FROM PASSIVE RETRIEVAL TO ACTIVE NAVIGATION: LEARNING TO USE LONG-TERM USER MEMORY AS A STRUCTURED ACTION SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term user memory is essential for personalized conversational agents, but exposing memory only as pre-selected context limits the agent's control over when and how to access user history. We introduce NapMem, a framework for *active memory navigation* that treats long-term user memory as a structured action space. NapMem organizes user histories into a multi-granularity pyramid of raw conversations, typed memory records, topic tracks, and user profiles, with provenance links connecting abstractions to supporting evidence. These levels are exposed through memory tools, and reinforcement learning over memory-tool trajectories trains the agent to decide whether to access memory, which level to consult, and when to refine or stop navigation based on intermediate evidence. On PersonaMem-v2, LongMemEval, and LoCoMo, a 9B NapMem agent achieves the highest average score among the evaluated systems, outperforming a 397B variant without memory-navigation training. Besides, the learned policy maintains performance on the evaluated non-memory reasoning and tool-use tasks, while reducing the memory-call rate. Ablation and trajectory analyses support the value of coupling structured memory with a learned, query-conditioned navigation policy.

1 INTRODUCTION

Large language model (LLM) agents in long-running interactive settings are expected to maintain continuity and personalize their responses across sessions. This requires persistent access to user information beyond the current conversation, making long-term user memory an essential component of personalized agents (Hu et al., 2025; Packer et al., 2024; Xu et al., 2026b). Effective memory use, however, involves more than retaining past interactions: different queries require different kinds and granularities of evidence (Zhong et al., 2023). Recalling a specific event may require the original exchange, understanding an evolving preference may require evidence across sessions, and adapting a response to the user may rely on a concise profile. The challenge is therefore not only what to remember, but also how to access the relevant information for a particular query.

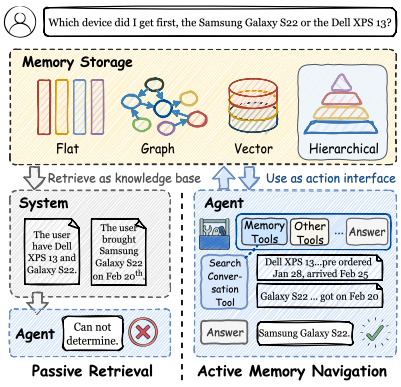

Figure 1: Passive retrieval supplies pre-selected context, whereas NapMem lets the agent gather memory evidence through tools.

Existing work improves both memory organization and retrieval. Memory systems organize user histories through finer-grained categories and vector, graph, hybrid, or hierarchical storage structures (Chhikara et al., 2025; Rasmussen et al., 2025; Li et al., 2025; Kang et al., 2025). Complementary approaches improve retrieval through reflection, query optimization, and reranking (Tan et al., 2025; Jiang et al., 2025b; Zhang et al., 2026b). When memory is exposed only through passive retrieval, however, the answering agent receives a pre-selected context rather than control over subsequent access (Figure 1). It cannot revisit the memory bank to refine its query or inspect supporting conversations when the provided context is incomplete. If retrieval is invoked for every query, memory is also supplied when the answer does not depend on user history. Thus,

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

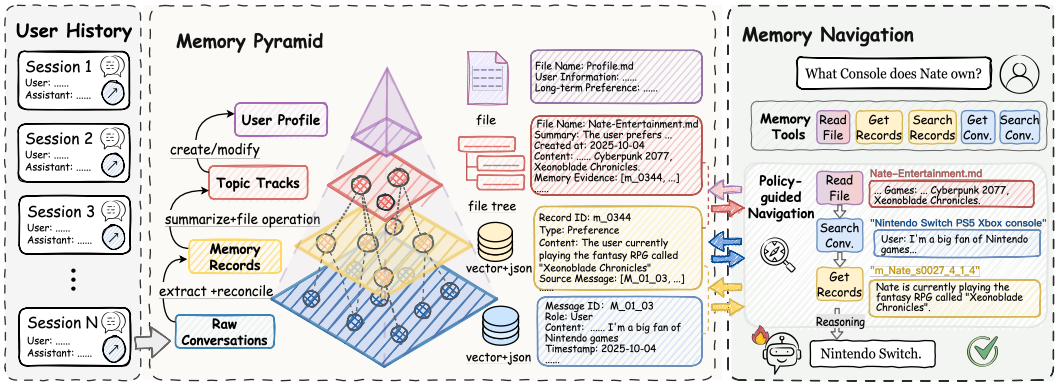

Figure 2: Overview of NapMem. A learned policy navigates a provenance-linked memory pyramid through tools, selecting abstraction levels and refining access based on intermediate evidence.

richer storage and stronger retrieval do not automatically translate into adaptive memory use by the answering agent.

We argue that ***long-term user memory use should move from system-level passive retrieval to agent-native memory navigation***. As illustrated in Figure 1, passive retrieval reduces different memory storage structures to pre-selected context and delivers it to the agent, limiting the agent's ability to inspect additional evidence when the retrieved context is incomplete. In contrast, active memory navigation formulates long-term user memory as a structured action space and equips the agent with memory tools, allowing it to decide whether memory is needed, which level of abstraction to consult, and whether the acquired evidence is sufficient. We instantiate this view with **NapMem** (**Na**vigate over **P**yramid **Mem**ory), a framework that trains agents to navigate long-term user memory as a structured action space.

NapMem pairs a multi-granularity memory pyramid with a learned navigation policy (Figure 2). The pyramid organizes user histories into four levels: raw conversations for original evidence, typed memory records for compact facts and preferences, topic tracks for cross-session aggregation, and user profiles for stable user-level information. Provenance links preserve paths from abstracted memories to their supporting evidence, allowing the agent to consult summaries for context and descend to original exchanges when finer detail is needed. Memory tools expose these levels for sequential access. We train the policy with reinforcement learning (RL) over memory-tool trajectories using both memory-intensive and memory-irrelevant queries, rewarding correct answers and query-appropriate memory access rather than unconditional tool use.

Across PersonaMem-v2, LongMemEval, and LoCoMo, the trained 9B agent achieves the highest average score among the evaluated systems, exceeding a 397B variant without memory-navigation training. On the evaluated non-memory benchmarks, it maintains reasoning and task-tool performance while reducing unnecessary memory calls. Ablations show that removing policy learning, replacing navigation with passive retrieval, or restricting the available memory levels each reduces performance. Together, these results support the value of coupling structured memory with a learned, query-conditioned access policy. Our contributions are threefold:

- We formulate long-term user-memory access as *active memory navigation* over linked abstraction levels, making evidence acquisition and the decision to skip memory explicit parts of the answering agent's policy.
- We develop NapMem, which combines a provenance-linked memory pyramid, tools for accessing different granularities, and reinforcement learning for query-conditioned navigation.
- We evaluate NapMem on three memory-intensive and three non-memory benchmarks, demonstrating improved memory-task performance and selective memory access, and analyze the roles of memory granularity and policy learning through ablations and navigation trajectories.

2 Related Work

Memory construction and organization. Early user-memory systems introduce persistent user history and virtual context management for long-running interactions (Zhong et al., 2023; Packer et al., 2024). Recent systems explore different memory architectures: Mem0 constructs compact memory records for scalable retrieval, Zep organizes memory as a temporal knowledge graph, MemOS treats memory as a system-level resource, and MemoryOS proposes a hierarchical short-/mid-/long-term personal memory (Chhikara et al., 2025; Rasmussen et al., 2025; Li et al., 2025; Kang et al., 2025). Complementary work links and evolves memories agentically or emphasizes lightweight compression and consolidation (Xu et al., 2026a; Fang et al., 2026; Liu et al., 2026). These systems invest in what is stored; how the agent reaches it at query time remains a fixed interface.

Adaptive retrieval for user memory. Another line improves the retrieval stage. Reflective Memory Management refines retrieval through online RL on cited evidence (Tan et al., 2025), while retrieval-oriented training such as Search-R1 and DeepRetrieval optimizes query generation during reasoning (Jin et al., 2025; Jiang et al., 2025b), and Zhang et al. (2026b) expands queries using user expression style and corpus structure. These methods make retrieval adaptive, but the agent still consumes whatever a single retrieval call returns and cannot choose a level of abstraction or continue searching when the evidence is insufficient.

