Learning to Remember: End-to-End Training of Memory Agents for Long-Context Reasoning

Anonymous ACL submission

Abstract

Long-context LLMs and Retrieval-Augmented Generation (RAG) systems process information passively, deferring state tracking, contradiction resolution, and evidence aggregation to query time, which becomes brittle under ultra-long streams with frequent updates. We propose the Unified Memory Agent (UMA), an end-to-end reinforcement learning framework that unifies memory operations and question answering within a single policy. As a key technical contribution, we further propose Task-Stratified Group Relative Policy Optimization (GRPO) to optimize heterogeneous objectives across memory maintenance and QA with more stable and effective credit assignment. UMA maintains a dual memory representation: a compact core summary for global context and a structured Memory Bank that supports explicit CRUD (create, read, update, delete) over key-value entries, enabling proactive consolidation during streaming. To evaluate long-horizon memory behavior, we introduce Ledger-QA, a diagnostic benchmark for continuous state tracking where answers are latent values derived from accumulated updates rather than local span retrieval. Across 13 datasets spanning Ledger-QA, Test-Time Learning (TTL), and Accurate Retrieval (AR), UMA outperforms long-context and RAG baselines on dynamic reasoning and learning tasks while remaining competitive on standard retrieval benchmarks, highlighting the value of learned, end-to-end memory management.

1 Introduction

Context windows of Large Language Models (LLMs) now span millions of tokens(Google, 2025; Anthropic, 2025; OpenAI, 2025; Qwen, 2025; Liu et al., 2024). However, long-context models face fundamental challenges beyond window size. First, computational complexity scales quadratically with sequence length in standard attention mechanisms(Vaswani et al., 2017), making processing extremely long contexts prohibitively expensive. Second, empirical studies report *context rot*(Hong et al., 2025): as input length increases, performance degrades even when relevant information is present, because critical details become harder to locate and use.

Retrieval-Augmented Generation (RAG) addresses scalability by retrieving only relevant passages at query time(Xu et al., 2024), but inherits a deeper limitation shared with long-context approaches: **passive information processing**. Both paradigms treat context as static raw material, deferring all reasoning to query time. As illustrated in Figure 1, RAG systems must reprocess raw retrieved passages for every query. For reasoning-intensive tasks (e.g., state tracking, contradiction resolution, and evidence aggregation), this causes accuracy degradation and computational redundancy.

Memory-augmented agents offer a fundamentally different paradigm: proactive reasoning that consolidates information as it arrives rather than reprocessing raw inputs per query. However, existing approaches face critical limitations. **Heuristic-based systems** like Mem0(Chhikara et al., 2025), Zep(Rasmussen et al., 2025) rely on hand-crafted prompts and memory schemas, lacking task adaptivity. **Decoupled architectures** such as Memory-R1(Yan et al., 2025) and Mem-α (Wang et al., 2025b) separate memory management from task execution through independent training, preventing end-to-end optimization—credit from task outcomes cannot flow back to shape memory decisions.

We propose **Unified Memory Agent (UMA)**, an end-to-end reinforcement learning framework that jointly optimizes memory operations (create, update, delete, reorganize) and task execution within a unified policy. To diagnose proactive reasoning capabilities, we introduce **Ledger-QA**, a benchmark

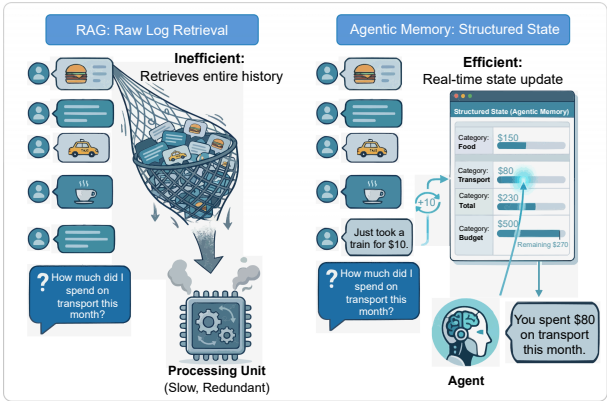

Figure 1: Expense-tracking example: RAG reprocesses retrieved logs per query, while an agentic memory system maintains a structured state and answers by reading the relevant fields.

requiring continuous state tracking where performance directly reflects the ability to consolidate information proactively rather than reprocess raw history per query. We conduct extensive evaluation across 13 datasets spanning three task categories: continuous reasoning (Ledger-QA), Test-Time Learning (TTL), and Accurate Retrieval (AR). Experiments show that UMA outperforms long-context and RAG baselines on dynamic reasoning tasks while remaining competitive on standard retrieval benchmarks, with ablations supporting the importance of end-to-end optimization.

2 Method

2.1 Problem Formulation as MDP

We formulate the long-context reasoning task as a Markov Decision Process (MDP). To handle sequences exceeding the context window, we decompose the input into a stream of chunks $C = \{c_1, \ldots, c_n\}$. The MDP tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ is defined as follows:

State Space ($\mathcal{S}$). At step t, the state $s_t = (\mathcal{M}_t, x_t, h_t)$ consists of:

- $\mathcal{M}_t = (m^{core}, \mathcal{B})$: The dual-component memory, comprising a high-level summary (m^{core}) and a structured key-value bank ($\mathcal{B}$).
- x_t: The current input focus. To circumvent context window limits, this is dynamically switched between a local text chunk c_k (during maintenance) and a specific query q (during QA).
- h_t: The immediate interaction history, which buffers recent observations and is periodically cleared to manage context length.

Action Space ($\mathcal{A}$). We unify all operations into a single action space, categorized by the phase of interaction. Crucially, read-only tools are available in both phases to ensure the agent can verify beliefs or query the ledger at any time. Detailed specifications for all tools (inputs/outputs) are provided in Appendix A.

- **Memory Operations ($\mathcal{A}_{mem}$):** Used during the chunk processing phase to maintain the state.
  - `Add`, `Update`, `Delete`: Write operations to modify the structured bank $\mathcal{B}$.
  - `Retrieve` (read) and `List`: Read-only operations to inspect keys or values in $\mathcal{B}$ before modification.
  - `UpdateCore`: A special terminal action that commits the current progress to m^{core} and triggers the transition to the next chunk.
- **QA Operations ($\mathcal{A}_{qa}$):** Used during the query phase to synthesize the answer.
  - `BM25(query)`, `Embedding(query)`: Hybrid retrieval over raw text chunks C.
  - `Retrieve(key)`, `List()`: Access to the structured memory bank $\mathcal{B}$.
  - `Answer(text)`: The terminal action to generate the final response.

Transition Dynamics ($\mathcal{P}$). The system evolves through two nested loops plus a final phase switch:

- **Inner Loop (within one session/chunk):** The agent performs multiple tool calls before committing the chunk. Each step appends interaction traces (tool calls and observations) to h_t and may update the memory bank $\mathcal{B}$, while the current input focus $x_t = c_k$ remains unchanged.
- **Outer Loop (across chunks):** A terminal `UpdateCore` action ends the current inner-loop episode, writes the consolidated update to m^{core}, advances x_t from c_k to c_{k+1}, and re-initializes h_t for the next chunk.
- **Phase Transition:** After processing c_n, the system switches from maintenance to QA, where x_t becomes a query q sampled from $P(q|C)$.

This transition structure implies that the policy should both optimize short-horizon inner-loop edits on $\mathcal{B}$ and optimize outer-loop consolidation into m^{core}, so that the final memory state $\mathcal{M}_n$ supports future queries.

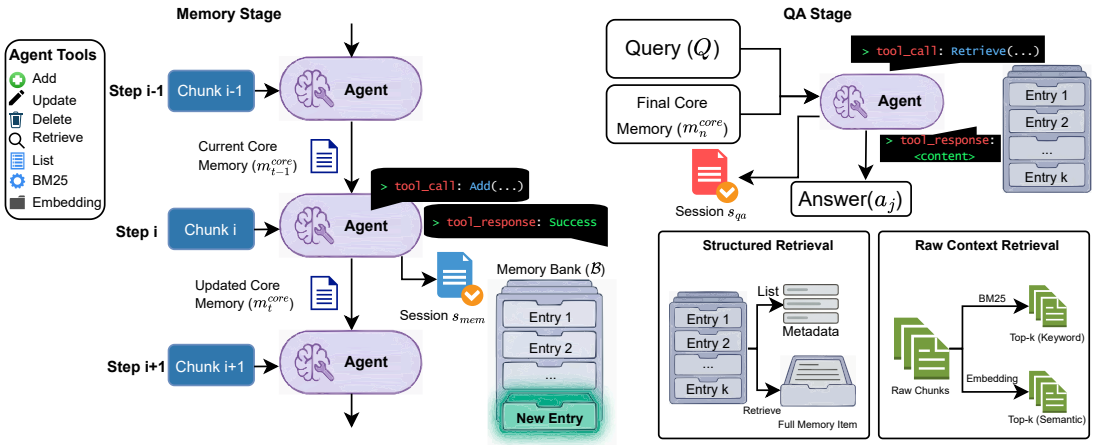

Figure 2: **Overview of UMA.** Phase I incrementally maintains a structured Memory Bank and core summary via CRUD over chunks; Phase II answers queries using both structured retrieval from the bank and raw-context retrieval.

