self-consistency, graph traversal). ATR is the only system that combines the first five, and runs at $sc=1$ without graph traversal; the dataset-specific ODYSSEY adds graph traversal at

the cost of every other axis. Figure 7 translates the same comparison into the (mean executed attempts per question, HybridQA accuracy) plane: ATR sits on the Pareto front with highest accuracy at a moderate attempt budget; TableRAG-S trades more attempts for similar Relaxed accuracy but worse F1; NaiveRAG and ReAct sit below ATR on accuracy at comparable attempt counts.

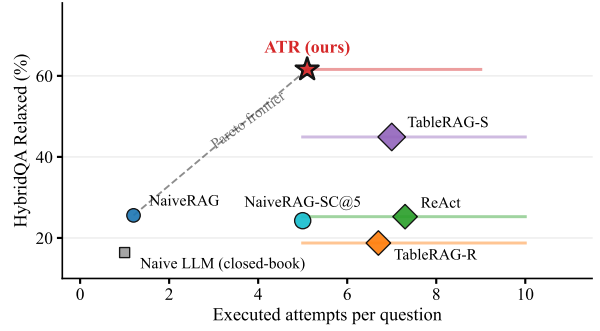


Figure 7: ATR sits on the Pareto front: top accuracy at a modest attempt budget. Whiskers mark each system’s min-to-max attempt range.

Value-linker telemetry. On the HybridQA dev set, the HYBRIDVALUELINKER classifies its successful outputs as EXACT (44.3%, the entity already matches a cell verbatim), PARTIAL (34.8%, substring or numeric-format overlap), or REWRITE (20.9%, canonical swap). On TAT-QA dev the corresponding fractions are 57.2%/20.5%/22.4%. The REWRITE+PARTIAL share, 42 to 56% across the two benchmarks, quantifies the slice of entities that pure NL2SQL would emit verbatim and miss, which is where the full-system ablation of §6.3 places the linker’s effect. Figure 8 visualises the same per-class distribution.

Table 5 samples representative alignments. REWRITE covers canonical-name swaps such as the ordinal `second` $\rightarrow$ `2nd` or the abbreviation `GP` $\rightarrow$ `Grand Prix`, cases where the question’s lexical form and the cell’s storage form are lin-

Table 4: Design-space comparison. ●/◐/◑ = has / partial / absent. ATR is the only system covering the first five axes.

System	Cell index	NL2SQL	Multi-route	Verifier + Esc.	Iter. decomp.	Self-cons.	Graph traversal
Naive LLM	◑	◑	◑	◑	◑	◑	◑
NaiveRAG	◑	◑	◑	◑	◑	◑	◑
NaiveRAG-SC@5	◑	◑	◑	◐	◑	●	◑
ReAct (≤ 10 steps)	◑	◐	◐	◐	◐	◑	◑
TableRAG-R	◐	◑	◑	◐	●	◐	◑
TableRAG-S	◑	●	◑	◐	●	◑	◑
ODYSSEY	◑	◑	◐	◑	◐	◑	●
ATR (ours)	●	●	●	●	●	◑	◑

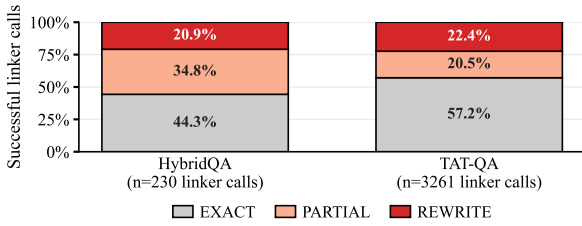


Figure 8: HYBRIDVALUELINKER output distribution. EXACT: entity matches a cell verbatim; PARTIAL: substring or format normalisation; REWRITE: lexical translation (e.g. 'second' $\rightarrow$ '2nd').

guistically equivalent but lexically disjoint. PARTIAL covers substring and numeric-format normalisations such as 2011-05-17 $\rightarrow$ 2011 (year extraction) or Lindsay $\rightarrow$ Lindsay Gilbee (first-name expansion), all cases where the question token is a sub-token of the cell or vice versa. TAT-QA's financial filings add a third flavour, 2019 $\rightarrow$ 19, where the cell stores a short-form year header that the question uses in long form.

TAT-QA per-answer_type breakdown. Table 6 decomposes the TAT-QA dev results of Table 1 along TAT-QA's answer_type label (arithmetic, count, multi-span, span). ATR leads NaiveRAG by +5.43 pp F1 on arithmetic ($n=718$, the largest type) and +6.25 pp on count ($n=32$), the two types that require explicit multi-step computation over table values. On multi-span ($n=217$) the two systems are within ± 2 pp F1. On span ($n=701$, 42% of dev) NaiveRAG leads by 4.01 pp F1: these questions have their answer present verbatim in the retrieved passage, so a retrieve-and-extract pipeline with a strong backbone (Gemini 2.5 Flash) already saturates the metric; ATR's iterative decomposition

Table 5: Representative HYBRIDVALUELINKER alignments. REWRITE: canonical-name swap. PARTIAL: substring or numeric-format normalization.

Class	Question entity	Cell value
<i>HybridQA</i>		
REWRITE	GP	Grand Prix
REWRITE	fifth season	Season 5
REWRITE	midfielder	MF
REWRITE	St. Louis	Saint Louis
PARTIAL	2011-05-17	2011
PARTIAL	Lindsay	Lindsay Gilbee
PARTIAL	NY	NY Yankees
PARTIAL	1.83 m	1.83
<i>TAT-QA</i>		
REWRITE	Solutions seg.	Industrial Sol.
REWRITE	net revenues	Net sales
REWRITE	domestic disc.	4.00%
PARTIAL	2019	19
PARTIAL	\$1,234 mil.	1234
PARTIAL	Q4 2019	2019-Q4

adds little here and occasionally over-paraphrases the span, which TAT-QA's strict TagOp scorer punishes. The aggregate F1 gap (+0.55 pp) is therefore a weighted average dominated by the span bucket; the architectural contribution lives in the computation-heavy buckets where ATR's value-linking and constrained SQL execution actually matter.