RL-based and agentic memory systems. Learning-based systems replace hand-crafted memory routines with trainable policies. One line trains agents to decide what to store, how to structure it, or when to update it from downstream feedback (Wang et al., 2025; Kang et al., 2026). Another treats memory management as operation selection, learning to write, update, delete, summarize, or retrieve across stores (Yan et al., 2026; Yu et al., 2026; Zhang et al., 2026a), and routing-based systems assign queries to memory paths or modules (Chen et al., 2026). These learn the write side or route among modules. NapMem instead learns the read side: a single conversational agent selects levels of a structured user-memory pyramid and refines access based on intermediate evidence.

3 NapMem

As shown in Figure 2, NapMem organizes a user's interaction history into a multi-granularity memory pyramid spanning raw conversational evidence to high-level profiles. At inference, the agent interacts with the pyramid through memory tools, sequentially deciding which level to access. We optimize this behavior with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) under rewards for accurate answers, valid format, and appropriate memory use within a tool-call budget.

3.1 Multi-Granularity Memory Pyramid

NapMem constructs a separate memory pyramid per user, processing the interaction history in chronological order. Construction is incremental and bottom-up: new sessions are appended to the raw-conversation layer, then used to extract and reconcile memory records, which may in turn update topic tracks and the profile, so lower-layer changes propagate upward when new evidence extends, supersedes, or contradicts existing memories. Each level abstracts the same history to a different degree, and adjacent levels are connected by provenance links that let the agent move between detailed evidence and summarized understanding at inference time. The two lower layers are stored as JSONL records with unique identifiers and vector plus keyword indices for exact lookup and hybrid retrieval; the two upper layers are Markdown files read directly through memory tools. Algorithm 1 in Appendix B gives the full procedure.

Raw conversations. The base layer stores the original interactions as message-level entries with raw content, speaker role, timestamp, and a unique identifier. This append-only layer preserves the highest-fidelity evidence, letting the agent recover fine-grained details when needed.

Memory records. The second layer converts raw evidence into compact records, the basic semantic unit of user memory, abstracting away conversational redundancy while staying retrievable and verifiable. We define four record types, *fact*, *event*, *instruction*, and *preference*, and update the layer

in two incremental stages. Once a batch of complete turns accumulates, the agent reads the new conversations together with recent records and writes new records supported by the current dialogue, each following a fixed schema with a unique identifier, type, content, creation or update time, source message identifiers, and type-specific metadata. The new records are then reconciled against the existing bank through hybrid retrieval, which removes redundancy, updates outdated or conflicting information, and merges related records. Reconciled records retain their update history and source message identifiers, providing a bridge from compact memory back to raw evidence.

Topic tracks. The third layer organizes related records into evolving user-centric topic tracks. Records capture atomic information, but many dimensions of user understanding develop across sessions and require longitudinal tracking over multiple pieces of evidence. Given newly created or revised records, the agent decides whether to update an existing track, merge related tracks, or start a new one. Each track is a file with metadata, a short summary, and a narrative description, grounded by explicit links to supporting record identifiers, so tracks capture cross-session evolution while preserving a downward path to the record layer.

User profile. The top layer maintains a global profile of stable user attributes, long-term preferences, and interaction patterns. It is initialized after the first session and revised under a length budget once enough new records or topic-track changes accumulate, giving a concise global view of the user for personalization and high-level contextualization.

3.2 Tool-Based Memory Navigation

NapMem exposes the pyramid levels through memory tools, making memory access explicit: the agent must decide which operation to invoke from the query and the information gathered so far. Formally, given a user query q and a memory pyramid $\mathcal{M}$, at each step t the agent either invokes a memory tool or terminates navigation and answers, sampling $a_t \sim \pi_\theta(\cdot \mid q, \mathcal{M}, a_{<t}, o_{<t})$, where o_t is the observation returned by a tool call. This yields a memory-use trajectory $\tau = (a_1, o_1, \ldots, a_k, o_k, y)$ ending in the final answer y.

NapMem is equipped with five tools tailored to different memory layers: `get_conversations`, `search_conversations`, `get_records`, `search_records`, and `read_files`. Search tools retrieve candidate raw conversation snippets or memory records through hybrid search. Get tools access exact messages or records through their persistent identifiers. The file-reading tool allows the agent to inspect topic tracks and the user profile, which provide higher-level abstractions of user history. These tools support both top-down and bottom-up navigation: the agent may begin with global user context, refine its search through topic- or record-level memories, or descend to raw conversations for evidence-level verification.

3.3 Learning Memory Navigation with GRPO

We optimize the memory-navigation policy with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Each training sample pairs a query with a memory pyramid, and the agent answers within a fixed tool-call budget. Training combines memory-intensive queries with a smaller set of memory-irrelevant queries, allowing the policy to learn both memory navigation and direct answering. Each trajectory receives a terminal reward that jointly evaluates answer quality and query-appropriate memory use.

Rubric reward. We score each trajectory using three binary criteria: format validity F, answer correctness C, and memory-use appropriateness U. Let $m(\tau) \in \{0, 1\}$ indicate whether trajectory τ invokes at least one memory tool, and let $m^\star(q) \in \{0, 1\}$ indicate whether query q requires user memory. We define $U = \mathbb{1}[m(\tau) = m^\star(q)]$, encouraging memory access on memory-intensive queries and direct answering otherwise. Trajectories that exceed the tool-call budget or fail to produce a valid final answer are treated as invalid ($F = 0$). For memory-intensive queries ($m^\star(q) = 1$), the

terminal reward is

$$r(\tau) = \begin{cases} -1, & F = 0, \\ 1, & F = 1,\ C = 1,\ U = 1, \\ 0, & F = 1,\ C = 1,\ U = 0, \\ -0.5, & F = 1,\ C = 0,\ U = 1, \\ -1, & F = 1,\ C = 0,\ U = 0. \end{cases}$$

For memory-irrelevant queries ($m^\star(q) = 0$), we use the same rubric except that a correct answer with unnecessary memory access receives 0.5 rather than 0. Answer correctness is determined by exact matching for multiple-choice tasks and by an LLM judge for open-ended tasks, following the corresponding benchmark protocol. While U supervises whether to access memory, finer-grained navigation decisions are optimized through the trajectory-level reward.

Group-relative optimization. For each query, we sample a group of trajectories $\mathcal{G}$ and assign each trajectory τ_i a terminal reward $r_i = r(\tau_i)$. We compute mean-centered group-relative advantages:

$$A_i = r_i - \frac{1}{|\mathcal{G}|} \sum_{j=1}^{|\mathcal{G}|} r_j.$$

We then maximize the clipped GRPO objective:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \frac{1}{T_i} \sum_{t=1}^{T_i} \ell_{i,t}(\theta) - \beta D_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big) \right],$$

where

$$\ell_{i,t}(\theta) = \min\left(\rho_{i,t} A_i,\ \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) A_i\right).$$

Here, $\rho_{i,t}$ is the token-level importance ratio between the current and rollout policies, T_i is the number of agent-generated tokens in τ_i, ϵ is the clipping threshold, and β controls regularization toward the reference policy π_{ref}. The same trajectory-level advantage is applied to all agent-generated tokens, including those specifying memory-tool calls, jointly optimizing navigation and final-answer generation.

4 Experiments

4.1 Experiment Setup

Datasets. We use three **memory-intensive** benchmarks: LongMemEval (Wu et al., 2025) and LoCoMo (Maharana et al., 2024) target user-specific factual memory through long-range recall, temporal change, and cross-session reasoning, while PersonaMem-v2 (Jiang et al., 2025a) targets implicit preferences and persona understanding. To test whether memory-use training preserves broader agentic ability without inducing unnecessary memory calls, we add three **non-memory** benchmarks: GPQA-Diamond (GPQA-D) (Rein et al., 2023) for reasoning, BFCL-v3 (Patil et al., 2025) for function calling, and V*Bench (Wu & Xie, 2024) for visual tool use. Appendix A gives dataset details and splits.

