STRATTR: ONE STRATEGY DOES NOT FIT ALL IN TEMPORAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal reasoning involves a range of different tasks, e.g. determining the order of events, reasoning about the duration or overlap of events, reconstructing timelines. These correspond to distinct operations including event-graph reasoning for relation questions, date arithmetic for duration questions, and even direct information extraction. Existing systems, however, typically use a single prompting scaffold or policy across these question types, causing a mismatch between the reasoning process and the operation need for a given temporal question. We introduce **StratTR**, a trainable multistrategy agentic framework that supports four different strategies (graph construction, code generation, direct extraction, and hybrid reasoning). StratTR distills routed, multi-call trajectories from a proprietary teacher into a single-call 4B policy. To train the strategy-conditioned policy, StratTR leverages key deterministic temporal signals across heterogeneous artifacts with reinforcement learning. Across seven benchmarks comprising 22,728 questions, StratTR with a 4B-scale backbone outperforms the strongest same-backbone baseline with macro-averaged exact match gains of 16.23%, 11.46% over the top-performing API baseline and 3.04% over its teacher model with a reduction of external model calls from 6.0 to 1.0. Further analysis suggests that learned verifiers and LLM judges can support candidate selection when deterministic checks are unavailable, although their effectiveness varies across datasets.

1 INTRODUCTION

Temporal reasoning in natural language aims at answering questions that require reasoning about when events happen, how long they last, and how they relate to one another in time. While such questions are all temporally motivated they require different operations. Consider three examples: *"Did event A happen before event B?"*, *"How many days elapsed between X and Y?"*, and *"When was Z elected president of the US?"* The first requires reasoning on event relations, the second date arithmetic, and the third a direct lookup in the context.

Related work takes two broad approaches. Early efforts focus on strengthening a shared prompting scaffold for temporal questions, for instance through timeline construction and iterative self-reflection (Bazaga et al., 2025). However, such single-scaffold methods apply one undifferentiated reasoning procedure to every question, regardless of the reasoning operation the question actually requires. A timeline does not directly perform exact arithmetic, direct answering may omit necessary relational structure, and graph construction can add unnecessary computation to fact lookup. Recent systems optimize tailored temporal policy via reinforcement learning (RL) with outcome rewards (Liu et al., 2025; Gong et al., 2025). However, such methods optimize a monolithic strategy over an heterogeneous question distribution, so that the learned policy is dominated by the most frequent question types.

In both approaches, the reasoning strategy remains shared across temporal question types, leading to a *strategy mismatch* issue: the selected procedure does not match the operation needed for resolving the question. We analyze this strategy mismatch problem with a factorial experiment on a fixed sample of questions, while evaluating four strategies on each of nine temporal subtypes. As shown in Table 1, the highest-scoring strategy varies across temporal subtypes and no single strategy performs uniformly best: graph construction performs best on relation classification, whereas code generation performs best on duration computation. These results motivate us to develop a system that routes each question according to its temporal subtype.

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

Table 1: Factorial experiment: exact-match accuracy (%) for each strategy applied to every subtype (90 questions each). Bold marks the highest observed score in each row.

Subtype	Graph	Code	Direct	Hybrid
Co-temporality / Overlap	67.8	71.1	**73.3**	71.1
Counterfactual Reasoning	**21.1**	5.6	5.6	3.3
Duration Computation	47.8	**80.0**	72.2	73.3
Explicit / Relative Reasoning	3.3	8.9	**10.0**	7.8
Fact Extraction	33.3	95.6	**100.0**	92.2
Point-in-Time Lookup	38.9	81.1	**84.4**	80.0
Relation Classification	**42.2**	20.0	21.1	6.7
Temporal Localization	5.6	24.4	**28.9**	25.6
Temporal Ordering	52.2	**80.0**	78.9	80.0

Here we introduce **StratTR**, a framework that adapts the reasoning procedure to the temporal question type. Our key insight is that different temporal question types present different bottlenecks and therefore require different capabilities, including information retrieval (Chu et al., 2023), structural event reasoning (Fatemi et al., 2025), and program execution (Su et al., 2024). The modular design of agentic systems (Li et al., 2026b) provides a natural way to meet these requirements through strategies that invoke specialized tools and operations. Accordingly, StratTR selects among four strategies (graph construction, code generation, direct extraction, and hybrid reasoning) across nine temporal subtypes:. A proprietary teacher model executes the selected strategy over multiple calls, and supervised fine-tuning distills the resulting trajectories into a single-call open-weight policy. A final reinforcement-learning stage uses strategy-specific verification signals derived from the corresponding reasoning artifacts, including consistency checks for generated event graphs and execution outcomes for generated programs.

We extensively evaluate StratTR across seven benchmarks comprising 22,728 questions. It achieves 74.92% macro-averaged exact match, compared with 64.46% for TISER on the same backbone and 67.22% for TISER with the API-based model used as the teacher. Component comparisons show that the routed teacher achieves 72.71%. Distillation reaches 74.12% while reducing the mean number of external model calls per question from 6.0 to 1.0, and the final reinforcement-learning stage obtains 74.92%. Exploratory reranking experiments further examine an LLM judge and a trained process reward model for settings in which deterministic checks are unavailable. These alternatives improve candidate selection on some datasets, and their effects vary across settings, revealing trade-offs among verifier choices.