Table 6: TAT-QA dev per-answer_type breakdown (ATR learned router vs. NaiveRAG, Gemini 2.5 Flash). n is the question count per type. **Bold** = better per cell.

Type	n	EM		F1		$\Delta F1$
		RAG	ATR	RAG	ATR	
arithmetic	718	36.21	41.64	36.21	41.64	+5.43
count	32	53.13	59.38	53.13	59.38	+6.25
multi-span	217	29.49	31.34	45.11	43.39	-1.72
span	701	38.80	36.66	51.64	47.64	-4.01
Agg.	1668	36.75	38.55	44.18	44.73	+0.55

Full backbone catalogue. Table 7 reports the full backbone swap on HybridQA dev across nine backbones, split into closed-weight (Gemini 2.5 Flash anchor, Gemini 2.5 Pro, Gemini 3.1 Pro Preview, Claude Haiku 4.5, Claude Opus 4.7, GPT-4o-mini) and open-weight self-hosted via vLLM (Llama-3.3-70B, Qwen3.5-122B, Llama-4 Scout). Three patterns emerge.

Within-family variation is small at competitive scale. Within the Gemini family, Pro is within +0.56 pp F1 of the Flash anchor, while Gemini 3.1 Pro Preview adds another +9.34 pp on top of Pro, so almost all of the family’s spread comes from the frontier release, not from the Pro/Flash split at the previous generation. Within Claude, Opus 4.7 leads Haiku 4.5 by +12.02 pp F1; the gap is wider than the Gemini family because Haiku is positioned as a small/efficient model rather than a frontier sibling. Within Llama, Llama-3.3-70B (dense) leads Llama-4-Scout (MoE, 17B active) by +2.89 pp F1 on this workload despite Scout’s 1.6× larger total parameter count (109B against 70B): structured-JSON agentic prompts favour the dense model’s far larger active compute over sparse MoE routing (§6.2).

Cross-family ordering tracks active compute, not total parameters. The top of the table (3.1 Pro Preview, Opus 4.7, Llama-3.3-70B) and the bottom (GPT-4o-mini, Haiku 4.5) split along the active-parameter axis: frontier closed models and dense 70B open-weights win, while small reasoning-tuned checkpoints (GPT-4o-mini’s 8B-class budget, Haiku 4.5’s small-tier positioning) lose by −4 to −9 pp F1. Total parameters in MoE models (Llama-4-Scout’s 109B, Qwen3.5’s 122B) do not close this gap because only ∼10 to 17 B are active per token on a structured JSON workload.

Open-weight ATR is competitive with closed-weight Flash. Llama-3.3-70B-AWQ self-hosted via vLLM (+0.66 pp F1 vs. Flash) and Qwen3.5-122B-A10B-GPTQ-Int4 (−1.26 pp F1) bracket the Flash headline within ±1.3 pp at zero API cost. This is the empirical claim behind the floor/ceiling reading: ATR’s architectural floor (the persistent gap to baselines under *any* backbone choice) does not require closed-weight LLM access on Wikipedia-style tables.

Decomposer- and verifier-model robustness. Table 8 isolates how much ATR depends on the *model* behind its two single-LLM roles, the decomposer and the verifier verdict, by holding the rest of

the pipeline at Gemini 2.5 Flash and swapping one role’s model at a time. Across seven checkpoints per role, spanning closed (Gemini, Claude) and self-hosted open-weight (Llama-3.3-70B, Llama-4 Scout, Qwen3.5-122B) models, the decomposer swap moves headline F1 within $[-5.9, +1.3]$ pp of the Flash anchor and the verifier-verdict swap within $[-4.2, +2.1]$ pp; both spans sit far inside the −35 to −49 pp cost of removing a component in Table 2, so neither role is a fragile single point of failure. The effect is non-monotonic in model scale, since Gemini 3.1 Pro helps while Gemini 2.5 Pro hurts in both roles, and the extremes do not follow weight class either: the largest drop in the verifier block is a closed-weight model (Claude Haiku 4.5, −4.24) and in the decomposer block a self-hosted one (Llama-3.3-70B, −5.86). Both patterns indicate that decomposition and verdict *behaviour*, not raw capability, drives the residual variance. The *Oracle verifier* row replaces the verdict with a gold match and upper-bounds the accuracy a perfect verifier could recover from the candidates ATR actually produces: it adds only +2.23 pp F1 (+3.98 pp Relaxed) over the anchor, so the verifier’s recoverable cost is small and the larger share of residual error is sub-answers ATR never generates rather than correct sub-answers the verifier discards. A deployable verdict model already reaches that ceiling: the Gemini 3.1 Pro verifier’s +2.13 pp sits 0.10 pp below the oracle’s +2.23, so verdict quality is close to exhausted as a source of further gain.

Router error modes. Beyond the global agreement rate (98.6% between DistilBERT student and LLM teacher on HybridQA dev), the residual disagreements concentrate at three boundaries. SQL↔HYBRID disagreements (∼0.8% of sub-queries) occur on row filters with at most one entity, where both primitives are plausible and the teacher’s preference is subtle (whether the cell needs linking or already matches). TEXT↔RETRIEVE disagreements (∼0.4%) occur on biographical look-ups whose target attribute might or might not have a dedicated cell. HYBRID→TEXT false negatives (∼0.2%) are the most consequential failure: the student under-routes to HYBRID when an entity *would* have been linkable but the student’s confidence is too low. Re-routing on failure absorbs the first two classes: $\nu=0.5$ on the wrong first choice triggers a re-route within the same outer iteration, and the third class is the main residual error behind the teacher-student gap of Ta-

Table 7: Full backbone swap on HybridQA dev (learned router fixed). Top: closed-weight; bottom: open-weight (self-hosted via vLLM). $\Delta F1$ vs. the Gemini 2.5 Flash anchor.

