Parameters as Memory: Internalizing Long-Horizon Memories for Efficient LLM Agents

Zhenheng Tang [*1] **Fanjunduo Wei** [*2] **Zichen Tang** [1] **Peijie Dong** [3] **Xiang Liu** [3] **Qian Wang** [4] **Xiaowen Chu** [3] **Bo Li** [1]

Abstract

LLM agents need long-term memory, but most current systems keep memory outside the model and repeatedly append retrieved text to the prompt, which burdens both long-context reasoning and serving efficiency. We propose Parameters as Memory (PAM), a high-level methodology that treats durable agent knowledge as memory that can be internalized into parameter-efficient adapters. Inspired by neuroscience views of replay and consolidation, we instantiate PAM with Reflective Memory Optimization (REMO), a data rewriting and training pipeline that treats reflection over dialogue history as memory-oriented optimization. REMO reconstructs overlapping dialogue windows, rewrites them through intrinsic, cross-turn, network-structuring, and interaction-oriented views, and then trains LoRA adapters with supervised fine-tuning so the model better internalizes the rewritten memories. On LoCoMo, PAM with REMO outperforms retrieval-only and learned-memory baselines, with especially strong gains on temporal reasoning and answer extractability.

1. Introduction

Large language model (LLM) agents are increasingly expected to act as persistent assistants rather than isolated prompt responders. As autonomous tasks stretch across many tool-use steps and sessions (Kwa et al., 2025; Model Evaluation and Threat Research, 2025), useful agents must remember user preferences, prior decisions, task constraints, tool outcomes, and evolving plans. This requirement has made memory a central abstraction for agent systems (Zhang et al., 2025; Du et al., 2025; Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025c).

The dominant implementation of agent memory stores memories outside the model and appends retrieved entries to each input context (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025c; Zhang et al., 2025; Du et al., 2025). The base LLM call is effectively stateless, so long-term memory must be reconstructed by placing past memory fragments back into the prompt. This resembles a memory-loss protagonist who wakes up reset and must consult an external notebook or archive (Nolan, 2001; Chiang, 2013). Longer contexts with more tokens increase this cognitive burden and reduce model performance across long-context and multi-turn studies (Liu et al., 2023; Laban et al., 2026; Kwa et al., 2025).

External or context memory also creates a serving-side challenge. Repeatedly injecting dynamic memories changes the prompt prefix, weakens prefix-cache reuse, increases prefill work, and expands KV-cache pressure. Recent serving systems study cache reuse, offloading, migration, scheduling, and multi-agent cache management under these pressures (Yao et al., 2025; Wang et al., 2025; Li et al., 2025). Context memory therefore makes both reasoning and serving harder, motivating the parameter-memory formulation in Section 2.

These limitations motivate a complementary direction: writing stable memory into model parameters. Neuroscience offers a light design analogy here: human experience is not only logged, but replayed and consolidated into durable internal representations, often through transformations that support future use (McClelland et al., 1995; Kumaran et al., 2016; Diekelmann & Born, 2010; Lewis & Durrant, 2011). This suggests that agent memory should not be trained only from raw transcripts; it should be rewritten into multiple reflective views before being internalized.

We call this high-level methodology Parameters as Memory (PAM). PAM treats durable agent knowledge as parameter memory stored in parameter-efficient adapters. We instan-

[1]The Hong Kong University of Science and Technology [2]College of Computer Science and Technology, Zhejiang University [3]The Hong Kong University of Science and Technology (Guangzhou) [4]National University of Singapore. Correspondence to: Xiaowen Chu <xwchu@hkustgz.edu.cn>, Bo Li <bli@ust.hk>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

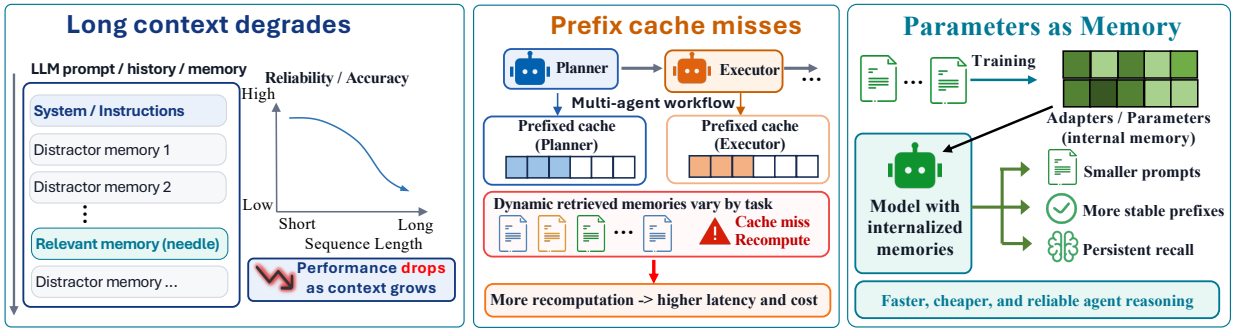

Figure 1. **Motivation overview.** External or context memory creates two coupled bottlenecks for LLM agents: long prompts make relevant memories harder to use reliably, while dynamic memory and agent switching reduce prefix-cache reuse.

tiate PAM with Reflective Memory Optimization (REMO), a concrete data rewriting and training pipeline. REMO reconstructs dialogue history into overlapping windows, rewrites each window into views for direct recall, inverse access, temporal reasoning, structured aggregation, and interaction-style retrieval, and then trains LoRA adapters with supervised fine-tuning so the model better internalizes the rewritten memories (Mecklenburg et al., 2024; Ovadia et al., 2023; Xu et al., 2025b; Lu et al., 2024; Zhao et al., 2025). We evaluate against A-Mem, Mem0, MemoryOS, Memory-R1, and retrieval-only baselines (Maharana et al., 2024; Xu et al., 2025c; Chhikara et al., 2025; Kang et al., 2025), and find the best overall performance on both LLaMA-3.1-8B-Instruct and Qwen-2.5-7B-Instruct. Our contributions are:

- We formulate Parameters as Memory (PAM), a high-level methodology that stores durable agent knowledge in adapters rather than repeatedly replaying all historical context at inference time.
- We design Reflective Memory Optimization (REMO), a concrete sliding-window rewriting and LoRA SFT pipeline for memory internalization, motivated by replay- and reflection-like transformations and tailored to contextual agent memories.
- We show on LoCoMo that PAM instantiated with REMO outperforms strong agentic memory baselines, demonstrating that data construction is central to effective parameterized memory.

2. Parameters as Memory (PAM)

PAM is the high-level methodology in this paper. It treats durable agent history as knowledge that can be written into model parameters, rather than replayed as retrieved text in every future LLM call. For a long conversation, this means that stable facts, preferences, decisions, and recurring constraints are internalized through training. During later task execution, the model can retrieve the relevant knowledge from parameter memory, so the same history does not need to repeatedly appear in the prefill context across many calls.