2.2 Unified Memory Agent Architecture

Based on the MDP formulation, UMA employs a unified policy $\pi_\theta(a_t|s_t)$ to generate actions. The architecture is designed to handle the inputs explicitly as follows:

Input Representation. At any time step t, the input context provided to the LLM is a concatenation of four segments:

$$\text{Input}_t = [I_{sys}, m_t^{core}, x_t, h_t] \tag{1}$$

- **System Instruction (I_{sys}):** Specifies the phase-specific operational guidelines—governing either memory maintenance or question answering—and defines the available toolset ($\mathcal{A}_{mem}$ or $\mathcal{A}_{qa}$).
- **Core Memory (m_t^{core}):** The evolving high-level summary that provides global context.
- **Current Focus (x_t):** The raw text of the current chunk c_k (in Phase I) or the user query q (in Phase II).
- **Interaction History (h_t):** The trajectory of recent tool calls and their execution results.

The Unified Memory Agent framework contains two phases, as illustrated in Figure 2.

Phase I: Sequential Memory Maintenance. For each chunk c_k, the agent initializes $h_t = \emptyset$. It iteratively samples actions

$$a_t \sim \pi_\theta(\cdot|I_{sys}, m^{core}, c_k, h_t) \tag{2}$$

The environment executes the tool and appends the result to h_t. This loop continues until the agent outputs `UpdateCore`, which compresses relevant insights into m^{core} and advances the process to c_{k+1}.

Phase II: Hybrid Retrieval-Augmented QA. After processing the final chunk, the input focus x_t switches to the query q. The agent utilizes the fully constructed memory $\mathcal{M}_n$ and an empty history h_t. It can invoke hybrid retrieval tools (accessing both the structured bank via `Retrieve` and raw text via `BM25/Embedding`) to gather evidence. The process terminates when the agent generates the `Answer` action.

2.3 Training: Task-Stratified GRPO

We propose a **Task-Stratified Group Relative Policy Optimization** algorithm. By viewing GRPO as a Monte Carlo (MC) advantage estimator without a value network, we adapt it to handle the hierarchical structure of our task as illustrated in Figure 3.

2.3.1 Nested Trajectory Sampling

To estimate the value of memory actions effectively, we employ a nested sampling strategy during training. For a given context C:

1. **Memory Rollout (trajectory level):** We sample N distinct trajectories $\{\tau^1, \ldots, \tau^N\}$. Each trajectory τ^i contains multiple memory-tool-calling sessions $\{s_{mem}^{i,1}, s_{mem}^{i,2}, \ldots, s_{mem}^{i,n}\}$ and yields one final memory state $\mathcal{M}_n^i$, where n is the number of chunks in C, which also equals the number of memory sessions in trajectory τ^i.
2. **Query Sampling:** For each final memory state $\mathcal{M}_n^i$, we sample M questions $\{q_1, \ldots, q_M\}$ from the dataset.

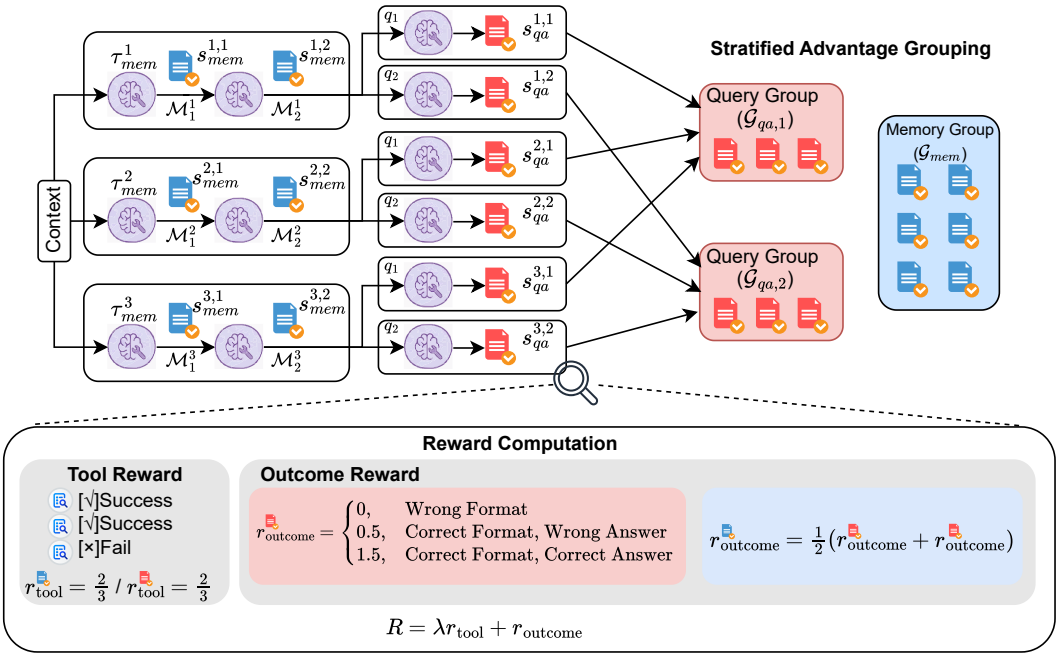

Figure 3: **Illustration of Task-Stratified GRPO.** For a given input, we first sample multiple *memory trajectories* (blue), each producing a final memory state; then for each memory state and query, we sample a *QA session* (red). **(Right)** The reward function combines immediate tool execution feedback (r_{tool}) with outcome assessments ($r_{outcome}$). Crucially, memory steps receive a *Future Utility Signal* derived from subsequent QA rewards. **(Bottom)** Advantages are normalized within distinct groups: all memory steps are aggregated into a global pool ($\mathcal{G}_{mem}$), while QA steps are normalized strictly within their specific query groups ($\mathcal{G}_{qa,j}$).

3. **QA Rollout (session level):** For each pair $(\mathcal{M}_n^i, q_j)$, we generate one multi-turn QA-tool-calling session $s_{qa}^{i,j}$ and compute its reward $R_{qa}^{i,j}$.

Thus, each update contains N memory trajectories and $N \times M$ QA sessions.

2.3.2 Reward Function

Rewards are computed at the *session* level, with different outcome assignments for QA and memory sessions.

For each QA session $s_{qa}^{i,j}$, the reward is

$$\begin{aligned} R_{qa}^{i,j} &= \lambda r_{tool}^{qa,i,j} + r_{outcome}^{i,j}, \\ r_{outcome}^{i,j} &= r_{outcome}(a_{ans}^{i,j}, y_j) \end{aligned} \tag{3}$$

where $r_{tool}^{qa,i,j}$ is the tool success signal of this QA session.

For each memory session $s_{mem}^{i,k}$ in trajectory τ^i, we keep an independent tool reward and use the averaged QA outcome under the same trajectory as its outcome signal:

$$\begin{aligned} \bar{r}_{outcome}^{i} &= \frac{1}{M} \sum_{j=1}^{M} r_{outcome}^{i,j}, \\ R_{mem}^{i,k} &= \lambda r_{tool}^{mem,i,k} + \bar{r}_{outcome}^{i} \end{aligned} \tag{4}$$

where $r_{tool}^{mem,i,k}$ is computed from the tool-call validity of memory session $s_{mem}^{i,k}$ (e.g., valid-call ratio N_{valid}/N_{total} for that session). Here, λ controls the trade-off between tool-use quality and answer outcome, and is set to 0.5 by default.

2.3.3 Task-Stratified Advantage Estimation

Given the session-level rewards defined above, we compute GRPO advantages via group-wise normalization. Our extension uses two stratified groups for memory and QA, respectively.

Advantage for Memory Steps (A_{mem}). For memory steps, we assign advantages at the *session level*. Specifically, each memory-session reward is normalized against all memory sessions sampled in the current update:

$$\begin{aligned} A_{mem}^{i,k} &= \frac{R_{mem}^{i,k} - \bar{R}_{mem}}{\sigma(R_{mem}^{\cdot,\cdot})}, \\ \bar{R}_{mem} &= \frac{1}{Nn} \sum_{u=1}^{N} \sum_{v=1}^{n} R_{mem}^{u,v} \end{aligned} \tag{5}$$

This provides fine-grained credit assignment for each memory session.

Advantage for QA Steps (A_{qa}). For a specific question q_j, the difficulty is intrinsic. Therefore, we normalize each QA-session reward only against other sessions answering the *same* question q_j (but with different constructed memory banks).

$$A_{qa}^{i,j} = \frac{R_{qa}^{i,j} - \bar{R}_{qa,j}}{\sigma(R_{qa}^{\cdot,j})}, \quad \bar{R}_{qa,j} = \frac{1}{N}\sum_{k=1}^{N} R_{qa}^{k,j} \quad (6)$$

This isolates the reasoning capability from the question difficulty, ensuring fair credit assignment.

Finally, the policy π_θ is updated by maximizing a unified GRPO objective over all sampled sessions:

$$\mathcal{J}(\theta) = \frac{1}{Nn + NM}\big(\mathcal{S}_{mem} + \mathcal{S}_{qa}\big) \quad (7)$$

$$\begin{aligned} \mathcal{S}_{mem} &= \sum_{i=1}^{N}\sum_{k=1}^{n}\sum_{t \in s_{mem}^{i,k}} \ell(\rho_t, A_{mem}^{i,k}), \\ \mathcal{S}_{qa} &= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t \in s_{qa}^{i,j}} \ell(\rho_t, A_{qa}^{i,j}) \end{aligned} \quad (8)$$

$$\begin{aligned} \ell(\rho_t, A) = \ & \min\big(\rho_t A, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A\big) \\ & - \beta D_{KL,t} \end{aligned} \quad (9)$$

Here, $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{ref}(a_t|s_t)}$ denotes the importance sampling ratio between the current policy π_θ and the behavior policy π_{ref} used for trajectory collection, and $D_{KL,t} = D_{KL}(\pi_\theta \| \pi_{ref})_t$.