Variant	Exact	Norm	Relax	F1	Judge	$\Delta F1$
<i>Closed-weight</i>						
Gemini 2.5 Flash (anchor)	37.02	40.83	61.59	54.02	54.02	—
Gemini 2.5 Pro	38.36	42.30	63.28	54.58	54.10	+0.56
Gemini 3.1 Pro Preview	47.87	53.11	69.51	63.92	64.92	+9.90
Claude Haiku 4.5	35.74	39.67	58.69	50.15	54.28	−3.87
Claude Opus 4.7	47.21	51.15	65.90	62.17	62.62	+8.15
GPT-4o-mini	28.20	35.08	47.87	45.03	46.23	−8.99
<i>Open-weight</i>						
Llama-3.3-70B-AWQ	39.67	44.92	56.39	54.68	55.08	+0.66
Qwen3.5-122B-A10B-GPTQ-Int4	40.33	42.62	57.05	52.76	51.15	−1.26
Llama-4-Scout-17B-16E-W4A16	38.36	43.28	53.77	51.79	53.29	−2.23

Table 8: Decomposer- and verifier-model robustness on HybridQA dev. **Bold** = best per block; $\Delta F1$ vs. the Flash anchor.

Variant	Exact	Norm	Relax	F1	Judge	$\Delta F1$
Full ATR (Gemini 2.5 Flash)	37.02	40.83	61.59	54.02	54.02	—
<i>Decomposer model swapped (verifier and executors fixed at Flash)</i>						
<i>closed-weight</i>						
Gemini 3.1 Pro Preview	41.31	43.93	61.64	55.35	55.41	+1.33
Claude Opus 4.7	38.69	41.31	59.34	52.64	53.11	−1.38
Claude Haiku 4.5	38.69	41.64	59.67	51.51	53.77	−2.51
Gemini 2.5 Pro	36.39	40.33	56.72	50.73	50.49	−3.29
<i>open-weight (self-hosted)</i>						
Llama-3.3-70B-AWQ	35.08	37.70	53.11	48.16	49.51	−5.86
Llama-4-Scout-17B-16E-W4A16	38.69	42.95	56.39	52.29	53.77	−1.73
Qwen3.5-122B-A10B-GPTQ-Int4	37.38	40.00	57.38	51.64	52.46	−2.38
<i>Verifier verdict model swapped (decomposer and executors fixed at Flash)</i>						
<i>closed-weight</i>						
Gemini 3.1 Pro Preview	42.95	45.90	62.95	56.15	57.38	+2.13
Claude Opus 4.7	38.69	41.97	59.02	52.73	53.44	−1.29
Gemini 2.5 Pro	37.70	40.98	58.03	51.60	52.13	−2.42
Claude Haiku 4.5	35.41	39.02	56.07	49.78	49.18	−4.24
<i>open-weight (self-hosted)</i>						
Llama-3.3-70B-AWQ	39.34	43.61	58.36	53.49	53.44	−0.53
Llama-4-Scout-17B-16E-W4A16	36.39	39.67	56.72	50.40	52.13	−3.62
Qwen3.5-122B-A10B-GPTQ-Int4	39.34	42.62	59.34	53.18	52.79	−0.84
<i>Verifier accuracy ceiling</i>						
Oracle verifier	40.00	43.28	65.57	56.25	55.26	+2.23

ble 1: a route the student declined but the teacher would have taken.

Oracle-context control. Table 9 reports the full-context control of §5. For each dev question we concatenate the gold table, serialised row-wise, with all linked passages into a single Gemini 2.5 Flash call at temperature 0 under the same answer-format instructions, with no retrieval and no decomposition. On HybridQA dev the median question carries 120 table cells, 40 linked passages, and 10,924 input tokens, and every question fits inside the context window, so the control applies to the whole dev set rather than a favourable subset. We report it as a diagnostic upper bound rather

than a deployable system, since it assumes gold-document access that is unavailable at inference.

Teacher-labeled versus oracle-labeled student. Table 10 gives the comparison summarised in §6.2. Both students share the architecture, training schedule, and inference path; only the label source differs. The oracle-labeled student is not merely less accurate but behaviourally different, firing escalation 9.3 pp more often, which is the signature of first choices that are locally admissible but poorly recoverable downstream.

Table 9: Oracle-context control: the gold table serialised row-wise plus every linked passage in a single Gemini 2.5 Flash call at temperature 0, with no retrieval and no decomposition.

Metric	Oracle-ctx	ATR
HybridQA EM	36.5	40.83
HybridQA F1	51.8	54.02
TAT-QA EM	37.9	38.55
TAT-QA F1	46.1	44.73
arithmetic F1	37.2	41.64
count F1	50.0	59.38

Table 10: Distilled students trained on the LLM teacher’s labels versus on *oracle* labels (HybridQA dev), everything else fixed.

Student	Agree.	F1	Relax	Escal.
Teacher-labeled	98.6%	54.02	61.59	38.3%
Oracle-labeled	91.4%	51.6	58.4	47.6%

HybridValueLinker ablation under the full system. Table 11 gives the linker ablation of §6.3, including the selective normalisation-level rows. Disabling REWRITE alone costs 1.1 pp F1 on HybridQA and 0.8 pp on TAT-QA; disabling PARTIAL as well costs 2.3 and 1.5 pp, which places roughly 85% of the linker’s total effect on the non-exact alignment classes.

Table 11: HybridValueLinker ablation under the *full* system, with backbone, retrieval, and prompts held fixed. The lower block disables normalisation levels selectively.

Variant	Δ F1 vs Full ATR	
	HybridQA	TAT-QA
<i>Linker replaced</i>		
Verbatim NL2SQL	−2.7	−1.8
<i>Normalisation levels disabled</i>		
REWRITE only	−1.1	−0.8
+ PARTIAL	−2.3	−1.5

Distilled router size. Table 12 sweeps the student architecture with the rest of the pipeline held fixed. Agreement with the teacher and downstream accuracy both saturate at the DistilBERT scale: TinyBERT-4L gives up 1.1 pp F1 for $4.5\times$ fewer parameters, while BERT-base buys 0.09 pp, well inside the ± 0.41 pp seed spread reported in Limitations, for $1.7\times$ the size and $1.8\times$ the per-sub-query latency. The routing decision is therefore cheap to compute but not cheap to omit: DistilBERT-scale capacity suffices, while forcing a single first-choice

primitive costs 8.26 pp F1 (Table 2). We keep DistilBERT as the default.

