RL-Guided Temporal Localization for Dual-Channel Retrieval in Long-Horizon Agent Memory

Dabin Sheng[1], Zhe Wu[1], Jinming Zhao[2], Junliang Xing[1]†, Yuanchun Shi[1]
[1]Key Laboratory of Pervasive Computing, Ministry of Education,
Department of Computer Science and Technology, Tsinghua University
[2]Qiyuan Lab
{sdb26,wu-z24}@mails.tsinghua.edu.cn
{jlxing,shiyc}@tsinghua.edu.cn zhaojinming@qiyuanlab.com

Abstract

Time-aware retrieval is crucial for long-horizon LLM agents to effectively leverage ever-growing long-term memory. Existing methods focus on lexical or semantic similarity, leading to target memories being crowded out by temporally mismatched yet semantically similar candidates. Even time-aware approaches inject temporal cues only during memory organization or answer generation, overlooking temporal constraints at the retrieval stage, where candidate selection occurs—the root cause of the displacement. Inspired by the temporal contiguity effect in human episodic memory, we propose **TIDER** (**T**emporal **I**nterval-driven **D**ual-channel **E**vidence **R**etrieval), a time-aware agentic retrieval framework. Central to TIDER is a reinforcement learning (RL)-trained temporal interval locator that infers the temporal windows in which the evidence lies. RL optimizes window localization as a discrete, non-differentiable retrieval decision, using composite rewards to convert sparse answer supervision into fine-grained retrieval feedback. These temporal windows first define the temporal retrieval channel; the retrieved candidates are then merged with results from the semantic channel and reranked by proximity to the windows. This temporal grounding enables TIDER to retrieve temporally aligned evidence that would otherwise be drowned out by the volume of semantically similar yet temporally mismatched memories. With a lightweight 1.7B locator, TIDER achieves state-of-the-art accuracy on both long-horizon memory benchmarks—61.52% on LifeBench (+8.82%) and 78.96% on LoCoMo (+3.66%), which indicates that temporal cues are crucial for pinpointing target evidence amid a vast pool of semantically similar historical events in long-term daily-life memory.

1 Introduction

In recent years, long-term personalized assistants powered by large language model agents have advanced rapidly [16, 18, 14, 2, 34]. Sustaining such long-horizon interactions requires a foundational capability: retrieving pertinent evidence from an ever-growing history of daily events to address user queries [5, 10]. However, daily events are highly homogeneous in surface semantics—*"had dinner with family"* can recur hundreds of times within a year—and the cue that genuinely tells one occurrence from another is often *when* it took place. This makes time-aware retrieval the central bottleneck: the system must locate the right evidence among many semantically similar events recurring at different times. In long-horizon personalized memory benchmarks such as LifeBench [4] and ATM-Bench [17], 80–90% of questions involve temporal expressions such as *"what did I have for lunch after returning from Japan?"*, making time-aware retrieval indispensable rather than incidental.

†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

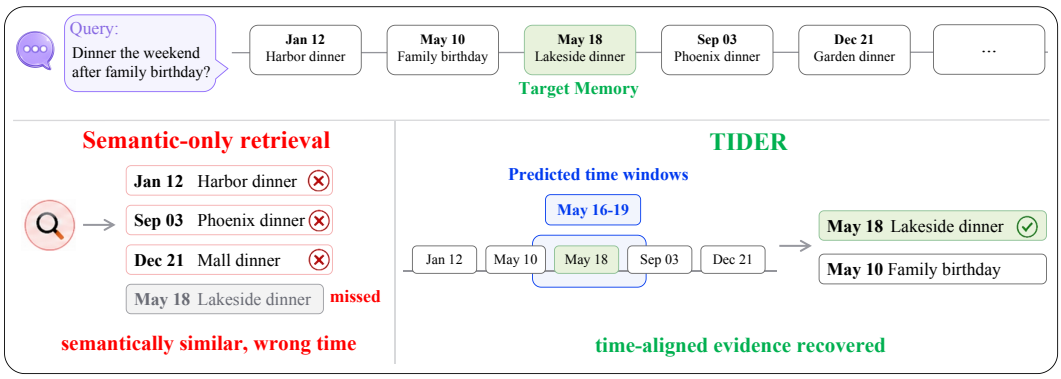


Figure 1: Time-sensitive queries require evidence from a specific window, but semantic-only retrieval ranks semantically similar memories from the wrong time. TIDER predicts a temporal window to define the temporal channel, and promotes temporally aligned evidence into the retrieved context.

Existing agentic memory systems incorporate temporal information mostly outside the retrieval stage. Some inject temporal cues during memory organization with timestamps, temporal knowledge graphs, or time-aware structures [11, 21, 26]; others strengthen temporal reasoning during answer generation with neuro-symbolic reasoning, program solvers, or temporal operators [7, 25, 13, 29]; recent RL-based methods optimize memory operations or post-retrieval candidate filtering [31, 27, 6]. None of these methods addresses the root cause: once the correct evidence has been crowded out at retrieval, downstream reasoning and post-retrieval filtering can only operate on an incomplete or contaminated candidate set. Borrowing prior time-aware retrieval is also a poor fit: rule-based normalizers [33] fail on personal, fuzzy phrasings such as *"the week of her birthday"*, while time-sensitive encoders [1, 28, 20] couple temporal and semantic signals in one vector space, demanding extra supervision and degrading semantic recall. Long-term memory, therefore, calls for a retrieval-stage, query-adaptive temporal mechanism decoupled from the semantic encoder.

Motivated by the temporal contiguity effect in human episodic memory, where retrieval is shaped by temporal distance as well as semantic association [9], we propose **TIDER** (**T**emporal **I**nterval-driven **D**ual-channel **E**vidence **R**etrieval), a time-aware agentic retrieval framework for long-term memory; the basic idea is illustrated in Figure 1. TIDER decouples temporal cues from the semantic encoder and represents them as temporal windows at the retrieval stage. Central to TIDER is a temporal interval locator π_θ trained via reinforcement learning: given a query and a timeline summary, π_θ decides whether the query is time-sensitive and predicts the windows that contain the target evidence. Window localization is a discrete, non-differentiable decision, so we cast it as an RL problem. The predicted windows define an independent temporal retrieval channel, while the semantic retrieval channel runs in parallel. Results from the two channels are merged and reranked by their distance to the predicted windows, promoting temporally aligned evidence into the top-k and suppressing semantically close but temporally mismatched distractors. π_θ is trained under a composite reward over temporal need, window localization, evidence coverage, and answer correctness. When the query is non-temporal, π_θ closes the temporal channel, and TIDER falls back to the semantic retrieval.

We evaluate TIDER on LifeBench [4] and LoCoMo [15], where it consistently outperforms prior agentic memory frameworks, time-aware retrieval methods, and temporal reasoning methods, with the largest gains on hard queries that require distinguishing semantically similar events recurring at different times. Our contributions are:

1. **A time-aware agentic retrieval framework for long-term memory.** TIDER predicts query-adaptive temporal windows that define a temporal channel parallel to the semantic one, and reranks merged candidates by window distance, resolving the bottleneck that semantically similar events from different times are hard to distinguish at retrieval.

2. **A reinforcement-learning algorithm for training the temporal interval locator.** Rewards over temporal need, window localization, evidence coverage, and answer correctness convert sparse answer-only supervision into fine-grained retrieval-stage feedback, enabling the locator to predict evidence-covering windows that improve downstream retrieval.

3. **State-of-the-art performance on long-horizon memory benchmarks.** With only a 1.7B locator, TIDER reaches 61.52% on LifeBench ($+8.82\%$) and 78.96% on LoCoMo ($+3.66\%$), indicating that temporal cues are crucial for pinpointing target evidence amid a vast pool of semantically similar historical events in long-term daily-life memory.

2 Related Work

2.1 Long-term Agentic Memory

Long-term agentic memory is an active area of research for LLM agents. One line focuses on the organization of memories: MemGPT [19] virtualizes context paging, A-MEM [30] maintains a Zettelkasten-style dynamic index, Mem0 [5] and MemoryBank [35] extract and consolidate episodic facts, Zep [21] encodes events as a temporal knowledge graph with explicit validity intervals, and TiMem [11] builds a multi-granularity temporal tree. Another line applies reinforcement learning to memory operations: Memory-R1 [31] and Mem-α [27] train policies with GRPO [23] to decide when to create or update memory, while Memory-T1 [6] applies RL to multi-session temporal reasoning at the answer stage. All of these methods act either *upstream* of retrieval (deciding what to store and when to update) or *downstream* (filtering candidates after retrieval), and none directly optimizes the retrieval stage itself. TIDER fills this gap by applying reinforcement learning to the generation of retrieval-time temporal constraints rather than to memory operations or post-retrieval reasoning.

2.2 Time-aware Retrieval and Temporal Reasoning

Conventional time-aware retrieval models temporal signals inside a vector space: TempRetriever [1] trains separate temporal and semantic encoders, TsContriever [28] trains a temporal query encoder with a temporal-need gate, TSM [8] merges multiple time-specifier-conditioned retrievers, and TimeR4 [20] trains temporal embedders or rerankers over temporal knowledge graphs. Coupling temporal and semantic signals within a single feature space requires large-scale temporal supervision and degrades semantic recall. Heuristic-based methods such as MRAG [33] instead split the query into temporal and semantic parts and rely on hand-crafted templates or date-distance scoring; they cannot parse relative or fuzzy personal-history phrasings such as *"the week of her birthday"*. In contrast, TIDER's temporal windows live outside any vector space: the locator emits explicit temporal windows that constrain an independent temporal channel, leaving the semantic encoder untouched. A complementary line strengthens *downstream* temporal reasoning given retrieved evidence: TReMu [7] translates evidence into Python via neuro-symbolic reasoning, CoTime [29] uses SQL-style temporal operators, and NeSTR [13] performs neuro-symbolic abductive reasoning over symbolic temporal relations. These methods take the retrieved set as given and ask how to compute over it, while we ask the upstream question of *whether the retrieved set covers the correct temporal windows at all*.

3 Method

TIDER is a time-aware agentic retrieval framework for long-term memory QA that decouples temporal constraints from the semantic embedding space and expresses them as explicit temporal windows. The pipeline is built around a reinforcement-learned locator π_θ: given a query and a timeline summary, π_θ predicts the windows that contain the required evidence (Section 3.1). The predicted windows define an independent temporal channel that runs in parallel with the always-on semantic channel, and the merged candidates are reranked by their temporal distance to the predicted windows (Section 3.2). The locator is trained with a composite reward over temporal need, window localization, evidence coverage, and answer correctness (Section 3.3). When the query is non-temporal, π_θ closes the temporal channel, and TIDER falls back to the semantic retrieval.

3.1 Temporal Constraint Generation

TIDER takes a user's multi-turn dialogue history $\mathcal{C}$ and a query q as input, and produces an evidence set $\hat{\mathcal{E}}$ for a downstream LLM, which generates an answer $\hat{a}$ conditioned on $\hat{\mathcal{E}}$ and compared against the gold answer a^*. This subsection describes the two steps that turn $\mathcal{C}$ into the inputs consumed at the retrieval stage: first, the dialogue is compressed into a temporally anchored event memory pool $\mathcal{M}$, then a locator π_θ rewrites the query's temporal need into an explicit temporal localization z.

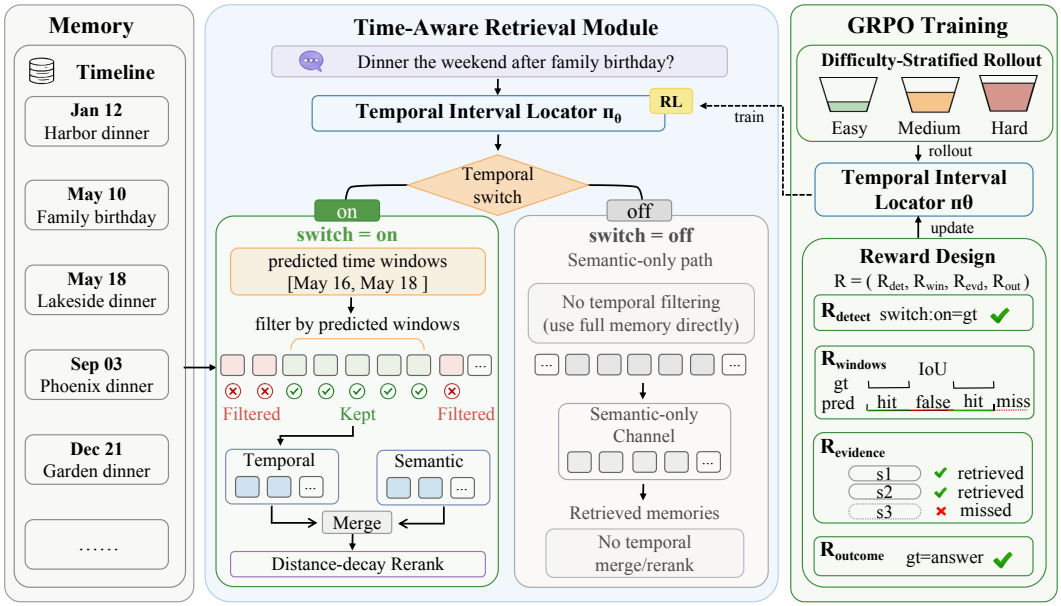


Figure 2: Overview of TIDER. TIDER uses a temporal interval locator to decide whether a query requires temporal constraints and, if so, to predict explicit temporal windows. The windows define the temporal channel, whose candidates are merged with semantic candidates and reranked by temporal distance; non-temporal queries fall back to the semantic-only path. The locator is trained with GRPO under rewards for switch detection, window localization, evidence coverage, and answer correctness.

We convert the raw dialogue $\mathcal{C}$ into a temporally anchored event memory pool, so that events sharing a temporal neighborhood and semantic similarity are explicitly grouped into the same cluster and become amenable to window-based retrieval:

$$\mathcal{M} = \{m_i = (d_i, s_i, \mathcal{I}_i)\}_{i=1}^{N}, \tag{1}$$

where $d_i = [d_i^{\text{start}}, d_i^{\text{end}}]$ is cluster i's date span, taken as the union of its members' timestamps; s_i is the cluster's event summary; and $\mathcal{I}_i$ is the set of raw memory items inside the cluster. The construction is performed by an LLM that extracts items with timestamps from each dialogue turn, then clusters them first by date and then by semantic similarity; Appendix B provides the construction details and corpus-level cluster statistics.

Implicit or relative temporal phrasings such as "the week of her birthday" or "after the last get-together" are hard to cover with rule templates and hurt semantic retrieval when forced into a shared embedding. We therefore introduce a locator $\pi_\theta(\cdot \mid q, \mathcal{S}_q)$ that takes the query q together with a query-adaptive timeline summary $\mathcal{S}_q = \{(d_i, s_i)\}_{i \in \mathcal{N}_q} \subseteq \mathcal{M}$, and emits a structured temporal localization

$$z = (\texttt{detect}, W_{\text{pred}}), \qquad W_{\text{pred}} = \{w_\ell = [w_\ell^{\text{start}}, w_\ell^{\text{end}}]\}_{\ell=1}^{L}, \tag{2}$$

where $\texttt{detect} \in \{\text{true}, \text{false}\}$ gates the temporal channel and W_{pred} is a set of temporal windows where the evidence may lie. Because the constraint is expressed as explicit windows rather than vectors, TIDER neither needs to train a time-sensitive encoder nor contaminates the semantic embedding; π_θ is trained via GRPO [23] under the composite reward of Section 3.3.