3 The Ledger-QA Benchmark

Existing long-context benchmarks (e.g., LongMemEval) primarily focus on *retrieval-centric* tasks, where the answer is explicitly located within a specific text span. In contrast, many applications require *state tracking*: continuously updating a latent structured state from sequential, partially noisy observations. To isolate this core requirement, we introduce **Ledger-QA**, a controllable and extensible synthetic benchmark for long-horizon state updates. Rather than claiming full realism, Ledger-QA is intentionally built with simple, transparent generation rules that capture key aspects of state tracking while remaining challenging for modern long-context agents.

3.1 Dataset Construction

The construction pipeline contains three simple programmatic stages:

Long-Horizon Timeline Simulation. We sample transaction events over a continuous timeline (e.g., one year) and organize each example as a chronological stream of N sessions. Sessions are date-stamped to induce temporal dependencies across updates.

Natural Dialogue Synthesis. For each session, we use a strong LLM (e.g., Gemini-3-Pro) to verbalize sampled events into natural user-assistant dialogue. The surface form is linguistically diverse (e.g., chit-chat, clarifications, informal phrasing), while the underlying state-transition rules remain deliberately simple. This setup preserves natural language difficulty while keeping benchmark semantics explicit.

Ground Truth & Query Generation. In parallel, we maintain a structured list of `Transaction` objects as exact supervision. We then generate queries that require multi-step aggregation over accumulated updates (instead of local span matching), covering 8 question categories.

A concrete construction example and full generation details are provided in Appendix B.

3.2 Why Ledger-QA is Challenging

Ledger-QA fundamentally differs from standard RAG benchmarks by demanding **long-horizon state aggregation** rather than local information retrieval. Answers (e.g., "Total spending") are latent values derived from summing scattered transactions ($v_1 + \cdots + v_n$) across the timeline. This requires continuous belief-state maintenance: local extraction errors can propagate to final answers, and simple retrieval or approximate summarization is often insufficient. We view this benchmark as an initial step toward systematic evaluation of long-horizon state tracking, with future work needed to further improve real-world realism.

4 Experiment

4.1 Experimental Setup

4.1.1 Evaluation Benchmarks

We evaluate UMA on three complementary categories: **Dynamic State Tracking** (our Ledger-QA in Section 3), **Test-Time Learning (TTL)** (*TREC-Coarse*, *TREC-Fine*, *NLU*, *Clinic*, *Banking77*, *PubMed-RCT*), and **Accurate Retrieval (AR)** (*HotpotQA*, *LoCoMo*, *LongMemEval*, *MSC*, *PearlTQA*, *SQuAD*, *ConvoMem*) (Hu et al., 2025; Yang et al., 2018; Maharana et al., 2024; Wu et al.,

2025; Xu et al., 2022; Du et al., 2024; Rajpurkar et al., 2016; Pakhomov et al., 2025).

Task focus. Ledger-QA emphasizes persistent state maintenance under continual updates; TTL tests whether the agent can store and apply task rules from in-context supervision; AR evaluates fine-grained factual recovery from long documents or conversations. This combination covers both proactive memory construction and precise downstream access.

Evaluation metric. We use an LLM-as-a-Judge protocol with `DeepSeek-V3.2`. Prompt templates, scoring criteria, and dataset taxonomies are provided in Appendix C and Appendix D.

4.1.2 Baselines

We compare UMA against a diverse set of baselines. First, we evaluate standard approaches: **Concat**, which inputs the full context to test the backbone's native capacity, and **RAG**, which retrieves the top-20 chunks using a hybrid strategy combining dense embeddings and sparse BM25 via Reciprocal Rank Fusion (RRF). Second, we include recurrent summarization models: **MemAgent** (Yu et al., 2025) iteratively updates a running memory state based on the query, while **MemAgent-woq** performs query-agnostic compression, challenging the agent to identify salient information without guidance. Finally, we compare against advanced agentic frameworks: **Mem1** (Zhou et al., 2025) interleaves chunk reading with dynamic retrieval to maintain an evolving internal state, **Mem-α** (Wang et al., 2025b) utilizes an RL-optimized policy to manage a composite memory system (core, episodic, and semantic), and **A-MEM** (Xu et al., 2025) is an additional external-memory baseline with explicit memory operations. Implementation details are provided in Appendix E.

4.1.3 Implementation Details

Training uses a mixed corpus combining multi-hop retrieval data (HotpotQA-derived), general long-context data (adapted from Mem-α), and synthetic Ledger-QA state-tracking data; statistics are in Appendix F. For dense retrieval, we use `sentence-transformers/all-MiniLM-L6-v2` served via `infinity` (Feil, 2023). We implement UMA on customized `veRL` (Sheng et al., 2024), initialize from `Qwen/Qwen3-4B-Instruct`, and optimize with Task-Stratified GRPO on 32 NVIDIA H200 GPUs. Hyperparameters are reported in Appendix G.

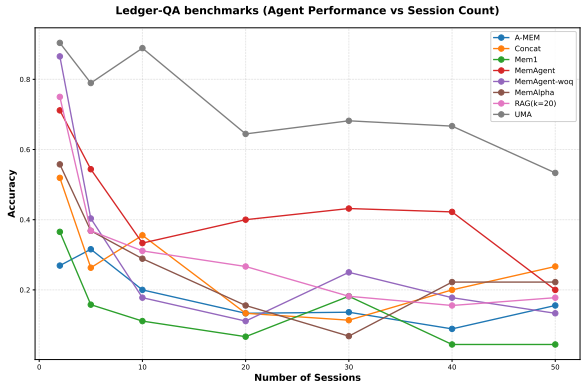

Figure 4: **Performance comparison on Ledger-QA across varying session counts.** The x-axis represents the number of dialogue sessions (simulating increasing time horizons), and the y-axis denotes accuracy. **Detailed numerical results are provided in Table 11 in Appendix J.**

4.2 Main Results

Table 1 presents the comparative performance across 13 standard benchmarks (6 TTL and 7 AR tasks). We observe that:

Superiority in Test-Time Learning (TTL). Our Unified Memory Agent shows strong performance on TTL tasks (e.g., Banking77, Clinic, TREC-Fine), and remains competitive with or superior to prior memory-agent baselines. Notably, UMA performs well on Banking77 and Clinic, which are unseen during training, suggesting good generalization.

Robustness in Retrieval & Reasoning. On Accurate Retrieval (AR) tasks, our method remains competitive; efficiency and scalability analysis is provided in Appendix I.

4.3 Results on Ledger-QA

To evaluate the capability of dynamic state tracking over long horizons, we test all baselines on the Ledger-QA benchmark with varying session counts $N \in \{2, 5, \ldots, 50\}$. As shown in Figure 4, the results reveal a distinct separation in performance trends as the timeline extends.

Vulnerability of Baselines. In short-horizon settings ($N \leq 5$), standard approaches like RAG and MemAgent-woq achieve competitive accuracy (70–90%), indicating that static retrieval or simple

	Method	Test-Time Learning (TTL)						Accurate Retrieval (AR)							Avg.
		Bank77	Clinic	NLU	Pub	T-C	T-F	Convo	Hotpot	LoCo	LME	MSC	Perl	SQuAD	
Baselines	Concat-16k	72.00	79.00	71.00	57.80	71.00	18.00	2.46	63.28	30.01	31.60	58.40	36.00	63.38	50.30
	RAG-16k	80.00	70.00	65.00	51.20	9.00	14.00	51.12	78.12	54.03	**59.60**	54.60	**83.75**	25.35	53.52
	MemAgent	25.00	13.00	22.00	49.90	61.00	37.00	59.60	51.56	**60.37**	55.40	50.20	65.25	77.02	48.25
	MemAgent-woq	27.00	41.00	34.00	44.40	70.00	44.00	1.56	11.72	25.43	7.60	29.40	23.50	25.01	29.59
	Mem1	0.00	6.00	0.00	9.00	7.00	0.00	1.34	5.47	21.60	6.40	8.40	7.25	15.09	6.73
	A-MEM	83.00	81.00	72.00	53.70	36.00	16.00	52.46	67.97	45.87	31.80	62.80	73.00	**88.12**	58.75
	Mem-α	83.00	77.00	68.00	56.50	78.00	62.00	7.14	**79.69**	44.26	51.60	56.80	74.50	81.92	63.11
Ablation	UMA (w/o RL & Phase I)	26.00	27.00	23.00	39.30	13.00	12.00	79.02	64.06	50.40	38.60	36.20	57.25	82.33	42.17
	UMA (w/o Phase I)	84.00	80.00	75.00	47.40	81.00	28.00	86.16	69.53	44.21	48.80	27.40	59.00	83.47	62.61
	UMA (w/o RL)	77.00	72.00	61.00	38.20	69.00	20.00	84.82	64.84	44.26	48.60	63.80	51.50	81.19	59.71
	UMA (w/o Raw-Context)	69.00	71.00	59.00	36.30	71.00	44.00	73.12	60.15	40.59	35.80	62.20	52.00	76.52	57.74
	UMA (Global Group)	69.00	77.00	**84.00**	**72.50**	88.00	53.00	81.47	60.94	56.55	37.60	**74.40**	73.00	81.98	69.96
	UMA (2 Stage)	86.00	84.00	70.00	44.60	77.00	43.00	84.82	67.19	42.75	49.20	65.60	60.75	65.59	64.65
	UMA (8k Window)	79.00	88.00	75.00	65.60	95.00	70.00	80.80	64.06	44.91	44.80	64.60	71.25	85.81	71.45
	UMA (32k Window)	90.00	92.00	84.00	68.60	97.00	83.00	89.06	74.22	47.48	49.20	67.00	69.00	87.32	76.76
	UMA	**87.00**	**91.00**	82.00	67.20	**94.00**	**75.00**	**87.05**	76.56	46.58	51.60	66.20	74.50	87.53	**75.86**