2.1. Problem setup

Let a multi-session interaction history be

$$\mathcal{H} = \{h_t\}_{t=1}^{T}, \qquad h_t = (x_t, \tau_t, m_t), \tag{1}$$

where x_t is a user or assistant utterance generated by a frozen base model θ_0, τ_t is its timestamp after time normalization, and m_t denotes optional side information such as tool outcomes or image descriptions. A conventional external-memory agent first builds a memory store

$$\Omega_{\mathcal{H}} = \mathrm{Construct}(\mathcal{H}), \tag{2}$$

and then retrieves a memory payload for each future call:

$$h_{\text{mem}} = \mathrm{Retrieve}(q, s; \Omega_{\mathcal{H}}, \mathcal{H}), \qquad P_{\theta_0}(y \mid p, h_{\text{mem}}, q, s), \tag{3}$$

where p is the role or system prompt and s is the current task state. Here h_{mem} denotes the retrieved memory text appended to the prompt. In realistic agents, h_{mem} can be long because it may contain summaries, raw dialogue spans, tool traces, and prior observations (Packer et al., 2023). Constructing $\Omega_{\mathcal{H}}$ can also consume substantial time and compute, especially when the system extracts facts, builds graphs, embeds chunks, or updates summaries after each session (Chhikara et al., 2025). At serving time, the retrieved payload can change across calls, which destabilizes the prompt prefix, reduces prefix-cache hits, and forces more KV-cache prefill recomputation (Liu et al., 2025). In a multi-call task, the same durable memories may therefore be reconstructed and replayed many times.

Different from conventional external-memory agents, PAM aim to learn adapter parameters $\phi_{\mathcal{H}}$ on top of the base model θ_0:

$$\phi_{\mathcal{H}} = \mathrm{Write}_{\text{PAM}}(\mathcal{H}; \theta_0), \qquad P_{\theta_0, \phi_{\mathcal{H}}}(y \mid p, q, s). \tag{4}$$

The future call is conditioned on the current task context, not on a long replay of $\mathcal{H}$. The goal is that stable knowledge from $\mathcal{H}$ is recalled implicitly through the adapted parameters when it is needed for answering q.

This paper studies only this parameter-memory write path on LoCoMo. In particular, we do not yet learn a complete router that decides which fresh, uncertain, private, or rapidly changing memory fragments should remain retrieval-only rather than being written into adapters. That selective volatile-memory routing problem is left to future full-agent systems and discussed further in Section 6.

2.2. Challenges

The first challenge is reliable memory internalization. Training on raw history text is not enough. Prior work on knowledge injection shows that naive SFT can produce superficial memorization, copy surface patterns, or mimic training formats without making the injected knowledge robustly usable under inverse queries, multi-hop reasoning, or conflicts with pretrained beliefs (Berglund et al., 2023; Gekhman et al., 2024; Mecklenburg et al., 2024; Ovadia et al., 2023; Yamin et al., 2025). Simple paraphrasing and semantic rewrites help, but they still often fall short of deep internalization unless the fact is exposed through diverse access paths and reasoning contexts (Xu et al., 2025b; Lu et al., 2024; Zhao et al., 2025). Our experiments show the same pattern in agent memory: **Memory-NaiveSFT** uses the same parameter-memory training recipe but lacks our rewriting pipeline, and it is substantially weaker than PAM with REMO.

The second challenge is computing costs in LLM serving. A literal implementation of PAM would fine-tune a new backbone LLM for every user, project, or agent task. That is unrealistic for providers serving millions or billions of users, because each full model copy would add large GPU memory, loading, scheduling, and maintenance costs. A practical PAM system therefore needs parameter-efficient memory units, adapter selection, batching, and hot-swapping. In Section 3, we instantiate PAM with REMO, which writes memory into LoRA adapters through a data rewriting pipeline. The LoRA design does not solve serving by itself, but it makes PAM compatible with emerging multi-adapter serving systems and gives system builders a plausible path for future deployment.

3. Reflective Memory Optimization (REMO)

REMO is our concrete implementation of PAM. In neuroscience, complementary learning systems and replay-based consolidation suggest that experiences can be revisited and transformed before becoming durable internal knowledge (McClelland et al., 1995; Kumaran et al., 2016; Diekelmann & Born, 2010; Lewis & Durrant, 2011). REMO turns this intuition into an optimization pipeline for agent memory: it reconstructs long-horizon dialogue into overlapping windows, rewrites each window through multiple reflective views, and trains LoRA adapters so stable history can be retrieved from parameter memory during future calls.

The central idea is that durable agent memory should not be learned from a single transcript form. The same underlying memory should be exposed through multiple retrieval pathways, so the adapted model can recover it under direct lookup, inverse lookup, temporal questions, cross-turn composition, and multi-hop reasoning.

3.1. Window construction and source units

We do not write isolated facts independently. Isolated extraction can sever nearby dialogue fragments that jointly define a memory: a later correction may change an earlier claim, a tool result may explain a prior decision, and adjacent turns may provide the entities, time, and motivation needed to interpret a fact. Training on such detached fragments risks teaching incomplete or misleading memory units. We therefore split the history into overlapping sliding windows so each training unit preserves as much local semantic and narrative context as possible:

$$\mathcal{W} = \{w_j\}_{j=1}^{J}, \tag{5}$$

where each w_j contains a bounded sequence of temporally adjacent turns and associated metadata. Overlap preserves relational dependencies across turns: a correction may revise an earlier statement, a later tool observation may explain a prior decision, and multiple utterances may jointly define the fact needed by a future query.

Each window is normalized into a compact context block c_j that keeps speaker identity, resolved timestamps, and optional visual descriptions. Relative dates such as "yesterday" are converted into absolute dates when rewriting, and all entities are preserved with their original names. This normalized window set defines the **naive SFT** corpus

$$\mathcal{D}_{\text{naive}} = \{c_j\}_{j=1}^{J}, \tag{6}$$

which serves as the no-augmentation baseline in our experiments.