With $\mathcal{M}$ from preprocessing and z from the locator, TIDER's retrieval can be formalized as a scoring function conditioned on the temporal localization,

$$f : (q, \mathcal{M}; z) \mapsto \hat{\mathcal{E}}, \tag{3}$$

whose computation is described in Section 3.2: it uses $\texttt{detect}$ and W_{pred} inside z to drive parallel semantic and temporal recalls, merges their candidates, and reranks the merged set by date distance to W_{pred}, yielding the final top-k set $\hat{\mathcal{E}}$.

3.2 Dual-channel Constrained Retrieval

A single-channel retriever cannot balance semantic and temporal evidence: a semantic-only channel floods the top-k with semantically similar events from wrong times, while a temporal-only channel misses temporally distant yet relevant evidence. TIDER therefore runs the two channels in parallel and merges their candidates. The semantic channel encodes the query q and each memory m_i into vectors $\mathbf{e}_q$ and $\mathbf{e}_{m_i}$ via the dense retriever and uses their cosine similarity as the semantic score,

$$\text{sem}(q, m_i) = \frac{\mathbf{e}_q^\top \mathbf{e}_{m_i}}{\|\mathbf{e}_q\| \, \|\mathbf{e}_{m_i}\|}, \tag{4}$$

The temporal channel is gated by `detect`: when `detect` = false, it is closed and retrieval degrades to only semantic; when `detect` = true, for each $m_i \in \mathcal{M}$ we check whether its temporal window d_i overlaps any window in W_{pred} and keep only the overlapping candidates.

The candidates from the two channels are unified by a single ranking score that combines (i) the semantic similarity, (ii) a boost for entering the temporal channel, and (iii) a temporal distance decay that suppresses semantically-close but temporally-mismatched distractors:

$$\text{score}(q, m_i) = \gamma(\text{dist}_i) \cdot \big[\, \text{sem}(q, m_i) + \lambda_{\text{boost}} \cdot \mathbf{1}[m_i \in \text{temporal channel}] \,\big], \tag{5}$$

Here dist_i denotes the smallest date distance from m_i's event date d_i to W_{pred}: zero when d_i overlaps any window, and the number of days to the nearest window boundary otherwise. The coefficient γ takes a piecewise form with no decay inside a buffer, linear decay outside the buffer:

$$\gamma(\text{dist}_i) = \begin{cases} 1, & \text{dist}_i \le \tau_{\text{buf}}, \\ \max\big(\gamma_{\min},\ 1 - \rho\,(\text{dist}_i - \tau_{\text{buf}})\big), & \text{dist}_i > \tau_{\text{buf}}. \end{cases} \tag{6}$$

Here τ_{buf} is the tolerance buffer that absorbs week-level misalignments, ρ is the per-day decay rate, and $\gamma_{\min}$ is the floor that keeps distant-yet-relevant evidence from vanishing. When `detect` = false, the temporal channel is closed, and the decay is bypassed, so $\text{score}(q, m_i)$ reduces to $\text{sem}(q, m_i)$. The system takes the top-k by score as $\hat{\mathcal{E}}$ and feeds it to the downstream LLM.

3.3 Reward Design and GRPO Training

The pipeline of Sections 3.1–3.2 is driven by π_θ, but z is discrete and blocks end-to-end gradient flow from the retriever; we therefore train π_θ with reinforcement learning. Each training sample carries four gold supervisions: `gold_temporal` $\in \{\text{true}, \text{false}\}$; the gold window set W_{gold}, present only when `gold_temporal` = true; the gold evidence $\mathcal{E}^* = \{e_k\}_{k=1}^{|\mathcal{E}^*|}$; and the gold answer a^*. Appendix C details the supervision provenance and split construction.

Sparse supervision from answer correctness r_{out} alone is insufficient for stable learning of *when to trigger the temporal channel* and *which window to fetch from*: a correct answer obtained from a wrong window would still be reinforced by r_{out}. We therefore decompose the training signal into four complementary rewards that correspond to different aspects of the retrieval decision: temporal need, window localization, evidence coverage, and answer correctness. With fixed reward weights, these rewards are combined as

$$R = w_{\text{det}}\, r_{\text{det}} + w_{\text{win}}\, r_{\text{win}} + w_{\text{evd}}\, r_{\text{evd}} + w_{\text{out}}\, r_{\text{out}}, \qquad \sum_{\bullet} w_\bullet = 1, \tag{7}$$

The detection reward r_{det} supervises whether the temporal channel is correctly triggered, given by the agreement between `detect` and `gold_temporal`:

$$r_{\text{det}} = \begin{cases} +1, & \texttt{detect} = \texttt{gold_temporal}, \\ -1, & \text{otherwise}. \end{cases} \tag{8}$$

The window-localization reward r_{win} measures the geometric alignment of W_{pred} with W_{gold}: for any window pair (P, G) we take the IoU $\varphi(P, G) = |P \cap G| / |P \cup G|$, and aggregate via a symmetric, bidirectional best match. For non-temporal queries or when $W_{\text{pred}} = \emptyset$, we set $r_{\text{win}} = 0$.

$$r_{\text{win}} = \tfrac{1}{2} \left(\tfrac{1}{|W_{\text{gold}}|} \sum_{G \in W_{\text{gold}}} \max_{P \in W_{\text{pred}}} \varphi(P, G) + \tfrac{1}{|W_{\text{pred}}|} \sum_{P \in W_{\text{pred}}} \max_{G \in W_{\text{gold}}} \varphi(P, G) \right). \tag{9}$$

The evidence reward r_{evd} feeds the retrieval outcome back to the locator: r_{win} only scores the geometric overlap between W_{pred} and W_{gold}, while r_{evd} checks whether the retriever conditioned on W_{pred} actually places $\mathcal{E}^*$ within top-k. An independent LLM judge tests, item by item, whether e_k is semantically covered by $\hat{\mathcal{E}}$, and the evidence reward is the coverage rate

$$r_{\text{evd}} = \frac{1}{|\mathcal{E}^*|} \sum_{k=1}^{|\mathcal{E}^*|} \mathbf{1}\big[e_k \text{ covered by } \hat{\mathcal{E}}\big]. \tag{10}$$

The outcome reward r_{out} is the direct signal of whether the final answer $\hat{a}$ is correct, evaluated per question type:

$$r_{\text{out}} = \begin{cases} \mathbf{1}\big[\mathrm{ExactMatch}(\hat{a}, a^*)\big], & \text{multiple-choice}, \\ \mathbf{1}\big[\mathrm{maj}_{j=1,2,3}\,\mathrm{judge}^{(j)}(\hat{a}, a^*) = 1\big], & \text{open-ended}, \end{cases} \tag{11}$$

where $\mathrm{ExactMatch}$ checks string equality after normalization, and $\mathrm{judge}^{(j)}$ denotes three independent LLM-judge calls whose majority vote suppresses single-evaluation variance. Appendix J.6 audits judge consistency across model families.

A finite annotated $(q, \texttt{detect}, W_{\text{pred}})$ set covers neither the long tail of natural-language temporal phrasings nor the actual downstream retrieval and reasoning effects, so we adopt a two-stage scheme of SFT cold-start followed by GRPO online optimization: SFT aligns the locator to a fixed JSON schema and seeds an initial window guess, while GRPO uses R as the policy-gradient signal to self-correct over end-to-end rollouts,

$$\mathcal{J}(\theta) = \mathbb{E}_{q,\,\{z^{(j)}\}\sim\pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{j=1}^{G} \min\big(\rho^{(j)} A^{(j)},\ \mathrm{clip}\big(\rho^{(j)}, 1-\epsilon, 1+\epsilon\big)\, A^{(j)}\big) \right] - \beta\, \mathbb{D}_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big), \tag{12}$$

Rollouts whose output cannot be parsed by the schema are clamped to $R = R_{\min}$ and skip the downstream judge; rollouts are U-shape difficulty-stratified by $|W_{\text{gold}}|$.

4 Experiments

4.1 Datasets

LifeBench [4] is a benchmark for long-term, multi-source, fragmented personal memory that spans both declarative and non-declarative memory. It contains one-year synthetic personal histories for ten users, with on average roughly 13,000 entries per user and a total of 2,003 questions spanning five categories: Information Extraction (IE), Multi-hop Reasoning (MR), Nondeclarative Memory Reasoning (ND), Temporal and Knowledge Updating (TKU), and Unanswerable (UA); following the LoCoMo evaluation protocol, the Unanswerable category is excluded from evaluation.

LoCoMo [15] focuses on extremely long multi-session dialogues with QA spanning five reasoning categories—single-hop, multi-hop, temporal, open-domain, and adversarial; following prior work, we evaluate on its 1,540-question subset excluding the adversarial category. Expanded benchmark descriptions and evaluation protocols are provided in Appendix A.

4.2 Experiments Setup

Baselines. We compare against three families of baselines; aside from Oracle, all methods share the same downstream answer model and top-k retrieval budget. *Agentic memory frameworks:* **MemOS** [12] is a memory operating system that builds unified mechanisms across parametric, activation, and plaintext memory types; **TiMem** [11] is a temporal-hierarchical memory framework that consolidates conversations into a temporal memory tree for long-horizon agents; **Zep** [21] is a memory layer service for LLM agents that synthesizes history into a temporally-aware knowledge graph; **Mem0** [5] is a memory-centric architecture that extracts, consolidates, and retrieves salient information from ongoing conversations. *Time-aware retrieval:* **TsContriever** [28] applies supervised contrastive learning with temporally-constrained negatives to fine-tune the query encoder, routing queries through a classifier to handle temporal constraints. *Temporal reasoning:* **NeSTR** [13]

Table 1: Performance comparison on the LifeBench benchmark. Categories include Information Extraction, Multi-hop Reasoning, Nondeclarative Memory Reasoning, and Temporal and Knowledge Updating. Best non-Oracle results are **bolded**; underline indicates second best.

Method	IE ↑	MR ↑	ND ↑	TKU ↑	Overall ↑	F1 ↑
TsContriever [28]	31.28±1.05	23.33±0.00	27.43±0.60	23.91±0.00	27.21±0.42	49.36
MemOS [12]	54.57±1.43	48.89±0.48	42.01±1.20	31.16±1.26	47.30±0.88	45.90
Zep [21]	43.61±0.40	39.44±0.96	32.99±0.60	30.43±2.17	38.40±0.37	38.55
Mem0 [5]	57.99±1.05	48.33±2.89	50.35±1.59	44.20±6.64	51.80±1.26	51.94
TiMem [11]	17.58±1.43	17.50±1.44	21.53±3.01	10.87±2.17	17.73±0.57	18.89
NeSTR [13]	56.16±1.37	52.22±1.27	53.12±1.80	42.03±3.32	52.70±0.65	61.25
TIDER	**67.12**±0.40	**60.83**±2.10	**60.42**±0.60	**47.83**±2.51	**61.52**±0.86	**69.23**
Oracle (upper bound)	*87.44±1.05*	*82.78±0.96*	*81.94±3.18*	*74.64±2.51*	*83.33±0.74*	*76.79*

performs neuro-symbolic abductive reasoning over symbolic temporal representations with reflection and consistency checking. We do not include training-based temporal-reasoning methods such as Memory-T1 [6] as direct baselines, since they fine-tune the response model, whereas our evaluation keeps the downstream LLM fixed across methods. We additionally report **Oracle**, which directly provides the gold-evidence context, as a non-comparable upper bound.

Implementation. The locator π_θ uses **Qwen3-1.7B** [32] as its backbone and is trained with GRPO [23] via the veRL framework [24], with KL coefficient $\beta = 0.1$ and group size $G = 8$; the query-adaptive timeline summary $\mathcal{S}_q$ fed to π_θ is restricted to the top-30 most query-relevant timeline entries for computational efficiency. The semantic channel uses the BGE-M3 [3] dense retriever together with BM25 [22]; the boost coefficient is $\lambda_{\text{boost}} = 0.3$, and the distance-decay function γ uses buffer $\tau_{\text{buf}} = 14$, decay rate $\rho = 0.05$, and floor $\gamma_{\min} = 0.3$. All methods retrieve top-$k = 20$, except Oracle which directly supplies the gold evidence. The downstream answer model and LLM judge are GPT-5-mini on LifeBench and GPT-4o-mini on LoCoMo, fixed across all baselines for fairness. The evaluation is re-executed three times, and the reported metric is the sample mean $\pm$ sample standard deviation across the three runs. The detailed introduction and implementation of baselines can be found in Appendix E. We evaluate on the user-0 and user-1 split of LifeBench (408 questions) and on the full LoCoMo set; the remaining LifeBench splits (user-2 through user-9) are used to synthesize training samples, so the training and evaluation users on LifeBench are disjoint, while LoCoMo serves as a zero-shot generalization test.

4.3 Results

Main results. On LifeBench (Table 1, 408 questions) and LoCoMo (Table 2, 1,540 questions), TIDER simultaneously attains the best overall accuracy: **61.52**% on LifeBench (second-best NeSTR 52.70%, +8.82%) and **78.96**% on LoCoMo (second-best TiMem 75.30%, +3.66%); the locator is only **1.7B**, far smaller than the shared downstream answer models GPT-5-mini / GPT-4o-mini, confirming that the gain comes from a structured retrieval-time temporal localization rather than from a stronger response model. Compared with agentic-memory frameworks (MemOS, TiMem, Zep, and Mem0), TIDER achieves the best performance on both datasets. On LifeBench, it outperforms the strongest baseline in this group, Mem0 (51.80%), by 9.72%; on LoCoMo, it surpasses the strongest baseline in this group, TiMem (75.30%), by 3.66%. The time-aware retriever TsContriever performs poorly on both datasets, with a substantially larger gap to TIDER on LifeBench than on LoCoMo. This result supports the concern raised in §1: coupling temporal and semantic signals in a shared representation space can weaken semantic recall, especially in large-scale long-term memory settings such as LifeBench, where relative and indirect temporal expressions are more prevalent and place greater stress on encoder-fused retrieval. As a temporal reasoning baseline, NeSTR processes temporal information at the answer stage rather than during retrieval. It underperforms TIDER by 8.82% and 5.48% on LifeBench and LoCoMo, respectively, highlighting the importance of addressing temporal mismatch before evidence enters the downstream context. The effectiveness of TIDER stems from its dual-channel retrieval architecture with explicit window constraints: by moving temporal constraints upstream into the retrieval stage as an explicit window, TIDER suppresses semantically similar but temporally mismatched candidates without degrading semantic recall, offering a structural advantage over both embedding-stage temporal supervision and reasoning-stage post-processing.

Table 2: Performance comparison on the LoCoMo benchmark. Reported categories include Single-hop, Multi-hop, Temporal Reasoning, and Open-domain Knowledge. Best results are **bolded**; underline indicates second best.