Table 1: Main results and ablations (Accuracy %). **Pub**: PubMed-RCT; **Convo**: ConvoMem; **LoCo**: LoCoMo; **LME**: LongMemEval; **T-C/T-F**: TREC-Coarse/Fine. Detailed ablation settings and additional metrics (EM/F1/ROUGE and LLM-as-a-Judge with `Qwen3-30B-A3B-Instruct-2507`) are provided in Section 4.4 and Appendix H.

summarization can be sufficient for limited contexts. However, performance drops markedly as N increases. RAG falls below 20% at $N = 50$, indicating difficulty in aggregating scattered updates. Similarly, despite moderate mid-range performance, MemAgent declines to $\sim$20% at longer horizons. These trends suggest that without a persistent structured store, neither retrieval nor implicit compression reliably sustains long-horizon consistency.

Robustness of Unified Memory Agent. In contrast, UMA (gray line) remains comparatively stable. It outperforms baselines by a clear margin, particularly in the long-horizon regime ($N \geq 20$). Even at the most challenging setting of 50 sessions, UMA maintains accuracy above 50%, whereas the best baseline remains below 25%. This evidence indicates that explicit memory operations help decouple state maintenance from sequence length, enabling more robust bookkeeping in ultra-long contexts.

4.4 Ablation Studies

To disentangle the contributions of our unified architecture and the reinforcement learning paradigm, we conduct a comprehensive ablation analysis as shown in the bottom section of Table 1.

Ablation Settings. **w/o Phase I** removes sequential memory maintenance and relies on QA-time retrieval only. **w/o RL** keeps the full architecture but removes RL optimization. **w/o RL & Phase I** removes both mechanisms. **w/o Raw-Context** disables `BM25/Embedding` and only allows MemoryBank retrieval at inference. **Global Group** normalizes advantages across all steps without task stratification. **2 Stage** trains memory and QA separately (no end-to-end coupling). **8k/32k Window** changes context budget from the 16k default to 8k or 32k.

Impact of Memory Maintenance (Phase I). We first examine the necessity of the sequential memory maintenance phase by comparing the full architecture against the retrieval-only variants.

- **Without RL:** Even before RL optimization, enabling explicit memory maintenance yields a clear gain over the retrieval-only variant, indicating that proactive write/update operations already improve long-horizon retention.
- **With RL:** The advantage remains after training: the full UMA still outperforms the **w/o Phase I** variant by a substantial margin (13.3 points on average), showing that retrieval alone is insufficient for sustained state tracking.

Impact of RL Training. Next, we evaluate the effectiveness of our Task-Stratified GRPO by comparing the base models with their RL-trained counterparts.

- **On Retrieval-Only Agent:** RL consistently improves the retrieval-only setting, showing that the reward design helps the policy use retrieval tools more effectively.

- **On Unified Agent:** RL also provides a large gain for the full architecture (+16.2 points on average), suggesting that training is crucial for learning stable memory lifecycles such as timely updates and accurate lookups.

Impact of Stratified Advantage Grouping. We further validate our Task-Stratified GRPO by comparing it with a **Global Group** variant that normalizes advantages across all steps indiscriminately. As shown in Table 1, the Global Group variant is worse than our stratified design (5.9 points lower on average). This is consistent with our claim that mixing heterogeneous rewards (binary QA outcomes vs. aggregated memory utility) weakens the learning signal.

Two-Stage vs. Unified End-to-End Training. We further investigate whether UMA can be learned via a decoupled two-stage pipeline that trains memory maintenance and QA separately (**UMA (2 Stage)**). This variant remains noticeably below the jointly optimized UMA (11.2 points on average), indicating a clear train–test mismatch: isolated optimization of Phase I cannot anticipate downstream QA utility, and Phase II cannot retroactively improve upstream memory decisions.

Effect of Context Window Size. We additionally evaluate UMA under different context-window budgets. Moving from 8k to 16k yields a clear improvement, while increasing further to 32k gives only modest additional gains. This indicates diminishing returns at larger windows despite higher compute and memory cost, so we use 16k as the default trade-off.

Ablation: w/o Raw-Context Retrieval. We add an ablation that disables raw-context retrieval tools (`BM25/Embedding`) and keeps only Memory-Bank access at inference. The drop is substantial on retrieval-heavy TTL/AR tasks, but minimal on Ledger-QA. This supports two points: (i) Ledger-QA mainly measures long-horizon *state tracking* rather than local span lookup; (ii) hybrid retrieval remains important for fine-grained evidence access, and key-only bank retrieval is relatively weaker for detailed factual grounding.

5 Related Work

Internal Memory and Context Compression. Several approaches extend context by modifying model internals, such as embedding information in parameters (Wang et al., 2025a; Zhang et al., 2025), compressing KV caches (Qian et al., 2025; Li et al., 2024), or utilizing soft prompts (Burtsev et al., 2020; Ge et al., 2023). While offering high inference efficiency, these methods require white-box access and suffer from inherent capacity bottlenecks compared to external architectures (Wang and Chen, 2025), limiting their utility for ultra-long horizons.

Prompt-based External Memory. To bypass internal limits, frameworks like MemGPT (Packer et al., 2023) and SCM (Wang et al., 2023) manage external storage via OS-like hierarchies or graph schemas. While compatible with black-box models, they rely heavily on zero-shot instruction following. This creates a dependency on large proprietary models, as smaller models struggle to reliably execute complex memory updates via prompting alone (Wang et al., 2025b), often leading to suboptimal retrieval in dynamic settings.

Trainable Memory Agents. Recent work shifts towards *training* models for autonomous memory management. Approaches include recurrent compression via RL (Yu et al., 2025), interleaving retrieval with reasoning (Zhou et al., 2025), and utilizing composite memory structures (Yan et al., 2025; Wang et al., 2025b). Unlike our unified approach, these methods typically decouple memory maintenance from reasoning or rely on transient summaries. Our work advances this domain by unifying memory operations and reasoning into a single policy, enabling more effective cross-session state tracking.

6 Conclusion

We propose the Unified Memory Agent, an end-to-end framework that jointly optimizes memory acquisition and utilization via reinforcement learning. Task-Stratified GRPO aligns the heterogeneous objectives of long-horizon memory maintenance and downstream question answering. To address the lack of benchmarks for dynamic state tracking, we introduce **Ledger-QA**, a dataset requiring persistent updates to an evolving ledger. Experimental results show that our agent outperforms baselines on state-tracking tasks while remaining competitive on standard retrieval benchmarks. Ablation studies indicate the importance of the learned policy and explicit memory updates.

Limitations

While our Unified Memory Agent shows promising capabilities, several limitations remain. First, our current study focuses on single-agent settings with a fixed tool interface. Extending UMA to collaborative multi-agent environments or dynamically evolving tool APIs is an important direction for improving robustness and generality. Second, our evaluation primarily emphasizes long-context QA and state-tracking workloads; broader validation on additional real-world interactive domains (e.g., open-ended planning and multi-session human-agent workflows) is still needed. Finally, our training relies on a specific set of tools (CRUD + Retrieval); extending this framework to support multi-modal memory (e.g., images, audio) or more complex tool-use patterns requires further investigation.

References

Anthropic. 2025. Claude opus 4.5. https://www.anthropic.com/news/claude-opus-4-5. Accessed: 2026-01-04.

Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. 2020. Memory transformer. *arXiv preprint arXiv:2006.11527*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. 2024. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and synthesis in question answering. *arXiv preprint arXiv:2402.16288*.

Michael Feil. 2023. Infinity - to embeddings and beyond.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Google. 2025. Gemini 3. Technical report, Google DeepMind. Accessed: 2026-01-04.

Kelly Hong, Anton Troynikov, and Jeff Huber. 2025. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.

OpenAI. 2025. Gpt-5.2. https://openai.com/index/introducing-gpt-5-2/. Accessed: 2026-01-04.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *CoRR*.

Egor Pakhomov, Erik Nijkamp, and Caiming Xiong. 2025. Convomem benchmark: Why your first 150 conversations don't need rag. *arXiv preprint arXiv:2511.10523*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025*, pages 2366–2377.

Qwen. 2025. Qwen3 technical report: Unified thinking and non-thinking modes for massive scaling. Technical report, Alibaba Group. ArXiv:2511.21631 [cs.CL].

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

you need. *Advances in neural information processing systems*, 30.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Scm: Enhancing large language model with self-controlled memory framework. *arXiv e-prints*, pages arXiv–2304.

Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*.