Metrics. We report F1 and LLM-judge accuracy (L-J) for open-ended memory QA, accuracy for multiple-choice tasks, and task success or accuracy for BFCL-v3 and V*Bench following their standard protocols.

Baselines and models. We compare against five memory baselines spanning both architectures and agentic policies: Mem0 (Chhikara et al., 2025) for flat vector memory, Zep (Rasmussen et al., 2025) for temporal graph memory, MemOS (Li et al., 2025) for a memory operating system, MemoryOS (Kang et al., 2025) for hierarchical memory, and AgeMem (Yu et al., 2026) for RL-trained agentic memory. All use Qwen3.5-9B (Team, 2026) as the base LLM; NapMem denotes this model trained with our framework unless stated otherwise, and we also report untrained scaling variants NapMem-122B and NapMem-397B.

Three aspects of the protocol matter for attribution. First, since this is an end-to-end comparison of memory systems, each baseline keeps its published retrieval settings: the passive-retrieval systems define no trainable navigation policy, and imposing our reward on them would replace rather than evaluate the original method. AgeMem provides the matched learned comparison, trained with the same backbone, the same LoCoMo and PersonaMem-v2 user split, and the same optimization budget as NapMem while keeping its own tool interface and reward. Second, every memory bank here is built by that same Qwen3.5-9B model, for NapMem and baselines alike, so gains cannot come from a stronger offline builder; AgeMem is the only exception, building memory with its own RL-trained model as its method specifies. The builder sees only interaction histories, never benchmark questions, gold answers, or development examples (Appendix C.1). Third, NapMem is trained on LoCoMo and PersonaMem-v2 queries together with 500 memory-irrelevant MMLU questions, so LongMemEval is never seen during training and provides the out-of-distribution evidence.

Implementation details. We train with memory-tool RL on a mixture of memory-intensive queries from LoCoMo and PersonaMem-v2, split 60/20/20 by user to avoid leakage, and 500 memory-irrelevant MMLU questions for which the rubric rewards answering without consulting memory. The memory-irrelevant portion supplies the negative supervision that the criterion U needs. At inference NapMem and AgeMem may take at most four tool steps per query, with over-budget trajectories counted incorrect, and each NapMem retrieval returns up to five memory items; non-agentic baselines may return up to twenty and otherwise follow their original settings. All embedding-based retrieval shares Qwen3-Embedding-0.6B (Yang et al., 2025). Open-ended QA is judged by `Claude-Opus-4.6`, which matches the majority of three human annotators on 99.0% of 100 sampled judgments (Cohen's $\kappa = 0.980$). Results are averaged over three seeds; hyperparameters and prompts are in Appendices C and H.

4.2 Main Comparison

Results on long-term user-memory tasks. Table 1 reports results on the three memory-intensive benchmarks. NapMem-9B w/ RL obtains the highest average score among the evaluated systems (63.19), compared with 59.62 for NapMem-397B and 59.25 for the strongest external baseline, Mem0. It ranks first on LoCoMo F1, on both LongMemEval metrics, and on PersonaMem-v2, and second on LoCoMo LLM-judge accuracy behind Mem0. On LongMemEval, which is held out from navigation training, it exceeds NapMem-397B by 2.00 L-J points and 1.47 F1 points, providing evidence of cross-benchmark transfer despite the smaller model size. Overall, the trained 9B model is competitive on factual recall, cross-session reasoning, and implicit preference understanding.

Table 1: Memory-intensive results. L-J is LLM-judge accuracy. Avg. averages LoCoMo L-J, LongMemEval L-J, and PersonaMem-v2 accuracy.

Method	LoCoMo		LongMemEval		PersonaMem-v2	Avg.
	F1	L-J	F1	L-J	Acc.	
Mem0	41.19 ± 0.36	**60.86 ± 0.38**	53.86 ± 0.54	78.00 ± 0.82	38.89 ± 0.34	59.25 ± 0.32
Zep	36.09 ± 0.38	52.21 ± 0.31	52.61 ± 0.20	73.33 ± 0.94	36.33 ± 0.81	53.96 ± 0.43
MemOS	35.22 ± 0.38	55.82 ± 0.65	39.16 ± 0.40	54.33 ± 0.47	37.58 ± 0.44	49.24 ± 0.30
MemoryOS	31.40 ± 0.53	44.28 ± 1.22	22.68 ± 0.30	23.67 ± 0.58	35.82 ± 1.87	34.59 ± 0.05
AgeMem	38.00 ± 0.12	45.02 ± 0.12	37.83 ± 1.38	51.33 ± 0.94	24.19 ± 0.67	40.18 ± 0.37
NapMem-122B	35.39 ± 0.26	51.69 ± 0.84	48.86 ± 2.58	70.00 ± 1.73	41.38 ± 0.77	54.36 ± 0.74
NapMem-397B	38.31 ± 0.13	54.42 ± 0.81	53.85 ± 1.25	78.33 ± 2.08	46.10 ± 1.15	59.62 ± 1.14
NapMem-9B w/ RL	**41.59 ± 0.23**	59.92 ± 0.52	**55.32 ± 1.02**	**80.33 ± 0.58**	**49.32 ± 0.61**	**63.19 ± 0.57**

Gains concentrate where a single retrieval is insufficient. Table 2 breaks the two open-ended benchmarks down by question type against Mem0. NapMem gains where an answer must be assembled from evidence no single retrieval returns: LoCoMo multi-hop improves by 14.96 and LongMemEval single-session-user by 14.29, while categories answerable from one matched record are essentially unchanged. The remaining gap is LoCoMo adversarial/unanswerable, which is scored on abstention rather than evidence use and accounts for the slightly lower overall LoCoMo LLM-judge score despite the multi-hop gain.

Table 2: Sub-task breakdown against Mem0 (LLM-judge accuracy).

Benchmark	Sub-task	Mem0	NapMem	Δ
LongMemEval	Single-session-user	80.95	**95.24**	+14.29
	Single-session-assistant	91.67	**94.44**	+2.77
	Single-session-preference	72.22	72.22	0.00
	Multi-session	65.38	65.38	0.00
	Knowledge-update	**97.78**	95.56	−2.22
	Temporal-reasoning	**80.25**	77.78	−2.47
	Overall	78.00	**80.33**	+2.33
LoCoMo	Multi-hop	37.19	**52.15**	+14.96
	Open-domain / commonsense	41.56	**45.45**	+3.89
	Single-hop	**73.01**	72.64	−0.37
	Temporal	**63.69**	61.04	−2.65
	Adversarial / unanswerable	**68.83**	45.93	−22.90
	Overall	**60.86**	59.92	−0.94

Table 3: Generalization to non-memory tasks. "Mem." is the memory-tool call rate, measuring unnecessary memory access; the base model has no memory tools.

Model	GPQA-D		BFCL-v3				V*Bench			
	Acc.	Mem.	Task	Tool	Exec	Mem.	Ans.	Tool	Exec	Mem.
Base	53.03	–	**95.00**	97.50	93.50	–	84.82	97.91	97.91	–
NapMem w/o RL	55.74	34.51	93.50	97.50	94.00	**0.00**	89.53	98.95	98.95	**0.00**
NapMem w/ RL	**61.26**	**3.52**	**95.00**	97.50	**94.50**	**0.00**	**91.10**	**99.48**	**99.48**	**0.00**

NapMem uses memory only when the query calls for it. A memory-tool policy is useful only if it also knows when *not* to consult memory, so we evaluate on benchmarks whose answers do not depend on user history. For each query we randomly assign a user memory pyramid and enable all memory tools, while the task itself requires none of it. Table 3 shows the policy largely abstains: memory calls fall from 34.51% without RL to 3.52% on GPQA-D, and to zero on BFCL-v3 and V*Bench. This follows directly from defining U against the query rather than against tool use, which lets the memory-irrelevant part of the training mixture reward answering without memory. Suppressing unnecessary access also removes a source of distraction rather than merely saving calls: GPQA-D accuracy reaches 61.26, above both the base model and the untrained tool-using variant, and V*Bench improves on answer, tool, and execution accuracy. Learning memory navigation therefore does not trade away general reasoning or tool use, and how much the agent relies on memory is set by the training distribution rather than by hand-tuned retrieval thresholds.