Method	Single-Hop ↑	Multi-Hop ↑	Temporal ↑	Open-Domain ↑	Overall ↑	F1 ↑
TsContriever [28]	77.41±0.45	57.92±0.33	62.93±0.25	39.58±0.85	68.47±0.16	48.94
MemOS [12]	76.07±0.10	56.85±0.69	69.47±0.25	45.14±0.49	69.24±0.11	45.02
Zep [21]	50.42±0.29	42.20±0.77	19.11±0.29	38.19±0.49	41.62±0.21	26.88
Mem0 [5]	62.09±0.42	50.14±0.32	59.25±0.41	37.70±0.41	57.79±0.34	42.52
TiMem [11]	81.43±0.05	62.20±0.82	**77.63**±0.34	**52.08**±0.74	75.30±0.16	**54.40**
NeSTR [13]	82.60±0.25	63.00±1.02	66.56±0.18	47.57±0.60	73.48±0.23	53.54
TIDER	**86.44**±0.79	**74.11**±2.41	71.65±0.31	**52.08**±2.40	**78.96**±0.30	51.72

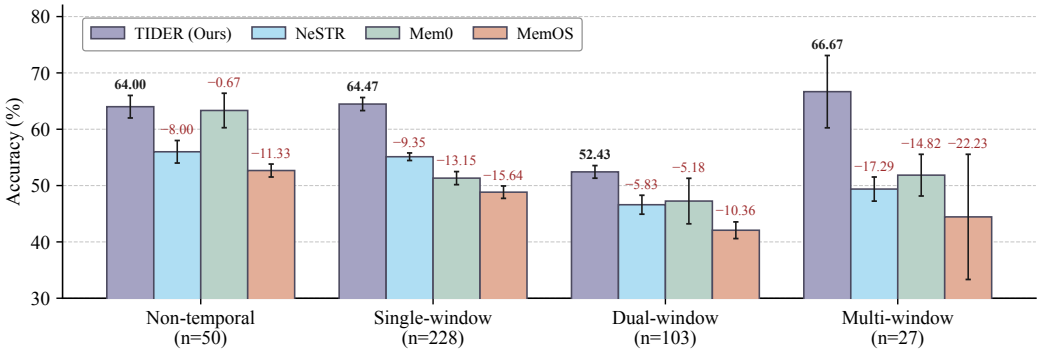


Figure 3: LifeBench accuracy stratified by gold-window count. **Bold** labels report TIDER's absolute accuracy; colored labels report each baseline's delta from TIDER within the same category.

Although TIDER achieves the highest overall accuracy on LoCoMo, its Temporal accuracy and token-level F1 are slightly below TiMem's. Our analysis suggests this gap stems from cross-dataset window-localization mismatch on LoCoMo's sparser timeline: predicted windows cover 76.25% of gold dates, compared with 93.77% on LifeBench. Using oracle windows raises Temporal accuracy from 71.65% to 79.13% and F1 from 51.72% to 57.83%, surpassing TiMem on both.

Stratified analysis by window count. To unpack overall accuracy by question type, we stratify the 408 LifeBench questions into four categories (Fig. 3). Time-independent queries form the *non-temporal* category. For the remaining questions, we segment gold-evidence dates into temporal windows, where any gap of at least three days starts a new window, and group them into *single-window*, *dual-window*, and *multi-window* categories by window count. As shown in Fig. 3, TIDER attains the best accuracy in every category. On *non-temporal* queries, its margin over the next-best baseline is only 0.67%, suggesting that closing the temporal channel does not hurt pure semantic recall. Across the three temporal categories, TIDER improves over the strongest baseline by 9.78% on average, with the largest gains arising when evidence must be localized across separated temporal windows (+14.8%), suggesting that long-horizon agent memory benefits from treating temporal cues as an independent retrieval dimension rather than folding them into semantic similarity. Appendix F provides more detailed analysis results with standard deviations. Appendix G.2 provides case studies from different temporal categories, illustrating how baselines can retrieve same-entity events from the wrong time, miss one of two required temporal anchors, or fall back to long-running persona facts, whereas TIDER recovers the correct evidence from long-term memory.

4.4 Ablation Study

Module ablations. Table 3 reports cumulative ablations of TIDER's main components. Overall accuracy decreases monotonically from 61.52% to 51.45% as we remove the distance-decay reranker, the temporal channel, and the timeline summary $\mathcal{S}_q$. Removing the distance-decay reranker reduces Overall accuracy by 3.92%, with the largest drop on the time-sensitive TKU category (6.53%), which is especially sensitive to temporal alignment. Further removing the temporal channel yields

Table 3: Module ablations on LifeBench. Successively removing the distance-decay reranker, the temporal channel, and the timeline summary $\mathcal{S}_q$ from TIDER; the bottom row is a pure semantic baseline. Best in column **bold**.

Variant	$\mathcal{S}_q$	Temporal	Decay γ	IE ↑	MR ↑	ND ↑	TKU ↑	Overall ↑	F1 ↑
TIDER (full)	✓	✓	✓	**67.12**	**60.83**	**60.42**	**47.83**	**61.52**	**69.23**
w/o decay rerank	✓	✓	×	63.70	58.33	55.21	41.30	57.60	67.02
w/o temporal channel	✓	×	×	58.22	52.50	51.04	34.78	52.21	47.62
w/o summary (semantic)	×	×	×	55.48	51.67	48.96	39.13	51.45	63.05

an additional 5.39% drop, while removing $\mathcal{S}_q$ from the remaining semantic-only setting accounts for another 0.76%. F1 shows a non-monotonic trend: removing the temporal channel reduces F1 by 19.40 points, while further removing $\mathcal{S}_q$ recovers 15.43 points. This suggests that the timeline summary is most effective when paired with the temporal channel; without temporal constraints, it can introduce noisy context that hurts token-level matching. Overall, TIDER's gains arise from the coupled operation of the temporal interval locator, the dual-channel retriever, and the distance-decay reranker, rather than from any single component in isolation.

To separate temporal retrieval gains from memory construction, we cross two memory representations with two retrieval methods while keeping all other settings fixed. TIDER improves Overall accuracy over semantic-only retrieval by 8.09% on flat fixed chunks and 9.31% on the full memory pool. Full memory construction adds 1.72% and 2.94% under semantic-only and TIDER retrieval, respectively. These results indicate that TIDER's gains come primarily from temporal retrieval rather than memory construction. The complete factorial comparison is provided in Appendix D.2.

We also ablate the reward components while keeping the remaining training and retrieval pipeline fixed. Jointly removing the detection reward r_{det} and window-localization reward r_{win} reduces Overall accuracy by 5.64% and TKU accuracy by 15.22%. Removing the outcome reward r_{out} or evidence reward r_{evd} reduces Overall accuracy by 2.45% or 1.96%, respectively. These results show that the detection and window-localization rewards are particularly important for learning effective temporal retrieval, especially on time-sensitive questions. Detailed results are reported in Appendix C.6.

Locator substitution. We isolate the locator by fixing the dual-channel retriever and distance-decay reranker (Fig. 4). Removing GRPO from the Qwen3-1.7B locator drops Overall to 55.64% (-5.88%, with the largest loss on TKU at -13.05%), while the zero-shot Qwen3-1.7B locator reaches only 53.19%, barely above the No-Temporal lower bound ($+0.98\%$), confirming that the temporal path requires a trained window policy. A much larger GPT-5.2 zero-shot locator improves to 57.84% but remains -3.68% below TIDER, showing scale alone cannot replace reward-optimized temporal localization. Conversely, an Oracle Window upper bound (W_{gold}) adds only $+1.70\%$ Overall because TIDER's predicted windows already cover 93.77% of gold dates. These results indicate the temporal path provides the structural capacity, while GRPO learns the window-alignment policy needed to activate it.

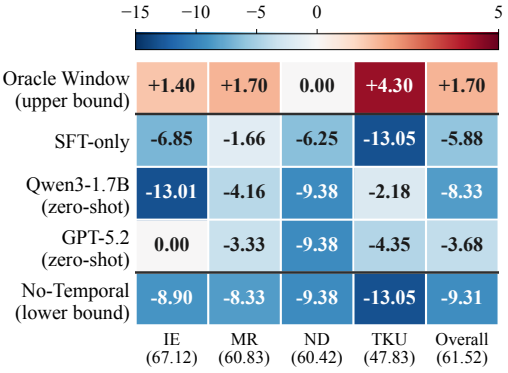
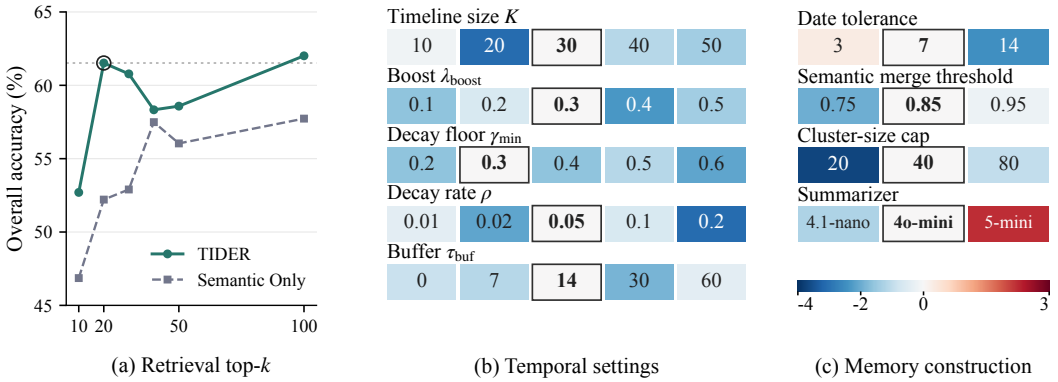


Figure 4: Locator substitution on LifeBench. Cells show accuracy differences (%) from TIDER; column headers show TIDER's baseline scores.

Hyperparameter robustness. We examine TIDER's sensitivity to retrieval budget, temporal settings, and memory construction on LifeBench (Fig. 5). Each sweep varies one setting while keeping the others fixed. Cells show accuracy differences(%) from the default score, with outlines marking default settings. Detailed results are provided in Appendix D.



(a) Retrieval top-k (b) Temporal settings (c) Memory construction

Figure 5: Hyperparameter sensitivity analysis on LifeBench.

Retrieval top-k. Figure 5(a) varies the number of retrieved memories passed to the answer model. Accuracy changes non-monotonically with the retrieval budget, indicating that retrieving more memories does not consistently improve answer quality. At $k = 20$, TIDER achieves 61.52% accuracy, exceeding the best semantic-only result at $k = 100$ by 3.79% while retrieving one fifth as many memories. Increasing TIDER's budget from 20 to 100 improves accuracy by only 0.49%. These results support $k = 20$ as an effective balance between retrieval volume and answer accuracy.

Temporal settings. Figure 5(b) examines the timeline-input size K, boost coefficient, decay floor, decay rate, and temporal buffer. Here, K controls the timeline information supplied to the locator, whereas k in panel (a) controls the retrieved memories supplied to the answer model. Across the displayed settings, Overall accuracy ranges from 58.45% to 61.52%. The largest within-sweep variation is 3.07%, with only 2.10% for the decay floor and 1.86% for the buffer. These results demonstrate the robustness of temporal retrieval to hyperparameter choices within the tested ranges.

Memory construction. Figure 5(c) varies the date-merge tolerance, semantic merge threshold, cluster-size cap, and summarizer model. Overall accuracy ranges from 57.84% to 63.73%, with every tested configuration remaining above the semantic-only baseline of 52.21% on the default full memory pool. The construction choices nevertheless affect performance: replacing GPT-4o-mini with GPT-5-mini raises accuracy to 63.73%, while reducing the cluster-size cap to 20 items lowers it to 57.84%. Thus, memory construction remains an important design choice.

5 Conclusion

In this work, we introduced **TIDER**, a time-aware agentic retrieval framework that addresses temporal-mismatch failures at retrieval time, where correct evidence is buried by semantically similar yet temporally mismatched candidates before any downstream reasoner can act on it. At the core of TIDER is a query-adaptive temporal interval locator, decoupled from the semantic encoder, that emits explicit temporal windows. These windows define a parallel temporal channel, whose candidates are merged with semantic candidates and reranked by temporal distance, suppressing temporally mismatched distractors while preserving semantic recall. The locator is trained via GRPO, with a composite reward that balances temporal need, window localization, evidence coverage, and answer correctness. Extensive evaluation on LifeBench and LoCoMo shows that TIDER attains state-of-the-art accuracy with only a 1.7B locator, and locator-substitution analysis confirms that the gain comes from retrieval-time temporal constraints rather than model scale. These results suggest temporal cues are crucial for pinpointing target evidence amid a vast pool of semantically similar historical events in long-term daily-life memory. We discuss limitations and broader impact in Appendix I.

Acknowledgments and Disclosure of Funding

This work is supported by the National Key Research and Development Plan under Grant No. 2024YFB4505500 & 2024YFB4505503.

References

[1] Abdelrahman Abdallah, Bhawna Piryani, Jonas Wallat, Avishek Anand, and Adam Jatowt. Tempretriever: Fusion-based temporal dense passage retrieval for time-sensitive questions, 2025. URL https://arxiv.org/abs/2502.21024.

[2] Jiangjie Chen, Xintao Wang, Rui Xu, Siyu Yuan, Yikai Zhang, Wei Shi, Jian Xie, Shuang Li, Ruihan Yang, Tinghui Zhu, Aili Chen, Nianqi Li, Lida Chen, Caiyu Hu, Siye Wu, Scott Ren, Ziquan Fu, and Yanghua Xiao. From persona to personalization: A survey on role-playing language agents, 2024. URL https://arxiv.org/abs/2404.18231.

[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. URL https://arxiv.org/abs/2402.03216.

[4] Zihao Cheng, Weixin Wang, Yu Zhao, Ziyang Ren, Jiaxuan Chen, Ruiyang Xu, Shuai Huang, Yang Chen, Guowei Li, Mengshi Wang, Yi Xie, Ren Zhu, Zeren Jiang, Keda Lu, Yihong Li, Xiaoliang Wang, Liwei Liu, and Cam-Tu Nguyen. Lifebench: A benchmark for long-horizon multi-source memory, 2026. URL https://arxiv.org/abs/2603.03781.

[5] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

[6] Yiming Du, Baojun Wang, Yifan Xiang, Zhaowei Wang, Wenyu Huang, Boyang XUE, Bin Liang, Xingshan Zeng, Fei Mi, Haoli Bai, Lifeng Shang, Jeff Z. Pan, Yuxin Jiang, and Kam-Fai Wong. Memory-t1: Reinforcement learning for temporal reasoning in multi-session agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vQf2YR2Kpd.

[7] Yubin Ge, Salvatore Romeo, Jason Cai, Raphael Shu, Yassine Benajiba, Monica Sunkara, and Yi Zhang. TReMu: Towards neuro-symbolic temporal reasoning for LLM-agents with memory in multi-session dialogues. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18974–18988, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.972. URL https://aclanthology.org/2025.findings-acl.972/.

[8] SeungYoon Han, Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, Huije Lee, and Jong C. Park. Temporal information retrieval via time-specifier model merging. In Yuji Zhang, Canyu Chen, Sha Li, Mor Geva, Chi Han, Xiaozhi Wang, Shangbin Feng, Silin Gao, Isabelle Augenstein, Mohit Bansal, Manling Li, and Heng Ji, editors, *Proceedings of the 3rd Workshop on Towards Knowledgeable Foundation Models (KnowFM)*, pages 1–13, Vienna, Austria, August 2025. Association for Computational Linguistics. ISBN 979-8-89176-283-1. doi: 10.18653/v1/2025.knowllm-1.1. URL https://aclanthology.org/2025.knowllm-1.1/.