3.2. Design goals and universal constraints of Data Augmentation

Our augmentation design is driven by four goals. First, the adapted model should preserve **exact factual content**: specific names, dates, places, and list items must remain explicit rather than being replaced by vague summaries. Second, the answer-bearing span should remain **easy to extract**, so the key fact is not buried in a subordinate clause. Third, memories should become accessible through **multiple reasoning paths**, including direct recall, inverse lookup, and cross-turn composition. Fourth, the model should resist **entity confusion and temporal ambiguity**,

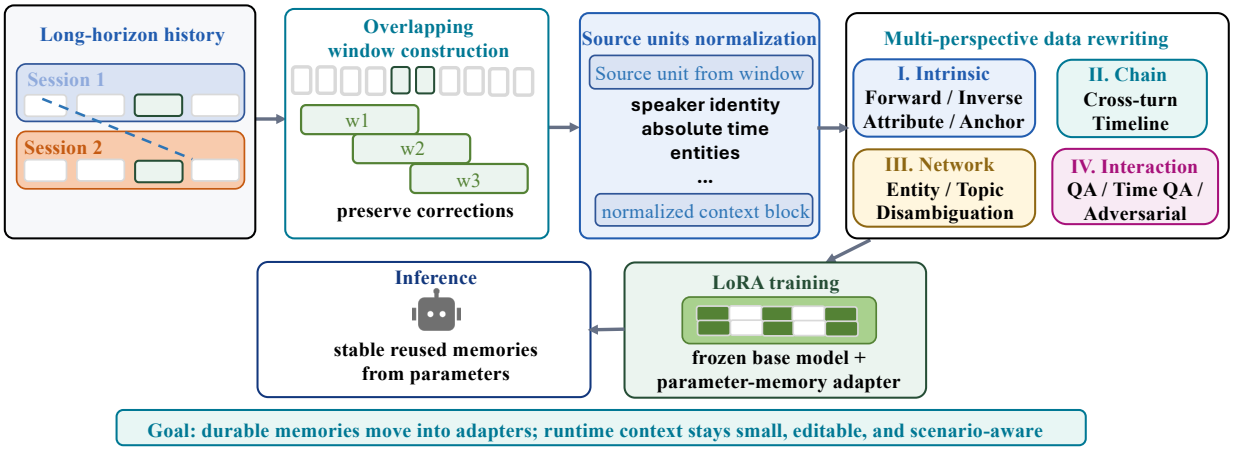

Figure 2. **Method overview.** Under the PAM methodology, REMO reconstructs long-horizon dialogue into overlapping windows, builds source units with speaker, temporal, entity, and metadata context, and rewrites each unit through four complementary augmentation families. The resulting corpus trains a parameter-memory adapter with supervised fine-tuning.

two recurring failure modes in long-term conversational memory.

These goals translate into universal prompt constraints used across all generators: (i) definitive language only, without speculative hedging; (ii) exact entity-name preservation; (iii) absolute-time resolution for training text; (iv) self-contained outputs; and (v) explicit enumeration when a fact contains multiple items. We further apply lightweight quality filtering after generation to remove outputs with vague wording or unresolved relative time.

3.3. Detailed Data Augmentation Strategy

Let $\mathcal{R}$ denote the set of rewriting strategies. For each normalized window $c_j \in \mathcal{D}_{\text{naive}}$ and strategy $r \in \mathcal{R}$, we prompt a generator model $\mathcal{M}$ with a strategy-specific template $\mathcal{T}_r$ and sample rewritten text

$$x_{j,r} \sim \mathcal{M}\big(\mathcal{T}_r(c_j, \mathcal{C}(c_j))\big), \tag{7}$$

where $\mathcal{C}(c_j)$ contains the relevant local or global context, such as resolved timestamps, nearby utterances, entity inventory, and image descriptions. The final training corpus is

$$\mathcal{D}_{\text{REMO}} = \mathcal{D}_{\text{naive}} \cup \bigcup_{j=1}^{J} \bigcup_{r \in \mathcal{R}} \mathcal{A}_r(c_j), \tag{8}$$

where $\mathcal{A}_r(c_j)$ denotes the set of accepted generations from strategy r. Appendix E lists the concrete prompt templates and constraints.

We organize $\mathcal{R}$ into four categories, each targeting a distinct memory failure mode. Taken together, these categories define REMO. A memory is not merely repeated, but rephrased, temporally ordered, grouped with related evidence, contrasted against confusable alternatives, and rehearsed through future-style queries. This is the concrete point at which the PAM methodology becomes an implementable pipeline, because reflective augmentation defines the distribution of parameter-memory training data.

Category I: Local Fact Re-expression (Intrinsic). This category rewrites a single utterance into multiple factual views so that the same memory is accessible beyond one surface form. 1) **Forward Assertion** expresses the canonical fact directly; 2) **Inverse Indexing** reverses the retrieval direction to mitigate one-way recall; 3) **Attribute-Centric** converts the fact into a stable property of an entity; 4) **Premise Injection** appends a short, plausible consequence while keeping the original fact dominant; 5) **Consequence Extraction** starts from a concrete clue and reconstructs the fact; 6) **Comparative Bridging** contrasts the fact against a confusable peer; 7) **Enumeration Drill** forces exhaustive listing for multi-item facts; and 8) **Exact Answer Anchor** foregrounds the precise answer span.

Category II: Cross-Turn Composition (Chain). This category uses multiple nearby turns jointly, since many conversational memories are distributed across a local dialogue segment rather than stated in one utterance. 1) **Cross-Turn Fact Composition** merges compatible facts from different turns into a single coherent statement; 2) **Structured Multi-Hop Chain** organizes facts into verifiable multi-step dependencies such as event $\rightarrow$ action $\rightarrow$ outcome; and 3) **Chronological Timeline** linearizes cross-turn events into a temporally ordered representation.

Category III: Retrieval-Oriented QA Encoding (Interaction). This category exposes memories in question-answer forms so the model learns to retrieve stored facts under evaluation-style prompts without requiring a separate instruction-tuning stage. 1) **Rhetorical QA Statement** embeds the answer in natural-language question-answer framing; 2) **Direct QA as Factual Text** writes explicit "Ques-

tion / Answer" pairs; 3) **Fact Verification Triple** combines a statement with its verification question and concise answer; and 4) **Dual-Format Time QA** teaches the same temporal fact in both absolute-date and relative-reference form.

Category IV: Global Memory Structuring (Network). This category operates on a full conversation or entity-centric slice, producing high-coverage summaries and contrastive supervision that connect local facts into broader entity- and session-level memory structure. 1) **Entity Profile Extraction** gathers all facts about an entity into a structured profile; 2) **Temporal Event Registry** enumerates all time-anchored events in chronological order; 3) **Topic-Clustered Fact Sheet** groups facts by semantic topic; 4) **Entity Boundary Guard** pairs correct attribution with explicit denial of the wrong entity; and 5) **Contrastive Entity Disambiguation** places two entities side by side within the same semantic domain.

This taxonomy follows a simple principle. Durable agent memory should be reflected on before it is written. Operationally, reflection means that a memory is repeated, inverted, composed, structured, verified, and contrasted rather than merely copied. In particular, **Memory-NaiveSFT** retains only $\mathcal{D}_{\text{naive}}$, whereas REMO uses the full multi-perspective rewrite set $\mathcal{D}_{\text{REMO}}$.

3.4. Training and inference

We fine-tune the base model θ_0 from Section 2 on $\mathcal{D}_{\text{REMO}}$ using LoRA adapters and the standard autoregressive objective, while keeping the backbone weights fixed. The resulting adapter parameters $\phi_{\mathcal{H}}$ implement the parameter-memory write operation in Section 2. At inference time, the adapter provides a stable parameter-memory layer, so the model can answer future queries without re-injecting the full dialogue history. Our experiments evaluate this parameter-memory layer directly; they do not implement a full routing policy that chooses between adapter memory and external retrieval for volatile agent state.