Thus our work makes three key contributions: (1) We provide a systematic factorial analysis of strategy mismatch: different temporal subtypes favour different evaluated procedures, and no strategy applied uniformly matches the question-level oracle over the four observed outputs (Section 1). (2) We propose StratTR, which represents four procedures through strategy-specific artifacts and distills routed, multi-call teacher episodes into a single-call policy (Section 3). (3) Through comprehensive experiments we show that StratTR with a 4B backbone outperforms the strongest same-backbone baseline and even larger proprietary models. Further analyses reveal that learned verifiers and LLM judges can support candidate selection when deterministic checks are unavailable, but their effectiveness varies across datasets.(Sections 5–8).

2 Related Work

Temporal representations and approaches. Temporal systems vary in how they represent and address temporal reasoning: TISER constructs timelines with self-reflection (Bazaga et al., 2025), TG-LLM translates text into a temporal graph (Xiong et al., 2024), and TempAgent couples a ReAct-style agent with temporal constraints (Hu et al., 2025). Time-R1 and Temp-R1 train temporal-reasoning policies with verifiable rewards (Liu et al., 2025; Gong et al., 2025), while Timo transfers mathematical reasoning ability (Su et al., 2024). Evaluation is fragmented across benchmarks probing distinct abilities, including TempReason (Tan et al., 2023), TimeQA (Chen et al., 2021), TGQA (Xiong et al., 2024), TIME-Lite (Wei et al., 2025), and NarrativeReason (Song et al., 2025) (event-event relations). Within each method, however, the same procedure is applied across question types, so prior evaluations do not directly compare procedure utility across temporal operations. StratTR makes this comparison explicit, allows the procedure to vary, and distills the resulting multi-call executions.

Table 2: Unified subtype taxonomy and strategy statistics.

Unified subtype	Strategy	Test n
Co-temporality / Overlap	direct_extraction	665
Counterfactual Reasoning	graph_construction	150
Duration Computation	code_generation	993
Explicit / Relative Reasoning	graph_construction	300
Fact Extraction	direct_extraction	3,057
Point-in-Time Lookup	direct_extraction	8,778
Relation Classification	graph_construction	1,965
Temporal Localization	hybrid	150
Temporal Ordering	hybrid	6,670
Total		22,728

Strategy selection and test-time compute. Snell et al. (2025) show that the optimal test-time strategy is prompt-dependent, motivating adaptive allocation of computation. Agentic RL frameworks train policies that interleave reasoning with tool calls (Zhang et al., 2025; Feng et al., 2025), and recent in-flow systems train the planning module itself within a modular planner–executor–verifier loop (Li et al., 2026a). Learned process reward models provide dense supervision from human or search-derived labels (Lightman et al., 2024; Setlur et al., 2025); reinforcement learning with verifiable rewards (RLVR) avoids a learned reward model, but prior work finds that it may improve consistency without increasing solution coverage (Yue et al., 2025; DeepSeek-AI, 2025). StratTR studies a different axis: selecting among temporal procedures whose heterogeneous reasoning artifacts expose different verification signals.

3 The StratTR Framework

StratTR addresses strategy mismatch through four components: strategy routing, strategy-specific execution, trajectory distillation, and policy optimization with strategy-conditioned temporal signals. We describe these components in pipeline order and then specify the reward used in our implementation. Appendix A provides the role templates used during strategy execution.

3.1 Subtype Taxonomy and Strategy Routing

We unified evaluation schemata so that each question is associated with one of nine subtypes defined by the operation required. Each generated trajectory begins with a strategy tag that selects one of four procedures. *Graph construction* builds an event graph with Allen-style relations; *code generation* emits an executable date-arithmetic program; *direct extraction* returns a localized fact; and *hybrid reasoning* combines extraction with timeline, date, or overlap reasoning. Table 2 reports the strategy statistics.

3.2 Strategy Execution and Distillation

The teacher model executes the selected strategy through a sequence of external model calls. Let r denote the route specified by the initial strategy tag and $x_t = (q, c, r, z_t, h_t)$ the state at step t, where q is the question, c is the context, z_t contains the temporal artifacts accumulated so far (event nodes, relations, code, or extracted spans), and h_t records the preceding steps. For the teacher model, each step is a separate policy invocation that reads x_t and appends new artifacts to z_t. A teacher episode terminates when the trace indicates that sufficient evidence has been collected, a step produces no new temporal artifact, or the episode reaches the step budget K_{teacher}. Graph-building strategies repeatedly invoke planner, extractor, linker, and evaluator roles. Code generation uses a sandboxed Python interpreter restricted to date arithmetic, whereas direct extraction completes in a single external model call. We train the student model using supervised fine-tuning to distill the teacher's procedures via reasoning trajectories. The distilled student model generates the strategy tag, reasoning steps, and artifacts as a single serialized trace within one external model call.

3.3 Strategy-Conditioned Rewards and Policy Optimization

StratTR uses the heterogeneous artifacts produced by different strategies to define route-conditioned rewards. For a trajectory τ generated under route r, let $\mathcal{K}_r$ denote the reward terms applicable to that route. We define

$$R_r(\tau) = \frac{1}{Z_r} \sum_{k \in \mathcal{K}_r} \lambda_{r,k} R_k(\tau), \tag{1}$$

where $\lambda_{r,k}$ is the weight of reward term k and Z_r is a fixed route-specific normalization constant. In our implementation, $\mathcal{K}_r$ is a subset of $\{\mathrm{ans}, \mathrm{exec}, \mathrm{graph}, \mathrm{eff}, \mathrm{attempt}, \mathrm{date}\}$. The selected subset depends on the artifacts produced by route r.