Yu Wang, Xinshuang Liu, Xiusi Chen, Sean O'Brien, Junda Wu, and Julian McAuley. 2025a. Self-updatable large language models by integrating context into model parameters. In *The Thirteenth International Conference on Learning Representations*.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025b. Mem-α: Learning memory construction via reinforcement learning. *arXiv e-prints*, pages arXiv–2509.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 5180–5197.

Peng Xu, Wei Ping, Xianchao Wu, and 1 others. 2024. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv e-prints*, pages arXiv–2507.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

A Tool Definitions

Our agent interacts with the environment and its internal state via a structured API. Below we provide the detailed specifications for each tool available in the **Memory Toolset** ($\mathcal{T}_{mem}$) and the **Retrieval Toolset** ($\mathcal{T}_{qa}$).

A.1 Memory Toolset ($\mathcal{T}_{mem}$)

This toolset is used during the *Sequential Memory Maintenance* phase to manage the persistent Memory Bank ($\mathcal{B}$).

- `Add(key: str, value: str)` $\rightarrow$ `str`: Creates a new entry in the memory bank. **Input:** A unique title (`key`) and the content (`value`). **Output:** Returns "Success" if the key is new; returns an error message if the key already exists.
- `Update(key: str, value: str)` $\rightarrow$ `str`: Modifies an existing entry. **Input:** The target `key` and the new `value`. **Output:** Returns "Success" if the update is applied; returns an error if the key is not found.
- `Delete(key: str)` $\rightarrow$ `str`: Removes an entry. **Input:** The target `key`. **Output:** Returns "Success" upon deletion; returns an error if the key does not exist.
- `Retrieve(key: str)` $\rightarrow$ `str`: Reads the full content of a specific entry. **Input:** The target `key`. **Output:** Returns the stored `value` string; returns an error if the key does not exist.
- `List()` $\rightarrow$ `List[str]`: Scans the memory bank. **Input:** None. **Output:** Returns a list of all current keys (titles), enabling the agent to survey stored topics before performing operations.

A.2 Retrieval Toolset ($\mathcal{T}_{qa}$)

Used during the *Hybrid Retrieval-Augmented QA* phase to fetch information from both the structured memory and the raw context. Note that `Retrieve` and `List` are also available in this phase to access the Memory Bank.

- `Embedding(query: str, top_k: int)` → `List[str]`: Performs dense retrieval over the raw context chunks. **Input:** A search query and the number of chunks to retrieve (`top_k`). **Mechanism:** Encodes the query using `sentence-transformers/all-MiniLM-L6-v2` and retrieves the top-k chunks based on cosine similarity. **Output:** A list of the most semantically relevant text chunks.

- `BM25(query: str, top_k: int)` → `List[str]`: Performs sparse retrieval over the raw context chunks. **Input:** A search query and the number of chunks to retrieve (`top_k`). **Mechanism:** Uses the BM25 algorithm to match keywords. **Output:** A list of the top-k chunks with the highest keyword overlap.

B Details of Ledger-QA Construction

The construction of the Ledger-QA benchmark employs a three-stage pipeline designed to simulate realistic, long-horizon bookkeeping scenarios. In our implementation, we adopt a **joint synthesis strategy**: we provide the LLM with high-level user intents (e.g., "wants to buy coffee"), and the LLM simultaneously generates both the natural dialogue and the corresponding structured transaction records. This design helps ensure that details such as amounts and descriptions are contextually appropriate and naturally woven into the conversation.

B.1 Consumption Scene Taxonomy

To simulate a diverse personal finance environment, we define a hierarchical taxonomy of consumption scenes. The dataset covers 8 major categories, each with multiple sub-scenes:

- **Dining:** Fast Food, Restaurant, Coffee, Bubble Tea, BBQ, Hot Pot, Snacks, Takeout.
- **Transportation:** Subway, Bus, Taxi, Gas, Parking, Train, Flight.
- **Shopping:** Clothing, Electronics, Daily Necessities, Cosmetics, Books, Groceries, Furniture.
- **Entertainment:** Movie, KTV, Gaming, Gym, Travel, Concert, Escape Room.
- **Utilities:** Water & Electricity, Property Fee, Phone Bill, Internet, Gas Bill, Rent.
- **Medical:** Medicine, Doctor Visit, Health Checkup, Dental, Glasses.
- **Education:** Training Course, Books & Materials, Online Course, Exam Registration, Tuition.
- **Other:** Transfer, Red Envelope, Donation, Pet, Beauty & Salon.

B.2 Data Generation Pipeline

Stage 1: Timeline & Scenario Sampling. For each sample, we first generate a chronological timeline spanning a full year. We randomly sample N dates (sessions) from this range. For each session, we stochastically select a subset of consumption scenes (e.g., *Dining-Coffee*, *Transportation-Taxi*) to form the user's "intent" for that day.

Stage 2: Joint Dialogue & Transaction Synthesis. We feed the selected date and scene intents into an advanced LLM (e.g., Gemini-3-Pro) to obtain natural language realizations while keeping generation controls explicit. The model is instructed to act as a user-assistant pair and generate a JSON object containing two aligned fields:

1. **Dialogue:** The natural conversation turns.
2. **Transactions:** The structured ground truth (Amount, Description, etc.) corresponding to the dialogue.

Crucially, we do not pre-determine the amounts. The LLM decides reasonable amounts based on the context (e.g., $5 for coffee vs. $500 for a flight), ensuring semantic consistency.

Prompt for Joint Synthesis (Simplified)
Please generate a natural conversation between a user and an AI assistant about expense tracking. **Context:** - Date: {date} - The user wants to record: {selected_scenes} - *[Instruction: First Session vs. Continuation]* **Requirements:** - The dialogue should be natural, containing {min_turns}-{max_turns} turns. - Each consumption scene needs a specific amount (**you decide a reasonable amount**). - Return JSON with two fields: 1. "dialogue": List of turns. 2. "transactions": List of records (scene, amount, description, date).

Table 2: Prompt template used for the joint generation of dialogue and structured data.

Example Multi-Session Sample from Stage 2 (Excerpt). Table 3 shows a shortened multi-session sample generated by the Stage-2 joint synthesis pipeline. We include two sessions from different dates, aligned transaction records with explicit date fields, and key metadata, to illustrate the temporal structure used by Ledger-QA.

Session Dates: 2024-02-23 and 2024-04-29
Dialogue Excerpts (cross-date) `[2024-02-23][User]` Hi there! I'd like to start using this chat to track my daily expenses. `[2024-02-23][Assistant]` Great initiative. I can help log each expense with category and amount. `[2024-02-23][User]` I paid my monthly Internet bill. It cost 75. `[2024-02-23][Assistant]` Got it. Should I categorize this under Utilities? `[2024-02-23][User]` Yes, please put it under Utilities. `[2024-02-23][Assistant]` Recorded: Utilities–Internet, 75. Do you have other expenses? . . . `[2024-04-29][User]` I have a few more receipts. First, I took a cab to the city center. `[2024-04-29][Assistant]` Transportation–Taxi noted. How much was the fare? `[2024-04-29][User]` 25.5. Also, a clinic co-pay (50), a gas bill (85.20), and skincare products (120). `[2024-04-29][Assistant]` Done. I've saved all four transactions for April 29th. . . .
Aligned Structured Transactions (date, scene, subscene, amount) 1) (2024-02-23, Utilities, Internet, 75.00) 2) (2024-02-23, Transportation, Subway, 2.90) 3) (2024-02-23, Entertainment, Escape Room, 40.00) 4) (2024-02-23, Education, Tuition, 1500.00) 5) (2024-02-23, Transportation, Bus, 1.50) 6) (2024-04-29, Transportation, Taxi, 25.50) 7) (2024-04-29, Medical, Doctor Visit, 50.00) 8) (2024-04-29, Utilities, Gas Bill, 85.20) 9) (2024-04-29, Shopping, Cosmetics, 120.00)
Sample Metadata num_sessions=2; num_turns=48; num_transactions=9; date_range=[2024-02-23, 2024-04-29]; total_amount=1900.10.

Table 3: Shortened multi-session Stage-2 generation example. The excerpt preserves cross-date dialogue evidence, date-aware transaction records, and core metadata.

Stage 3: Question Generation. We extract the ground-truth transaction ledger generated in Stage 2 and programmatically generate questions across 8 distinct templates.

The question types include:

1. **Time-Range Scene Amount:** "How much was spent on Dining from Jan to Mar?"
2. **Time-Range Multi-Scene:** "Total spending on Dining and Travel in Q1?"
3. **Global Total:** "What was the total spending across all records?"
4. **Max Scene:** "Which category had the highest spending?"
5. **Max Frequency Date:** "On which date were there the most transactions?"
6. **Max Single Amount:** "What was the largest single transaction?"
7. **Point Query:** "How much was spent on Coffee on 2024-05-01?"
8. **Single Date Scene Amount:** "How much was spent on Dining on 2024-05-01?"

B.3 Dataset Statistics

To evaluate the agent's robustness across varying temporal horizons, we constructed a comprehensive suite of datasets ranging from short-term interactions (2 sessions) to long-horizon histories (50 sessions).

Training Set. We utilize the 10-session configuration for training, consisting of 200 samples with over 6,000 questions. This provides a balanced difficulty level for the agent to learn state-tracking policies.