Serving-oriented design. The serving challenge in Section 2 rules out a full-model copy for every user or task. REMO therefore writes memory into LoRA adapters instead of updating all backbone parameters (Hu et al., 2021). The base model can remain shared, while small memory adapters are selected, loaded, batched, or swapped at request time. Recent LoRA serving systems show that many adapters can share the same base model with specialized batching, memory management, and adapter scheduling (Sheng et al., 2024; Chen et al., 2024; Wu et al., 2024). This does not make billion-user PAM serving solved, but it gives a practical direction for cluster and personal serving systems to improve over time. Appendix B gives a more concrete serving discussion.

4. Experiments

We evaluate PAM on LoCoMo, a long-term multi-session dialogue benchmark that includes questions spanning **Single Hop**, **Multi-Hop**, **Open Domain**, and **Temporal** reasoning (Maharana et al., 2024). In this setting, the model must answer questions grounded in information distributed across many prior sessions rather than relying on a short local context.

4.1. Setup

Backbones and metrics. We report results on two widely used open-source backbones: **LLaMA-3.1-8B-Instruct** and **Qwen-2.5-7B-Instruct**. For each benchmark split, we report **F1**, **BLEU-1 (B1)**, and **LLM-as-a-Judge (J)**. F1 measures lexical overlap at the answer level, BLEU-1 captures unigram precision, and Judge reflects semantic correctness beyond n-gram matching.

Baselines. We compare against three categories of baselines. **LoCoMo (RAG)** performs retrieval over the raw dialogue history without parameterizing memory. **Zep**, **A-Mem** (Xu et al., 2025c), and **Mem0** (Chhikara et al., 2025) are representative memory-agent systems that store and retrieve long-term memory externally. **Memory-NaiveSFT**, **Memory-R1-PPO**, and **Memory-R1-GRPO** are learned-memory baselines. Among them, **Memory-NaiveSFT** is the key ablation for our method: it uses the same parameter-memory training recipe as ours but is trained only on normalized conversation text, *without* the data augmentation and rewriting pipeline described in Section 3.

Training details. All models are fine-tuned with LoRA (Hu et al., 2021) via LLaMA Factory. We use rank r=128, α=256, dropout 0.05, learning rate 1e$-$4, cosine scheduling, 10 epochs, batch size 8 with gradient accumulation 4, cutoff length 1200, and fp16. Unless otherwise noted, these settings are kept fixed across all experiments.

4.2. Main results

PAM achieves the best overall performance on both backbones (Table 1). The most informative comparison is against **Memory-NaiveSFT**: both methods use the same parameter-memory training setup, but PAM replaces raw conversation text with rewritten supervision. The consistent gap shows that the main bottleneck is not the fine-tuning recipe itself, but the quality and coverage of the training signal.

The strongest gains appear in **temporal reasoning** and **BLEU-1**, which is consistent with our augmentation design. Timeline construction, time normalization, dual-

Model	Method	Single Hop			Multi-Hop			Open Domain			Temporal			Overall		
		F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑
LLaMA-3.1-8B Instruct	LoCoMo (RAG)	12.25	9.77	13.81	13.69	10.96	20.48	11.59	8.30	15.96	9.38	8.15	4.65	11.73	9.29	13.73
	Zep	30.15	17.15	52.38	15.04	11.56	33.33	26.67	18.44	45.36	3.49	2.68	27.58	18.84	12.46	39.66
	A-Mem	21.62	16.93	44.76	13.82	11.45	34.93	34.67	29.13	49.38	25.77	22.14	36.43	23.97	19.91	41.38
	Mem0	27.29	18.63	43.93	18.59	13.86	37.35	34.03	24.77	52.27	26.90	21.06	31.40	26.70	19.58	41.24
	Memory-SFT	34.64	23.73	56.90	20.80	16.26	37.35	46.47	37.35	63.27	47.18	34.58	54.65	37.27	27.98	53.04
	Memory-R1-PPO	32.52	24.47	53.56	26.86	23.47	42.17	45.30	39.18	64.10	41.57	26.11	47.67	36.56	28.31	51.88
	Memory-R1-GRPO	35.73	27.70	59.83	**35.65**	30.77	53.01	47.42	41.24	68.78	49.86	38.27	51.55	42.16	34.50	58.29
	ours	**36.35**	**40.80**	**60.31**	34.30	**35.53**	**55.32**	**49.49**	**55.06**	**70.52**	**53.84**	**54.70**	**65.59**	**43.50**	**46.52**	**62.94**
Qwen-2.5-7B Instruct	LoCoMo (RAG)	9.57	7.00	15.06	11.84	10.02	19.28	8.67	6.52	12.79	8.35	8.74	5.43	9.61	8.07	13.14
	Zep	31.02	21.39	42.85	20.42	15.76	23.81	25.25	21.34	42.26	8.94	8.42	29.31	21.41	16.73	34.56
	A-Mem	18.96	12.86	40.78	14.73	12.66	31.32	30.58	26.14	46.90	23.67	20.67	28.68	21.98	18.08	36.92
	Mem0	24.96	18.05	61.92	20.31	15.82	48.19	32.74	25.27	65.20	33.16	26.28	38.76	27.79	21.36	53.52
	Memory-SFT	27.81	20.25	57.74	24.62	22.28	46.99	43.33	34.06	66.85	44.41	34.32	52.71	35.04	27.73	56.07
	Memory-R1-PPO	34.22	23.61	57.74	**32.87**	29.48	**53.01**	44.78	38.72	66.99	42.88	30.30	42.25	38.69	30.53	55.00
	Memory-R1-GRPO	33.64	26.06	**62.34**	23.55	20.71	40.96	**46.86**	40.92	**67.81**	47.75	38.49	49.61	37.95	31.55	55.18
	ours	**35.61**	**39.76**	58.76	27.62	**29.82**	52.84	42.73	**47.21**	67.34	**49.67**	**51.42**	**61.93**	**38.9**	**42.05**	**60.22**

Table 1. **Main results on LoCoMo.** We compare retrieval-based memory systems, learned-memory baselines, and PAM on two backbones: LLaMA-3.1-8B-Instruct and Qwen-2.5-7B-Instruct. Performance is reported on four evaluation categories, **Single Hop**, **Multi-Hop**, **Open Domain**, and **Temporal**, using F1, BLEU-1 (B1), and Jaccard (J). PAM achieves the best overall performance on both backbones, with especially strong gains in temporal reasoning and answer extractability. Bold numbers denote the best result within each backbone and metric column.

format temporal QA, and exact-answer anchoring make stored memories easier to recover in the form expected by evaluation. More broadly, retrieval-only memory systems remain weaker than learned-memory approaches, suggesting that repeated inference-time retrieval alone is not a sufficient substitute for internalized memory.