Answer-outcome reward (R_{ans}). For graph construction, hybrid reasoning, and direct extraction, R_{ans} is the mean of exact match and token-level F1 against the reference answer. Direct-extraction trajectories receive no artifact-level process reward because they contain no intermediate representation that can be verified independently of the final answer.

Program-execution reward (R_{exec}). For code generation, R_{exec} assigns progressively higher rewards when the generated program parses, executes successfully, and produces the reference answer. Because this term evaluates both execution validity and output correctness, it replaces R_{ans} for code-generation trajectories.

Graph-consistency reward (R_{graph}). For graph construction, R_{graph} checks the generated event graph for self-loops and violations of symmetry and transitivity over a constrained seven-relation vocabulary based on Allen's interval algebra (Allen, 1983). The reward decreases with the number of consistency violations and also penalizes degenerate graphs.

Trace-efficiency reward (R_{eff}). For trajectories with a correct final answer, R_{eff} rewards shorter serialized reasoning traces. Trace length counts generated reasoning steps rather than external model invocations, which remain fixed at one for the student. Appendix defines the remaining answer-attempt and date-precision terms and gives the complete route-specific objectives and weights.

Group-relative policy optimization. Starting from the supervised policy π_{SFT}, we optimize the route-conditioned reward with GRPO (Shao et al., 2024). For each input x, the behavior policy samples G trajectories $\{\tau_i\}_{i=1}^{G}$, where each trajectory emits a route r_i and receives reward $R_i = R_{r_i}(\tau_i)$. We compute the group-relative advantage

$$\widehat{A}_i = \frac{R_i - \mathrm{mean}(\{R_j\}_{j=1}^{G})}{\mathrm{std}(\{R_j\}_{j=1}^{G}) + \delta}, \tag{2}$$

and optimize the standard clipped GRPO objective:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t}\min\Big(\rho_{i,t}\widehat{A}_i,\ \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i\Big)\right], \tag{3}$$

where $\rho_{i,t}$ is the token-level likelihood ratio between the current and behavior policies. All tokens in a trajectory share its advantage, jointly optimizing the route tag, reasoning trace, and final answer. Detailed hyperparameters are reported in Appendix.

Applicability of artifact-level checks. In the production GRPO run, every active reward term is computed by a deterministic scoring function. We include an artifact-level reward only when the strategy emits an intermediate representation that is retained in the trajectory, can be parsed reliably, and admits a checker that covers all valid representations. Otherwise, the route uses its answer-outcome and other applicable shaping rewards without an artifact-level term. The same optimization interface could incorporate learned or model-based verification signals.

4 Experimental Setup

Benchmarks. We evaluate on seven temporal-reasoning benchmarks: NarrativeReason (Song et al., 2025), TIME-Lite (Wei et al., 2025), TGQA (Xiong et al., 2024), TimeQA-easy and TimeQA-hard (Chen et al., 2021), and TempReason L2 and L3 (Tan et al., 2023). We convert all benchmarks to a common free-form question-answering schema, producing 22,728 test questions grouped into the nine subtypes in Table 2. Teacher trajectories are generated only from published training splits, and neither trajectory filtering nor checkpoint selection uses test data.

Evaluation protocol. We score the raw outputs of all systems with a common scoring script. Exact match (EM) is computed over extracted labels for relation classification and over normalized

Table 3: Main results across seven benchmarks comprising 22,728 questions, scored from raw outputs with a common scoring script (EM and F1 in %). TimeQA and TempReason are combined into displayed benchmark groups, whereas Macro EM averages the seven original benchmark scores. Baselines use direct prompting or TISER prompting (Bazaga et al., 2025) with open-weight models from 1B to 32B and a proprietary model. StratTR is the single-call 4B policy trained with the combined deterministic objective. The best result in each column is in bold and the second-best is underlined. For benchmark-group columns only, † marks an unadjusted two-sided McNemar test with $p < 0.05$ against the second-best system on the column's pooled items.

Method	NR	TIME-Lite	TGQA	TimeQA	TempReason	Macro EM	Macro F1
Direct prompting							
Qwen3-4B-Instruct-2507	<u>39.44</u>	8.91	35.65	65.66	73.45	51.75	70.81
TISER prompting (Bazaga et al., 2025), open-weight backbones							
Llama-3.2-1B-Instruct	0.00	0.00	0.03	0.74	0.24	0.29	0.46
Llama-3.2-3B-Instruct	1.12	4.39	12.55	51.52	44.97	30.15	34.12
Gemma-3-4B-it	7.53	5.62	16.47	43.64	26.82	24.36	30.84
Qwen3-4B-Instruct-2507	19.90	12.91	58.23	87.21	**92.88**	64.46	73.03
OLMo-3-7B-Instruct	2.29	2.78	25.78	27.75	17.21	17.25	23.60
API, Gemini-3.1-flash-lite							
TISER	31.35	13.11	64.81	90.25	<u>90.38</u>	67.22	72.07
StratTR	<u>45.24</u>	<u>19.88</u>	<u>76.06</u>	<u>94.10</u>	89.80	<u>72.71</u>	<u>88.35</u>
Ours							
StratTR (Qwen3-4B)	**51.76**†	**21.43**†	**77.26**†	**97.12**†	89.87	**74.92**	**89.67**