Table 12: Distilled router size sweep on HybridQA dev, rest of the pipeline fixed: TinyBERT, DistilBERT, and BERT-base (Devlin et al., 2019). *Agree.* is agreement with the LLM teacher; *ms/sq* is GPU latency per sub-query.

Student	Params	Agree.	F1	Relax	ms/sq
TinyBERT-4L	14.5M	96.1%	52.9	60.2	0.4
DistilBERT	66M	98.6%	54.02	61.59	1.0
BERT-base	110M	98.9%	54.11	61.72	1.8

Official metrics and surface-form artefacts.

We headline each benchmark’s official metric even where it costs us. On HybridQA we report the official normalized EM even though it charges ATR 20.8 pp against the substring-tolerant Relaxed, and Tables 7 and 8 report a raw exact match stricter than the official metric, which the same comparison charges 24.6 pp. The gap is dominated by two patterns in which ATR’s output preserves the gold’s meaning but fails exact matching (Table 13): *adjacent-cell bundling*, where ATR returns the gold value together with the neighbouring cell in the same row, and *noun-phrase expansion*, where it prepends a determiner to each item of a list answer. On WTQ it runs the other way, favouring the baseline on the tolerant metrics (Table 14): TableRAG-S embeds the gold span inside a longer sentence, which the official Normalized EM rejects but both Relaxed EM and the Judge rubric credit in full, in one case crediting a clause the table does not support. Substring containment is therefore no substitute for the official metric: it rewards verbosity over ATR’s mean of 1.7 tokens and credits unsupported text.

MultiHiertt and systems outside the shared protocol.

Table 15 collects the numbers that do not belong in Table 1. The top block reports MultiHiertt (Zhao et al., 2022), run with the configuration unchanged from HybridQA (router distilled on HybridQA-train, $K_{\max}=5$, $\tau=0.1$) under the shared protocol against NaiveRAG, which is by a wide margin the strongest shared-protocol baseline on the other arithmetic-heavy benchmark (TAT-QA, Table 1) and therefore the relevant comparator here. ATR leads it by 4.9 pp EM and 5.4 pp F1 on a benchmark that neither the router nor the hyperparameters ever saw. The bottom block lists HProPro’s and ODYSSEY’s published HybridQA

Table 13: Surface-form patterns where ATR’s output preserves the gold’s meaning but fails strict EM; both are credited by Relaxed and LLM-Judge.

Pattern	Mechanism	Gold	ATR output	EM	Relax/Judge
Adjacent-cell bundling	ATR returns the gold value together with the next-column cell value	“Saint Lawrence”	“Gananoque River and St. Lawrence River”	×	✓
Noun-phrase expansion	ATR prepends a determiner to each item in a list answer	“Houses of York and Lancaster”	“The House of York and the House of Lancaster”	×	✓

Table 14: Logged WTQ dev cases the official Normalized EM credits to ATR but not to TableRAG-S, though Relaxed EM and the Judge score TableRAG-S fully correct: the baseline wraps the gold span in a longer sentence.

System	Prediction	Norm	Relax	Judge
<i>Q1 (superlative lookup): “which country had the most riders that placed in the top 20 of the 1971 trans-ama final standings?”</i>				
Gold	“Sweden”			
ATR	“Sweden”	✓	✓	✓
TableRAG-S	“Based on the final standings, Sweden had the most riders finishing in the top 20 of the 1971 Trans-AMA series, with four riders placing in those positions.”	×	✓	✓
<i>Q2 (count, scalar fast path): “what was piotr’s total number of 3rd place finishes?”</i>				
Gold	“3”			
ATR	“3”	✓	✓	✓
TableRAG-S	“Piotr finished 3rd a total of 3 times in the competitions listed, and he also recorded 2 second-place finishes.”	×	✓	✓

numbers; each uses a different evaluation sample and a GPT-4-family backbone, so they index a different experimental setting rather than a stronger system under ours.

Table 15: *Top*: MultiHiertt (Zhao et al., 2022) under the shared protocol, ATR configuration unchanged. *Bottom*: HybridQA numbers as published by HProPro (Shi et al., 2024) (gpt-4-0613) and ODYSSEY (Agarwal et al., 2025) (GPT-4-1106-preview) on the dev subsets named, so not comparable to Table 1.

System	EM	F1
<i>MultiHiertt, shared protocol</i>		
NaiveRAG	28.9	34.2
ATR (learned router)	33.8	39.6
<i>HybridQA, as published</i>		
HProPro, 200-sample dev, GPT-4	48.0	54.6
test set, GPT-4	48.7	57.7
ODYSSEY, 500-sample dev, GPT-4	58.4	71.8
200-sample, GPT-4	51.5	66.0

HProPro under given and retrieved evidence.

Table 16 separates the two conditions on all three benchmarks. HProPro’s released pipeline is built around a per-question table id, so *full table in prompt* is the setting it assumes; *retrieved* substitutes the top-1 table from the shared BGE-M3 index, whose accuracy is 11.8% on HybridQA,

17.6% on TAT-QA and 37.5% on WTQ. The substitution costs it 42.3, 41.1 and 46.7 pp EM. The two given-side rows are not a head-to-head: HProPro reads the question’s entire table under no budget, while ATR *gold table id* still answers from the cells, rows, and passages its retrieval surfaces under the shared budget, so the gap in that block prices the evidence privilege, not the reasoning. The evidence-matched comparison is the retrieved pair, where neither system is told which table holds the answer: ATR leads by 13.1 pp EM on HybridQA and 1.3 on TAT-QA, and the two are level on WTQ (−0.6), the one benchmark whose linked passages are not central. The advantage is therefore in recovering from a missed table rather than in reading a correct one.

Oracle-context size distribution. Table 17 gives the per-question context the oracle-context control consumes, measured over the HybridQA release; our preprocessed copy of the same tables and passages agrees to within one passage at the median. Two properties of the release shape the distribution: tables are capped at 20 rows, so cell counts concentrate at 120, and passage counts vary widely because they follow the hyperlink density of the table. Every question fits the context window under either token convention, so the control is applicable

Table 16: HProPro on all three benchmarks under both evidence conditions, with ATR in both of its own. The evidence-matched pair is HProPro *retrieved table* against ATR *no table id*. Backbone and scorers as in Table 1.