4.3. Ablation studies

To understand what drives PAM's performance, we isolate the contribution of each design axis below. All ablations use LLaMA-3.1-8B-Instruct.

4.3.1. Strategy group ablation

Design. We group the 20 rewriting strategies into four categories and train one model per group, then compare each variant against the full system.

Findings. No single group matches the full model (Table 2), which shows that the four groups are complementary rather than interchangeable. G3 is the strongest standalone component, indicating that QA-style encoding is especially well aligned with the benchmark format. By contrast, G4 is weak in isolation because its structured extraction outputs are far from the final conversational QA distribution, but it still helps when combined with the other groups by adding global entity and temporal structure.

4.3.2. Compatibility with external memory frameworks

Design. We test whether our gains depend on a particular retrieval backend by pairing the same augmented training corpus with two external memory frameworks, **A-Mem** and **Mem0**, and comparing each to its vanilla version.

Findings. The augmentation pipeline improves both backends across all categories (Table 3). This suggests that the value of PAM lies in how memories are encoded before fine-tuning, rather than in any single retrieval framework. In that sense, the rewriting pipeline is largely framework-agnostic and improves the parametric memory substrate itself.

4.3.3. Oracle vs. vanilla evaluation

Design. We compare a setting with oracle context at inference time against the standard vanilla setting where the model must rely entirely on internalized memory.

Findings. The oracle setting shows that access to the right evidence is not enough by itself: answer-distribution alignment still matters. Oracle context yields strong semantic judgment, but the augmented model expresses answers far better because its training data is already aligned with QA-style evaluation. The two effects are complementary: internalized knowledge improves what the model knows, while rewriting improves how that knowledge is expressed.

4.3.4. LoRA rank sensitivity

Design. We sweep the LoRA rank while keeping all other hyperparameters fixed to study how adapter capacity interacts with the breadth of the rewriting pipeline.

Findings. Performance is not monotonic with rank (Table 5). Very small ranks underfit the diversity of the 20-strategy training signal, while larger ranks improve capacity but do not always improve generalization uniformly

Strategy Group	Single Hop			Multi-Hop			Open Domain			Temporal			Overall		
	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑
G1 Only	15.04	13.91	19.31	30.44	29.41	32.24	34.10	34.25	47.24	32.39	44.08	45.56	27.99	30.41	36.09
G2 Only	22.92	19.77	24.71	12.03	13.10	15.36	33.36	30.49	39.65	13.70	10.24	16.38	20.50	18.40	24.03
G3 Only	30.22	33.36	49.63	19.34	32.97	29.54	35.71	45.31	57.36	**58.35**	**64.86**	**74.57**	35.91	44.13	52.78
G4 Only	2.44	1.39	4.88	9.41	5.69	4.58	9.16	5.72	7.15	2.73	1.55	4.58	5.94	3.59	5.30
Full (PAM)	**36.35**	**40.80**	**60.31**	**34.30**	**35.53**	**55.32**	**49.49**	**55.06**	**70.52**	53.84	54.70	65.59	**43.50**	**46.52**	**62.94**

Table 2. Ablation by strategy group. G1: single-utterance; G2: session-context (3); G3: QA-format (4); G4: full-session (5). Full uses all 4 groups of strategies.

Method		Single Hop			Multi-Hop			Open Domain			Temporal			Overall		
		F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑
A-Mem	Vanilla	21.62	16.93	44.76	13.82	11.45	34.93	34.67	29.13	49.38	25.77	22.14	36.43	23.97	19.91	41.38
	+PAM	**29.52**	**32.46**	**58.77**	**23.46**	**27.43**	**50.47**	**37.92**	**44.21**	**56.37**	**47.62**	**39.91**	**57.41**	**34.63**	**36.00**	**55.76**
Mem0	Vanilla	27.29	18.63	43.93	18.59	13.86	37.35	34.03	24.77	52.27	26.90	21.06	31.40	26.70	19.58	41.24
	+PAM	**34.17**	**39.25**	**59.41**	**26.71**	**30.47**	**52.85**	**41.57**	**45.68**	**64.41**	**54.35**	**53.74**	**62.57**	**39.20**	**42.29**	**59.81**

Table 3. Compatibility with external memory frameworks on LLaMA-3.1-8B-Instruct.

across categories. The best trade-off appears at an intermediate rank, suggesting that effective parameter memory depends on matching adapter capacity to the complexity of the rewriting distribution rather than simply increasing rank.

4.4. Discussion

A common thread runs through all of the ablations: *no single component accounts for PAM's gains; the improvement comes from their composition*. G3 provides the strongest standalone result, yet still falls 7.6 F1 short of the full pipeline. G4 performs poorly in isolation, but contributes entity-boundary signals that improve open-domain and multi-hop performance when combined with the other groups. The augmentation pipeline improves every memory backend it touches, and the Oracle experiment shows that the mechanism is twofold—knowledge internalization *and* answer-distribution alignment, rather than either one alone. Even the LoRA rank results follow the same pattern: the best performance comes not from maximizing capacity, but from choosing a rank that is well matched to the training signal. Overall, these results suggest that future gains in long-term memory for LLMs will come less from asking *where* memories are stored, and more from understanding *how* they should be encoded before reaching the model.

5. Related work

Agent memory. LLM agent memory systems usually store experiences or facts outside the model and retrieve them for later reasoning. MemGPT frames memory as an operating-system-like hierarchy with explicit movement between context and storage (Packer et al., 2023). MemoryBank and Lifelong Memory systems study persistent user modeling and cross-session storage (Zhong et al., 2023; Wang et al., 2023). Mem0, A-Mem, and MemoryOS develop more scalable long-term memory mechanisms for agents, including structured extraction, agentic memory management, and system-level memory orchestration (Chhikara et al., 2025; Xu et al., 2025c; Kang et al., 2025). Zep further emphasizes temporally structured knowledge-graph memory for agent histories (Rasmussen et al., 2025). These systems demonstrate the importance of memory extraction, organization, and retrieval, but they still primarily rely on runtime memory access rather than writing durable memories into parameters.

Long-context reliability and agent serving. Long-context and multi-turn studies show that LLM performance degrades as relevant evidence becomes harder to locate, conversations grow longer, and tasks span more interaction steps (Liu et al., 2023; Bai et al., 2023; Yen et al., 2024; Zhang et al., 2024; Yan et al., 2025; Wu et al., 2025; Laban et al., 2026; Kwa et al., 2025; Xu et al., 2025a; Sinha et al., 2025; Zheng et al., 2025). At the systems level, long and dynamic prompts also increase prefill cost and KV-cache complexity, motivating cache-reuse, offloading, and scheduling systems for agent workloads (Vaswani et al., 2017; Kwon et al., 2023; Qin et al., 2024; Yao et al., 2025; Wang et al., 2025; Liu et al., 2025; Jiang et al., 2026; Li et al., 2025; Bian et al., 2025; Yang et al., 2025; Pan et al., 2025).