answer strings for all other questions; token-level F1 is computed over the raw answer strings. Because these two metrics use different representations for relation questions, we treat F1 as descriptive and base the main comparisons on EM. Our primary aggregate is exact match macro-averaged over the seven original benchmark scores, which we report as Macro EM (%). For compact presentation, Table 3 combines TimeQA-easy with TimeQA-hard and TempReason L2 with L3. We also report descriptive macro averages for the six *structural* and three *extraction* subtypes, grouped by the majority strategy in the teacher trajectories. We use two-sided McNemar tests for paired EM outcomes and paired-bootstrap confidence intervals with 10,000 resamples over the common question set (Section 5). Each bootstrap replicate samples the 22,728 paired per-question score differences with replacement, and we report the central 95% interval of the resulting mean differences. These tests compare fixed model outputs at the pooled item level; they do not test differences in the seven-dataset macro average or account for variation across training runs. Unless stated otherwise, per-column and per-subtype tests are not adjusted for multiple comparisons.

Compared systems. Table 3 compares StratTR with direct prompting and TISER prompting (Bazaga et al., 2025) using open-weight backbones from 1B to 32B and a proprietary API model under the same scoring protocol. Table 4 compares the routed teacher scaffold, the distilled StratTR-SFT policy, and the StratTR-RL policy trained with the strategy-conditioned objective. We use official baseline implementations when available and otherwise follow the settings reported in the corresponding papers.

Training. StratTR-SFT fine-tunes Qwen3-4B-Instruct-2507 (Qwen Team, 2025) with LoRA on teacher trajectories filtered only by final-answer quality, using a two-stage curriculum. GRPO then optimizes the strategy-conditioned objective in Section 3.3 with a group size of 4 and a three-stage curriculum on GH200 hardware. Appendix lists the hyperparameters taken directly from the released training configurations.

5 RESULTS

StratTR achieves the highest performance among the evaluated systems. On the same Qwen3-4B backbone, StratTR reaches 74.92% macro-averaged exact match, compared with 64.46% for TISER prompting. It obtains the highest score in four of the five displayed benchmark groups and in the macro average. For each of the four benchmark-group wins, a two-sided McNemar test on the corresponding pooled item-level outputs rejects equality with the second-best system at $p < 0.05$. It also obtains a higher macro-averaged score than TISER with the proprietary model

Table 4: Component comparisons across seven benchmarks comprising 22,728 questions (EM and F1 in %). Structural and extraction macros average the subtypes grouped by their majority teacher strategy. "Calls" is the measured mean number of external model invocations per question, not the serialized trace-step count used by R_5. All rows are scored from raw outputs with a common scoring script; pooled item-level comparisons between adjacent fixed checkpoints appear in Table 5. The best result in each column is in bold and the second-best is underlined.

Method	NR	TIME-Lite	TGQA	TimeQA	TempReason	Macro EM	Macro F1	Struct. EM	Extr. EM	Calls
Naive prompting (Qwen3-4B)	39.44	8.91	35.65	65.66	73.45	51.75	70.81	28.00	70.57	1.0
TISER prompting (Qwen3-4B)	19.90	12.91	58.23	87.21	**92.88**	64.46	73.03	30.20	84.47	1.0
TISER prompting (Gemini-3.1-flash-lite)	31.35	13.11	64.81	90.25	90.38	67.22	72.07	29.97	86.65	1.0
StratTR scaffold (Gemini-3.1-flash-lite)	45.24	19.88	76.06	94.10	89.80	72.71	88.35	42.95	87.91	6.0
StratTR-SFT (Qwen3-4B)	48.19	20.27	77.05	96.77	89.91	74.12	89.13	44.56	88.95	1.0
StratTR-RL (Qwen3-4B)	**51.76**	**21.43**	**77.26**	**97.12**	89.87	**74.92**	**89.67**	**44.68**	**89.19**	1.0

Table 5: Paired item-level comparisons on the shared 22,728 questions: mean change in EM (percentage points) with 10,000-resample paired-bootstrap 95% CIs and two-sided McNemar tests on EM flips. These pooled micro-averages differ from the macro-averaged changes in Table 4. Intervals quantify variation over test items, not training runs.

Comparison	ΔEM	95% CI	McNemar p
TISER − Naive (Qwen3-4B)	+16.38	[+15.73, +17.03]	$< 10^{-16}$
StratTR scaffold − TISER (same Gemini)	+4.12	[+3.73, +4.52]	3.03×10^{-90}
SFT student − teacher scaffold	+1.22	[+0.91, +1.52]	2.98×10^{-15}
RL − SFT	+0.51	[+0.25, +0.77]	1.72×10^{-4}
RL − TISER (same Qwen3-4B)	+7.44	[+6.96, +7.91]	1.36×10^{-197}

(74.92% versus 67.22%). TempReason, which primarily contains extraction-oriented questions, is the only displayed benchmark group on which StratTR does not rank first.