4.3 Efficiency

Storage and construction cost. Long-term memory must stay affordable both to keep and to build. Table 4 shows NapMem uses less storage than most baselines despite exposing multiple granularities; AgeMem is smaller but scores far lower in Table 1, so footprint alone is not the relevant axis. Building the pyramid does cost more than flat extraction, since NapMem also deduplicates records and rewrites topic files and the profile, though it stays well below Zep. That price is paid offline rather than at inference, construction is incremental so each update processes only new sessions, and the builder is the same 9B open model rather than a stronger closed one.

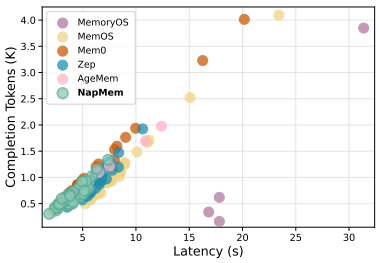

Figure 3: Latency and completion tokens on 100 successful PersonaMem-v2 runs.

Inference efficiency. Inference cost matters for long-running agents, where memory access should improve personalization without raising latency or verbosity. Figure 3 reports latency and completion tokens on 100 PersonaMem-v2 samples, computed over

successful runs. The two quantities are approximately linear, so generation length is a major contributor to cost. Compared with baselines that reason over a fixed retrieved context, NapMem navigates until the evidence suffices and then stops, yielding shorter completions and lower latency despite using memory tools.

Table 4: Offline memory cost. Storage is the total memory-bank size over the three benchmarks; construction cost is per user over 50 PersonaMem-v2 users. Appendix E gives the per-dataset breakdown.

Method	Storage (GiB)	Wall-clock (s)	LLM calls	Tokens (M)
NapMem	4.83	181	190	2.05
Mem0	10.44 (2.16×)	159	115	1.15
Zep	14.10 (2.92×)	838	859	2.49
MemOS	23.10 (4.78×)	**67**	**33**	**0.10**
MemoryOS	6.68 (1.38×)	133	377	0.21
AgeMem	**2.99** (0.62×)	89	76	0.19

4.4 Ablation Study

NapMem couples a structured memory representation with a learned access policy, so the ablations in Table 5 separate the two. ***w/o navigation*** keeps the constructed pyramid but removes iterative access: instead of calling tools, the model receives one fixed passive context with the top-5 memory records, the top-1 topic file, and the user profile, so it still sees the upper levels and only evidence-seeking is removed. ***w/o RL*** is the complement, keeping the identical pyramid and tool interface but dropping policy learning. ***records-only tools*** keeps the interface but restricts the agent to compact records, and ***w/o upper levels*** exposes conversations and records but removes topic tracks and the profile.

Neither component carries the result alone, and we state this precisely because it bounds the claim. Read passively, the pyramid reaches 54.08: ahead of MemoryOS (34.59), which also stores long-term fragments and a profile, but below Mem0 (59.25), so the representation by itself does not beat a well-tuned flat memory. Giving the agent the tool interface without training it is worse still (48.39), costing 5.69 points relative to passive access over the identical pyramid, so untrained tool use is actively harmful. Only the combination reaches 63.19. The pyramid's contribution is therefore not that summaries exist, but that separated, provenance-linked levels form an action space a policy can be trained on. Consistently, *records-only tools* and *w/o upper levels* both trail Full NapMem, so high-level abstraction and low-level evidence are complementary rather than redundant.

Table 5: Ablation of navigation, memory granularity, and RL. LCM and LME are LLM-judge accuracy; PMem is accuracy.

Variant	LCM.	LME.	PMem.	Avg.
Full NapMem	**59.92**	**80.33**	**49.32**	**63.19**
w/o RL	52.09	72.33	20.75	48.39
w/o navigation	48.37	68.00	45.88	54.08
records-only tools	39.77	60.00	35.02	44.93
w/o upper levels	49.13	75.00	38.20	54.11

4.5 Tool-Use Behavior Analysis

Navigation is query-directed, not exhaustive enumeration. A natural concern is that the agent may simply sweep every memory level regardless of the query. The trajectories say otherwise. Navigation is short, averaging 1.82–2.24 tool calls against a budget of four, and one pattern dominates everywhere: *records* → *conversations* → *answer*, covering 32–43% of trajectories. The learned default is to locate compact records and then descend to raw conversations to verify them, which is precisely the movement provenance links exist to support and that passive retrieval cannot express. Benchmarks then depart from this default in the expected direction. Reading topic tracks or the profile is common on PersonaMem-v2 (15.00%), marginal on LoCoMo (5.65%), and absent from

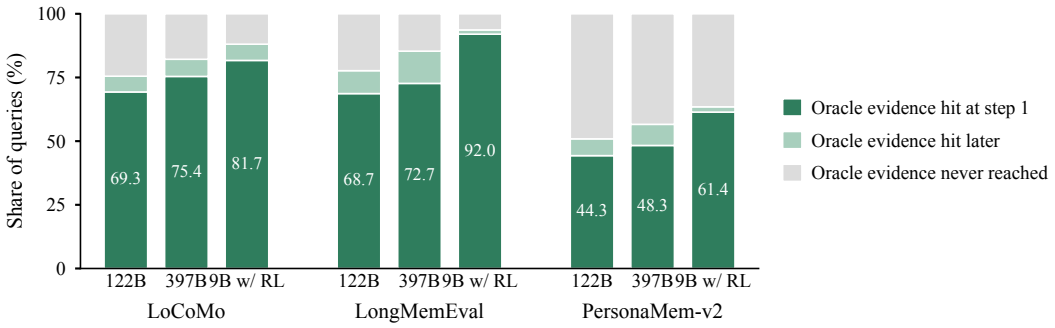

Figure 4: Where navigation first reaches oracle supporting evidence, for three NapMem variants on each benchmark. All variants navigate the same memory pyramid, so differences reflect the navigation policy alone.

LongMemEval's top patterns, matching the fact that only PersonaMem-v2 asks about stable preferences; LongMemEval instead has the shortest trajectories and the lowest multi-level rate (57.33%), consistent with questions answerable from one located exchange. One policy therefore yields different access strategies per query type, as an action-space view of memory predicts (Appendix D).

Navigation reaches the right evidence sooner. Figure 4 measures navigation quality directly, by checking whether the first tool call already returns an oracle supporting item. The trained 9B agent hits oracle evidence at step one in 81.7% (LoCoMo), 92.0% (LongMemEval), and 61.4% (PersonaMem-v2) of queries, above both untrained variants, including NapMem-397B, which is over forty times larger; trajectories that never reach oracle evidence drop correspondingly. Since all variants navigate the same pyramid, this is attributable to the policy rather than to memory quality, and explains how a 9B model outperforms far larger untrained ones. PersonaMem-v2 stays hardest, consistent with preference questions whose evidence is diffuse rather than localized in one record.

RL makes navigation shorter, not shallower. RL could in principle improve accuracy by making the agent search more. Table 6 shows it does the opposite. The trained agent averages 2.10 tool calls against 3.97 without RL, while its evidence-hit ratio, the number of returned tool results matching supporting evidence per 100 tool uses, rises from 20.66% to 34.92%: it issues roughly half as many calls and a larger fraction of them land on supporting evidence. This is not a collapse onto a single memory level, since 57–75% of its trajectories still cross levels depending on the benchmark (Appendix D); the saving comes from stopping once the evidence suffices.

Table 6: Tool-use behavior over the three memory benchmarks, weighted by test-question count. Per-benchmark depth is in Appendix D.