Parametric knowledge and model editing. Prior work has shown that factual knowledge can be localized and modified in model parameters, especially in MLP modules (Geva et al., 2020; Meng et al., 2022a;b). Model editing provides targeted updates, while parameter-efficient fine-tuning provides a practical way to attach task- or user-specific memory adapters. Our focus is not single-fact editing, but agent memory internalization under long-horizon

Setting		Single Hop			Multi-Hop			Open Domain			Temporal			Overall		
		F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑
Oracle	LLaMA-3.1-8B	14.53	8.66	83.59	21.48	12.90	**90.38**	22.64	14.13	89.64	21.32	13.01	81.48	19.99	12.18	86.27
	+PAM	**80.44**	**80.50**	**89.06**	**59.58**	**97.43**	82.69	**75.06**	**83.36**	**90.15**	**74.35**	**73.87**	**89.62**	**72.36**	**83.79**	**87.13**
Vanilla	A-Mem	21.62	16.93	44.76	13.82	11.45	34.93	34.67	29.13	49.38	25.77	22.14	36.43	23.97	19.91	41.38
	Mem0	27.29	18.63	43.93	18.59	13.86	37.35	34.03	24.77	52.27	26.90	21.06	31.40	26.70	19.58	41.24
	Memory-R1-GRPO	35.73	27.70	59.83	35.65	30.77	53.01	47.42	41.24	68.78	49.86	38.27	51.55	42.16	34.50	58.29
	PAM	36.35	40.80	60.31	34.30	35.53	55.32	49.49	55.06	70.52	53.84	54.70	65.59	43.50	46.52	62.94

Table 4. Oracle vs. Vanilla evaluation on LLaMA-3.1-8B-Instruct. *Oracle*: ground-truth relevant context is provided at inference time. *Vanilla*: the model relies solely on internalized knowledge.

LoRA Rank	Single Hop			Multi-Hop			Open Domain			Temporal			Overall		
	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑	F1↑	B1↑	J↑
r=8	27.50	28.99	42.16	11.30	10.58	19.24	25.38	26.30	31.53	46.15	42.60	49.21	27.58	27.12	35.54
r=16	30.70	34.54	44.52	10.59	11.58	20.51	27.51	28.07	35.32	41.15	37.56	47.43	27.49	27.94	36.95
r=64	19.05	20.17	29.82	17.32	15.31	22.85	32.95	32.36	57.31	51.42	52.78	60.61	30.19	30.16	42.65
r=128	**36.35**	**40.80**	60.31	**34.30**	**35.53**	**55.32**	**49.49**	**55.06**	**70.52**	53.84	54.70	65.59	**43.50**	**46.52**	**62.94**
r=256	34.21	37.63	**64.21**	29.91	27.36	50.28	39.20	44.59	62.67	**60.87**	**55.34**	**69.54**	41.05	41.23	61.68

Table 5. LoRA rank sensitivity on PAM.

evaluation.

Data-centric knowledge injection. Recent work shows that data construction strongly affects whether fine-tuning internalizes knowledge or only memorizes surface forms (Mecklenburg et al., 2024; Ovadia et al., 2023; Xu et al., 2025b). Rewriting strategies such as paraphrasing, inverse ordering, chain-of-thought augmentation, and contrastive correction are especially relevant for turning isolated facts into usable knowledge (Berglund et al., 2023; Lu et al., 2024; Zhao et al., 2025). We adapt this data-centric view to the agent-memory setting.

Human memory and neuroscience inspiration. Human memory provides a useful conceptual contrast to current LLM agents. Complementary learning systems, systems consolidation, and replay-based accounts all suggest that experiences can be revisited, reorganized, and gradually integrated into durable internal knowledge rather than preserved only as raw episodic traces (McClelland et al., 1995; Kumaran et al., 2016; Frankland & Bontempi, 2005; Squire, 2004; Sun et al., 2023; Diekelmann & Born, 2010; Lewis & Durrant, 2011; Tonegawa et al., 2018; Schapiro et al., 2018; Carr et al., 2011; Joo & Frank, 2018; Kaefer et al., 2022; Gattas et al., 2023). We use these works only as inspiration: for PAM and REMO, the main algorithmic lesson is that durable memory becomes more usable when experience is revisited through multiple access paths and rewritten into structured supervision before internalization.

6. Limitations and Future Work

This paper focuses on parameterizing stable memories and evaluates this direction on LoCoMo, a long-context memory benchmark. We do not yet study a complete agent memory router that decides which information should be written into adapters and which volatile information should remain retrieval-only. Such a router matters in realistic agents because tool outputs, temporary plans, private notes, and rapidly changing facts may be better served by explicit retrieval than by parameter updates. Future work should evaluate PAM in real agent workloads where REMO-style adapter writing is combined with selective retrieval for volatile state.

7. Conclusion

We proposed Parameters as Memory (PAM), a high-level methodology that treats parameters as a complementary memory layer for LLM agents. Our central claim is that durable agent memory should not remain purely in prompt text. Repeatedly replaying stable history burdens both reasoning and serving, while naive supervised writing fails to produce robust internalization. We instantiate PAM with Reflective Memory Optimization (REMO), a concrete rewriting and LoRA SFT pipeline that turns dialogue history into replay-like supervision for parameter-efficient memory adapters. Results on LoCoMo show that this memory-writing perspective improves over retrieval-only and learned-memory baselines, suggesting that effective agent memory depends not only on storage location, but also on how experiences are rewritten before they are learned.

Impact Statement

This paper studies how durable agent memories can be internalized into parameter-efficient adapters rather than repeatedly replayed as prompt text. Potential positive im-

pacts include more efficient and robust long-horizon assistants, while potential risks include stronger retention of sensitive user information and misuse in profiling or surveillance settings if such systems are deployed without appropriate controls.

References

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench: A bilingual, multitask benchmark for long context understanding. *CoRR*, abs/2308.14508, 2023. URL https://arxiv.org/abs/2308.14508.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *ArXiv*, abs/2309.12288, 2023. URL https://api.semanticscholar.org/CorpusID:262083829.

Bian, Z., Wu, F., Ma, T., and Zhuo, Y. Tokencake: A kv-cache-centric serving framework for llm-based multi-agent applications. *CoRR*, abs/2510.18586, 2025. URL https://doi.org/10.48550/arXiv.2510.18586.

Carr, M. F., Jadhav, S. P., and Frank, L. M. Hippocampal replay in the awake state: A potential substrate for memory consolidation and retrieval. *Nature Neuroscience*, 14(2):147–153, 2011. doi: 10.1038/nn.2732. URL https://doi.org/10.1038/nn.2732.