Routing improves the proprietary-model scaffold. Figure 1 illustrates three transition comparisons: replacing TISER with routed teacher scaffold, single-call distillation, and refining the student with policy optimization. As shown in Tables 4 and 5, we found that replacing TISER with the StratTR scaffold increases EM from 67.22% to 72.71% on the same proprietary model. The pooled item-level comparison estimates a difference of $+4.12$ percentage points (95% CI: $[+3.73, +4.52]$; Table 5). The structural-subtype macro increases from 29.97% to 42.95%, and exact-match accuracy on NarrativeReason relation classification increases from 31.35% to 45.24%. TISER changes 1,281 answers that direct prompting gets correct to incorrect answers, including 336 duration-computation questions. Together with the factorial result, these reversals are consistent with a fixed timeline procedure being poorly matched to date arithmetic.

Distillation reduces multi-call execution to one call. The proprietary teacher uses 6.0 external model calls per question and reaches 72.71% macro-averaged exact match; the distilled Qwen3-4B student uses 1.0 call and reaches 74.12%. The pooled item-level comparison estimates a difference of $+1.22$ percentage points (95% CI: $[+0.91, +1.52]$). The student's largest gains over the teacher occur on extraction-oriented datasets, with a small decrease on TempReason-L3. Because the model family and execution structure both change, this comparison establishes the resulting compression but does not isolate a causal effect of distillation. Both cold-start controls fail to produce parseable outputs, supporting SFT as an initialization step in this setting without isolating the value of strategy-structured trajectories.

RL provides a smaller final improvement. Starting from the SFT policy, GRPO with the combined objective increases macro-averaged exact match from 74.12% to 74.92%. This change is smaller than the observed differences associated with routing and distillation. The pooled item-level comparison estimates a difference of $+0.51$ percentage points (95% CI: $[+0.25, +0.77]$; Table 5). For pooled structural items, the estimate is $+0.72$ percentage points (95% CI: $[+0.20, +1.24]$), driven largely by NarrativeReason, which contains many more items than the other structural subtypes. NarrativeReason exact-match accuracy increases from 48.19% to 51.76%. By contrast, the macro over the six structural subtypes changes from 44.56% to 44.68%, and the extraction-subtype macro changes from 88.95% to 89.19%.

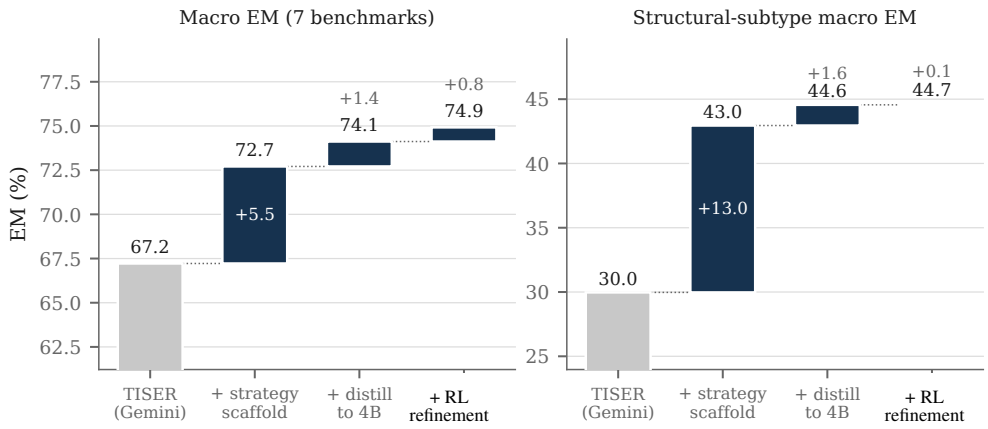

Figure 1: Observed system sequence under a common scoring protocol. The first comparison replaces TISER with the routed scaffold on the same proprietary model. The next comparisons change to the distilled 4B student and then to the checkpoint after strategy-conditioned RL. Statistical tests compare pooled item-level outputs from fixed checkpoints (Table 5).

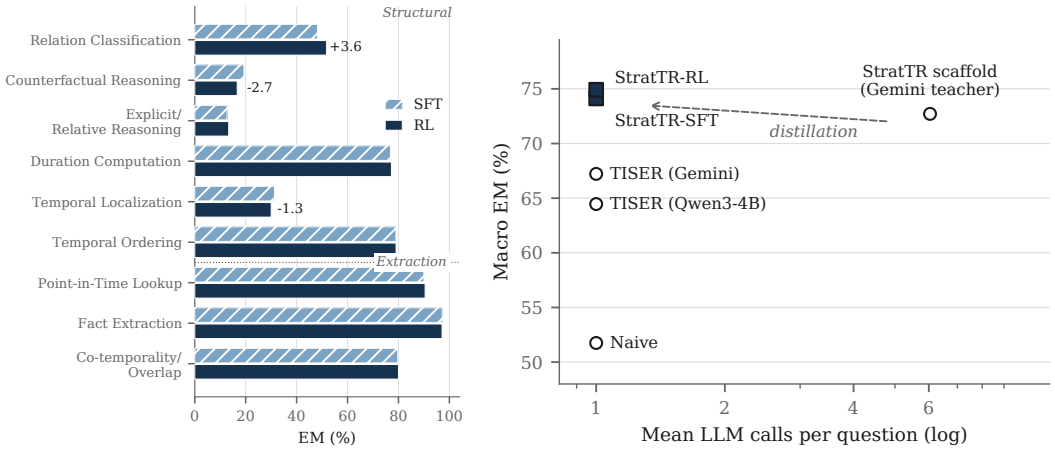

Figure 2: Left: per-subtype EM for the SFT and RL policies. Differences are shown for the three largest changes, and the dotted line separates structural from extraction subtypes. Right: accuracy versus mean external model calls per question. Filled squares denote the distilled policies, and the dashed arrow connects the teacher scaffold, which uses 6.0 calls per question on average, to the single-call student.