Model	Tools	Evid. Hit
NapMem-9B w/ RL	**2.10**	**34.92%**
NapMem-9B	3.97	20.66%
NapMem-122B	2.50	26.20%
NapMem-397B	2.69	26.42%

5 Conclusion

We presented NapMem, which treats long-term user memory as navigation over a multi-granularity pyramid of raw conversations, memory records, topic tracks, and user profiles, with a reinforcement-learned tool policy selecting the level to consult for each query. NapMem is competitive on user-memory tasks while largely preserving general reasoning and tool-use ability, and its gains concentrate on questions that require composing several pieces of evidence. Ablations and trajectory analyses indicate that neither the structured representation nor the learned policy is sufficient alone: memory becomes useful when storage and access are designed together.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data, implementing methods, and cleaning or reformatting datasets. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing experiments or methodology, or interpreting results. Assisting with proofs and translation are not applicable to this work. Additionally, we used generative AI tools to draft and edit the manuscript for readability, and better format. We have reviewed all AI-assisted work by checking generated text carefully. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Dataset construction, user-level splits, and benchmark protocols are described in Appendix A. Memory-pyramid construction is specified in Appendix B, including Algorithm 1. RL hyperparameters, the memory-tool interface, inference settings, and LLM-judge validation are in Appendix C. Prompt templates are in Appendix H. Main-text results are averages over three random seeds. An anonymous code supplement is the appropriate place to release the training and evaluation implementation; this statement does not assume that supplement is already attached.

REFERENCES

Jiayi Chen, Yingcong Li, and Guiling Wang. Memflow: Intent-driven memory orchestration for small language model agents. *arXiv preprint arXiv:2605.03312*, 2026.

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.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation, 2026. URL https://arxiv.org/abs/2510.18866.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory, 2025a. URL https://arxiv.org/abs/2512.06688.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*, 2025b.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent, 2025. URL https://arxiv.org/abs/2506.06326.

Jingyi Kang, Chunyu Li, Ding Chen, Bo Tang, Feiyu Xiong, and Zhiyu Li. Memreader: From passive to active extraction for long-term agent memory. *arXiv preprint arXiv:2604.07877*, 2026.

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.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

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 (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 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/.

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.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

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.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, et al. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

Penghao Wu and Saining Xie. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094, June 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026a.

Yue Xu, Qian Chen, Zizhan Ma, Dongrui Liu, Wenxuan Wang, Xiting Wang, Li Xiong, and Wenjie Wang. Toward personalized llm-powered agents: Foundations, evaluation, and future directions, 2026b. URL https://arxiv.org/abs/2602.22680.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents, 2026. URL https://arxiv.org/abs/2601.01885.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026a. URL https://arxiv.org/abs/2602.02474.

Yingyi Zhang, Pengyue Jia, Derong Xu, Yi Wen, Xianneng Li, Yichao Wang, Wenlin Zhang, Xiaopeng Li, Weinan Gan, Huifeng Guo, et al. Personalize before retrieve: Llm-based personalized query expansion for user-centric retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 16406–16414, 2026b.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023. URL https://arxiv.org/abs/2305.10250.

A Dataset Split Details

LoCoMo (Maharana et al., 2024). LoCoMo contains long multi-session conversations between two speakers, with questions that test long-term conversational memory. We convert each two-speaker conversation into speaker-specific user-agent memory banks by treating one speaker as the user and the other as the assistant, resulting in 20 users. We split users into train/validation/test with a 6/2/2 ratio. For each target speaker, we add a synthetic setup turn at the beginning of the memory bank, as shown below. If the first original utterance is already from the assistant/counterpart, only the user setup message is added. During evaluation, questions are asked from the corresponding speaker's perspective, and we remove questions whose evidence is missing or inconsistent with the provided answer in the original dataset. The final test set contains 1,315 questions.

LoCoMo speaker-specific setup turn

User: I am {target_speaker}. You are {counterpart_speaker}. Talk to me naturally like a friend.

Assistant: Got it.

LongMemEval. (Wu et al., 2025) LongMemEval evaluates long-term interactive memory across question types such as single-session recall, multi-session reasoning, temporal reasoning, knowledge updates, and preference-related questions. We use LongMemEval only as a held-out test benchmark to evaluate out-of-distribution generalization, with 100 test questions in total.

PersonaMem-v2. (Jiang et al., 2025a) PersonaMem-v2 evaluates personalized response selection from implicit user preferences and persona signals across long user-agent interactions. The benchmark contains 200 users, which we split by user into train/validation/test with a 6/2/2 ratio. The test set contains 911 questions. During evaluation, we randomly shuffle the answer options for each question to reduce positional bias.

GPQA-Diamond (Rein et al., 2023). We use the GPQA-Diamond (GPQA-D) dataset to evaluate memory-independent scientific reasoning. It is a graduate-level, Google-proof multiple-choice benchmark containing difficult biology, physics, and chemistry questions written by domain experts.

Algorithm 1 Incremental construction of a user memory pyramid.

Require: Chronologically sorted sessions $\mathcal{S} = \{s_1, \ldots, s_N\}$
Initialize raw store $\mathcal{C}$, record pool $\mathcal{R}$, pending buffer $\mathcal{B}$, topic tracks $\mathcal{T}$, profile P
Initialize topic and profile checkpoints $c_T, c_P \leftarrow 0$
for $s_i \in \mathcal{S}$ **do**
  if $i > 1$ and TOPICTRIGGER($\mathcal{B}, c_T, s_{i-1}, s_i$) **then**
    $\mathcal{T}, c_T \leftarrow$ UPDATETOPICTRACKS($\mathcal{T}, \mathcal{R}, \mathcal{B}$)
  end if
  if $i > 1$ and PROFILETRIGGER($\mathcal{T}, P, c_P$) **then**
    $P, c_P \leftarrow$ UPDATEPROFILE($P, \mathcal{T}$)
  end if
  Append non-empty messages in s_i to $\mathcal{C}$
  for each completed turn batch b in s_i **do**
    $\hat{\mathcal{R}} \leftarrow$ EXTRACTRECORDS(b, recent context, recent records)
    $\tilde{\mathcal{R}} \leftarrow$ RECONCILERECORDS($\hat{\mathcal{R}}, \mathcal{R}$)
    Add $\tilde{\mathcal{R}}$ to $\mathcal{R}$ and $\mathcal{B}$
    if TOPICTRIGGER($\mathcal{B}, c_T$) **then**
      $\mathcal{T}, c_T \leftarrow$ UPDATETOPICTRACKS($\mathcal{T}, \mathcal{R}, \mathcal{B}$)
    end if
    if PROFILETRIGGER($\mathcal{T}, P, c_P$) **then**
      $P, c_P \leftarrow$ UPDATEPROFILE($P, \mathcal{T}$)
    end if
  end for
end for
Flush remaining records, update topic tracks and profile if triggered
Finalize active records, retrieval indices, topic index, and build metadata

V*Bench. (Wu & Xie, 2024) We evaluate visual tool use on the official V* test set with 191 examples, including direct-attribute and relative-position questions. Each example contains an image, a multiple-choice visual question, and a ground-truth answer option. The model may call tools at most once and get the zoomed image as required for response.

BFCL-v3. (Patil et al., 2025) We evaluate text-based tool use on a 200-example executable subset of BFCL-v3, consisting of `exec_simple`, `exec_multiple`, and `exec_parallel` examples. Each example provides a user request, executable function schemas, and ground-truth tool calls. The evaluator executes the model-generated tool calls with deterministic local backends and returns the results to the model. We report tool-call accuracy, execution accuracy, task accuracy, and unnecessary memory tool-call count.

B MEMORY CONSTRUCTION DETAILS

NapMem builds an independent memory pyramid for each user by replaying normalized sessions in chronological order. The builder appends raw messages, extracts and reconciles memory records, periodically updates topic tracks, and refreshes the user profile. Construction is incremental: lower-layer changes are propagated upward only when update triggers are met.

Raw conversations. For each non-empty message, NapMem stores the speaker role, content, timestamp, session identifier, message identifier, user identifier, and available metadata. Raw messages are also indexed for retrieval using embeddings. This layer is append-only and serves as the evidence source for exact lookup and verification.

Memory records. Records are extracted from recently completed user-agent turns. The flush schedule expands from 1 to 2 to 4 turns and then uses a steady-state size of 5 turns, with a final flush at session end. Each record contains a persistent identifier, content, type, source message identifiers, timestamps, session information, and update metadata. We use four record types: *fact*, *event*, *instruction*, and *preference*. New records are deduplicated against existing records using vector retrieval. The reconciliation model chooses one of four actions: `store`, `skip`, `update`,

or `merge`. For `update` and `merge`, source message identifiers and timestamps are unioned to preserve provenance. The final records are indexed for later search.