[9] M Karl Healey, Nicole M Long, and Michael J Kahana. Contiguity in episodic memory. *Psychonomic bulletin & review*, 26(3):699–720, 2019.

[10] Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. Hello again! llm-powered personalized agent for long-term dialogue, 2025. URL https://arxiv.org/abs/2406.05925.

[11] Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents, 2026. URL https://arxiv.org/abs/2601.02845.

[12] Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhiqiang Yin, Qingchen Yu, Bo Tang, Hongkang Yang, Zhi-Qin John Xu, and Feiyu Xiong. Memos: An operating system for memory-augmented generation (mag) in large language models, 2025. URL https://arxiv.org/abs/2505.22101.

[13] Feng Liang, Weixin Zeng, Runhao Zhao, and Xiang Zhao. Nestr: A neuro-symbolic abductive framework for temporal reasoning in large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(38):31907–31915, Mar. 2026. doi: 10.1609/aaai.v40i38.40460. URL https://ojs.aaai.org/index.php/AAAI/article/view/40460.

[14] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges, 2025. URL https://arxiv.org/abs/2503.21460.

[15] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

[16] Andrea Matarazzo and Riccardo Torlone. A survey on large language models with some insights on their capabilities and limitations, 2025. URL https://arxiv.org/abs/2501.04040.

[17] Jingbiao Mei, Jinghong Chen, Guangyu Yang, Xinyu Hou, Margaret Li, and Bill Byrne. According to me: Long-term personalized referential memory qa, 2026. URL https://arxiv.org/abs/2603.01990.

[18] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2025. URL https://arxiv.org/abs/2402.06196.

[19] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

[20] Xinying Qian, Ying Zhang, Yu Zhao, Baohang Zhou, Xuhui Sui, Li Zhang, and Kehui Song. TimeR^4 : Time-aware retrieval-augmented large language models for temporal knowledge graph question answering. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6942–6952, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.394. URL https://aclanthology.org/2024.emnlp-main.394/.

[21] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

[22] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[24] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pages 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

[25] Xintong Song, Bin Liang, Yang Sun, Chenhua Zhang, Bingbing Wang, and Ruifeng Xu. Bridging time gaps: Temporal logic relations for enhancing temporal reasoning in large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 3040–3044, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3730173. URL https://doi.org/10.1145/3726302.3730173.

[26] Miao Su, Yucan Guo, Zhongni Hou, Long Bai, Zixuan Li, Yufei Zhang, Guojun Yin, Wei Lin, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. Beyond dialogue time: Temporal semantic memory for personalized llm agents, 2026. URL https://arxiv.org/abs/2601.07468.

[27] Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning Memory Construction via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2509.25911.

[28] Feifan Wu, Lingyuan Liu, Wentao He, Ziqi Liu, Zhiqiang Zhang, Haofen Wang, and Meng Wang. Time-sensitive retrieval-augmented generation for question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2544–2553, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679800. URL https://doi.org/10.1145/3627673.3679800.

[29] Jiaying Wu and Bryan Hooi. Chain-of-timeline: Enhancing LLM zero-shot temporal reasoning with SQL-style timeline formalization. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=ptHtrHwMpw.

[30] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=FiM0M8gcct.

[31] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2026. URL https://arxiv.org/abs/2508.19828.

[32] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[33] Siyue Zhang, Yuxiang Xue, Yiming Zhang, Xiaobao Wu, Anh Tuan Luu, and Chen Zhao. MRAG: A modular retrieval framework for time-sensitive question answering. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3080–3118, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.167. URL https://aclanthology.org/2025.findings-emnlp.167/.

[34] Zhehao Zhang, Ryan A. Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, Ruiyi Zhang, Jiuxiang Gu, Tyler Derr, Hongjie Chen, Junda Wu, Xiang Chen, Zichao Wang, Subrata Mitra, Nedim Lipka, Nesreen Ahmed, and Yu Wang. Personalization of large language models: A survey, 2025. URL https://arxiv.org/abs/2411.00027.

[35] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, Mar. 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Appendix

Appendix Contents

1 Introduction **1**

2 Related Work **3**

2.1 Long-term Agentic Memory 3

2.2 Time-aware Retrieval and Temporal Reasoning 3

3 Method **3**

3.1 Temporal Constraint Generation 3

3.2 Dual-channel Constrained Retrieval 5

3.3 Reward Design and GRPO Training 5

4 Experiments **6**

4.1 Datasets 6

4.2 Experiments Setup 6

4.3 Results 7

4.4 Ablation Study 8

5 Conclusion **10**

A Dataset Details **16**

A.1 LifeBench 16

A.2 LoCoMo 16

A.3 Evaluation Metrics 16

B Memory Pool Construction **17**

B.1 Date Extraction and Clustering 17

B.2 Cluster Statistics 18

B.3 Locator-Input Timeline Coverage Audit 18

C Locator Training Details **18**

C.1 Training Data Construction 18

C.2 SFT Cold-Start 19

C.3 GRPO Hyperparameters 20

C.4 Reward Weights 20

C.5 Difficulty-Stratified Rollout and R_{min} 20

C.6 Reward-Component Leave-One-Out Ablation 21

C.7 Training Curves 21

C.8 Lightweight Locator: 1.7B vs 4B 21

D Ablation Study **22**
D.1 Hyperparameter Robustness . . . 22
D.2 Memory Construction–Retrieval Factorial Comparison . . . 22
D.3 Retrieval Top-k Sweep . . . 23
D.4 Memory-Construction Hyperparameter Sweep . . . 23

E Baseline Details **24**
E.1 Method Summaries . . . 24
E.2 Excluded Baselines . . . 25

F Detailed Quantitative Results **25**
F.1 Stratified Accuracy by Category . . . 25
F.2 Paired Significance Tests . . . 25

G Diagnostic Analysis and Case Studies **25**
G.1 Error-Cause Analysis . . . 26
G.2 Case Studies . . . 26

H Computational Resources **27**
H.1 Training Cost . . . 27
H.2 Inference Latency . . . 27

I Limitations and Broader Impact **28**
I.1 Limitations . . . 28
I.2 Broader Impact . . . 28

J Prompt Templates and Judge Validation **28**
J.1 Memory Pool Construction Prompts . . . 29
J.2 Locator Prompt and Output Schema . . . 30
J.3 LLM Judge for Answer Correctness (r_{out}) . . . 32
J.4 LLM Judge for Evidence Coverage (r_{evd}) . . . 34
J.5 Error-Cause Judge Prompt . . . 35
J.6 Judge Consistency Audit . . . 38

A Dataset Details

This appendix describes the two long-horizon memory benchmarks used in our evaluation, the categories they expose, the train / evaluation splits we adopt, and the metrics under which all systems are compared.

A.1 LifeBench

LifeBench [4] is a long-horizon multi-source personalized memory benchmark designed to stress-test memory systems beyond chat-only settings. It contains synthetic year-long personal histories for ten users (`user-0` through `user-9`), with on average roughly $13{,}000$ entries per user. Each user trace integrates multi-source daily traces—multi-session chat dialogues, phone-operation logs, trip and location histories, exercise and health-tracking records, and free-form personal notes—generated through an anonymized-persona-driven simulator with explicit temporal anchoring and cross-source consistency checks, so semantically similar events recur across the year and force the memory system to disambiguate by time and context. Questions span five categories—Information Extraction (IE), Multi-hop Reasoning (MR), Nondeclarative Memory Reasoning (ND), Temporal and Knowledge Updating (TKU), and Unanswerable (UA)—summing to $2{,}003$ items in total, and are answered open-endedly under an LLM-as-judge protocol with GPT-5-mini. State-of-the-art memory systems that score around 90% on prior chat-only benchmarks (e.g., LoCoMo, LongMemEval) drop to 55% overall on LifeBench, confirming that multi-source temporal personal histories pose a substantially harder retrieval and reasoning challenge. Following the LoCoMo evaluation protocol, we exclude UA, which leaves $1{,}931$ effective questions. We evaluate on the user-0 / user-1 split (408 questions after UA exclusion) and synthesize training samples from user-2 through user-9, so the training and evaluation users on LifeBench are disjoint.

A.2 LoCoMo

LoCoMo [15] is a benchmark for very long-term conversational memory: each instance is a multi-session dialogue between two persona-grounded agents that, on average, spans 600 turns and 16K tokens across up to 32 sessions, far exceeding the five-session horizon of prior long-term-dialogue benchmarks. The dialogues are produced by a machine–human pipeline that grounds the speakers in personas and a temporal event graph, with optional image sharing for multi-modal turns and human annotators verifying long-range consistency and event-graph faithfulness. The associated QA benchmark covers five reasoning categories—single-hop, multi-hop, temporal, open-domain, and adversarial—and was designed to expose the difficulty of comprehending long-range temporal and causal dynamics within dialogues. Following prior work, we evaluate on the $1{,}540$-question subset after excluding the adversarial category. LoCoMo serves as a zero-shot generalization test: no LoCoMo conversations or questions enter our training pipeline.

A.3 Evaluation Metrics

We evaluate all systems with the official end-to-end question-answering protocol of LoCoMo [15] and LifeBench [4]: **Accuracy** and **F1**. This subsection gives the precise definitions and explains why we do not report classical retrieval-side metrics such as Recall@k as headline numbers.

Accuracy. For each question q with gold answer a^* and predicted answer $\hat{a}$, we obtain a binary correctness score $\mathrm{Acc}(q) \in \{0, 1\}$ via the official `locomo_grader` judge prompt (Appendix J.3). To suppress single-evaluation variance, we invoke the same judge prompt three independent times and adopt the majority vote ($\geq 2/3$ CORRECT) as the final label, matching the r_{out} definition used during GRPO training (Eq. 11). The judge model is GPT-5-mini on LifeBench and GPT-4o-mini on LoCoMo, fixed across all baselines for fairness; cross-judge variance is audited in Appendix J.6. Answer generation, in contrast, is dataset-specific—LifeBench follows its official `QA_PROMPT` and LoCoMo follows the Mem0-style `ANSWER_PROMPT_MEM0`, both reproduced verbatim in Appendix J.3.

F1. F1 is the token-level harmonic mean of precision and recall between the normalised prediction $\tilde{\hat{a}}$ and the normalised gold $\tilde{a^*}$, following the standard preprocessing inherited by the LoCoMo reference implementation: lowercasing, article and punctuation removal, and white-space tokenisation.

Why not Recall@k? Conventional IR metrics presuppose a canonical evidence unit shared across all systems, against which gold relevance labels can be defined uniformly. The agentic-memory baselines we compare against, as well as TIDER itself, do not consume raw conversations directly: each system ingests the raw data and emits a *method-specific* memory representation through compression, extraction, or rewriting. These resulting units differ substantially in granularity—from temporally anchored event clusters in TIDER's memory pool $\mathcal{M}$ (Eq. 1), to session-level summaries in TiMem [11] and MemoryBank [35], per-utterance dense chunks in TsContriever [28], and temporal knowledge-graph triples in Zep [21]. Consequently, top-k retrieval results are not directly comparable across systems: a top-k set of summaries covers substantially more raw evidence than a top-k set of utterances, and there is no principled normalization that aligns these heterogeneous units. Forcing a single unit, such as raw utterances, would penalize systems whose designs intentionally rewrite or condense memory, thereby conflating memory-construction choices with retrieval quality. We therefore follow the convention established by LoCoMo and LifeBench and evaluate all systems using the same downstream task performance metrics: QA accuracy and F1.

Retrieval Metrics over the Same Memory Pool. The comparability limitation above concerns systems with heterogeneous memory representations. For a controlled retrieval-side comparison, we additionally evaluate TsContriever, the semantic retriever used as NeSTR input, and TIDER on the same LifeBench memory pool $\mathcal{M}$. Because memory entries undergo extraction and rewriting, exact string matching cannot reliably align them with raw gold evidence. We therefore use an LLM coverage judge to assign binary relevance labels. Recall@k is the fraction of gold evidence covered by the top-k results, averaged over queries; nDCG@20 measures ranking quality using the binary relevance labels; and MRR@20 is the mean reciprocal rank of the first relevant memory within the top-20 results. These judge-based diagnostics supplement, rather than replace, the end-to-end QA metrics.

Table 4: Retrieval metrics on the same LifeBench memory pool $\mathcal{M}$. All values are reported as percentages and use LLM-judged binary relevance. Best results are **bolded**.

Method	Recall@10	Recall@20	nDCG@20	MRR@20
TsContriever	17.91	23.84	20.48	42.98
Semantic retriever (NeSTR input)	32.74	39.35	37.21	67.55
TIDER	**33.18**	**48.36**	**48.67**	**73.71**

Table 4 shows that TIDER improves Recall@20 and nDCG@20 over the shared semantic retriever by 9.01 and 11.46 percentage points, respectively. Recall@10 is similar, while the larger Recall@20 and MRR@20 support improved evidence coverage and earlier ranking of relevant memories within the same representation. These results are conditional on the LLM coverage labels and should not be interpreted as representation-independent retrieval scores across native agentic-memory systems.

B Memory Pool Construction

This appendix details the date extraction, clustering, and summarization steps that turn the raw dialogue $\mathcal{C}$ into the temporally anchored event memory pool $\mathcal{M}$ used by both retrieval channels; the full prompts driving these steps are listed in Appendix J.1.

B.1 Date Extraction and Clustering

Each dialogue turn is processed by an LLM that extracts atomic memory items together with their associated dates. Items are then aggregated into clusters by a temporally-bounded greedy merge: an item joins an existing cluster only when its timestamp falls within $\Delta_{\text{date}} = 7$ days of the cluster span and its semantic similarity (under the same dense retriever as the semantic channel) to the cluster centroid satisfies $\theta_{\text{sem}} \geq 0.85$; clusters are capped at 40 items each—once a cluster reaches the cap, subsequent items spawn a new cluster even if both temporal and semantic conditions hold. Each surviving cluster i records its date span $d_i = [d_i^{\text{start}}, d_i^{\text{end}}]$, an LLM-generated event summary s_i, and the raw items $\mathcal{I}_i$.

Table 5: LifeBench memory-cluster statistics. "#Items" counts raw turns absorbed into clusters; "#Clusters" is the number of surviving event clusters after temporally aware greedy merging; "Items / cluster (avg)" is the arithmetic mean cluster size; "Timeline span (days)" is the span between the earliest and latest cluster date.

User	#Items	#Clusters	Items / cluster (avg)	Timeline span (days)
`user-0`	136,208	15,603	8.73	990
`user-1`	128,538	14,847	8.66	833
`user-2`	132,862	15,369	8.64	2,726
`user-3`	131,961	15,243	8.66	823
`user-4`	121,168	14,036	8.63	7,014
`user-5`	88,335	10,656	8.29	1,308
`user-6`	123,917	14,708	8.43	1,794
`user-7`	20,683	3,136	6.60	897
`user-8`	50,944	6,509	7.83	499
`user-9`	129,014	14,705	8.77	1,071
Mean	106,363	12,481	8.52	1,795

B.2 Cluster Statistics

We report cluster statistics for all ten LifeBench users (EN). "#Items" counts raw chat turns absorbed by all clusters; "#Clusters" is the total number of LLM-summarized event clusters surviving the merge step described in Appendix B; "Items / cluster (avg)" is the arithmetic mean of cluster sizes; "Timeline span (days)" is the calendar span between the earliest and latest cluster date.