6 Analysis

We examine how the fixed SFT and RL checkpoints differ in their answers and emitted temporal artifacts.

6.1 Graph Outputs Change while Strategy Tags Remain Mostly Stable

We inspect per-subtype answer flips and the emitted temporal artifacts directly. Figure 2 show the largest answer change on relation classification, where exact-match accuracy increases from 48.19% to 51.76%. Explicit/relative reasoning and duration computation show smaller increases, temporal ordering changes little, and the two smallest structural subtypes show decreases (temporal localization and counterfactual reasoning, n=150 each). The trace-level diagnostics in Appendix Table 8 show that, on NarrativeReason, the RL policy emits more edges per graph (4.21 for SFT and 6.04 for RL), while the mean number of Allen violations remains near zero (0.001 per graph). The mean number of violations per graph also decreases on TIME-Lite (from 0.080 to 0.047) and TGQA

Table 6: Inference-time reranking on 200-question pools from each dataset (k=4, recomputed from raw samples). The samples come from the same-recipe SFT re-run in Table **??**; the StratTR-RL row reports the headline policy on the same questions under greedy decoding. Δ is judge best-of-four minus greedy accuracy, with a paired-bootstrap 95% CI. Judge best-of-four uses four rollout calls and four judge calls per question; greedy decoding uses one call.

	NR	TIME-Lite
SFT (re-run) greedy, pool	47.5	23.5
SFT (re-run) + judge-BoN-4	52.0	23.5
Δ vs greedy [95% CI]	+4.50 [-1.00,+10.00]	+0.00 [-3.50,+3.50]
StratTR-RL, single call	50.5	20.0
Oracle-BoN-4 (upper bound)	64.0	30.0

Table 7: Exploratory reranker comparison (Best-of-4) on small pools (n=42 NR / n=38 TIME-Lite; aggregate outputs only, hence unpaired Wilson 95% CIs), sampled with the same-recipe SFT re-run. Signs are unstable at this pool size; we report these for completeness and draw no per-reranker conclusion beyond the structural-reward null.

Reranker	Dataset	Greedy	BoN [95% CI]	Oracle
PRM-BoN	NR	40.48	45.24 [31.22, 60.05]	50.0
PRM-BoN	TIME-Lite	18.42	18.42 [9.22, 33.42]	26.32
PAV-BoN	NR	40.48	33.33 [21.01, 48.45]	50.0
PAV-BoN	TIME-Lite	18.42	26.32 [14.97, 42.01]	26.32
Structural-reward-BoN	NR	40.48	42.86 [29.12, 57.79]	50.0
Structural-reward-BoN	TIME-Lite	18.42	15.79 [7.44, 30.42]	26.32

(from 0.031 to 0.016). Strategy tags, by contrast, rarely change between the fixed checkpoints: only 5 of 22,728 questions (0.022%) receive different tags from the SFT and RL policies.

6.2 Efficiency and Exploratory Feedback Analysis

Distillation reduces external model calls. Figure 2 (right) plots accuracy against the mean number of external model calls per question over the full pool. The teacher uses 6.0 calls per question on average and obtains 72.71% macro-averaged exact match; the distilled student uses one call and obtains 74.12%. The RL checkpoint also uses one call and obtains 74.92%. These call counts do not include wall-clock latency, token usage, or monetary cost and should not be interpreted as direct measurements of deployment cost.

Exploratory reranking does not identify a preferred scorer. We evaluate inference-time reranking on the only stored multi-sample SFT run. An LLM judge reranks four rollouts using eight model calls per question in total, compared with one call for greedy decoding. On NarrativeReason, the EM point estimate increases, but its confidence interval includes zero (95% CI: $[-1.00, +10.00]$); an oracle over the same four samples reaches 64.00% EM (Table 6). The area under the receiver operating characteristic curve (AUROC) is 0.50 for the structural score and 0.66 for the judge; best-of-four reranking with the structural score does not outperform greedy decoding (Table 7). Smaller PRM, PAV, and structural-score probes also vary by dataset (Table 7). These limited candidate pools reveal both generation and selection errors, but they neither compare RL rewards nor identify a scorer that should be preferred for reranking or training.

7 Case Study

We present three worked traces from the stored artifacts, selected by fixed criteria (the first qualifying item in ID order; excerpts are verbatim but abridged).

Case 1: different outputs under forced strategies. Question: "Your task is to identify the temporal relation between Event: (team, recover, remains of Christopher Howes) and Event: (Gen. Nhek Bunchhay, sent, team to recover remains)" (gold relation: `AFTER`). Forced *direct extraction*, the strongest single strategy on the pooled factorial, labels the relation `Precedes`, a label outside the constrained seven-relation vocabulary; forced *graph construction*, the designed route for relation

classification, builds the event graph and answers `AFTER` correctly. The same backbone produces both outputs; in this example, only graph construction yields the correct relation label.

Case 2: an answer change within the same routed strategy (NarrativeReason). Question: "Your task is to identify the temporal relation between Event: (Pope's visit, causing, enormous expectations) and Event: (everyone, appears to believe, Cuba is going to change)" (gold: `IS_INCLUDED`). The SFT policy selects graph construction but predicts `OVERLAP`; the RL checkpoint selects the same strategy and predicts `IS_INCLUDED`. Its serialized trace begins with "<strategy>graph_construction</strategy> <step name="planner" iteration="1"> <thinking>1. Question . . . ".