Chen, L., Ye, Z., Wu, Y., Zhuo, D., Ceze, L., and Krishnamurthy, A. Punica: Multi-tenant LoRA serving. In *Proceedings of Machine Learning and Systems*, 2024. URL https://arxiv.org/abs/2310.18547.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Chiang, T. The truth of fact, the truth of feeling. In *Subterranean: Tales of Dark Fantasy 2*. Subterranean Press, 2013.

Diekelmann, S. and Born, J. The memory function of sleep. *Nature Reviews Neuroscience*, 11(2):114–126, 2010. doi: 10.1038/nrn2762. URL https://doi.org/10.1038/nrn2762.

Du, Y., Huang, W., Zheng, D., Wang, Z., Montella, S., Lapata, M., Wong, K.-F., and Pan, J. Z. Rethinking memory in ai: Taxonomy, operations, topics, and future directions, 2025. URL https://arxiv.org/abs/2505.00675.

Frankland, P. W. and Bontempi, B. The organization of recent and remote memories. *Nature Reviews Neuroscience*, 6(2):119–130, 2005. doi: 10.1038/nrn1607. URL https://doi.org/10.1038/nrn1607.

Gattas, S., Larson, M. S., Mnatsakanyan, L., Sen-Gupta, I., Vadera, S., Swindlehurst, A. L., Rapp, P. E., Lin, J. J., and Yassa, M. A. Theta mediated dynamics of human hippocampal-neocortical learning systems in memory formation and retrieval. *Nature Communications*, 14 (1):8505, 2023.

Gekhman, Z., Yona, G., Aharoni, R., Eyal, M., Feder, A., Reichart, R., and Herzig, J. Does fine-tuning llms on new knowledge encourage hallucinations? *ArXiv*, abs/2405.05904, 2024. URL https://api.semanticscholar.org/CorpusID:269635770.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. *ArXiv*, abs/2012.14913, 2020. URL https://api.semanticscholar.org/CorpusID:229923720.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiang, Y., Huang, Y., Cheng, L., Deng, C., Sun, X., and Mai, L. Contextpilot: Fast long-context inference via context reuse, 2026. URL https://arxiv.org/abs/2511.03475.

Joo, H. R. and Frank, L. M. The hippocampal sharp wave–ripple in memory retrieval for immediate use and consolidation. *Nature Reviews Neuroscience*, 19(12):744–757, 2018. doi: 10.1038/s41583-018-0077-1. URL https://doi.org/10.1038/s41583-018-0077-1.

Kaefer, K., Stella, F., McNaughton, B. L., and Battaglia, F. P. Replay, the default mode network and the cascaded memory systems model. *Nature Reviews Neuroscience*, 23(10):628–640, 2022. doi: 10.1038/s41583-022-00620-6. URL https://doi.org/10.1038/s41583-022-00620-6.

Kang, J., Ji, M., Zhao, Z., and Bai, T. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*, 2025.

Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

Kwa, T., West, B., Becker, J., Deng, A., Garcia, K., Hasin, M., Jawhar, S., Kinniment, M., Rush, N., von Arx, S., Bloom, R., Broadley, T., Du, H., Goodrich, B., Jurkovic, N., Miles, L. H., Nix, S., Lin, T., Parikh, N., Rein, D., Sato, L. J. K., Wijk, H., Ziegler, D. M., Barnes, E., and Chan, L. Measuring ai ability to complete long tasks. *CoRR*, abs/2503.14499, 2025. URL https://arxiv.org/abs/2503.14499.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Laban, P., Hayashi, H., Zhou, Y., and Neville, J. Llms get lost in multi-turn conversation. In *International Conference on Learning Representations*, 2026. Best paper.

Lewis, P. A. and Durrant, S. J. Overlapping memory replay during sleep builds cognitive schemata. *Trends in Cognitive Sciences*, 15(8):343–351, 2011.

Li, H., Mang, Q., He, R., Zhang, Q., Mao, H., Chen, X., Zhou, H., Cheung, A., Gonzalez, J., and Stoica, I. Continuum: Efficient and robust multi-turn llm agent scheduling with kv cache time-to-live. *CoRR*, abs/2511.02230, 2025. URL https://doi.org/10.48550/arXiv.2511.02230.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Liu, Y., Cheng, Y., Yao, J., An, Y., Chen, X., Feng, S., Huang, Y., Shen, S., Zhang, R., Du, K., et al. LMCache: An efficient kv cache layer for enterprise-scale llm inference. *CoRR*, abs/2510.09665, 2025. URL https://doi.org/10.48550/arXiv.2510.09665.

Lu, Z., Jin, L., Li, P., Tian, Y., Zhang, L., Wang, S., Xu, G., Tian, C., and Cai, X. Rethinking the reversal curse of llms: a prescription from human knowledge reversal. In *Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://api.semanticscholar.org/CorpusID:273901546.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Mecklenburg, N., Lin, Y., Li, X., Holstein, D., Nunes, L., Malvar, S., Silva, B. L. B., Chandra, R., Aski, V., Yannam, P. K. R., Aktas, T., and Hendry, T. Injecting new knowledge into large language models via supervised fine-tuning. *ArXiv*, abs/2404.00213, 2024. URL https://api.semanticscholar.org/CorpusID:268819437.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*, 2022a. URL https://api.semanticscholar.org/CorpusID:255825985.

Meng, K., Sharma, A. S., Andonian, A., Belinkov, Y., and Bau, D. Mass-editing memory in a transformer. *ArXiv*, abs/2210.07229, 2022b. URL https://api.semanticscholar.org/CorpusID:252873467.

Model Evaluation and Threat Research. Measuring ai ability to complete long tasks. https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long- 2025. Accessed: 2026-05-04.

Nolan, J. Memento mori. Short story, 2001.

Ovadia, O., Brief, M., Mishaeli, M., and Elisha, O. Fine-tuning or retrieval? comparing knowledge injection in llms. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:266162497.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Pan, Z., Patel, A., Shen, Y., Hu, Z., Guan, Y., Li, W.-L., Qin, L., Wang, Y., and Ding, Y. Kvflow: Efficient prefix caching for accelerating llm-based multi-agent workflows. In *Advances in Neural Information Processing Systems*, 2025.

Qin, R., Li, Z., He, W., Zhang, M., Wu, Y., Zheng, W., and Xu, X. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *CoRR*, abs/2407.00079, 2024. URL https://doi.org/10.48550/arXiv.2407.00079.

Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., and Chalef, D. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Schapiro, A. C., McDevitt, E. A., Rogers, T. T., Mednick, S. C., and Norman, K. A. Human hippocampal replay during rest prioritizes weakly learned information and predicts memory performance. *Nature Communications*, 9(1):3920, 2018.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., Gonzalez, J. E., and Stoica, I. S-LoRA: Serving thousands of concurrent LoRA adapters. In *Proceedings of Machine Learning and Systems*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/906419cd502575b617cc489a1a696a67-Paper-Conference.pdf.