Dataset	System / evidence	EM	Relax	F1	Judge
HybridQA	HProPro, full table in prompt	48.85	64.92	58.93	66.89
	HProPro, retrieved table	6.56	9.51	8.36	32.18
	ATR, gold table id	40.83	61.59	54.02	54.02
	ATR, no table id	19.67	32.79	28.22	28.85
TAT-QA	HProPro, full table in prompt	50.90	76.02	67.25	78.00
	HProPro, retrieved table	9.83	15.23	15.58	17.97
	ATR, gold table id	38.55	68.32	44.73	71.00
	ATR, no table id	11.09	24.70	19.37	24.76
WTQ	HProPro, full table in prompt	76.30	81.90	80.15	84.10
	HProPro, retrieved table	29.60	32.80	31.93	40.30
	ATR, gold table id	57.60	64.00	62.81	63.29
	ATR, no table id	29.00	32.40	31.20	31.60

on the whole dev set rather than a favourable subset.

Quantity	Min	Med	Mean	P90	P99	Max
Table cells	120	120	120.1	120	120	140
Linked passages	11	40	46.3	72	116	158
Context words	1922	7880	8273.2	13358	23594	33125
Subword tokens	2627	10924	11578.2	19378	31990	47038

Table 17: Per-question context size for the oracle-context control on HybridQA dev ($N=305$): the gold table plus every linked passage.

SPARTA: deeper multi-hop, and a three-way comparison. SPARTA (Park et al., 2026) poses tree-structured questions of up to four hops over three domains whose corpora are far too large to place in a prompt (tables reach 14,418 rows), which makes it the one benchmark here whose evidence must be assembled rather than given. We run ATR’s unchanged configuration on all three dev domains (565 questions each; 10 questions whose gold answer list is empty admit no F1 and are skipped by the benchmark’s scorer for every system alike, leaving 1,685) in two tracks (with and without the gold table identity), and re-run HProPro and ODYSSEY, the two systems the benchmark itself adapts, from their released code under our backbone. Because SPARTA’s gold answers are SQL result sets, we score set-based F1 over answer lists with item normalisation from the benchmark’s own metric code. Table 18 reports every condition; *retrieval* is the system’s own retrieval over the corpus index, *ATR’s retrieval* the tables and passages ATR surfaced, and *gold* the evidence a system is handed: the gold table identity for ATR (which still retrieves under the shared budget), the

gold SQL’s tables, serialised in full into the prompt, for HProPro, those tables with the full passage store and cell-passage link graph for ODYSSEY, traversed under no retrieval budget. Three observations. First, under matched evidence, with every system answering from the tables and passages ATR’s retrieval surfaced, ATR and HProPro are statistically indistinguishable (paired bootstrap over the pooled 1,685 questions: $\Delta=-0.68$, 95% CI $[-3.01, +1.62]$, $p=0.57$). The gold-evidence rows above them therefore measure evidence access rather than reasoning: moving off gold evidence costs HProPro 9.7 and 9.8 F1 on medical and movie against 2.8 and -0.6 for ATR, whose gold track is only the table identity, so HProPro’s standing rests on an oracle about four times more than ATR’s. Second, ODYSSEY’s advantage is confined to medical, whose corpus is small enough (a 60-entity traversal graph, 60 passages) that retrieval is not a problem; it does not survive to either larger domain, where it places last. Third, every system, including the resource-privileged references, drops steeply from single-hop to nested questions; the deficit at depth is common to retrieval-grounded and resource-privileged systems alike, which locates it in multi-hop reasoning rather than in evidence access and motivates the explicit subgoal planning named in Limitations.

Table 18: SPARTA dev, set-based F1 over answer lists (565/561/559 scored questions for medical/nba/movie). *Evidence* names what each condition is given.

System	Evidence	medical	nba	movie
ATR (learned router)	retrieval	34.42	32.43	30.14
ATR (learned router)	gold table id	37.23	35.09	29.57
HProPro	ATR’s retrieval	36.18	29.26	33.60
HProPro	full tables in prompt	45.85	32.15	43.36
ODYSSEY	ATR’s retrieval	44.96	21.03	10.61
ODYSSEY	gold + links	47.63	21.01	19.60

Cross-dataset transfer. The DistilBERT student is trained only on HybridQA-train and deployed zero-shot on TAT-QA and WTQ. The transfer holds because the four operator $\times$ modality $\times$ entity-mention slots are dataset-agnostic. TAT-QA arithmetic questions surface as `expected_operator` $\in$ {count, compare, aggregate, arithmetic} with `required_modalities` = table; the student dispatches these to SQL/HYBRID without having seen TAT-QA training data. WTQ filter/sort questions surface as `filter/sort` with `required_modalities` = table, dis-

patched to RETRIEVE/HYBRID depending on entity-mention presence. The Decomposer’s emit-time decision to fill these fields is what makes the cross-dataset jump possible; an operator-blind question encoder would have to learn the routing boundary separately on each benchmark.

Per-call inference load and dollar cost. Table 19 reports the LLM-call profile for one HybridQA dev run across every backbone in Table 7 for which we logged HTTP call counts. The Full ATR Gemini 2.5 Flash configuration issues 4,505 total LLM calls (14.77 per question on average, counting decomposition, routing, execution, verification, and synthesis) for a total of 11.26M input and 0.90M output tokens. Backbone choice changes the call budget by less than $1.3\times$ (min 3,442 for Gemini 3.1 Pro Preview, max 4,505 for Gemini 2.5 Flash), and the per-call prompt size is dominated by retrieval context rather than backbone-specific tokenisation.

The Cost column of Table 19 applies each provider’s listed per-token rates (May 2026, tabulated alongside) to the estimated token counts and reports the USD spend per HybridQA dev pass. The Flash headline costs \$5.63, GPT-4o-mini \$1.80, while Claude Opus 4.7 reaches \$62.80, roughly $11\times$ the Flash baseline, driven by Opus’s per-token rates across ATR’s 5-iteration decomposition. Self-hosted open-weight backbones (Llama-3.3, Qwen3.5, Llama-4 Scout) carry no per-call API charge. The sub-quadratic growth in calls with $K_{\max}$ is what makes a DistilBERT student attractive: a per-sub-query LLM router would add one extra LLM call per iteration, inflating the call count by $\approx K_{\max}$, while the distilled student adds zero.