Case 3: a change in the VAGUE label (NarrativeReason). Question: "Your task is to identify the temporal relation between Event: (manhunt, continues, for Rudolph in western North Carolina) and Event: (authorities, remember, Richard Jewell)" (gold: `VAGUE`). The SFT policy predicts the gold label `VAGUE`, whereas the RL checkpoint predicts `IS_INCLUDED`. This example illustrates the decrease in VAGUE recall reported in Section 6; it does not measure abstention or confidence calibration.

8 Discussion

Feedback choice depends on the strategy and its artifacts. Deterministic checks are reproducible and require no additional model calls, but cover only artifacts with encoded constraints; outcome feedback applies when references are available but does not diagnose process errors. Learned process verifiers and LLM judges cover less structured traces, but require verifier data or additional calls and may not transfer across tasks (Lightman et al., 2024; Setlur et al., 2025). Our small reranking studies illustrate rather than resolve these trade-offs and do not compare training rewards. We therefore treat deterministic checks as the evaluated implementation, not as a requirement for every strategy or future task.

Comparison with fixed subtype-level mappings. On the factorial sample, our designed subtype-level mapping obtains 57.41% macro EM, the best of all 262,144 fixed mappings obtains 57.78%, and an oracle over the four observed outputs for each question obtains 61.11%. For the one subtype on which the designed and best fixed mappings differ, we detect no significant difference between their selected strategies under the unadjusted test; this is not an equivalence result. No fixed subtype-level mapping can capture the observed-output oracle's question-specific advantage. Simple answer-agreement heuristics and prompted routers do not recover this advantage in the stored factorial sample, motivating further work on learned question-level selection (Li et al., 2026a).

Extension to new tasks does not require deterministic process checks. Routing and distillation require strategy representations and training outcomes, but not a deterministic verifier for every strategy. When a strategy emits a reliably parseable program or structure, deterministic checks can supplement outcome feedback; otherwise, refinement can retain outcome-only rewards or use learned or model-based scores when suitable supervision is available. Our experiments include outcome-only routes and evaluate alternative scorers only at inference time; they do not establish transfer to unseen tasks or compare these alternatives during training.

Both candidate generation and selection remain limited. On the stored four-sample pools, oracle accuracy is 64.00% on NarrativeReason and 30.00% on TIME-Lite, compared with greedy accuracy of 47.50% and 23.50%, respectively (Table 6). The oracle gains leave room for better selection, while oracle performance well below 100% shows that the sampled candidates also omit many correct answers; the current evidence does not establish a preferred scorer.

9 Conclusion

We show that different temporal subtypes favour different procedures: the best observed strategy varies by subtype. Motivated by this result, StratTR selects among four procedures and distills routed, multi-call teacher trajectories into a single-call 4B policy. Across seven benchmarks comprising 22,728 questions, StratTR achieves 74.92% macro-averaged exact match, compared with 64.46% for TISER on the same backbone. Distillation reduces the mean number of external model calls from 6.0 to 1.0, and strategy-conditioned RL provides a further improvement.

AI USE STATEMENT

We used generative AI tools primarily to polish the manuscript, including correcting grammar and improving clarity and readability. The authors reviewed all AI-assisted revisions to ensure accuracy and consistency with the intended meaning.

REFERENCES

James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26 (11):832–843, 1983.

Adrián Bazaga, Rexhina Blloshmi, Bill Byrne, and Adrià de Gispert. Learning to reason over time: Timeline self-reflection for improved temporal reasoning in language models. In *Proceedings of ACL*, 2025.

Wenhu Chen, Xinyi Wang, and William Yang Wang. A dataset for answering time-sensitive questions. In *Proceedings of NeurIPS Datasets and Benchmarks Track*, 2021.

Zheng Chu et al. TimeBench: A comprehensive evaluation of temporal reasoning abilities in large language models. *ArXiv preprint*, 2023.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 2025. URL https://arxiv.org/abs/2501.12948.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating LLMs on temporal reasoning. In *Proceedings of ICLR*, 2025. URL https://arxiv.org/abs/2406.09170.

Jiazhan Feng et al. ReTool: Reinforcement learning for strategic tool use in LLMs. *ArXiv preprint*, 2025.

Zhaoyan Gong, Zhiqiang Liu, Songze Li, Xiaoke Guo, Yuanxiang Liu, Xinle Deng, Zhizhen Liu, Lei Liang, Huajun Chen, and Wen Zhang. Temp-R1: A unified autonomous agent for complex temporal KGQA via reverse curriculum reinforcement learning. *ArXiv preprint*, 2025.

Qianyi Hu, Xinhui Tu, Cong Guo, and Shunping Zhang. Time-aware ReAct agent for temporal knowledge graph question answering. In *Findings of NAACL*, 2025.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *Proceedings of ICLR*, 2026a. URL https://arxiv.org/abs/2510.05592.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *Proceedings of ICLR*, 2026b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proceedings of ICLR*, 2024. URL https://arxiv.org/abs/2305.20050.

Zijia Liu, Peixuan Han, Haofei Yu, Haoru Li, and Jiaxuan You. Time-R1: Towards comprehensive temporal reasoning in LLMs. *ArXiv preprint*, 2025.