Test Set. The test set is stratified by session count ($N \in \{2, 5, 10, 20, 30, 40, 50\}$). Detailed statistics for each configuration are presented in Table 4.

B.4 Generation Algorithm

The core logic of our joint synthesis pipeline is formalized in Algorithm 1.

C Evaluation Prompts

To ensure consistent and fair evaluation, we employ an LLM-as-a-Judge protocol with two independent evaluators: `DeepSeek-V3.2` (main results) and `Qwen3-30B-A3B-Instruct-2507` (supplementary comparison). The specific prompts used for different benchmarks are detailed below.

General Prompt. For all benchmarks except LongMemEval, we utilize a unified evaluation template designed to assess semantic correctness:

> "I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response is equivalent to the correct answer or contains all the intermediate steps to get the correct answer, you should also answer yes. If the response only contains a subset of the information required by the answer, answer no.
>
> **Question:** {question}
>
> **Correct Answer:** {answer}
>
> **Model Response:** {prediction}
>
> Is the model response correct? Answer yes or no only."

LongMemEval Prompts. Following the official implementation (Wu et al., 2025), we apply task-specific prompts to handle the nuances of different reasoning types (e.g., temporal reasoning, knowledge updates).

Algorithm 1 Ledger-QA Data Generation Pipeline

Require: Date range $[D_{start}, D_{end}]$, Number of sessions N, Diversify flag $diversify$
Ensure: Context chunks $\mathcal{C}$, Question set $\mathcal{Q}$, Ground-truth Ledger $\mathcal{L}$

```
D ← SampleSortedDates(D_start, D_end, N)
L ← ∅                                          ▷ Initialize global transaction ledger
C ← ∅                                          ▷ Initialize dialogue history
for i ← 1 to N do
    d_i ← D[i]
    S_intent ← SampleScenes(Taxonomy)          ▷ Randomly select user intents
    P ← FormatPrompt(d_i, S_intent, is_first = (i == 1))   ▷ Construct prompt for joint synthesis
    Response ← LLM_gen(P)                      ▷ Generate dialogue and transaction simultaneously
    Dialogue_i, Trans_i ← ParseJSON(Response)
    C.append(FormatChunk(d_i, Dialogue_i))
    L.extend(Trans_i)
end for
Q_raw ← ApplyTemplates(L, Types = {1..8})      ▷ Generate questions based on the full ledger
if diversify then                              ▷ Optional: Diversify phrasing
    Q ← LLM_paraphrase(Q_raw)
else
    Q ← Q_raw
end if
return C, Q, L
```

- **Default / Fact Retrieval:** Same as the General Prompt above.
- **Temporal Reasoning:** Adds tolerance for off-by-one errors:

  > "...In addition, do not penalize off-by-one errors for the number of days. If the question asks for the number of days/weeks/months, etc., and the model makes off-by-one errors (e.g., predicting 19 days when the answer is 18), the model's response is still correct..."

- **Knowledge Update:** Allows inclusion of previous information if the update is correct:

  > "...If the response contains some previous information along with an updated answer, the response should be considered as correct as long as the updated answer is the required answer..."

- **Single-Session Preference:** Evaluates personalization based on a rubric:

  > "...Please answer yes if the response satisfies the desired response... The response is correct as long as it recalls and utilizes the user's personal information correctly.
  > **Rubric:** {rubric}..."

- **Abstention:** Checks for correct identification of unanswerable questions:

  > "...Please answer yes if the model correctly identifies the question as unanswerable. The model could say that the information is incomplete...
  > **Explanation:** {explanation}
  > Does the model correctly identify the question as unanswerable? Answer yes or no only."

D Evaluation Dataset Details

We employ a diverse set of benchmarks to evaluate the agent's capabilities. Detailed statistics for all evaluation datasets are summarized in Table 4. Below we detail the preprocessing and construction logic for each dataset, corresponding to the implementation in our codebase.

D.1 Dynamic State Tracking

Ledger-QA (Synthetic). As detailed in Appendix B, we generate synthetic bookkeeping data. In our experiments, we utilize the configuration of 10 sessions per sample to balance complexity and evaluation speed.

D.2 Test-Time Learning (TTL)

For TTL tasks, we source data from **MemoryAgentBench** (Hu et al., 2025). To ensure the agent understands the classification nature of these tasks, we prepend a specific instruction hint to each query if not already present:

Category	Dataset	Samples	Avg. Qs	Avg. Len (Tok)	Avg. Chunks
State Tracking	Ledger-QA (2 sessions)	5	20.6	1,693	2.0
	Ledger-QA (5 sessions)	4	30.8	4,204	5.0
	Ledger-QA (10 sessions)	3	34.3	8,192	10.0
	Ledger-QA (20 sessions)	3	34.0	16,487	20.0
	Ledger-QA (30 sessions)	3	34.3	24,772	30.0
	Ledger-QA (40 sessions)	3	35.0	32,792	40.0
	Ledger-QA (50 sessions)	3	35.0	40,610	50.0
Test Time Learning	Banking77	1	100.0	127,568	111.0
	Clinic150	1	100.0	130,702	38.0
	NLU	1	100.0	134,060	115.0
	PubMed-RCT	10	100.0	16,724	10.0
	TREC-Coarse	1	100.0	123,606	111.0
	TREC-Fine	1	100.0	125,531	108.0
Accurate Retrieval	ConvoMem	7	64.0	**522,602**	300.0
	HotpotQA	128	1.0	27,945	200.0
	LoCoMo	10	198.6	22,161	27.2
	LongMemEval	500	1.0	107,863	47.8
	MSC (Batch=5)	100	5.0	9,853	25.0
	PearlTQA	4	100.0	13,038	23.0
	SQuAD	30	96.8	10,561	10.0

Table 4: Detailed statistics of all evaluation datasets. **Avg. Len (Tok)** denotes the average total context length in tokens. **Avg. Chunks** denotes the average number of context chunks per sample processed by the agent.

"Sentence: {original_query}
What are the labels for the above sentence?"

D.3 Accurate Retrieval (AR)

HotpotQA. Following MemAgent (Yu et al., 2025), we ensure strict data isolation by using non-overlapping training and validation splits. We inject ground-truth documents into 200 shuffled distractors, requiring the agent to retrieve evidence from the chunked stream to solve multi-hop queries without any risk of data leakage.

LoCoMo. We adapt prompts for specific question categories to ensure valid evaluation:

- **Temporal (Cat 2):** We append *"Use DATE of CONVERSATION to answer with an approximate date"* to guide relative time reasoning.
- **Adversarial (Cat 5):** We convert these into a multiple-choice format *"(A) {distractor} (B) Not mentioned"* (with randomized order). The agent must select *"Not mentioned"* to demonstrate correct rejection of unanswerable queries.

LongMemEval. We utilize the `longmemeval_s` subset. Questions are timestamped (e.g., `[2024-01-01] Question...`) to provide temporal context.

MSC (Multi-Session Chat). We use a modified version of MSC introduced by MemGPT (Packer et al., 2023) provided at https://huggingface.co/datasets/MemGPT/MSC-Self-Instruct. To increase context length, we batch 5 base samples into a single evaluation instance (`batch_size=5`). Each instance thus contains 25 sessions (5 samples $\times$ 5 sessions), and the agent is queried about specific details from any of the batched conversations.

ConvoMem. We filter the `pre_mixed_testcases` provided at https://huggingface.co/datasets/Salesforce/ConvoMem/tree/main to select only those with a total conversation length exceeding 2 million characters to ensure they qualify as long-context tasks. Each conversation turn is formatted as "{Speaker}: {Text}". We choose the `user_evidence` category for evaluation.

E Baseline Implementation Details

We implement the baseline methods **Mem1** and **Mem-α** using the codebase provided at https://github.com/wangyu-ustc/Mem-alpha. For all experiments, we enforce a strict context window limit of 16k tokens. When the total input length (including memory/context, query, and instructions) exceeds this limit, we apply left-truncation to the memory or context sequence, retaining only the most recent tokens to fit within the 16k window.

Backbone Models. To isolate the contribution of the memory mechanism, we standardize the backbone model where possible:

- **Standard Backbone:** Unless otherwise specified (e.g., for Concat, RAG, MemAgent), we utilize `Qwen/Qwen3-4B-Instruct-2507` for both the context processing and final QA phases.
- **Specialized Memory Models:** For baselines that require specific pre-trained weights for memory operations, we follow their original configurations:
  - **Mem1:** Uses `Mem-Lab/Qwen2.5-7B-RL-RAG-Q2-EM-Release` for the memory formation phase.
  - **Mem-α:** Uses `YuWangX/Memalpha-4B` for the memory formation phase.

  Crucially, for the final Question Answering phase, both Mem1 and Mem-α revert to the standard `Qwen/Qwen3-4B-Instruct-2507` backbone to ensure a fair comparison of reasoning capabilities.

F Training Data Details

Table 5 summarizes the detailed statistics of the datasets utilized in our training phase.

Data Sources. Our training data is curated from three main sources: (1) **HotpotQA (Modified):** We construct a retrieval-intensive version of HotpotQA following the protocol in MemAgent (Yu et al., 2025), augmenting each sample with 200 documents. (2) **Ledger-QA (Synthetic):** We generate a specialized dataset for dynamic state tracking as described in Section 3. (3) **Mem-α Corpus:** For all other datasets (SQuAD, PerLTQA, NLU, TREC, PubMed, BookSum), we directly incorporate the high-quality training corpus provided by Mem-α (Wang et al., 2025b) to ensure broad coverage of reasoning tasks (provided at https://huggingface.co/datasets/YuWangX/Memalpha-full).