Topic tracks. Topic tracks consolidate related memory records into medium-range user-centric narratives. Each track contains metadata, a concise summary, a narrative description, and explicit evidence links to supporting record identifiers. Topic updates are triggered before a new session when unprocessed records exist, at date boundaries, when more than 20 new records have accumulated since the last topic checkpoint, at replay end, or during final retry recovery. The topic writer can inspect existing tracks, update or create tracks, perform localized edits, and finish the update round. The system maintains at most 20 topic tracks. Updates are committed only after format validation; invalid writes are rolled back.

User profile. The profile summarizes stable user attributes, long-term preferences, interaction patterns, and high-level contextual information. It is initialized once topic tracks become available and is refreshed when requested by the topic writer, after successful topic-track updates, or when more than 50 new records have accumulated since the last profile checkpoint. Each refresh revises the existing profile under a length budget. If validation fails, the previous profile is restored.

C Implementation Details

We train NapMem with multi-turn GRPO using the verl framework. At each step, the agent may either answer directly or invoke memory tools to inspect different levels of the memory bank. The memory bank is indexed with Qwen3-Embedding-0.6B embeddings, and search tools use hybrid reciprocal-rank fusion over keyword and vector retrieval with $k = 60$.

C.1 Memory Builder Configuration

All memory banks compared in this paper are constructed offline by Qwen3.5-9B, the same open model that serves as the base agent. This holds for NapMem and for the Mem0, Zep, MemOS, and MemoryOS baselines; AgeMem is the sole exception, where memory is written by its own RL-trained model as specified by the original method. We deliberately avoid using a stronger or closed-source model as the builder, so that differences in Table 1 cannot be attributed to an asymmetry in offline construction quality.

The builder is run with temperature 0.7, top-p 0.9, and a maximum output length of 4096 tokens, processing sessions in chronological order. Its only input is the user interaction history: benchmark questions, gold answers, and development examples are never visible during construction, and no evaluation signal is used to select what is stored. The record schema, topic-track format, and profile fields were fixed before running evaluation and are shared across all datasets; the extraction, deduplication, topic-update, and profile-update prompts are listed verbatim in Appendix H.

C.2 RL Hyperparameters

C.3 Tool Operation Protocol

Table 8 summarizes the memory tools exposed to the NapMem agent. The tools are aligned with different levels of the memory pyramid. Search tools support coarse retrieval over memory records and raw conversations, exact-fetch tools retrieve specific records or messages using identifiers returned by previous tool calls, and `read_file` provides access to higher-level topic tracks and the user profile.

C.4 LLM Inference Hyperparameters

For evaluation, we use the Qwen3-Coder tool-call format for parsing tool invocations. All the generations use temperature 1.0, top-p 0.9, and a maximum completion length of 4096 tokens.

Table 7: RL hyperparameters for NapMem-9B training.

Hyperparameter	Value
RL algorithm	GRPO
Base model	Qwen3.5-9B
Train batch size	8
Rollout group size	4
Learning rate	1×10^{-6}
Max assistant turns	5
KL loss coefficient	0.001
Entropy coefficient	0
Precision	bfloat16
Total epochs	5
Hardware	NVIDIA H20

Table 8: Memory tools exposed to the agent. Search tools provide coarse retrieval, while exact-fetch and file-reading tools allow the model to navigate between memory granularities.

Tool	Input	Functionality
`search_records`	free-text query	Search structured memory records with hybrid RRF retrieval and return top-5 records.
`search_conversation`	free-text query	Search raw conversation messages with hybrid RRF retrieval and return top-5 snippets.
`get_records`	record id list	Fetch exact structured memory records by `record_id`.
`get_conversation`	message id list	Fetch exact raw conversation messages by `message_id`.
`read_file`	file name	Read a high-level memory-bank file, such as a topic-track file or `profile.md`.

C.5 Judge Validation

For open-ended memory QA, we use `Claude-Opus-4.6` as the automatic judge. To validate the reliability of the judge, we manually annotate a random subset of 100 judged examples with three human annotators using the same prompt as LLM, and compare the LLM judgment against the human-majority label.

The LLM judge aligns closely with human-majority labels, achieving 99.0% accuracy, 100.0% precision, 98.0% recall, 99.0% F1, and Cohen's $\kappa = 0.980$. The only disagreement is a false negative case where the gold answer indicates that the conversation did not mention baking egg tarts, while the model answered "0 times"; the human majority considered this equivalent, but the LLM judge marked it incorrect. Human annotators also show high agreement: pairwise Cohen's κ ranges from 0.920 to 0.960, and Fleiss' κ across three annotators is 0.933. These results indicate that the LLM judge aligns with human-majority judgments at near-human reliability in our validation sample.

C.6 Judge Sensitivity

Human validation establishes that the judge agrees with people, but not that conclusions are invariant to the choice of judge. We therefore re-score all open-ended outputs with `GPT-5.5` as an independent second judge (Table 10). The second judge is more lenient on average, by $+1.09$ on LoCoMo and $+0.71$ on LongMemEval. The shift is close to constant across systems, except that NapMem-122B drops by 0.33 on LongMemEval, so relative comparisons are unaffected. Rankings are identical on LongMemEval (Spearman $\rho = \tau = 1.000$) and near-identical on LoCoMo and on the overall average ($\rho = 0.976$, Kendall $\tau = 0.929$ for both); in each case the only change is a swap between two adjacent systems ranked well below NapMem, MemoryOS and AgeMem on LoCoMo and Zep and NapMem-122B on the average. Under both judges NapMem-9B w/ RL is the best system on

Table 9: Validation of the LLM judge against human-majority labels on 100 open-ended QA examples.

Metric	LLM vs. Human Majority
N	100
Accuracy	0.990
Precision	1.000
Recall	0.980
F1	0.990
Cohen's κ	0.980
TP / TN / FP / FN	50 / 49 / 0 / 1

Table 10: Judge sensitivity. LLM-judge accuracy under the default `Claude-Opus-4.6` judge (C) and an independent `GPT-5.5` judge (G).

Method	LoCoMo			LongMemEval		
	C	G	Δ	C	G	Δ
Mem0	**60.86**	**61.53**	+0.67	78.00	79.00	+1.00
Zep	52.21	53.24	+1.03	73.33	75.67	+2.34
MemOS	55.82	56.29	+0.47	54.33	54.33	+0.00
MemoryOS	44.28	46.32	+2.04	23.67	24.67	+1.00
AgeMem	45.02	45.30	+0.28	51.33	51.67	+0.34
NapMem-122B	51.69	53.02	+1.33	70.00	69.67	−0.33
NapMem-397B	54.42	56.23	+1.81	78.33	79.67	+1.34
NapMem-9B w/ RL	<u>59.92</u>	<u>61.03</u>	+1.11	**80.33**	**80.33**	+0.00

LongMemEval, holds the highest overall average, and is second on LoCoMo behind Mem0. The conclusions in Section 4 therefore do not depend on the particular judge model.

D Navigation Trajectory Patterns

Table 11 reports the distribution of navigation trajectories per benchmark, where a trajectory is the ordered sequence of memory levels the agent touches before answering. We write *Rec* for memory-record tools, *Conv* for raw-conversation tools, and *File* for reads of topic tracks or the user profile.

Three observations follow. First, navigation is short: the agent averages 1.82–2.24 tool calls although the budget allows four, so it is not enumerating the interface. Second, one pattern dominates on every benchmark, *Rec* → *Conv* → *Answer*, covering 32–43% of trajectories. The agent's default strategy is to locate compact records and then descend to raw conversations to verify them, which is exactly the movement the provenance links are built to support and which passive retrieval cannot express. Third, the benchmarks differ in where they depart from that default, and they depart in the expected direction. File-level access is common on PersonaMem-v2 (*Rec* → *File* → *Answer*, 15.00%), marginal on LoCoMo (5.65%), and absent from the top patterns on LongMemEval, matching the fact that only PersonaMem-v2 asks about stable preferences. LongMemEval instead has the shortest trajectories, the lowest multi-level rate, and the only sizeable share of direct raw-conversation answers (*Conv* → *Answer*, 13.33%), consistent with questions answerable from a single located exchange. One policy therefore yields different access strategies across benchmarks without any per-benchmark tuning.