B.3 Locator-Input Timeline Coverage Audit

To check whether the clustered timeline summaries preserve the coarse temporal anchors needed by the locator, we audit the locator's actual input on the LifeBench user-0 / user-1 slice. For each question, an LLM reviewer receives the gold evidence events and the top-30 timeline summaries locator input, then judges whether those summaries contain enough coarse event-level clues to localize the evidence to the correct temporal window.

Table 6: Coarse temporal-window localization coverage of the locator's top-30 timeline-summary input on LifeBench. Coverage is judged by an LLM reviewer and averaged per question.

Category	**N**	**Top-30 localization coverage**
Non-temporal	50	87.60
Single-window	228	89.45
Dual-window	103	73.51
Multi-window	27	78.85
Overall	408	84.51

C Locator Training Details

This appendix specifies the training procedure for the locator π_θ: the SFT cold-start, the GRPO online phase, the composite-reward weighting, and the difficulty-stratified rollout.

C.1 Training Data Construction

We synthesize training samples from LifeBench user-2 through user-9. For each query q, we annotate four gold supervisions:

- `gold_temporal` $\in \{\text{true}, \text{false}\}$, indicating whether the query is time-sensitive.
- the gold window set W_{gold}, present only when `gold_temporal` $=$ true.

Table 7: GRPO hyperparameters for training the locator π_θ. The SFT-aligned Qwen3-1.7B policy is further optimized with GRPO using the configuration below.

Parameter	Value
Advantage estimator	GRPO
KL coefficient β (loss)	0.1
Group size G (rollouts per prompt)	8
Rollout temperature	0.7
Peak learning rate	5×10^{-6}
LR schedule	cosine, warmup ratio 0.1
PPO epochs per batch	1
Train / mini batch size	16 / 16
Micro batch size per GPU	8
Total epochs	6
R_{min} for unparseable rollouts	0

- the gold evidence $\mathcal{E}^* = \{e_k\}_{k=1}^{|\mathcal{E}^*|}$.
- the gold answer a^*.

Annotation provenance. The four supervisions above originate from heterogeneous sources rather than a single annotation pass.

Gold answer a^*. Each a^* is taken verbatim from LifeBench, which ships an authoritative answer for every QA instance: a single option letter (A/B/C/D) for multiple-choice questions and a short reference text for open-ended questions. No filtering or rewriting is applied.

Gold evidence $\mathcal{E}^*$. Each LifeBench QA instance carries an `evidence_resolved` field that lists one record per supporting evidence, with source identifier, type, date, and raw texts. We adopt the field verbatim and convert each record into a $(\text{date}, \text{text})$ tuple consumed by the evidence reward r_{evd}.

Gold windows W_{gold}. Derived deterministically from the dates in $\mathcal{E}^*$. Each evidence date is first cast as a degenerate single-day interval $[d, d]$; intervals are then sorted by start date and merged whenever the gap to the previous interval is fewer than three calendar days, otherwise emitted as a new window. Evidence with missing or unparseable dates is skipped. The resulting W_{gold} collapses dense evidence clusters into a single tight window and exposes sparse clusters as multi-window targets, providing the natural difficulty stratification used in Appendix C.

Temporal flag `gold_temporal`. A binary indicator of whether q is time-sensitive. We obtain an initial label by prompting GPT-5.2 to classify each query in isolation, and a human annotator subsequently verifies every label and overrides the model decision when warranted. This two-stage pipeline reduces both false positives (queries whose evidence happens to carry incidental dates) and false negatives (implicitly temporal queries whose surface form omits explicit time markers).

The resulting set contains $1{,}589$ samples spanning the four window-count buckets used by the difficulty-stratified rollout in Appendix C: 38 non-temporal ($|W_{\text{gold}}| = 0$, 2.4%), $1{,}014$ single-window ($|W_{\text{gold}}| = 1$, 63.8%), 383 dual-window ($|W_{\text{gold}}| = 2$, 24.1%), and 154 multi-window ($|W_{\text{gold}}| \geq 3$, 9.7%).

C.2 SFT Cold-Start

SFT aligns π_θ to the locator output schema (Appendix J) and seeds an initial window guess. Training data are pairs (q, z^*) derived from the gold annotations. We initialize from Qwen3-1.7B–and train for 2 epochs with peak learning rate 2×10^{-5}, per-GPU batch size 4, gradient-accumulation steps 2, and maximum sequence length 6144 tokens, on 2 GPUs; the effective batch size is therefore 16.

Table 8: Difficulty-aware rollout reweighting (DAR) by gold-window count. "Effective probability" applies the DAR factors to the empirical training distribution and renormalizes the result.

Bucket ($\lvert W_{\text{gold}}\rvert$)	DAR factor	Effective probability
0 (non-temporal)	10.0	0.159
1 (single-window)	1.0	0.424
2 (dual-window)	1.0	0.160
≥ 3 (multi-window)	4.0	0.257

C.3 GRPO Hyperparameters

For the GRPO objective in Eq. (12), the likelihood ratio for rollout j is

$$\rho^{(j)} = \frac{\pi_\theta(z^{(j)} \mid q, \mathcal{S}_q)}{\pi_{\theta_{\text{old}}}(z^{(j)} \mid q, \mathcal{S}_q)},$$

and $A^{(j)}$ is the group-normalized advantage computed from the composite rewards of the G rollouts sampled for the same query. This normalization follows GRPO: the rewards within a group are centered and scaled before clipping, while the KL term is applied against the reference policy π_{ref}.

C.4 Reward Weights

The composite reward (Eq. (7)) uses weights

$$w_{\text{det}} = 0.10, \quad w_{\text{win}} = 0.25, \quad w_{\text{evd}} = 0.20, \quad w_{\text{out}} = 0.45, \qquad \textstyle\sum_\bullet w_\bullet = 1.$$

The end-task signal w_{out} takes the largest share because answer correctness is the only term that depends jointly on detection, window prediction, and downstream synthesis; the three intermediate weights $w_{\text{det}}, w_{\text{win}}, w_{\text{evd}}$ supply dense supervision early in training, when r_{out} is still sparse. Among the intermediate signals, $w_{\text{win}} > w_{\text{evd}}$ because window-prediction errors propagate to evidence retrieval, so localizing the window is the higher-leverage upstream signal.

For r_{win} in Eq. (9), the bidirectional IoU score is computed only for temporal samples with nonempty W_{gold} and nonempty W_{pred}. Let $\mathcal{T}_{\text{win}}$ denote the case where `gold_temporal` = true, $W_{\text{gold}} \neq \emptyset$, and $W_{\text{pred}} \neq \emptyset$. The full boundary-condition form is:

$$b(W_{\text{pred}}, W_{\text{gold}}) = \tfrac{1}{2}\left(\tfrac{1}{|W_{\text{gold}}|} \sum_{G \in W_{\text{gold}}} \max_{P \in W_{\text{pred}}} \varphi(P, G) + \tfrac{1}{|W_{\text{pred}}|} \sum_{P \in W_{\text{pred}}} \max_{G \in W_{\text{gold}}} \varphi(P, G) \right),$$

$$r_{\text{win}} = \begin{cases} b(W_{\text{pred}}, W_{\text{gold}}), & \mathcal{T}_{\text{win}}, \\ 0, & \text{otherwise.} \end{cases}$$

Non-temporal samples, empty predictions, and samples whose gold evidence yields no parseable temporal window therefore receive $r_{\text{win}} = 0$. This convention avoids undefined maxima or normalizers and leaves the decision of whether the temporal channel should be opened to r_{det}.

C.5 Difficulty-Stratified Rollout and R_{min}

Rollouts are reweighted by an asymmetric U-shape difficulty-aware reweighting (DAR) schedule indexed by $|W_{\text{gold}}|$: the two extremes of the difficulty axis are up-weighted, while the easier middle buckets are kept at the base weight 1.0. The factor for non-temporal questions ($\times 10$) is much larger than that for multi-window questions ($\times 4$) because the non-temporal bucket is severely underrepresented in the training distribution (2.4% of samples; see Appendix C), whereas the multi-window bucket, while still skewed, is less so (9.7%). Concretely: The effective-probability column applies the DAR factors to the empirical training distribution reported in Appendix C (per-bucket sample counts $38/1{,}014/383/154$), then renormalizes. Rollouts whose locator output cannot be parsed by the JSON schema (Appendix J) are clamped to $R = R_{\text{min}} = 0$ and bypass the LLM judge.

C.6 Reward-Component Leave-One-Out Ablation

Table 9 reports leave-one-out ablations over the four composite-reward terms of Eq. (7). Each non-full row removes the indicated reward term during the GRPO phase while keeping the SFT cold-start, the locator backbone, the dual-channel retriever, and the downstream LLM fixed. Detection and window-IoU are removed jointly because r_{win} is gated on $r_{\text{det}}=1$.

Table 9: Reward-component leave-one-out ablations on LifeBench. The first row is full TIDER; subsequent rows remove the indicated term(s) during GRPO. Best results are **bolded**.

Variant	IE ↑	MR ↑	ND ↑	TKU ↑	Overall ↑	F1 ↑
TIDER (SFT+GRPO, 1.7B)	**67.12**	**60.83**	**60.42**	**47.83**	**61.52**	**69.23**
w/o $r_{\text{win}}, r_{\text{det}}$	60.96	56.67	58.33	32.61	55.88	67.46
w/o r_{out}	66.44	56.67	58.33	43.48	59.07	69.20
w/o r_{evd}	64.38	60.83	56.25	47.83	59.56	68.81

C.7 Training Curves

Figure 6 plots the per-step reward for three independent GRPO runs that share an identical training recipe and differ only in launch timing. Each dashed curve is one run's first 300 steps after EMA smoothing (α=0.95, bias-corrected); the solid line is the per-step mean across the runs—averaging all three for step. The three trajectories are visually superimposed during the early ascent and remain within a narrow band at the plateau, indicating that the GRPO gain reported in §4.3 is reproducible rather than an artifact of a single training trajectory.

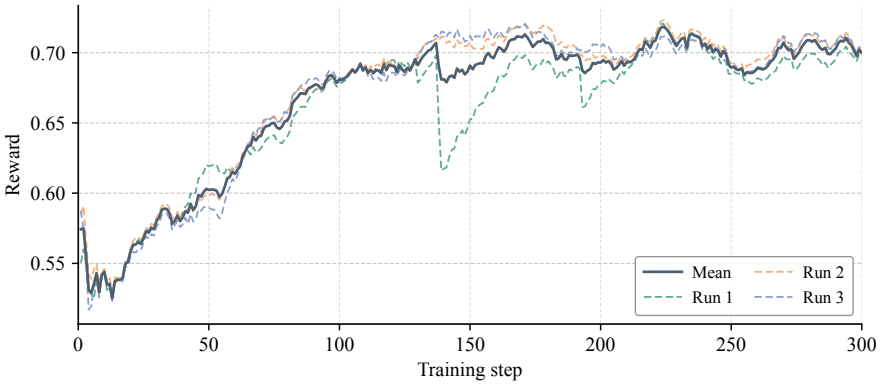


Figure 6: GRPO training curves of the 1.7B locator on LifeBench.

C.8 Lightweight Locator: 1.7B vs 4B

Locator capacity is not the source of TIDER's gains. We re-ran the full TIDER pipeline with a 4B locator under an identical training recipe; only the SFT cold-start and GRPO traces are model-specific, while the dual-channel retriever, prompts, and downstream LLM are unchanged. Figure 7 overlays the smoothed reward of the 4B run on the 1.7B run from Figure 6: the two trajectories rise on a similar shape and converge to plateaus differing by less than 0.02 in reward, with the 4B run ascending marginally faster in the early phase. On the evaluation side, Table 10 reports end-to-end accuracy on LifeBench. Halving the locator from 4B to 1.7B parameters leaves Overall accuracy and average F1 effectively unchanged (61.52 vs 61.03; 0.6923 vs 0.6957), with per-category swings within $\pm$2.2 pp. We therefore adopt the 1.7B locator as the default, attributing TIDER's headroom to the dual-channel temporal anchor rather than to locator scale. TIDER thus inherits a lightweight characteristic, achieving competitive end-to-end accuracy under a substantially smaller locator parameter budget.

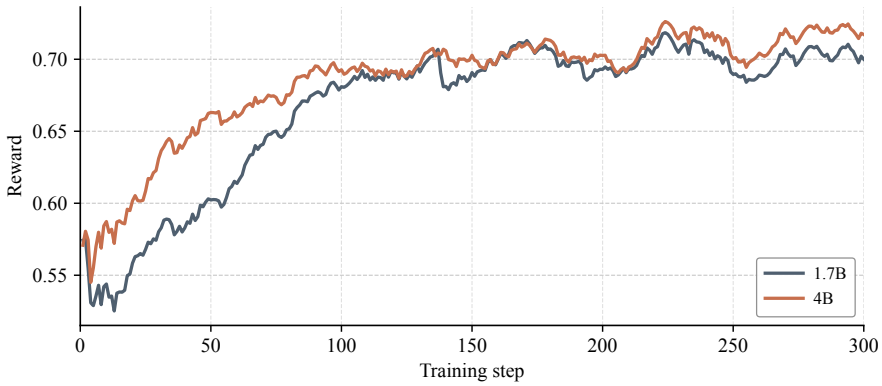


Figure 7: Capacity ablation on the GRPO training curve. Per-step reward (EMA α=0.95, bias-corrected) for the 4B locator and the 1.7B.

Table 10: Locator capacity ablation on LifeBench (user-0/1, n=408). Per-category accuracy is reported alongside Overall accuracy and average F1. Per-column best results are **bolded**; the 1.7B locator is our adopted default.

Locator	IE ↑	MR ↑	ND ↑	TKU ↑	Overall ↑	F1 ↑
TIDER (4B)	65.07	**61.67**	59.38	**50.00**	61.03	**69.57**
TIDER (1.7B)	**67.12**	60.83	**60.42**	47.83	**61.52**	69.23
Δ (1.7B−4B)	+2.05	−0.84	+1.04	−2.17	+0.49	−0.34

D Ablation Study

D.1 Hyperparameter Robustness

This subsection reports the full hyperparameter sweep referenced in §4.4. Across the 25 configurations summarized in Fig. 8, Overall accuracy stays within a narrow $[57.73, 61.52]\%$ band, far above the No-Temporal bound (52.21%). Per-hyperparameter ranges are: λ_{boost} (3.86%, most sensitive), top-K timeline entries (3.14%), decay rate ρ (3.14%), floor $\gamma_{\min}$ (2.10%), buffer τ_{buf} (1.86%, most stable). The chosen defaults sit at or near each per-panel optimum. We perform hyperparameter tuning on the User3–9 subset, use User2 as the validation set, and evaluate the model on User0 and User1 as the test set.