Data Isolation. We strictly enforce data isolation to prevent leakage. For **HotpotQA**, we follow the MemAgent protocol, constructing our training and evaluation sets from the original HotpotQA training and validation splits, respectively. For **Ledger-QA**, the training and test samples are synthesized independently with distinct seed parameters. For the **Mem-α Corpus**, we utilize the official training split provided by the authors, ensuring zero overlap with the evaluation benchmarks used in our experiments.

Consistent with our dual-track training strategy (see Appendix G), the Generalist Agent is trained on the combination of HotpotQA and Mem-α corpus to foster broad generalization, while the Specialist Agent is fine-tuned specifically on the Ledger-QA dataset. To maintain training efficiency, for Mem-α corpus containing a large number of queries per instance (e.g., SQuAD, NLU), we randomly sample up to 10 queries per training sample.

G Training Configuration

We implement our training framework based on a customized version of `veRL` (Sheng et al., 2024), optimized for multi-conversation and multi-turn agent rollouts. The policy model is initialized from `Qwen/Qwen3-4B-Instruct` and optimized using AdamW with a learning rate of 1×10^{-6}. We employ the Task-Stratified Group Relative Policy Optimization (GRPO) algorithm with a group size of $G = 16$ samples per context and a KL divergence penalty coefficient of $\beta = 0.001$.

To balance generalization across diverse tasks with the specific demands of intensive state tracking, we adopt a dual-track training strategy:

- **Generalist Agent (for TTL & AR):** To ensure robust performance across broad retrieval and reasoning tasks, this model is trained on the combined Multi-Hop and General Long-Context datasets for 2 epochs.
- **Specialist Agent (for Ledger-QA):** Given the unique requirement for continuous numerical updates and long-horizon consistency in Ledger-QA, we train a separate model specifically on the Ledger-QA training set. This model is trained for 6 epochs to fully adapt to the state-tracking dynamics, allowing us to probe the architectural upper bound for this specialized domain.

All experiments are conducted on a cluster of 32 NVIDIA H200 GPUs (4 nodes $\times$ 8 GPUs). To manage memory efficiency during long-horizon rollouts, we utilize vLLM for inference with a tensor parallelism size of 1 and set the GPU memory utilization to 0.5. The maximum sequence length for both prompts and responses is set to 8,192 tokens. The total training time spans approximately 2 days for the Generalist Agent and 0.5 days for the Specialist Agent.

Dataset	Samples	Avg. Qs	Avg. Len (Tok)	Avg. Chunks
HotpotQA	8192	1.0	25,667	61
SQuAD	264	10.0	10,780	10.0
PerLTQA	27	10.0	12,046	23.3
NLU	180	10.0	6,100	10.0
TREC-Coarse	180	10.0	3,900	10.0
PubMed-RCT	90	10.0	16,760	10.0
BookSum	1,387	1.0	15,328	8.0
Ledger-QA	200	34.0	8,254	10.0

Table 5: Detailed statistics of the training datasets. **Avg. Len (Tok)** denotes the average total context length in tokens. **Avg. Chunks** denotes the average number of context chunks per sample processed by the agent.

H Detailed Evaluation Metrics

In addition to the accuracy metric reported in the main text, we provide the Exact Match (EM), F1, ROUGE, and supplementary LLM-as-a-Judge scores to offer a more granular view of model performance. Table 6 presents the Exact Match scores, which measure the percentage of predictions that match the ground truth verbatim. Table 7 presents the F1 scores, which account for partial overlaps between the prediction and the reference answer. Table 8 presents the ROUGE-L scores, evaluating the lexical overlap and fluency of the generated responses. Table 9 further reports LLM-as-a-Judge summary scores using `Qwen3-30B-A3B-Instruct-2507` as the evaluator.

I Inference Efficiency and Resource Scalability

We benchmark inference latency on a single ConvoMem sample (528k-token context with 10 queries) using 8 NVIDIA A800 GPUs. Results are shown in Table 10. UMA reaches a practical sweet spot: it is substantially faster than recurrent baselines while preserving strong long-horizon reasoning capability.

The results indicate that UMA achieves a favorable efficiency–capability balance. Compared with recurrent baselines (MemAgent/Mem-α), UMA benefits from efficient chunk-wise processing and avoids substantial recurrent overhead while maintaining linear complexity $\mathcal{O}(N)$, unlike full-context models.

Resource Scalability. Beyond large-cluster training, we verify that the pipeline is compatible with smaller setups and can run on a single node with 8 NVIDIA A800 GPUs (80GB HBM each). Moreover, parameter-efficient fine-tuning methods (e.g., LoRA) offer a practical path to further reduce memory requirements, improving accessibility for broader research groups.

J Numerical Results on Ledger-QA

Table 11 provides the exact accuracy scores for the Ledger-QA benchmark across different session counts, corresponding to the trends illustrated in Figure 4.

	Method	Test-Time Learning (TTL)						Accurate Retrieval (AR)							
		Bank77	**Clinic**	**NLU**	**Pub**	**T-C**	**T-F**	**Convo**	**Hotpot**	**LoCo**	**LME**	**MSC**	**Perl**	**SQuAD**	**Avg.**
Baselines	Concat	66.00	79.00	71.00	57.80	70.00	18.00	0.45	30.47	13.95	18.20	27.00	5.50	40.68	38.31
	RAG ($k = 20$)	4.00	12.00	2.00	17.50	0.00	0.00	3.79	0.78	0.20	0.40	1.40	1.00	0.34	3.34
	MemAgent	24.00	13.00	23.00	11.70	61.00	37.00	2.90	14.06	2.92	8.80	5.20	1.00	3.31	15.99
	MemAgent-woq	27.00	41.00	34.00	0.00	69.00	42.00	0.00	0.00	0.05	0.00	0.40	0.00	0.17	16.43
	Mem1	0.00	4.00	0.00	0.00	7.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.85
	A-MEM	71.00	77.00	71.00	53.50	32.00	13.00	13.39	44.53	9.42	15.60	24.00	9.50	34.62	36.04
	Mem-α	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ablation	UMA (w/o RL & Phase I)	26.00	27.00	22.00	39.00	12.00	7.00	18.08	34.38	16.52	20.20	15.60	10.50	46.68	22.69
	UMA (w/o Phase I)	84.00	79.00	75.00	47.30	81.00	25.00	19.42	43.75	15.51	25.00	12.60	11.25	48.40	43.63
	UMA (w/o RL)	76.00	72.00	61.00	37.30	69.00	20.00	19.42	41.41	16.72	25.00	23.80	9.00	44.47	39.62
	UMA (w/o Raw-Context)	69.00	71.00	59.00	36.30	71.00	44.00	18.74	40.46	15.38	26.46	21.20	8.25	59.52	41.56
	UMA (Global Group)	69.00	77.00	**82.00**	**69.10**	87.00	47.00	19.20	40.62	**19.54**	14.80	28.40	6.50	30.73	45.45
	UMA (2 Stage)	84.00	84.00	69.00	44.30	77.00	43.00	19.20	43.75	13.75	24.40	28.00	11.00	37.62	44.54
	UMA (8k Window)	79.00	88.00	75.00	65.70	95.00	70.00	19.87	42.97	16.16	27.80	29.40	13.25	55.60	52.13
	UMA (32k Window)	90.00	92.00	84.00	68.60	97.00	83.00	21.65	47.66	18.23	31.20	31.20	12.25	56.46	56.40
	UMA	**87.00**	**91.00**	**82.00**	67.20	**95.00**	**75.00**	**20.98**	**48.44**	17.62	**30.60**	**29.20**	**12.25**	**56.32**	**54.82**

Table 6: Exact Match (EM) scores across all benchmarks. Values are percentages (%).

	Method	Test-Time Learning (TTL)						Accurate Retrieval (AR)							
		Bank77	**Clinic**	**NLU**	**Pub**	**T-C**	**T-F**	**Convo**	**Hotpot**	**LoCo**	**LME**	**MSC**	**Perl**	**SQuAD**	**Avg.**
Baselines	Concat	70.00	79.00	71.00	57.80	70.00	18.00	8.65	49.27	25.89	27.84	48.21	18.00	53.57	45.94
	RAG ($k = 20$)	25.12	30.82	20.90	18.92	3.63	2.04	38.66	15.57	16.90	17.62	23.33	13.92	5.29	17.90
	MemAgent	24.40	13.00	23.00	26.01	61.00	37.00	33.55	26.34	21.96	27.11	25.11	8.42	14.88	26.29
	MemAgent-woq	27.00	41.00	34.00	0.82	69.67	42.04	12.13	2.71	2.96	5.62	8.77	2.22	3.38	19.41
	Mem1	0.00	4.00	0.00	0.44	7.00	0.00	14.22	1.66	3.37	5.60	5.69	1.44	1.48	3.45
	A-MEM	79.00	79.00	71.67	53.57	32.00	13.00	37.70	61.99	28.61	28.84	49.10	30.05	57.46	47.87
	Mem-α	0.00	0.00	2.61	0.53	0.02	0.00	16.18	8.51	5.50	9.33	9.58	4.37	5.18	4.75
Ablation	UMA (w/o RL & Phase I)	26.00	27.50	22.00	40.20	12.00	7.00	40.94	52.73	36.70	28.82	28.36	26.01	62.08	31.56
	UMA (w/o Phase I)	84.67	79.25	75.00	47.35	81.00	25.00	44.21	59.53	34.57	36.66	21.08	28.22	63.18	52.28
	UMA (w/o RL)	78.12	72.00	61.67	39.64	69.00	20.00	43.38	56.81	32.96	36.32	46.73	24.01	60.38	49.31
	UMA (w/o Raw-Context)	69.00	71.00	59.00	36.30	71.00	44.00	43.72	56.64	30.78	33.86	40.20	29.39	59.52	49.57
	UMA (Global Group)	69.01	77.00	**82.01**	**69.11**	87.01	47.03	44.01	56.67	**40.18**	27.25	52.04	30.12	50.82	56.33
	UMA (2 Stage)	85.07	84.67	70.17	45.47	77.00	43.00	44.36	58.73	33.03	37.53	49.98	28.86	49.78	54.43
	UMA (8k Window)	79.00	88.00	75.00	65.70	95.00	70.00	44.16	56.51	37.05	37.34	51.03	32.61	70.64	61.69
	UMA (32k Window)	90.00	92.00	84.00	68.60	97.00	83.00	47.34	65.36	40.08	41.11	54.24	32.49	71.54	66.67
	UMA	**87.00**	**91.00**	82.00	67.20	**95.00**	**75.00**	**46.33**	**65.42**	39.39	**41.73**	**52.59**	**32.88**	**71.37**	**65.15**