Comparing against the aggregated statistics in Table 6, the untrained 9B model searches more (3.97 calls) and hits supporting evidence less often (20.66%). The 122B and 397B models also use more calls than the trained 9B agent (2.50 and 2.69) while landing on supporting evidence less often (26.20% and 26.42%). Per-benchmark navigation depth is reported only for the trained agent, in Table 11.

Table 11: Navigation trajectory statistics of NapMem-9B w/ RL. Multi-level is the fraction of queries touching more than one memory level. *Rec*: memory records; *Conv*: raw conversations; *File*: topic tracks or user profile.

Benchmark	Avg. tools	Multi-level	Most frequent trajectories
LoCoMo	2.24	74.86%	Rec → Conv → Answer (38.89%) Rec → Answer (15.94%) Conv → Rec → Answer (12.57%) Rec → File → Answer (5.65%)
LongMemEval	1.82	57.33%	Rec → Conv → Answer (32.33%) Rec → Answer (26.33%) Conv → Answer (13.33%) Conv → Rec → Answer (12.00%)
PersonaMem-v2	1.92	69.05%	Rec → Conv → Answer (43.14%) Rec → Answer (23.20%) Rec → File → Answer (15.00%) Rec → Conv → File → Answer (3.00%)

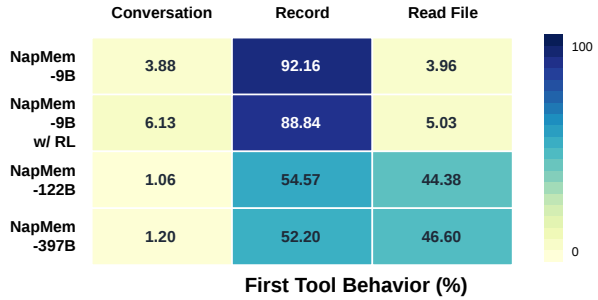

Figure 5: Distribution of the first memory tool invoked, across NapMem variants. Larger untrained variants more often open topic tracks or the profile first, while the 9B variants start from record-level tools, which return shorter and more targeted evidence.

E Offline Memory Cost

Construction cost in Table 4 is measured over 50 PersonaMem-v2 users and normalized per user, with all systems built by the same Qwen3.5-9B model (Appendix C.1) on identical hardware, so the comparison reflects the construction procedure rather than the builder. NapMem sits between the flat-extraction systems and the graph-based Zep: record extraction is followed by deduplication and by rewriting topic tracks and the profile, but this stays far below Zep's entity and edge resolution. MemOS is the cheapest to build and the most expensive to store, since it performs little abstraction and keeps largely unprocessed content. Table 12 breaks the footprint down by benchmark; the relative ordering is stable across datasets, so the aggregate comparison is not driven by any single benchmark.

F Case Study

Figure 6 shows a successful case where upward navigation helps recover missing context. Low-level search over raw messages and records returns noisy or underspecified evidence, while the topic track summarizes a Tampa beach visit and points the agent to the correct answer. This case illustrates that higher-level memory can help when direct retrieval fails to surface the needed evidence. Figure 7 shows a complementary failure mode. The agent retrieves relevant records, but the evidence is not strong enough to support the exact premise of the selected option. In particular, it over-generalizes an event-level memory about a successful salsa fundraiser into a stable preference for weekend salsa dance socials. Future navigation policies should better calibrate evidence strength, especially when converting event evidence into user preferences.

Table 12: Memory-bank storage size in GiB. Values in parentheses denote storage relative to NapMem.

Dataset	NapMem	Mem0	MemOS	Zep	MemoryOS	AgeMem
PersonaMem-v2	0.69	1.40 (2.03×)	3.61 (5.23×)	3.50 (5.07×)	1.98 (2.87×)	0.34 (0.49×)
LongMemEval	4.01	8.90 (2.22×)	18.85 (4.70×)	10.00 (2.49×)	4.30 (1.07×)	2.56 (0.64×)
LoCoMo	0.12	0.14 (1.17×)	0.64 (5.33×)	0.60 (5.00×)	0.40 (3.33×)	0.09 (0.75×)
Total	4.83	10.44 (2.16×)	23.10 (4.78×)	14.10 (2.92×)	6.68 (1.38×)	2.99 (0.62×)

Success Case: Upward Navigation to Topic-Level Memory

Query. *What state did Nate visit?*

Memory Trajectory.

`search_conversation`(*"Nate visit state travel"*)
→ no useful state-level evidence

`search_records`(*"Nate visit state location travel destination"*)
→ mostly irrelevant or underspecified records

`read_file`(`Nate-Identity-Interests.md`)

Relevant topic-track evidence.

- **Basic User Information:** Location includes "United States (Tampa beach visit Nov 11)."
- **Evolution Trajectory:** Nate traveled to Tampa beach for peace and relaxation on 2022-11-11.
- **Memory Evidence:** `m_Nate_s0029_2_1_1` records a trek to Tampa beach during a turtle trip.

Final answer. Florida, because Tampa is in Florida.
Judge. Correct.

Figure 6: Success case.

G Scope and Future Directions

Our evaluation spans three long-term memory benchmarks and three non-memory benchmarks, covering factual recall, cross-session reasoning, preference understanding, general reasoning, and both textual and visual tool use. Extending this to interactive and multi-modal settings, where the agent both reads and writes memory during a live session, is a natural next step.

Several extensions follow directly from the design of the framework. The rubric currently scores answer correctness and memory-use appropriateness; adding an explicit abstention criterion would let the policy also learn to recognize when memory does not contain the requested information. Because construction is bottom-up and provenance-linked, the same machinery can support user-configurable retention policies and periodic verification of upper-level entries against their supporting evidence, which would strengthen deployment on real user histories. Memory construction is also more involved than flat extraction (Table 4); it runs offline and incrementally, so it does not affect interaction latency, but reducing its cost would further improve practicality. Finally, broader scaling studies could characterize how model size shapes the memory-access strategies a policy prefers.

H Prompt

H.1 Judge Prompt

This prompt is used for LoCoMo and LongMemEval judged accuracy during evaluation and RL reward computation.

Failure Case: Over-generalizing Event Evidence into a Stable Preference

Query. *Do you have suggestions for peaceful weekend activities that can help me recharge and spark inspiration?*

Answer options. A. Since you already love spending weekends at lively salsa dance socials, you might enjoy adding a recharge element by strolling through the Pilsen murals in the afternoon, or sipping café de olla at a local spot while jotting down memories from your favorite dance nights.
B. Since you already love spending weekends on your favorite hobbies, you might enjoy adding a peaceful twist by taking morning walks in nature, or relaxing at a local café with a notebook, letting your thoughts wander into new ideas.
C. Since you already love spending weekends at silent creative writing retreats, you might enjoy extending that peace with sunrise walks along the lakefront, or taking a quiet afternoon to photograph hidden corners of the city. Pair that with journaling in a cozy coffee shop, letting the rhythms of the neighborhood inspire fresh story ideas.
D. Since you already love spending weekends experimenting with new vegan baking recipes, you might enjoy extending that creativity by visiting local farmers markets and chatting with growers about seasonal ingredients, or taking a peaceful afternoon to sketch in a sunny park while your latest bread rises at home."

Memory trajectory.

`search_records`(*weekend activities, hobbies, salsa*)
→ `search_records`(*Gabriela, preferences, relaxation*)
→ `get_records`([`m_persona_893_s0001_3_2_2`])
→ `get_records`(*energy from packed events record*)
→ Answer A

Retrieved evidence.

- A record states that the user considered changing a fundraiser entertainment lineup to salsa because the previous year's salsa set was successful.
- Another record states that the user feels energized by packed big events and community connections, but also needs alone time afterward to recover.
- Other retrieved records include forgetting constraints or negative preferences, such as no longer liking certain solitary or outdoor activities.

Failure mode. The agent treats a successful salsa set at a fundraiser as evidence that the user personally loves weekend salsa dance socials. This over-generalizes an event-level memory into a stable preference and makes option A appear more strongly supported than the evidence warrants.