D.2 Memory Construction–Retrieval Factorial Comparison

To separate the effects of memory construction and retrieval, we cross two memory representations with two retrieval methods on LifeBench. The fixed-chunk representation uses 512-token chunks with timestamps inherited from their source sessions, without LLM-based extraction, clustering, rewriting, or summarization; the locator directly consumes the timestamped chunks. The full representation uses the temporally anchored memory pool $\mathcal{M}$. Within each representation, we compare semantic-only retrieval with TIDER retrieval while holding the downstream answer model and retrieval budget fixed.

Table 11: A 2×2 factorial comparison of memory representation and retrieval method on LifeBench. Values are Overall accuracy (%).

Memory representation	Semantic retrieval	TIDER retrieval
Fixed chunks + session timestamps	50.49	58.58
Full temporally anchored memory pool $\mathcal{M}$	52.21	61.52

TIDER improves accuracy by 8.09 percentage points on fixed chunks and 9.31 points on the full memory pool. Memory construction adds 1.72 points under semantic retrieval and 2.94 points under

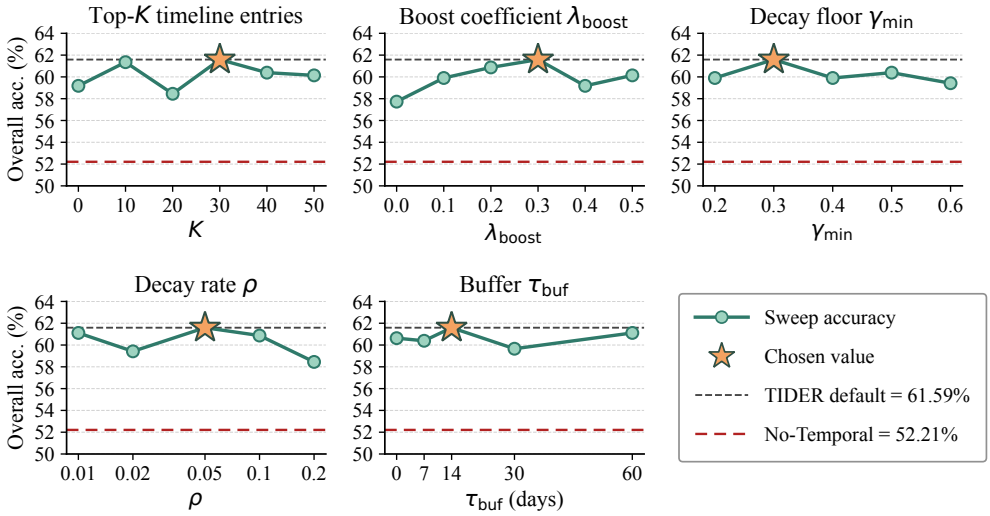


Figure 8: Hyperparameter robustness on LifeBench. Five hyperparameters are swept individually around the default configuration (K=30, λ_{boost}=0.3, $\gamma_{\min}$=0.3, ρ=0.05, τ_{buf}=14). Each panel shows Overall accuracy (%) versus the swept value; the orange star marks the chosen default; the upper dashed line is TIDER's default-config score; the lower dotted line is the No-Temporal lower bound.

TIDER retrieval. The retrieval-time mechanism therefore remains effective without LLM-based memory construction, while the full memory pool provides an additional, retrieval-dependent benefit.

D.3 Retrieval Top-k Sweep

We vary the number of retrieved memory units passed to the answer generator, k, while fixing the memory pool, answer generator, and all other settings. This retrieval budget differs from the top-K timeline-input size in Appendix D.1, which controls the information supplied to the locator.

Table 12: Retrieval-budget sweep on LifeBench with the same memory pool and answer generator. Values are Overall accuracy (%).

Retrieval budget k	10	20	30	40	50	100
TIDER	52.70	61.52	60.78	58.33	58.58	62.01
Semantic Only	46.86	52.21	52.90	57.49	56.04	57.73

Accuracy varies non-monotonically with k. Increasing the semantic retrieval budget narrows the gap, but the best semantic-only result within the tested range is 57.73% at $k = 100$, still 3.79 percentage points below TIDER at $k = 20$ (61.52%). Increasing TIDER's budget from 20 to 100 improves accuracy by only 0.49 points. These results support $k = 20$ as an effective operating point: TIDER achieves higher accuracy than the best tested semantic-only setting while retrieving one fifth as many memory units. This comparison concerns memory-unit counts, not measured token costs or end-to-end latency.

D.4 Memory-Construction Hyperparameter Sweep

We vary the date-merge tolerance Δ_{date}, semantic merge threshold θ_{sem}, cluster-size cap, and summarizer model individually, keeping the other settings at their defaults. The default construction uses $\Delta_{\text{date}} = 7$ days, $\theta_{\text{sem}} = 0.85$, a cap of 40 items, and GPT-4o-mini as the summarizer. Table 13 reports the corresponding LifeBench Overall accuracies.

Accuracy ranges from 59.07% to 61.76% for Δ_{date}, 59.56% to 61.52% for θ_{sem}, 57.84% to 61.52% for the cap, and 60.29% to 63.73% across summarizer models. All tested configurations remain above the default full-pool semantic-only accuracy of 52.21%. This comparison supports robustness relative to that reference operating point.

Table 13: Memory-construction sensitivity on LifeBench. Settings and accuracies in each row correspond in order; accuracy is reported in percent.

Parameter	Settings	Accuracy (%)
Δ_{date} (days)	3 / 7 / 14	61.76 / 61.52 / 59.07
θ_{sem}	0.75 / 0.85 / 0.95	59.56 / 61.52 / 61.27
Cluster-size cap	20 / 40 / 80	57.84 / 61.52 / 60.54
Summarizer	GPT-4.1-nano / GPT-4o-mini / GPT-5-mini	60.29 / 61.52 / 63.73

E Baseline Details

This appendix expands the one-line baseline descriptions in §4.2 and details the implementation choices that ensure a fair comparison.

E.1 Method Summaries

We group the six baselines into three families along their orthogonal design dimensions: agentic memory frameworks (MemOS, TiMem, Zep, Mem0) that target memory organization and lifecycle management, time-aware retrieval (TsContriever) that targets the retrieval encoder, and temporal reasoning (NeSTR) that targets answer-time inference over retrieved evidence. The summaries below paraphrase each paper's stated contribution; the closing line marks what we preserve from the original method versus what we adapt to keep the comparison on equal footing with TIDER.

MemOS [12] treats memory as a manageable system resource and proposes a memory operating system that unifies the representation, scheduling, and evolution of plaintext, activation-based, and parameter-level memories. Its basic unit, the MemCube, encapsulates memory content together with provenance and versioning metadata; MemCubes can be composed, migrated, and fused over time, bridging retrieval-style memory with parameter-based learning under explicit lifecycle control. *Adapted:* the downstream answer model is replaced by our shared LLM (GPT-5-mini on LifeBench, GPT-4o-mini on LoCoMo) so that gains attributable to memory organization are isolated from the choice of generator.

TiMem [11] is a temporal–hierarchical memory framework that organizes conversations through a Temporal Memory Tree (TMT), enabling systematic memory consolidation from raw observations to progressively abstracted persona representations. It is characterized by three properties: temporal–hierarchical organization through TMT, semantic-guided consolidation that integrates memory across hierarchical levels without fine-tuning, and complexity-aware memory recall that balances precision and efficiency across queries of varying complexity. *Adapted:* the downstream answer LLM is unified with the rest of the table, leaving TiMem's memory tree as the only varying component.

Zep [21] is a memory-layer service for AI agents whose core component, Graphiti, dynamically synthesizes unstructured conversational data and structured business data into a temporally-aware knowledge graph that maintains historical relationships with explicit validity intervals. Zep was originally evaluated on Deep Memory Retrieval (DMR) and LongMemEval, where it improves over MemGPT and shows substantial gains on cross-session synthesis and long-term context maintenance with reduced response latency. *Adapted:* the answer model is unified across baselines.

Mem0 [5] is a scalable memory-centric architecture that dynamically extracts, consolidates, and retrieves salient information from ongoing multi-session conversations, with an enhanced graph-based variant that captures relational structure among conversational elements. On the LOCOMO benchmark Mem0 reports a 26% relative improvement over an OpenAI memory baseline under LLM-as-Judge while reducing p95 latency by 91% and saving more than 90% of token cost. *Adapted:* the downstream answer LLM and top-k retrieval budget are unified.

TsContriever [28] (a.k.a. TS-Retriever) is a time-sensitive retrieval-augmented generation method built on top of Contriever. It augments dense retrieval with supervised contrastive learning using temporally-constrained negatives, performs query-side fine-tuning to make the encoder sensitive to explicit temporal expressions, and routes queries through a temporal-need classifier so that general queries fall back to standard semantic retrieval. The original paper reports clear gains on a custom time-sensitive QA benchmark while preserving general-query effectiveness. *Adapted:* We preserve the four official checkpoints (plain, document, query, router) and the routing classifier. We disable

the TIDER locator and plug TsContriever into our shared memory pool, retrieval pipeline, top-k, and answer LLM, so the difference reflects the encoder change alone.

NeSTR [13] is a Neuro-Symbolic Temporal Reasoning framework that integrates structured symbolic representations of temporal relations with hybrid reflective reasoning to enhance LLMs' temporal sensitivity. It explicitly preserves temporal relations through symbolic encoding, enforces logical consistency through verification, and corrects flawed inferences via abductive reflection, achieving zero-shot gains on temporal QA benchmarks without any fine-tuning. *Adapted:* We preserve the NeSTR for response generation, including symbolic temporal encoding, consistency verification, and abductive reflection. We disable TIDER's locator, temporal grounding and temporal reranker at the search stage, share the memory pool and dense retriever.

E.2 Excluded Baselines

We do not include training-based temporal-reasoning methods such as Memory-T1 [6] as direct baselines, since they fine-tune the response model whereas our evaluation keeps the downstream LLM fixed across methods.

F Detailed Quantitative Results

This appendix reports the detailed quantitative evidence behind the LifeBench comparisons: accuracy stratified by gold-window-count category (§F.1) and paired significance tests on per-question correctness (§F.2).

F.1 Stratified Accuracy by Category

Table 14 reports per-category accuracy of all seven baselines on the official LifeBench user-0 / user-1 slice (n=408), using the same four categories visualized in Fig. 3: non-temporal (n=50), single-window (n=228), dual-window (n=103), and multi-window (n=27). Means and standard deviations are reported over three independent evaluation runs.

Table 14: LifeBench accuracy stratified by gold-window-count category. Best results are **bolded**; underline indicates second best.

Method	Non-temporal	Single-window	Dual-window	Multi-window	Overall
MemOS	52.67±1.15	48.83±1.10	42.07±1.48	44.44±11.11	47.30±0.88
TiMem	23.33±1.15	16.96±1.10	17.80±1.48	13.58±4.28	17.73±0.57
Zep	48.67±7.02	38.74±0.67	33.98±1.68	33.33±0.00	38.40±0.37
Mem0	63.33±3.06	51.32±1.16	47.25±4.04	51.85±3.70	51.80±1.26
TsContriever	38.00±2.00	28.07±1.16	23.62±1.12	13.58±4.28	27.21±0.42
NeSTR	56.00±2.00	55.12±0.67	46.60±1.68	49.38±2.14	52.70±0.65
TIDER	**64.00**±2.00	**64.47**±1.16	**52.43**±1.12	**66.67**±6.42	**61.52**±0.86

F.2 Paired Significance Tests

Table 15 reports paired significance tests comparing TIDER against each baseline on the same 408 answerable LifeBench questions. We use the two-sided exact McNemar test on paired binary correctness outcomes and apply Holm correction across the six TIDER-versus-baseline comparisons. In the table, b counts questions answered correctly by TIDER but incorrectly by the baseline, while c counts the reverse direction.

All TIDER-versus-baseline differences remain significant after Holm correction, including the closest comparisons against Mem0 and NeSTR.

G Diagnostic Analysis and Case Studies

This appendix analyzes why methods fail on LifeBench. We first summarize LLM-judged error causes over incorrect answers (§G.1), then provide qualitative case studies covering single-, dual-, and multi-window queries (§G.2).

Table 15: Paired significance tests for TIDER against each baseline on LifeBench. ΔAcc is reported in percentage points.

Comparison	ΔAcc	b	c	Exact p	Holm p
TIDER vs. MemOS	13.48	92	37	1.40×10^{-6}	4.19×10^{-6}
TIDER vs. TiMem	43.14	188	12	8.11×10^{-42}	4.87×10^{-41}
TIDER vs. Zep	22.79	121	28	5.59×10^{-15}	2.24×10^{-14}
TIDER vs. Mem0	8.33	66	32	7.70×10^{-4}	8.94×10^{-4}
TIDER vs. TsContriever	34.07	149	10	6.25×10^{-33}	3.12×10^{-32}
TIDER vs. NeSTR	8.09	59	26	4.47×10^{-4}	8.94×10^{-4}

G.1 Error-Cause Analysis

For each incorrect answer, we ask an LLM reviewer to assign exactly one primary error cause by comparing the question, gold answer, model answer, gold evidence, and retrieved context (Appendix J.5). The labels are: *retrieval miss*, where the answer-bearing evidence is absent; *temporal mismatch*, where related evidence is retrieved from the wrong time; *partial evidence*, where only part of the required evidence is retrieved; and *reason failure*, where sufficient evidence is present but the answer model reasons or selects incorrectly. We define *retrieval-side failure* as the union of retrieval miss, temporal mismatch, and partial evidence.

Table 16: LLM-judged primary error causes over incorrect LifeBench answers. Percentages are computed within each method's incorrect answers.

Method	Wrong	Retrieval miss	Temporal mismatch	Partial evidence	Reason failure	Retrieval-side failure
TIDER	157	27 (17.2%)	107 (68.2%)	5 (3.2%)	18 (11.5%)	139 (88.5%)
MemOS	212	41 (19.3%)	132 (62.3%)	13 (6.1%)	26 (12.3%)	186 (87.7%)
TiMem	333	174 (52.3%)	142 (42.6%)	7 (2.1%)	10 (3.0%)	323 (97.0%)
Zep	250	88 (35.2%)	115 (46.0%)	18 (7.2%)	29 (11.6%)	221 (88.4%)
Mem0	191	45 (23.6%)	123 (64.4%)	9 (4.7%)	14 (7.3%)	177 (92.7%)
TsContriever	296	143 (48.3%)	122 (41.2%)	8 (2.7%)	23 (7.8%)	273 (92.2%)
NeSTR	190	38 (20.0%)	119 (62.6%)	2 (1.1%)	31 (16.3%)	159 (83.7%)

Table 16 shows that most failures are retrieval-side rather than pure answer-generation failures. Temporal mismatch is the dominant failure type for MemOS, Mem0, NeSTR, and TIDER, while TiMem and TsContriever still exhibit many retrieval misses, indicating that their retrievers often fail before temporal disambiguation can help. Overall, the diagnosis supports the paper's central claim: long-horizon memory QA is bottlenecked by retrieving the right evidence from the right time, not merely by answer-time reasoning.

G.2 Case Studies

We summarize the recurring baseline failure modes observed across cases and then walk through three representative cases covering single-, dual-, and multi-window queries.