Cost vs. accuracy interplay. Placing the per-run dollar cost alongside the headline F1 from Table 7 reveals two distinct cost-accuracy regimes. In the proprietary tier, frontier closed models trade a roughly 3 to $11\times$ cost increase for +8 to +10 pp F1 over the Flash anchor: Gemini 3.1 Pro Preview at $4.5\times$ cost buys +9.9 pp, Claude Opus 4.7 at $11\times$ cost buys +8.2 pp. The Opus premium is particularly steep because its per-token rates sit far above Flash’s (tabulated alongside), compounded across the iteration budget. In the open-weight tier, Llama-3.3-70B-AWQ delivers +0.66 pp F1 over Flash at \$0 marginal API cost, which positions the self-hosted variant as a strong default for cost-sensitive deployments. Qwen3.5 sits within -1.3 pp F1 at the same zero-API price and adds $2\times$

longer context support (262K tokens against Llama-3.3’s 128K), useful when the routing pipeline must carry long retrieval contexts through multiple sub-queries.

Practical guidance for backbone selection. The four (cost, accuracy) quadrants suggest a default policy. For tight latency and the simplest deployment, Gemini 2.5 Flash sits at the cost-accuracy Pareto frontier among proprietary backbones. For maximum accuracy regardless of cost, Gemini 3.1 Pro Preview’s +9.9 pp F1 jump is the largest single-knob return in our matrix. For privacy- or sovereignty-sensitive deployments, Llama-3.3-70B-AWQ on $2\times$ A6000 matches the Flash headline at zero per-call dollar cost while keeping inputs and outputs on-premise. Claude Opus 4.7 is the only backbone where the cost premium is not justified by the accuracy delta on this workload; it is worth using when its 1M context window or its non-Google provider story matters, but not as a default for HybridQA-style tasks.

D End-to-End Walkthrough

Figure 9 traces ATR on the HybridQA two-hop question “Which conflict caused the second place team of Campeonato Paulista in 1908 to not mention its German roots?”, whose gold answer is *World War II*. The question requires (i) locating the runner-up of one season in a Wikipedia results table and (ii) joining that club to a historical attribute that lives only in a cell-linked passage, a canonical multi-hop, cross-modal trajectory that breaks both pure NL2SQL and pure retrieval pipelines.

Iteration 1: cell-grounded SQL. The decomposer emits the sub-query q_1 = “What was the name of the second place team in Campeonato Paulista in 1908?” with operator *filter*, modality *table*, and entity mentions $E_1 = \{1908, \text{second place}\}$. The router selects HYBRID because q_1 asks for a row filter over a structured table and no header is named “second place”. The HYBRIDVALUELINKER rewrites “second place” to ‘Runner-up’ (a REWRITE at confidence 1.00) and aligns “1908” to the season cell ‘1908 LPF’. The rewrite is what resolves which column answers the sub-query: the two strings share no substring, so surface similarity alone never connects them. CONSTRAINED-SQL then issues `SELECT ‘Runner-up’ FROM Campeonato_Paulista_1 WHERE`

Table 19: Per-call inference load and per-run USD cost on HybridQA dev, at each provider’s May 2026 list rate applied to 2,500 input and 200 output tokens per call. **Bold** = Flash anchor; open-weight rows are self-hosted.

Type	Model	Ctx	Cutoff	Checkpoint / API ID	Calls	C/Q	\$/M in/out	Cost (\$)	×Flash	ΔF1
Proprietary	Gemini 2.5 Flash	1M	Jan 2025	gemini-2.5-flash	4,505	14.77	0.30/2.50	5.63	1.0	—
	Gemini 2.5 Pro	1M	Jan 2025	gemini-2.5-pro	3,818	12.52	1.25/10.00	19.54	3.5	+0.56
	Gemini 3.1 Pro Preview	1M	Jan 2025	gemini-3.1-pro-preview	3,442	11.29	2.00/12.00	25.50	4.5	+9.90
	GPT-4o-mini	128K	Oct 2023	gpt-4o-mini-2024-07-18	3,632	11.91	0.15/0.60	1.80	0.3	−8.99
	Claude Haiku 4.5	200K	Feb 2025	claude-haiku-4-5	4,438	14.55	1.00/5.00	15.55	2.8	−3.87
	Claude Opus 4.7	1M	Jan 2026	claude-opus-4-7	3,582	11.74	5.00/25.00	62.80	11.2	+8.15
Open-weight	Llama-3.3-70B-AWQ	128K	Dec 2023	llama-3.3-70b-instruct-awq	3,736	12.25	—	0.00	0	+0.66
	Qwen3.5-122B-A10B-GPTQ-Int4	262K	—	Qwen3.5-122B-A10B-GPTQ-Int4	3,687	12.09	—	0.00	0	−1.26
	Llama-4-Scout-17B-16E-W4A16	10M	Aug 2024	Llama-4-Scout-17B-w4a16	4,483	14.70	—	0.00	0	−2.23

Season = ‘1908 LPF’, restricted to columns retrieved from View 3 and seeded with the linked value, and returns *Germânia*. The verifier accepts with $\nu_1=1$.

Iterations 2 to 4: escalation and re-decomposition. The decomposer emits q_2 =“Which conflict caused *Germânia* to not mention its German roots?” with operator lookup, modality text, and $E_2=\{Germânia\}$. The router selects TEXT, but the verifier rejects the result at $\nu_2=0.5$, and the router is re-invoked with the failed-route history and dispatches primitives it has not tried, four executed attempts in all without an accepted answer, so sub-query 2 is recorded as failed rather than answered wrongly. The loop does not stop there: conditioned on that failure the decomposer re-poses the sub-query against the club’s linked article rather than against the conflict, TEXT accepts at $t=3$ and $t=4$, and the retrieved passage states that “during the course of World War II the club abandoned references to its German origin and in 1941 was renamed to *Pinheiros*.” With residual uncertainty $u_4<\tau$, the stop controller exits at $t=4$ of $K_{\max}=5$ and Final Synthesis returns *World War II*, matching the gold answer exactly.