Takeaway. Relevant memories may still be too weak to support the exact premise of an answer option. Future navigation policies should better distinguish event evidence from stable preference evidence and avoid upgrading weak associations into personal traits.

Figure 7: Failure case where relevant retrieved memories lead to over-generalization from an event-level memory to a stable user preference.

Open-QA LLM Judge Prompt

System:
You are a strict but fair evaluator for open-domain QA. Decide whether the predicted answer correctly answers the question given the ground truth answer. Accept paraphrases, aliases, equivalent dates, and equivalent numbers. Ignore extra explanation unless it contradicts the ground truth. Output only 1 if the prediction is correct; output only 0 otherwise.
User:
Question: {question}
Ground truth answer: {ground_truth_answer}
Predicted answer: {predicted_answer}
Return only 1 or 0.

H.2 Memory Pyramid Prompts

Figure 8–11 show the abbreviated prompt templates used to construct the memory pyramid, covering memory record extraction, record reconciliation, topic-track writing, and user-profile updating.

Memory Record Extraction Prompt

You are a scene segmentation and memory extraction expert.
Given the previous scene, background messages, and newly appended messages, perform two tasks:
Scene segmentation. Identify the current conversation scene. Inherit the previous scene if there is no clear topic transition; otherwise create a new concise scene name.
Memory extraction. Extract memory records only from the newly appended messages. Each memory must be self-contained, grounded in source messages, and belong to one of four types: *fact*, *event*, *instruction*, or *preference*.
Do not extract trivial small talk, temporary one-off requests without reusable information, repeated content, or unsupported subjective feelings. Preserve answerable details such as names, dates, numbers, prices, locations, status changes, and completed actions.
Output. Return a JSON array. Each item contains a scene name, message ids, and extracted memories:

```
[
  {
    "scene_name": "...",
    "message_ids": ["..."],
    "memories": [
      {
        "content": "...",
        "type": "fact|event|instruction|preference",
        "priority": 80,
        "source_message_ids": ["..."],
        "metadata": {}
      }
    ]
  }
]
```

Output only valid JSON and no extra text.

Figure 8: Abbreviated prompt template for memory record extraction.

H.3 Inference Prompts

Figure 12 shows the abbreviated prompt templates used to instruct the model to actively navigate the memory space.

H.4 Artifact Licenses

Table 13 summarizes the licenses of the external datasets, benchmarks, and model checkpoints used in this work. We use all artifacts for research and evaluation purposes, and follow the corresponding license terms and citation requirements.

Memory Record Reconciliation Prompt

You are a memory conflict detector.
Given a set of newly extracted memory records and a retrieved candidate pool of existing records, decide how each new record should be handled.
Actions. For each new record, choose one of:

- `store`: add the new record as new information.
- `skip`: discard the new record because it is redundant or less informative.
- `update`: replace older records when the new record is newer, more specific, or corrects outdated information.
- `merge`: combine complementary records that describe the same fact, state, event, instruction, or evolving process.

Decision criteria. Compare records by subject, theme, time, and scene context. Cross-type merging is allowed when records describe the same underlying information. Preserve exact values such as names, dates, prices, quantities, item lists, locations, and status changes.
Provenance. For `update` and `merge`, the final record must preserve the union of source message identifiers, timestamps, and metadata from the new and replaced records.
Output. Return a JSON array, one decision per new record:

```
[
  {
    "record_id": "...",
    "action": "store|skip|update|merge",
    "target_ids": ["..."],
    "merged_content": "...",
    "merged_type": "fact|event|instruction|preference",
    "merged_priority": 85,
    "merged_timestamps": ["..."],
    "merged_source_message_ids": ["..."],
    "merged_metadata": {}
  }
]
```

Output only valid JSON and no extra text.

Figure 9: Abbreviated prompt template for memory record reconciliation.

Topic Track Writing Prompt

You are a memory consolidation writer.
Given newly committed memory records, the existing topic-track index, and the current timestamp, consolidate the records into coherent topic tracks.
Goal.
Build medium-range user contexts rather than a list of isolated records. Each topic track should summarize a recurring user topic, such as an ongoing project, preference domain, relationship, routine, or long-term goal.
Actions.
Prefer updating an existing topic track when possible. Create a new track only when the new records cannot be integrated into any existing topic. Merge related tracks when they describe the same topic or narrative arc. The total number of topic tracks should not exceed 20.
Writing requirements.
Each topic track should be written as a Markdown document with metadata, a short summary, a narrative description, and an evidence section. The narrative should integrate new and existing information coherently rather than appending records mechanically.
Provenance.
Every integrated memory must be grounded with its supporting memory record identifier in the evidence section.
Output behavior.
Use file operations to read, update, create, or merge topic tracks. Finish the round after all necessary edits are completed.

Figure 10: Abbreviated prompt template for topic-track writing.

User Profile Update Prompt

You are a user-profile synthesis writer.
Given the existing user profile, newly changed topic tracks, update statistics, and the current timestamp, revise the global user profile.
Evidence constraints. Use only information grounded in topic tracks. Do not infer user information from file paths, workspace structure, system metadata, or unsupported assumptions. If evidence is insufficient, omit the field rather than speculate. Keep the output language consistent with the source records.
Synthesis goals. Build a concise user-level narrative by connecting information across topic tracks. Focus on stable information rather than episodic details. Scan for:

- **Base anchors**: confirmed facts, demographic traits, and current state.
- **Interest graph**: topics, activities, or domains that receive sustained attention.
- **Interaction protocol**: communication style, workflow preferences, and response constraints.
- **High-level patterns**: recurring decision logic, long-term tendencies, and supported evolution.

Output. Return a Markdown user profile within 2000 characters. The profile may contain sections for basic information, long-term preferences, current context, interaction protocol, and high-level evolution. Revise the existing profile rather than rewriting from scratch, preserving useful stable information and updating outdated content when new evidence supports the change.

Figure 11: Abbreviated prompt template for user-profile updates.

Inference Prompt

You are a personal assistant. Your goal is to answer the user's question correctly and grounded in evidence.
Workflow. First, determine whether the current information is sufficient. If the question concerns the user and the evidence is insufficient, use the structured memory bank through tools. Stop using tools once enough evidence has been gathered.
Tool budget. Use no more than four tool-calling turns per question. Do not continue searching for marginal extra evidence.
Memory bank. The memory bank contains four layers:

- **Raw conversations**: original dialogue evidence; use `search_conversation` or `get_conversation`.
- **Memory records**: compact structured memories; use `search_records` or `get_records`.
- **Topic tracks**: topic-level evolving summaries; use `read_file`.
- **User profile**: global user-level summary; use `read_file`.

Grounding. Do not guess user information. If retrieved memories conflict, strict instruction records take precedence. Use raw conversations when exact evidence is needed.
Final answer. Only when evidence is sufficient, return the answer in the required JSON format:

```
{"answer": "...", "reason": "..."}
```

Figure 12: Abbreviated prompt template for inference-time memory navigation.

Table 13: Licenses of external artifacts used in this work.

Artifact	Type	License	Use in this work
LoCoMo	Long-term conversational memory benchmark	CC BY-NC 4.0	Used for long-term user-memory evaluation and part of memory-tool RL training. The license is non-commercial, so we use it for research evaluation only.
PersonaMem-v2	Personalized user-memory benchmark	MIT	Used for implicit preference and persona-memory evaluation, and part of memory-tool RL training.
LongMemEval	Long-term memory benchmark	MIT	Used as a held-out out-of-distribution memory benchmark.
V*Bench	Visual tool-use benchmark	MIT project license	Used for non-memory visual tool-use evaluation. The official repository is MIT-licensed; image assets should be used following their original source terms when redistributed.
GPQA-Diamond	Scientific reasoning benchmark	MIT	Used as a non-memory reasoning benchmark.
BFCL-v3	Function-calling benchmark	Apache-2.0	Used as a non-memory text tool-use benchmark.
Qwen3.5-9B / 122B / 397B	Base language models	Apache-2.0	Used as base LLMs for NapMem and scaling variants.