Common baseline failure modes. Baseline mistakes fall into five recurring patterns: (i) *topic shell only*—retrieved context preserves the right person and broad theme but loses the specific event core (e.g., MemOS, Zep on `q100`, `q181`); (ii) *event substitution with temporal drift*—a same-person event from a nearby month replaces the queried one (e.g., TiMem and Mem0 on `q100`, `q181`); (iii) *long-term persona suppression*—high-frequency identity facts override the question's specific temporal window (e.g., MemOS and TsContriever on `q143`); (iv) *single-endpoint capture*—one of two required temporal anchors is retrieved while the other is dropped (e.g., MemOS, NeSTR on `q120`, `q193`); and (v) *near-total miss*—retrieval finds little relevant content (e.g., TsContriever on `q100`, `q164`). The three walk-throughs below illustrate (i)–(iv); failure mode (v) is largely confined to TsContriever and is consistent with its low overall accuracy in Table 14.

Case 1: same-entity multi-event disambiguation (`user0/q100`). The question asks what trouble Huang Wenhui ran into in March 2025; the gold answer is that her pre-sold apartment project halted construction amid rumors that the developer's cash flow had broken down. Within the same

conversational thread, Huang Wenhui appears in at least five negatively-toned events spread across the year—a January cafe scam (q181), the March apartment halt, an April game-project conflict (q28), April/July neighbor-dog disturbances (q98, q37), and a September app-launch slump (q49)—so the challenge is isolating the March incident from this same-entity cluster. TIDER retrieves the March 12 messages that explicitly mention the halted construction site, the broken developer cash flow, and the continuing mortgage pressure. MemOS and Zep keep only the topic shell ("A-Hui has a housing problem"); TiMem replaces the event with a later apartment-water-leak story; Mem0 drifts to the July neighbor-dog incident; NeSTR preserves the negative-affect shell but slides to the January cafe scam. All baselines identify the correct entity, but only TIDER recovers the correct month-event pair.

Case 2: dual-window endpoint coverage (user1/q120). The question asks for the user's core activities on the first weekend after moving (June 14–15). The gold answer requires a complete two-day chain: a park walk and a housewarming dinner on day one, then a Pilates session and a pottery-class breakthrough on day two. TIDER retrieves evidence for all four components, with both days' messages co-occurring in its retrieved context, and answers the chain completely. MemOS retains the park-walk fragment but drops the second-day activities; TiMem and Zep pull in broadly related "new-home" or "weekend" memories without preserving the precise two-day structure; NeSTR captures the park and pottery pieces but misses Pilates. The challenge here is not topical relevance but preserving every answer-bearing endpoint required by a temporally compositional question.

Case 3: stage-level multi-window summarization (user1/q143). The question asks what the user's main work content was at Zhengzhou Central Hospital between August 11 and August 27; the gold answer is a respiratory-medicine rotation, including community-acquired pneumonia admissions, a COPD exacerbation case, a difficult-case discussion meeting, and the associated discharge documentation. TIDER's retrieved context anchors the August 11 admission and initial-course record, the August 21 case-discussion presentation, and the August 27 differential-diagnosis preparation, and additionally surfaces a "reviewing the respiratory medicine section" note. MemOS, Zep, and TsContriever all collapse back to the user's long-running cardiology persona (resident physician, cardiology rounds, cardiology case work), missing the temporary respiratory rotation entirely; TiMem fails to retrieve a coherent August 11–27 chain; Mem0 surfaces individual clinical actions but drops the rotation-stage label. This case illustrates how high-frequency persona evidence can override a specific time-bounded stage, and how TIDER's window-anchored retrieval resists this drift.

H Computational Resources

H.1 Training Cost

We separate the SFT cold-start (Appendix C.2) from the GRPO main loop (Appendix C.3). All training uses NVIDIA A100 80GB GPUs and the veRL framework. The SFT cold-start consumes about 3 wall-clock hours on 2×A100 80GB; a single GRPO run trains for 300 steps in roughly 100 hours on 2×A100 80GB.

H.2 Inference Latency

We measure the marginal latency introduced by TIDER's temporal channel on the LifeBench-EN test split, holding the GPT-5-mini answer step fixed and toggling only the temporal pathway. The 1.7B locator is served on a single A100 80GB via vLLM. Averaged across 408 questions, enabling the temporal channel adds 0.41s of wall-clock latency per query relative to the no-temporal baseline; this overhead aggregates the locator forward pass, the timeline-summary retrieval branch, and the distance-decay rerank that fuses the two channels. The figure is well below the cost of the downstream answer call, which dominates end-to-end response time at the several-second scale, so the dual-channel design recovers the accuracy gap reported in Section 4 without bottlenecking the GPT-5-mini / GPT-4o-mini answer step.

I Limitations and Broader Impact

I.1 Limitations

We discuss three limitations of TIDER that are visible from the current setup and shape the most natural next steps; each isolates a distinct failure axis (data realism, upstream memory dependency, language coverage) so that the relative weight of remediation effort is clear to the reader.

Synthetic personal histories. LifeBench is generated by an anonymized-persona-driven simulator with explicit temporal anchoring and cross-source consistency checks, and LoCoMo is multi-session synthetic dialogue rather than real long-term user data. Real personal traces likely contain heavier noise, more frequent timestamp gaps, and stronger cross-source contradictions than either benchmark exposes. Validating TIDER on a real, consented personal-memory corpus—under the privacy and retention safeguards discussed in Appendix I.2—therefore remains an open direction, and the gains we report should be read as upper bounds achievable when the temporal signal is well-formed.

Upstream memory-item dependency. TIDER's temporal channel operates on a query-adaptive timeline summary $\mathcal{S}_q$ drawn from clustered memory items in the memory pool $\mathcal{M}$ (Appendix B). We deliberately keep this construction as simple as possible—temporally bounded clustering followed by a single-pass summarization prompt—so that the locator does not couple to a bespoke memory representation; even so, $\mathcal{S}_q$ still inherits the quality of upstream item extraction, since mis-dated, mis-attributed, or fragmented items propagate into the cluster boundaries and into the resulting summary. The locator can only filter on this view, so its window predictions degrade whenever the upstream memory pool is noisy.

Language scope. LifeBench and LoCoMo are both English-only, so TIDER's evaluation does not cover languages whose date conventions diverge from the Gregorian numerical form (e.g., Chinese lunar dates, Japanese era names) or whose temporal expressions carry richer morphological cues. The locator prompt and the gold-window construction both rely on the parser shipped with our extractor, which is calibrated for English; cross-lingual deployment will likely require multilingual date normalization upstream of the temporal channel before the locator's window predictions become meaningful.

I.2 Broader Impact

TIDER targets long-term personalized assistants whose factual grounding rests on a temporally anchored memory pool; we discuss its societal impact along three positive axes and two risk axes, and note one experimental safeguard. On the positive side: (i) the dual-channel design strengthens the reliability of long-term personal-assistant deployments that must reason over months or years of dialogue history; (ii) by filtering candidates against query-relevant temporal windows rather than reweighting them after generation, the temporal channel directly reduces cross-time evidence hallucinations at the retrieval stage; and (iii) the locator's window predictions W_{pred} are user-inspectable, exposing which time region the system treated as relevant and giving end users an auditable handle on retrieval decisions. Two principal risks remain: (i) an adversary who can write into the dialogue stream may craft contradictory or fabricated dates that push the locator toward an incorrect window, thereby suppressing or surfacing specific memories at retrieval time; and (ii) any bias in the upstream LLM-based extractor—including event misattribution along demographic or linguistic lines—propagates directly into the temporally bounded clusters and the resulting timeline summary $\mathcal{S}_q$, so locator decisions inherit those biases without an independent corrective step. As an experimental safeguard, all training and evaluation in this paper use synthetic benchmarks (LifeBench, LoCoMo); no real consented personal-memory corpus was used at any stage.

J Prompt Templates and Judge Validation

This appendix lists the prompts used by the memory pool builder, the locator, and the two LLM judges in the exact form fed to the underlying LLMs (§J.1–§J.4), and reports a cross-judge consistency audit (§J.6).

J.1 Memory Pool Construction Prompts

The two prompts that drive memory-pool construction are listed below.

Cluster summarization (`REORGANIZE_PROMPT`). Used to merge the items in a temporally bounded cluster (or sub-cluster) into a single chronologically-ordered event summary, with date extraction and entity disambiguation handled inline.

Cluster summarization prompt

```
You are a memory clustering and summarization expert.

Given the following child memory items:

{memory_items_text}

Please perform:
1. Identify information that reflects user's experiences, beliefs,
concerns, decisions, plans, or reactions -- including meaningful
input from assistant that user acknowledged or responded to.
2. Resolve all time, person, and event references clearly:
- Convert relative time expressions (e.g., "yesterday," "next Friday")
into absolute dates using the message timestamp if possible.
- Clearly distinguish between event time and message time.
- If uncertainty exists, state it explicitly (e.g., "around June
2025," "exact date unclear").
- Include specific locations if mentioned.
- Resolve all pronouns, aliases, and ambiguous references into full
names or identities.
- Disambiguate people with the same name if applicable.
3. This summary is a navigation aid for retrieval, not canonical
evidence. You may use third-person for clarity, but do not rewrite
into a polished narrative. Do not collapse distinct event instances
across different dates into one fact. Do not introduce any detail
not present in the child items.
4. Do not omit any information that user is likely to remember.
- Include all key experiences, thoughts, emotional responses, and
plans -- even if they seem minor.
- Prioritize completeness and fidelity over conciseness.
- Do not generalize or skip details that could be personally
meaningful to user.
5. Summarize all child memory items into one memory item.

Output format requirements:
- The value field must be a natural language paragraph, describing
events in chronological order.
- Absolutely forbidden to output JSON arrays, lists, or structured data
formats (e.g., [{...}, {...}]).
- Each independent event starts with "YYYY-MM-DD," followed by a
sentence. Sentences are separated by periods.

Language rules:
- The `key`, `value`, `tags`, `summary` fields must match the mostly
used language of the input memory items.
- Keep `memory_type` in English.

Return valid JSON:
{
    "key": <string, a concise title of the `value` field>,
    "memory_type": <string, "LongTermMemory" or "UserMemory">,
    "value": <natural language paragraph describing events in
    chronological order; preserve dates, amounts, names,
    locations; do not collapse different event instances into
    one abstract statement; do not introduce facts not present
    in the child items>,
    "tags": <list of relevant thematic keywords>,
```

```
    "summary": <natural paragraph from user's perspective, 120-200 words,
    same language as the input>
}
```

Sub-cluster split. Used to subdivide an oversized cluster into 2–10-item sub-clusters before summarization.

Sub-cluster split prompt

```
You are a memory organization expert.

You are given a cluster of memory items, each with an ID and content.
Your task is to divide these into smaller, semantically meaningful
sub-clusters.

Instructions:
- Identify natural topics by analyzing common time, place, people, and
event elements.
- Each sub-cluster must reflect a coherent theme that helps retrieval.
- Each sub-cluster should have 2-10 items. Discard singletons.
- Each item ID must appear in exactly one sub-cluster or be discarded.
No duplicates are allowed.
- All IDs in the output must be from the provided Memory items.
- Return strictly valid JSON only.

Example: If you have items about a project across multiple phases,
group them by milestone, team, or event.

Language rules:
- The 'key' fields must match the mostly used language of the
clustered memories.

Return valid JSON:
{
    "clusters": [
    {"ids": ["<id1>", "<id2>", ...], "key": "<concise memory title>"},
    ...
    ]
}

Memory items:
{joined_scene}
```

J.2 Locator Prompt and Output Schema

The locator π_θ emits a single JSON object with two fields per query: a temporal-sensitivity flag and a list of temporal windows.

Locator output schema

```
{
    "detect": true | false,
    "windows": [
    {"start": "YYYY-MM-DD", "end": "YYYY-MM-DD"}, ...
    ]
}
```

The `detect` field follows the temporal-sensitivity definition listed below, which is the prompt fed to GPT-5.2 to label `gold_temporal` during training-data construction (Appendix C); π_θ is trained to imitate this judgment. The prompt text itself uses the `detect` key so that the annotation format matches the locator schema. Window prediction is supervised directly from (q, W_{gold}) pairs and does not require a separate LLM prompt template at data-construction time.

Temporal-sensitivity detection prompt

```
You are a temporal constraint detector. Determine whether the user's
question requires searching memories within a specific time range.

## What is a temporal constraint
A temporal constraint limits the time range for memory retrieval. Two
types:
- Explicit: directly mentions time, e.g. "in June 2025", "last month",
"during Spring Festival", "in the last 3 days"
- Implicit: indirectly limits time via events, e.g. "after mom got
sick", "before moving", "after that business trip"
- Event-pair (time gap): asks how long between two events -- both
events are implicit constraints; extract the events, NOT the time-gap
word ("how long", "how many days")

## What is NOT a temporal constraint (important)
- Factual questions: "When is the birthday", "What time do you wake up"
-- asking for information, not limiting search range
- Pure semantic questions: "favorite restaurant", "how do you commute"
- Dates in multiple-choice options (e.g. "A. Jan 6  B. Jan 8") are
answer choices, not search constraints
- Time words that don't change meaning if removed: "the restaurant
mentioned before" -- "before" is casual speech, not a constraint
- Time-gap words are NOT expressions: "how many days", "how long",
"how far in advance" are the answer target -- do NOT put them in
expressions

## Expression extraction rules
1. Every temporal expression must be extracted separately -- do not
omit any.
2. Explicit dates (e.g. "June 2025", "April 26", "Spring Festival")
must never be omitted, even when other expressions like "recently"
are also present.
3. Do NOT merge time words with event content -- "recently changed
jobs" yields "recently", not "recently changed jobs".

## Decision method
Ask yourself: does answering this question require limiting the search
to a specific time period? If not, it's false.

## User question
${query}

Output strict JSON only (no other text):
{
    "detect": true/false,
    "expressions": []
}
Fill expressions with the original temporal expressions extracted from
the query. Empty array if no constraint.

## Examples
### Positive (explicit)
Input: In May 2025, what did Zhang Ming's cat eat?
Output: {"detect": true,
    "expressions": ["In May 2025"]}

### Positive (multiple temporal expressions)
Input: In June 2025, Zhang Ming mentioned on the phone that he
recently changed jobs. What was the new job?
Output: {"detect": true,
    "expressions": ["In June 2025", "recently"]}

### Positive (implicit)
Input: What pets did he get after moving?
Output: {"detect": true,
```

```
    "expressions": ["after moving"]}

### Negative (factual question)
Input: What is Zhang Ming's favorite restaurant?
Output: {"detect": false, "expressions": []}

### Negative (dates in options)
Input: When did Zhang Ming go to the hospital? A. Jan 6 B. Jan 8
C. Jan 10
Output: {"detect": false, "expressions": []}

### Positive (event anchors, not time words)
Input: How many days before checking in did she book the guesthouse?
Output: {"detect": true,
    "expressions": ["book the guesthouse", "checking in"]}

### Positive (event-pair time gap)
Input: How many days passed between Zhang Ming signing the lease and
actually moving in?
Output:
{
"detect": true,
"expressions": ["signing the lease", "actually moving in"]
}
```

J.3 LLM Judge for Answer Correctness (r_{out})

Answer generation and answer judging are kept separate. The downstream answer is generated by a frozen LLM, but the answer-generation prompt is dataset-specific: following the LifeBench paper's official evaluation requirement, LifeBench uses the official `QA_PROMPT` from the LoCoMo reference implementation, whereas LoCoMo uses the Mem0-style `ANSWER_PROMPT_MEMO`. In both datasets, answer correctness is judged with the official LoCoMo prompt reproduced below; we invoke the same judge prompt independently three times and use majority vote (CORRECT in at least two of three calls) as the final label.