This trace illustrates the operational signature of sub-query routing: a single question mixes HYBRID (cell-grounded SQL) and TEXT (passage retrieval), and a rejected sub-answer costs attempts inside one sub-query rather than corrupting the final response.

Failure modes of single-primitive baselines.

The same question highlights why a fixed-primitive baseline loses on either hop. A pure NL2SQL system has no column named *second place*, emits the phrase verbatim, and returns an empty row set for hop 1; even given the row it has no column holding the *conflict*, so hop 2 has no route at all. A pure retrieval system (chunks + reranker) sur-

faces the *Pinheiros* passage, which names both the German origin and the war, but cannot compute “second place in 1908” against the structured table, so it anchors on the wrong club or on none. Single-primitive systems pay this as a hard failure, whereas ATR pays it as attempts inside a bounded loop.

Connection to value-linker statistics.

This walkthrough’s REWRITE (*second place* → ‘Runner-up’) and PARTIAL (*1908* → ‘1908 LPF’) alignments are the two majority classes on HybridQA dev (20.9% + 34.8%; §6.3), the 56% of first-ranked cells needing an LLM-verified rewrite before the executor accepts the row. The question is also one where that machinery is load-bearing rather than incidental: under the *no linker* and the REWRITE-disabled ablations of Table 11, ATR returns *not found* instead of *World War II*.

Algorithm 1 ATR online inference (§3.1): decompose, route, verify; re-route on $\nu_t < 1$.

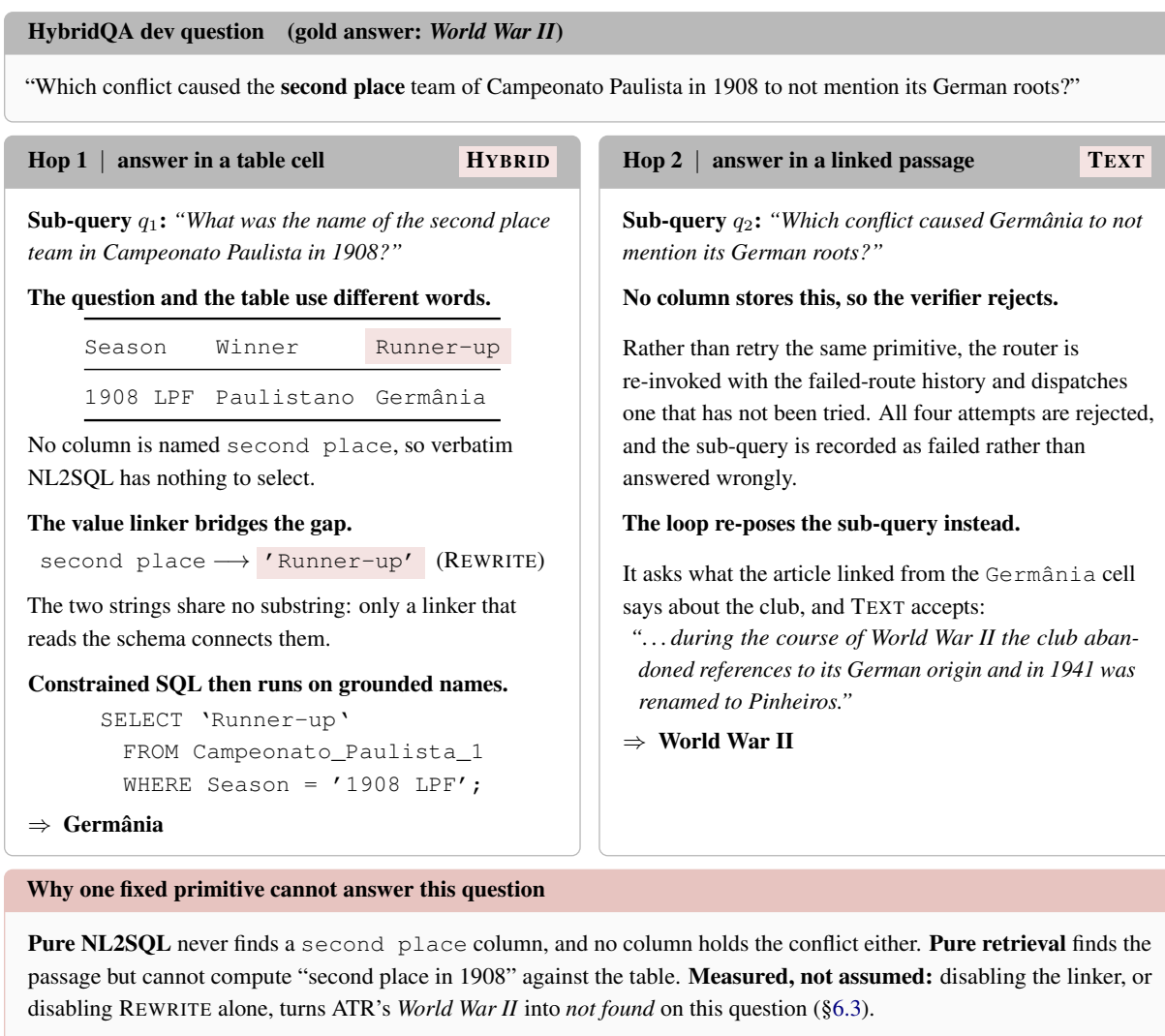
Require: Question q ; multiview indices (Views 1 to 5; View 4 hosts both a cell index and a row index, §3.2); max iterations $K_{\max}$; max escalations per sub-query M ; verifier threshold τ