Qwen Team. Qwen3 technical report. *ArXiv preprint*, 2025.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *Proceedings of ICLR*, 2025. URL https://arxiv.org/abs/2410.08146.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *Proceedings of ICLR*, 2025. URL https://arxiv.org/abs/2408.03314.

Jiayu Song, Mahmud Elahi Akhter, Dana Atzil-Slonim, and Maria Liakata. Temporal reasoning for timeline summarisation in social media. In *Proceedings of ACL*, 2025. URL https://aclanthology.org/2025.acl-long.1362/.

Zhaochen Su, Jun Zhang, Tong Zhu, Xiaoye Qu, Juntao Li, Min Zhang, and Yu Cheng. Timo: Towards better temporal reasoning for language models. In *Proceedings of COLM*, 2024. URL https://arxiv.org/abs/2406.14192.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of ACL*, 2023.

Shaohang Wei, Wei Li, Feifan Song, Wen Luo, Tianyi Zhuang, Haochen Tan, Zhijiang Guo, and Houfeng Wang. TIME: A multi-level benchmark for temporal reasoning of LLMs in real-world scenarios. In *Proceedings of NeurIPS Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2505.12891.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. Large language models can learn temporal reasoning. In *Proceedings of ACL*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Proceedings of NeurIPS*, 2025. URL https://arxiv.org/abs/2504.13837.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, et al. AgentRL: Reinforcing multi-task LLM agents from zero. *ArXiv preprint*, 2025.

A STRATEGY AND EPISODE TEMPLATES

Each agentic episode instantiates a fixed sequence of role templates; the strategy determines which solver is called between extraction and evaluation. The verbatim opening role statement of every template is reproduced below (abridged); the full Jinja templates ship with the release.

Episode roles.

- `planner`: You are an expert Temporal Reasoning Planner. Your goal is to analyze a question, context, and the current state of a Temporal Knowledge Graph to decide which events need to be extracted next.
- `extractor`: You are an expert Temporal Event Extractor. Your goal is to extract event information with precise temporal boundaries to support complex reasoning.
- `linker`: You are an expert Temporal Relation Analyst. Your task is to determine the temporal relation between events using all available evidence: explicit timestamps, temporal connectors, narrative structure, causal ordering, and contextual cues.
- `evaluator`: You are a Skeptical Graph Logic Evaluator. Your task is to rigorously determine if the current Temporal Knowledge Graph contains sufficient AND well-supported information to answer the user's Question.

Strategy solvers.

- `graph_solver`: You are a Precise Temporal Question Solver. Answer the question based strictly on the constructed Temporal Graph and the original Context.
- `timeline_solver`: You are a Timeline Solver specialized in temporal ordering tasks. Your task is to sort events chronologically using the constructed temporal graph and answer ordering questions.
- `code_solver`: You are a Temporal Code Solver. Your task is to generate Python code that computes the answer to a temporal reasoning question using extracted temporal events.
- `date_solver`: You are a Date Solver specialized in temporal localization tasks. Your task is to anchor events to specific dates, times, or time periods using the constructed temporal graph.
- `direct_solver`: You are a Direct Temporal Fact Solver. Your task is to directly extract the answer to a temporal question from the given context. No graph construction is needed — the answer can be found directly.
- `overlap_solver`: You are an Overlap Solver specialized in co-temporality and overlap detection tasks. Your task is to identify events that co-occur, overlap, or share temporal intervals using the constructed temporal graph.

Hyperparameters (from the released configurations). SFT: LoRA rank 64, $\alpha = 128$, dropout 0.05, two epochs, effective batch size 128, two-stage curriculum over teacher trajectories filtered by final-answer quality only. GRPO (headline stage 3 configuration): group size 4, temperature 0.7, top-p 0.95, learning rate 5×10^{-6}, LoRA rank 64, three-stage curriculum.

B DATA CARD AND RELEASE

The release comprises (i) the unified free-form test pool (22,728 questions, seven benchmarks, shared schema with subtype and split fields), (ii) the teacher-trajectory corpus (raw and quality-filtered) used for distillation, (iii) verifier and parser implementations, (iv) the evaluation pipeline that regenerates every table, figure, and in-text number from raw outputs, and (v) the train–test audit script with its as-trained output. Source datasets retain their original licenses; contexts are drawn from the published benchmark releases, and no additional personal data is introduced.

Table 8: Trace diagnostics on graph-emitting datasets, computed once with the production parser and Allen checker: parse-success rate, mean edges per emitted graph, and mean violations of Allen's interval-algebra constraints (self-loop, symmetry, and transitivity) per graph.

Dataset	System	Parse	Graph rate	Edges/graph	Viol./graph
NR	Teacher scaffold (Gemini)	1.000	1.000	4.58	0.009
	StratTR-SFT	1.000	1.000	4.21	0.007
	StratTR-RL	1.000	0.998	6.04	0.001
TIME-Lite	Teacher scaffold (Gemini)	1.000	0.954	5.72	0.136
	StratTR-SFT	1.000	0.664	4.43	0.080
	StratTR-RL	1.000	0.657	4.59	0.047
TGQA	Teacher scaffold (Gemini)	1.000	0.753	5.75	0.042
	StratTR-SFT	1.000	0.753	5.71	0.031
	StratTR-RL	1.000	0.751	5.79	0.016