LifeBench answer generation. For LifeBench, the retrieved context is prepended before this prompt suffix.

LifeBench answer generation prompt

```
Based on the above context, write an answer in the form of a short
phrase for the following question. Answer with exact words from the
context whenever possible.

Question: {} Short answer:
```

LoCoMo answer generation. The following Mem0-style answer prompt is used for LoCoMo, not for LifeBench.

LoCoMo answer generation prompt

```
You are an intelligent memory assistant tasked with retrieving accurate
information from conversation memories.

# CONTEXT:
You have access to memories from two speakers in a conversation. These
memories contain timestamped information that may be relevant to
answering the question.

# INSTRUCTIONS:
1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
```

```
3. If the question asks about a specific event or fact, look for
direct evidence in the memories
4. If the memories contain contradictory information, prioritize the
most recent memory
5. If there is a question about time references (like "last year",
"two months ago", etc.), calculate the actual date based on the
memory timestamp. For example, if a memory from 4 May 2022 mentions
"went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months,
or years. For example, convert "last year" to "2022" or "two months
ago" to "March 2023" based on the memory timestamp. Ignore the
reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do
not confuse character names mentioned in memories with the actual
users who created those memories.
8. The answer should be less than 5-6 words.

# APPROACH (Think step by step):
1. First, examine all memories that contain information related to
the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events
that answer the question
4. If the answer requires calculation (e.g., converting relative time
references), show your work
5. Formulate a precise, concise answer based solely on the evidence
in the memories
6. Double-check that your answer directly addresses the question
asked
7. Ensure your final answer is specific and avoids vague time
references

{context}

Question: {question}

Answer:
```

Answer-correctness judge.

Answer-correctness judge prompt

```
[system]
You are an expert grader that determines if answers to questions match
a gold standard answer

[user]
Your task is to label an answer to a question as 'CORRECT' or 'WRONG'.
You will be given the following data:
(1) a question (posed by one user to another user),
(2) a 'gold' (ground truth) answer,
(3) a generated answer
which you will score as CORRECT/WRONG.

The point of the question is to ask about something one user should
know about the other user based on their prior conversations.
The gold answer will usually be a concise and short answer that
includes the referenced topic, for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous
with your grading - as long as it touches on the same topic as the
gold answer, it should be counted as CORRECT.

For time related questions, the gold answer will be a specific date,
month, year, etc. The generated answer might be much longer or use
```

```
relative time references (like "last Tuesday" or "next month"), but
you should be generous with your grading - as long as it refers to
the same date or time period as the gold answer, it should be counted
as CORRECT. Even if the format differs (e.g., "May 7th" vs "7 May"),
consider it CORRECT if it's the same date.

Now it's time for the real question:
Question: {question}
Gold answer: {gold_answer}
Generated answer: {response}

First, provide a short (one sentence) explanation of your reasoning,
then finish with CORRECT or WRONG.
Do NOT include both CORRECT and WRONG in your response, or it will
break the evaluation script.

Just return the label CORRECT or WRONG in a json format with the key
as "label".
```

J.4 LLM Judge for Evidence Coverage (r_{evd})

The evidence-coverage judge r_{evd} is a reward that measures item-level evidence recall: for each question, the reviewer independently checks every gold evidence item and returns $r_{\text{evd}} = m/n$, where n is the number of gold evidence items and m is the number of items explicitly covered by the retrieved memory set $\hat{\mathcal{E}}$. The reward is therefore a floating-point value in $[0, 1]$.

Evidence-coverage judge.

Evidence-coverage judge prompt

```
[system]
You are a strict evidence-coverage reviewer.

Given a list of gold evidence facts and retrieved memories, decide
what fraction of gold evidence facts are covered by the retrieved memories.

Coverage criteria (strict):
- Each gold evidence fact is judged independently.
- A gold fact is covered only if at least one retrieved memory
explicitly mentions semantically identical content.
- The retrieved text must mention the gold fact's specific person
names, places, events, numbers, dates, and key modifiers.
- Conceptually related text, partial hits, loose paraphrases, and
superordinate substitutions do NOT count as coverage.
- Synonyms are allowed.
For example, "xiao-long-bao" = "steamed soup dumplings".
- Category broadening is not allowed.
For example, "xiao-long-bao" != "dim sum".
- Missing a key modifier counts as not covered.
For example, "Li Ming ate xiao-long-bao" !=
"Li Ming ate something".

Compute:

coverage = number of covered gold facts / total number of gold facts

Return only JSON in this format:

{
    "coverage": <float between 0.0 and 1.0>,
    "reason": "<one short English sentence>"
}

Positive 1:
gold:
```

```
["Li Ming ate xiao-long-bao in Lujiazui, Shanghai"]

retrieved:
["I went with Li Ming to try xiao-long-bao in Lujiazui,
Shanghai -- it was delicious"]

output:
{
    "coverage": 1.0,
    "reason": "the food and location are identical"
}

Negative 1, superordinate substitution:
gold:
["Li Ming ate xiao-long-bao in Lujiazui, Shanghai"]

retrieved:
["Li Ming ate dim sum in Shanghai"]

output:
{
    "coverage": 0.0,
    "reason": "dim sum is not equal to xiao-long-bao"
}

Negative 2, partial coverage:
gold:
[
"Li Ming went to Shanghai on April 1",
"Li Ming ate xiao-long-bao in Shanghai"
]

retrieved:
["Li Ming went to Shanghai on April 1 for a meeting"]

output:
{
    "coverage": 0.5,
    "reason": "only the Shanghai trip on April 1 is covered"
}

Negative 3, number mismatch:
gold:
["Li Ming spent 200 yuan on a book"]

retrieved:
["Li Ming bought a book"]

output:
{
    "coverage": 0.0,
    "reason": "the price is missing"
}

[user]
gold_evidence:
{gold}

retrieved_memories:
{retrieved}

Output JSON following the rules above.
```

J.5 Error-Cause Judge Prompt

The following prompt is used for the LLM reviewer in Appendix G.1.

Error-cause judge prompt

```
[system]
You are a strict retrieval failure diagnosis reviewer for
long-horizon personal-memory QA.

Role:
You diagnose why an incorrect answer happened. Your main focus is
retrieval quality, especially whether retrieval selected evidence from
the wrong time or wrong event instance.

Task:
Given one incorrect QA case, compare:
1) question,
2) gold answer,
3) model answer,
4) gold evidence,
5) retrieved context.
Assign exactly one primary_error.

Allowed labels:

1. temporal_mismatch
Use when the retrieved context contains a salient distractor that is
semantically related to the question but temporally incompatible with
the gold evidence. This requires at least one of:
- same person/entity/topic, but a clearly different date/window/event
instance than the gold evidence;
- stale state or old preference used when the question requires a later
updated state;
- future plan/reminder/preview used instead of the actual completed
event;
- long-term persona or routine activity replacing a temporary
time-bounded stage;
- model answer is visibly based on a retrieved wrong-time event.

2. partial_evidence
Use when retrieved context contains correct evidence from the right
event/window but misses other necessary correct evidence. Typical cases:
one hop of a multi-hop query, one endpoint of a dual-window query, one
item in a list, or correct event but missing a key detail.

3. retrieval_miss
Use when the answer-bearing evidence is absent and there is no explicit
wrong-time same-entity/same-topic distractor. This includes off-topic
retrieval, generic persona summaries, vague related memories without a
date/event-instance conflict, and near-empty contexts.

4. reason_failure
Use when retrieved context contains enough correct evidence from the
right event/window to answer correctly, but the model answer is wrong
due to reasoning, aggregation, option selection, or wording.

5. other
Use only when none of the above fits.

Decision procedure:
Step 1: Is there enough correct retrieved evidence from the right
event/window for a careful reader to answer? If yes and answer is still
wrong, choose reason_failure.
Step 2: Is some correct right-event/right-window evidence present but
incomplete? If yes, choose partial_evidence unless a wrong-time
distractor is clearly dominant.
Step 3: Is there an explicit wrong-time/stale/future/wrong-stage
distractor related to the same entity/topic and likely driving the wrong
answer? If yes, choose temporal_mismatch.
Step 4: Otherwise choose retrieval_miss.
```

```
Important constraints:
- Do NOT label temporal_mismatch just because the benchmark is temporal.
- Do NOT label temporal_mismatch for non-temporal questions unless the
failure truly depends on stale/updated state or a wrong event instance.
- Do NOT label temporal_mismatch when retrieved context is merely
broad/generic and lacks a clear wrong date/window/event.
- The explanation for temporal_mismatch must name the wrong-time
distractor type, e.g., "July dog incident replaces March apartment
halt" or "cardiology persona replaces August respiratory rotation".

Example A: temporal_mismatch
question: "What trouble did Huang Wenhui face in March?"
gold evidence: "2025-03-12: her pre-sold apartment project halted
construction and the developer's cash flow broke down."
retrieved context: "2025-07-10: Huang Wenhui complained about a
neighbor's dog disturbing her sleep."
model answer: "A neighbor's dog bothered her."
output: {"primary_error":"temporal_mismatch",
    "secondary_errors":["retrieval_miss"],
    "retrieval_contains_core_evidence":false,
    "retrieval_contains_wrong_time_related_evidence":true,
    "explanation":"the July dog incident replaces the queried March
    apartment halt"}

Example B: retrieval_miss, not temporal_mismatch
question: "What trouble did Huang Wenhui face in March?"
gold evidence: "2025-03-12: her pre-sold apartment project halted
construction."
retrieved context: "Huang Wenhui has been stressed about housing and
work."
model answer: "She had general housing stress."
output: {"primary_error":"retrieval_miss","secondary_errors":[],
    "retrieval_contains_core_evidence":false,
    "retrieval_contains_wrong_time_related_evidence":false,
    "explanation":"the context is generic and lacks a specific wrong-time
    event"}

Example C: partial_evidence
question: "What did she do on both days of the first weekend after
moving?"
gold evidence: ["Saturday: park walk and housewarming dinner",
"Sunday: Pilates and pottery class"]
retrieved context: "Saturday: she took a park walk and had a
housewarming dinner."
model answer: "She walked in the park and had dinner."
output: {"primary_error":"partial_evidence","secondary_errors":[],
    "retrieval_contains_core_evidence":true,
    "retrieval_contains_wrong_time_related_evidence":false,
    "explanation":"the correct weekend is retrieved but only the Saturday
    endpoint is covered"}

Example D: reason_failure
question: "Which option is the appointment date?"
gold evidence: "The app notification confirmed January 10."
retrieved context: "The app notification confirmed the appointment on
January 10."
model answer: "B: January 8."
output: {"primary_error":"reason_failure","secondary_errors":[],
    "retrieval_contains_core_evidence":true,
    "retrieval_contains_wrong_time_related_evidence":false,
    "explanation":"the right evidence is present but the option selection
    is wrong"}

Return only valid JSON:
{
```

```
    "primary_error": "temporal_mismatch | partial_evidence | retrieval_miss
     | reason_failure | other",
    "secondary_errors": ["temporal_mismatch | partial_evidence |
    retrieval_miss | reason_failure | other"],
    "retrieval_contains_core_evidence": true/false,
    "retrieval_contains_wrong_time_related_evidence": true/false,
    "explanation": "one concise English sentence"
}

[user]
question:
{question}

gold_answer:
{gold_answer}

model_answer:
{model_answer}

gold_evidence:
{gold_evidence}

retrieved_context:
{retrieved_context}
```

J.6 Judge Consistency Audit

This appendix audits the variance of the LLM judges used for GRPO.

To audit cross-model agreement, we re-evaluate the same response set ($n = 408$ from LifeBench user-0 and user-1) with two additional judges drawn from different model families—`claude-sonnet-4-6` (Anthropic) and `gemini-3.1-flash-lite-preview-nothink` (Google)—using the exact same system and user prompts as the `gpt-5-mini` judge reported in the main paper.

	`gpt-5-mini`	`claude-sonnet-4-6`	`gemini-3.1-lite`
`gpt-5-mini`	—	0.894	0.836
`claude-sonnet-4-6`	0.894	—	0.894
`gemini-3.1-lite`	0.836	0.894	—

All three pairwise Cohen's κ values lie in $[0.836, 0.894]$, well above the 0.81 threshold conventionally interpreted as "almost perfect" agreement. We therefore report a single judge (`gpt-5-mini`) throughout the main paper without loss of generality.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the TIDER framework, the RL-trained temporal interval locator, the dual-channel retrieval mechanism, and the reported gains on LifeBench and LoCoMo. These claims are supported by the main results in Section 4.3 and ablations in Section 4.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix I.1 discusses limitations from synthetic personal histories, dependence on upstream memory-item extraction, and English-only evaluation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes an empirical retrieval framework and does not claim theoretical results requiring formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 4.1–4.2 and Appendices A, B, C, and J describe the datasets, splits, memory-pool construction, training procedure, hyperparameters, prompts, and evaluation protocol needed to reproduce the main results. The supplementary Markdown guide and anonymized code package provide executable scripts and configuration files for reproducing the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide an anonymized supplementary code package with a Markdown reproduction guide, including environment setup, data preparation instructions, configuration files, and scripts for reproducing the main experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.2 specifies datasets, baselines, retrieval budget, model choices, split construction, and key hyperparameters; Appendix C gives SFT and GRPO training details, and Appendix E details baseline adaptation. The supplementary code package includes the corresponding configuration files and run scripts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The main result tables report means and standard deviations over independent evaluation runs, and Appendix F reports stratified accuracy with standard deviations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix H reports training hardware, wall-clock cost, and the per-query inference latency.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses synthetic benchmarks and does not use real personal-memory data; privacy and misuse risks are discussed in Appendix I.2. The submission is anonymized and does not disclose identifying author information.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix I.2 discusses positive impacts for long-term personalized assistants as well as risks from fabricated dates, biased extraction, and propagation of upstream memory errors.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk foundation model, scraped dataset, image generator, or real personal-memory corpus. As an experimental safeguard, all training and evaluation use synthetic benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Existing assets used in the paper, including LifeBench, LoCoMo, Qwen3-1.7B, BGE-M3, BM25, veRL, and API-based LLMs, are credited through citations in the paper. The supplementary Markdown guide lists their project links, versions when available, and license or terms-of-use information, and our use follows those terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We introduce an anonymized supplementary code package as a new asset. It is documented with a Markdown guide describing its purpose, setup, reproduction commands, dependencies, and license.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The work does not conduct crowdsourcing experiments or human-subject studies. The only human involvement described is verification of synthetic benchmark-derived temporal labels, not collection of human-subject data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research or real personal-memory data, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are core components of the method and evaluation: they are used for memory-item extraction and summarization, temporal-label construction, downstream answer generation, and LLM-based judging. These uses are described in Sections 3–4.2 and Appendix J.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.