Ensure: Final answer a

```

1:  $z_{<1} \leftarrow \emptyset$ ;  $H \leftarrow \emptyset$  ▷ sub-query / answer history; persistent failed-route history
2: for  $t = 1$  to  $K_{\max}$  do
3:    $(q_t, meta_t, E_t, u_t) \leftarrow \text{DECOMPOSE}(q, z_{<t})$ 
4:   if  $q_t = \text{TERMINATE}$  then
5:     break
6:   end if
7:    $chunks \leftarrow \text{DOCUMENTRETRIEVAL}(q_t)$ 
8:    $schema \leftarrow \text{RESTORESCHEMAVIAMAPPING}(chunks)$ 
9:   for  $k = 0$  to  $M$  do
10:     $r \leftarrow \text{ROUTER}(q_t, meta_t, schema, H)$  ▷ excludes  $H$ ; TEXT if  $H=\mathcal{R}$ 
11:    if  $r = \text{TEXT}$  then
12:       $y_t \leftarrow \text{ANSWERFROMDOCCHUNKS}(q_t, chunks)$ 
13:    else if  $r = \text{RETRIEVE}$  then
14:       $(C, V_{\text{raw}}) \leftarrow \text{SCHEMACELLRETRIEVAL}(q_t)$  ▷ Views 3+4
15:       $V^* \leftarrow \text{HYBRIDVALUELINKER}(E_t, C, V_{\text{raw}}, H)$ 
16:       $y_t \leftarrow \text{SYNTHESIZEFROMRETRIEVAL}(q_t, C, V^*)$ 
17:    else if  $r = \text{SQL}$  then
18:       $C \leftarrow \text{SCHEMARETRIEVAL}(q_t)$  ▷ View 3
19:       $y_t \leftarrow \text{EXECUTENL2SQL}(q_t, C)$  ▷ View 5
20:    else if  $r = \text{HYBRID}$  then
21:       $(C, V_{\text{raw}}) \leftarrow \text{SCHEMACELLRETRIEVAL}(q_t)$ 
22:       $V^* \leftarrow \text{HYBRIDVALUELINKER}(E_t, C, V_{\text{raw}}, H)$ 
23:       $y_t \leftarrow \text{CONSTRAINEDSQL}(q_t, C, V^*, chunks)$ 
24:    end if
25:     $\nu_t \leftarrow \text{VERIFIER}(q_t, y_t)$ 
26:    if  $\nu_t \geq 1$  then
27:      break
28:    end if
29:     $H \leftarrow H \cup \{(r, \nu_t)\}$  ▷ record failed route; re-route on next attempt
30:  end for
31:   $z_{\leq t} \leftarrow z_{<t} \cup \{(q_t, y_t)\}$ 
32:  if  $\nu_t = 1$  and  $u_t < \tau$  then
33:    break ▷ stop controller
34:  end if
35: end for
36: if  $\text{FINALSYNTHESIS}$  enabled and  $|z_{\leq t}| \geq 2$  then
37:    $a \leftarrow \text{FINALSYNTHESIS}(q, z_{\leq t})$  ▷ multi-hop only; single-hop skips synthesis
38: else
39:    $a \leftarrow y_{\text{last verifier-accepted}}$ 
40: end if
41: return  $a$ 

```



Prompt 1: Decomposer

Decompose the question into atomic sub-queries, each answerable by a single retrieval step.

Input: {question}, {history}.

If history suffices $\rightarrow$ sub_query=TERMINATE.
Otherwise emit one new sub-query.

Output JSON:

```
sub_query (string),
required_modalities  $\in$  {text, table, both},
expected_operator  $\in$  {lookup, filter, count,
compare, aggregate, sort, arithmetic},
need_global_table_view (bool),
entity_mentions (set of spans),
uncertainty  $\in$  [0,1].
```

Strategy hints:

BRIDGE: find X , then look up Y .
MULTI-HOP LOOKUPS: chain across rows / passages.
COMPARATIVES: each comparand separately.
NEVER repeat an already-answered sub-query.

(a) Decomposer (§3.3): emits the next sub-query q_t as a structured SUBQUERY object.

Prompt 3: Verifier

Judge whether the candidate answer is faithfully supported by the evidence.

Input: question, candidate answer, retrieval evidence, SQL result (if any).

Three verification dimensions:

D1 slot alignment: answer type matches the question slot (name / number / date / count).
D2 evidence-answer consistency: every claim in the answer is derivable from the retrieval evidence or the SQL result.
D3 route faithfulness: every entity name and cell value in the answer appears verbatim in some evidence source (no hallucinated spans).

Verdict:

$\nu=1$: all three pass.
 $\nu=0.5$: partial support, or one minor failure (also when SQL returns empty but evidence is consistent).
 $\nu=0$: answer contradicts evidence or names an unseen entity.

(c) Verifier (§3.6): returns $\nu_t \in \{0, 0.5, 1\}$ for the escalation chain.

Prompt 2: LLM Router (teacher)

Select the single best execution route for the sub-query.

Input features:

```
sub_query,
expected_operator,
required_modalities,
entity_mentions,
schema_available (bool),
already_failed_routes (set).
```

Routes: TEXT / RETRIEVE / SQL / HYBRID.

Selection rules:

R1: modality=text, or no schema and no entities $\rightarrow$ TEXT.
R2: count/aggregate/sort/arithmetic/compare + schema + no entities $\rightarrow$ SQL.
R3: count/aggregate/sort/arithmetic/compare + entities $\rightarrow$ HYBRID.
R4: lookup/filter + entities $\rightarrow$ RETRIEVE.
R5: schema available $\rightarrow$ SQL, otherwise $\rightarrow$ TEXT.
R6: any route in already_failed_routes is excluded; apply the next rule that matches, or TEXT if none does.

Output one token from {TEXT, RETRIEVE, SQL, HYBRID}.

(b) LLM Router (§3.4): teacher for the DistilBERT student distillation.

Prompt 4: LLM Judge (evaluation only)

Rate whether the predicted answer is suitable for the question against the gold answer.

Criteria:

Accuracy / Hallucinations: the prediction must be semantically consistent with the gold; numerical values and ordering must match; no fabricated entities or facts.
Completeness: the prediction must contain all key points required by the question; further elaboration may be omitted.

Overall score on a {0,1} scale (0=wrong, 1=correct). Output a single line.

Format: Rating: [[score]].

Input fields: {question}, {golden}, {gen}.

(d) LLM Judge: evaluation-only prompt for the supplementary *Judge* metric in Tables 1 and 2; retained verbatim from Yu et al. (2025) so the metric is strictly comparable.

Figure 10: ATR prompt templates (abridged). Prompts 1–3 drive the inference loop; Prompt 4 (LLM Judge) is used only for the supplementary *Judge* metric.