Table 7: F1 scores across all benchmarks. Values are percentages (%).

	Method	Test-Time Learning (TTL)						Accurate Retrieval (AR)							
		Bank77	Clinic	NLU	Pub	T-C	T-F	Convo	Hotpot	LoCo	LME	MSC	Perl	SQuAD	Avg.
Baselines	Concat	70.00	79.00	71.00	57.80	70.00	18.00	8.63	50.44	25.97	27.76	48.79	18.36	54.00	46.13
	RAG ($k = 20$)	25.13	30.82	20.90	18.92	3.63	2.04	36.55	15.47	16.55	16.30	23.17	13.92	5.40	17.60
	MemAgent	24.40	13.00	23.00	26.04	61.00	37.00	32.31	25.71	21.51	26.59	25.17	8.35	15.15	26.10
	MemAgent-woq	27.00	41.00	34.00	0.83	69.67	42.04	10.80	2.74	2.88	4.81	8.78	2.33	3.44	19.25
	Mem1	0.00	4.00	0.00	0.45	7.00	0.00	11.63	1.77	3.28	4.70	5.36	1.54	1.54	3.17
	A-MEM	79.00	79.00	71.67	53.57	32.00	13.00	37.56	64.99	28.62	28.80	50.01	30.53	57.80	47.97
	Mem-α	0.00	0.00	2.61	0.54	0.02	0.00	14.33	8.49	5.26	8.43	9.71	4.29	5.22	4.53
Ablation	UMA (w/o RL & Phase I)	26.00	27.50	22.00	40.20	12.00	7.00	40.86	52.78	36.34	28.76	29.18	26.24	63.03	31.68
	UMA (w/o Phase I)	84.67	79.25	75.00	47.35	81.00	25.00	44.16	59.34	34.15	36.51	21.29	28.57	64.20	52.34
	UMA (w/o RL)	78.12	72.00	61.67	39.64	69.00	20.00	43.28	56.79	32.87	36.17	48.05	24.14	61.38	49.47
	UMA (w/o Raw-Context)	69.00	71.00	59.00	36.30	71.00	44.00	43.42	56.44	31.68	33.80	41.20	28.19	59.25	49.56
	UMA (Global Group)	69.01	77.00	**82.01**	**69.11**	87.01	47.03	43.91	57.61	**39.85**	27.33	52.41	29.97	51.05	56.37
	UMA (2 Stage)	85.07	84.67	70.17	45.47	77.00	43.00	44.40	59.47	32.72	37.40	50.89	29.36	50.59	54.63
	UMA (8k Window)	79.00	88.00	75.00	65.70	95.00	70.00	44.30	57.25	36.79	37.20	51.90	32.88	71.28	61.87
	UMA (32k Window)	90.00	92.00	84.00	68.60	97.00	83.00	47.44	66.01	39.85	40.81	54.97	32.93	72.34	66.84
	UMA	**87.00**	**91.00**	82.00	67.20	**95.00**	**75.00**	**46.39**	**66.16**	39.13	**41.52**	**53.28**	**33.14**	**72.04**	**65.30**

Table 8: ROUGE-L scores across all benchmarks. Values are percentages (%).

	Method	Test-Time Learning (TTL)						Accurate Retrieval (AR)							
		Bank77	Clinic	NLU	Pub	T-C	T-F	Convo	Hotpot	LoCo	LME	MSC	Perl	SQuAD	Avg.
Baselines	Concat	72.00	79.00	71.00	57.80	71.00	18.00	5.36	65.62	33.43	33.00	61.20	42.50	64.42	51.87
	RAG ($k = 20$)	81.00	70.00	65.00	53.30	10.00	10.00	51.34	77.34	57.35	**61.20**	55.80	**85.25**	25.56	54.09
	MemAgent	26.00	14.00	24.00	51.20	61.00	37.00	60.27	51.56	**64.15**	59.40	53.00	71.50	78.06	50.09
	MemAgent-woq	27.00	41.00	34.00	47.00	70.00	44.00	2.68	14.84	29.71	9.80	29.60	26.75	25.80	30.94
	Mem1	0.00	4.00	0.00	45.40	7.00	0.00	1.56	6.25	22.41	8.00	9.60	7.75	18.43	10.03
	A-MEM	83.00	80.00	72.00	53.70	32.00	15.00	53.35	72.22	50.25	51.61	64.21	80.00	**89.46**	61.29
	Mem-α	83.00	78.00	71.00	57.40	78.00	60.00	8.48	**78.91**	48.64	55.60	56.60	79.50	83.40	64.50
Ablation	UMA (w/o RL & Phase I)	34.00	28.00	24.00	40.20	23.00	15.00	79.91	62.50	54.88	41.00	38.20	65.50	82.23	45.26
	UMA (w/o Phase I)	84.00	79.00	74.00	47.70	81.00	25.00	86.61	71.09	47.43	51.40	28.80	69.75	83.71	63.81
	UMA (w/o RL)	76.00	72.00	61.00	38.80	70.00	20.00	85.49	66.41	47.99	50.80	66.20	62.50	80.71	61.38
	UMA (w/o Raw-Context)	71.00	73.00	63.00	40.30	73.00	45.00	71.43	62.50	44.86	37.60	63.00	51.25	71.97	59.07
	UMA (Global Group)	72.00	79.00	**82.00**	**74.00**	89.00	55.00	83.71	76.56	61.98	44.00	**75.40**	83.00	80.92	73.58
	UMA (2 Stage)	84.00	84.00	69.00	45.00	78.00	43.00	85.49	69.53	45.97	52.20	67.60	71.50	66.45	66.29
	UMA (8k Window)	79.00	88.00	75.00	65.70	95.00	70.00	83.04	66.41	48.04	46.80	65.40	79.75	86.53	72.97
	UMA (32k Window)	90.00	92.00	84.00	68.60	97.00	83.00	90.18	74.22	50.86	51.80	69.60	82.00	87.87	78.55
	UMA	**87.00**	**91.00**	**82.00**	67.20	**95.00**	**75.00**	**88.17**	77.34	50.10	54.80	68.00	82.50	87.32	**77.34**

Table 9: LLM-as-a-Judge summary scores using `Qwen3-30B-A3B-Instruct-2507` as the evaluator across all benchmarks. This table complements Table 1 by reporting the same benchmark setup under an alternative judge model.

Method	Latency (s)	Relative Speed
UMA	**107.32**	**1.0x**
Mem1	368.46	3.4x Slower
A-MEM	3251.75	30.3x Slower
Mem-α	1503.87	14.0x Slower
MemAgent-woq	2718.88	25.3x Slower
MemAgent	4753.26	44.3x Slower
Concat-16k	2.58	41.6x Faster
Concat-256k	61.16	1.8x Faster
RAG	58.01	1.9x Faster

Table 10: Inference latency comparison on a single ConvoMem sample (528k tokens context, 10 queries) using 8 NVIDIA A800 GPUs.

Method	Number of Sessions							Avg.
	2	5	10	20	30	40	50	
Concat	51.92	26.32	35.56	13.33	11.36	20.00	26.67	26.45
RAG ($k = 20$)	75.00	36.84	31.11	26.67	18.18	15.56	17.78	31.59
MemAgent	71.15	54.39	33.33	40.00	43.18	42.22	20.00	43.47
MemAgent-woq	86.54	40.35	17.78	11.11	25.00	17.78	13.33	30.27
Mem1	36.54	15.79	11.11	6.67	18.18	4.44	4.44	13.88
A-MEM	20.00	26.92	13.33	13.64	8.89	31.58	15.56	18.56
Mem-α	55.77	36.84	28.89	15.56	6.82	22.22	22.22	26.90
UMA	**90.38**	**78.95**	**88.89**	**64.44**	**68.18**	**66.67**	**53.33**	**72.98**

Table 11: Detailed accuracy scores (%) on Ledger-QA across varying session counts. Columns represent the number of sessions in the input stream.