Sinha, A., Arun, A., Goel, S., Staab, S., and Geiping, J. The illusion of diminishing returns: Measuring long horizon execution in llms. *CoRR*, abs/2509.09677, 2025. URL https://arxiv.org/abs/2509.09677.

Squire, L. R. Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3):171–177, 2004. doi: 10.1016/j.nlm.2004.06.005. URL https://doi.org/10.1016/j.nlm.2004.06.005.

Sun, W., Advani, M., Spruston, N., Saxe, A., and Fitzgerald, J. E. Organizing memories for generalization in complementary learning systems. *Nature Neuroscience*, 26 (8):1438–1448, 2023.

Tonegawa, S., Morrissey, M. D., and Kitamura, T. Memory engram storage and retrieval. *Nature Reviews Neuroscience*, 19(8):485–498, 2018. doi: 10.1038/s41583-018-0031-2. URL https://doi.org/10.1038/s41583-018-0031-2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL https://arxiv.org/abs/1706.03762.

Wang, J., Han, J., Wei, X., Shen, S., Zhang, D., Fang, C., Chen, R., Yu, W., and Chen, H. KVCache cache in the wild: Characterizing and optimizing KVCache cache at a large cloud provider. *CoRR*, abs/2506.02634, 2025. URL https://doi.org/10.48550/arXiv.2506.02634.

Wang, Y., Yang, Y., and Ren, M. Lifelongmemory: Leveraging llms for answering queries in long-form egocentric videos. *arXiv preprint arXiv:2312.05269*, 2023.

Wu, B., Zhu, R., Zhang, Z., Sun, P., Liu, X., and Jin, X. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pp. 911–927, 2024. URL https://www.usenix.org/conference/osdi24/presentation/wu-bingyang.

Wu, Y., Hee, M. S., Hu, Z., and Lee, R. K.-W. Longgenbench: Benchmarking long-form generation in long context llms. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=3A71qNKWAS.

Xu, F. F., Song, Y., Li, B., Tang, Y., Jain, K., Bao, M., Wang, Z. Z., Zhou, X., Guo, Z., Cao, M., Yang, M., Lu, H. Y., Martin, A., Su, Z., Maben, L. M., Mehta, R., Chi, W., Jang, L. K., Xie, Y., Zhou, S., and Neubig, G. Theagentcompany: Benchmarking llm agents on consequential real world tasks. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025a. URL https://openreview.net/forum?id=LZnKNApvhG.

Xu, R., Ji, Y., Cao, B., Lu, Y., Lin, H., Han, X., He, B., Sun, Y., Li, X., and Sun, L. Memorizing is not enough: Deep knowledge injection through reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28682–28693, Vienna, Austria, July 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1392. URL https://aclanthology.org/2025.acl-long.1392/.

Xu, W., Mei, K., Gao, H., Tan, J., Liang, Z., and Zhang, Y. A-mem: Agentic memory for llm agents, 2025c. URL https://arxiv.org/abs/2502.12110.

Yamin, K., Ghosal, G. R., and Wilder, B. Can llms reconcile knowledge conflicts in counterfactual reasoning. 2025. URL https://api.semanticscholar.org/CorpusID:279464170.

Yan, K., Ling, Z., Liu, K., Yang, Y., Fan, T.-H., Shen, L., Du, Z., and Chen, J. Mir-bench: Benchmarking llm's long-context intelligence via many-shot in-context inductive reasoning, 2025. URL https://openreview.net/forum?id=dwQ0AYwT0q.

Yang, J., Hou, B., Wei, W., Bao, Y., and Chang, S. Kvlink: Accelerating large language models via efficient kv cache reuse, 2025. URL https://arxiv.org/abs/2502.16002.

Yao, J., Li, H., Liu, Y., Ray, S., Cheng, Y., Zhang, Q., Du, K., Lu, S., and Jiang, J. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In

Proceedings of the Twentieth European Conference on Computer Systems, pp. 94–109, 2025. URL https://doi.org/10.1145/3689031.3696098.

Yen, H., Gao, T., Hou, M., Ding, K., Fleischer, D., Izsak, P., Wasserblat, M., and Chen, D. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*, 2024.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M. K., Han, X., Thai, Z. L., Wang, S., Liu, Z., and Sun, M. infty-bench: Extending long context evaluation beyond 100k tokens. *CoRR*, abs/2402.13718, 2024. URL https://arxiv.org/abs/2402.13718.

Zhang, Z., Dai, Q., Bo, X., Ma, C., Li, R., Chen, X., Zhu, J., Dong, Z., and Wen, J.-R. A survey on the memory mechanism of large language model based agents. *ACM Trans. Inf. Syst.*, July 2025. ISSN 1046-8188. doi: 10.1145/3748302. URL https://doi.org/10.1145/3748302. Just Accepted.

Zhao, E., Awasthi, P., and Haghtalab, N. From style to facts: Mapping the boundaries of knowledge injection with finetuning. *ArXiv*, abs/2503.05919, 2025. URL https://api.semanticscholar.org/CorpusID:276902745.

Zheng, W., Ye, X., Xia, P., Wu, F., Li, L., Zhang, W., Wang, L., Choi, Y., Li, Y., and Yao, H. The agent's marathon: Probing the limits of endurance in long-horizon tasks, 2025. URL https://openreview.net/forum?id=dAn82lpLx4.

Zhong, W., Guo, L., Gao, Q., and Wang, Y. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

A. Runtime Motivation Details

Table 6. Motivation scenarios and design target. PAM focuses on shifting durable memory toward LoRA adapters. The decision of which volatile information should remain retrieval-only is not evaluated in this paper and is left to future full-agent systems.

Runtime regime	**Prompt contents at round t**	**Implication**
External memory only	Role prompt + current subtask + retrieved history/tool outputs + prior observations	Flexible and editable, but it lengthens the runtime context and keeps the prefix dynamic.
Prefix-cached external memory	Same as above, but with reuse only when the prompt prefix remains sufficiently stable	Helps serving, but retrieval order changes, memory churn, and agent switching still weaken cache locality.
Practical PAM hybrid	Role prompt + current subtask; durable memory is shifted toward LoRA adapters	Our evaluated regime focuses on parameterizing stable memories. Volatile retrieval is a future full-agent extension.

Table 6 summarizes the runtime regimes behind PAM. If an agent repeatedly replays durable memory as prompt text, both reasoning pressure and serving pressure accumulate across calls. PAM targets the practical regime where stable memories are shifted toward LoRA adapters, while the policy for routing volatile memories to retrieval is left to future full-agent systems.

A.1. Bottleneck 1: long context is not reliable memory

Appending all relevant history to a prompt is the simplest way to give an LLM access to memory, but access is not utilization. Long-context benchmarks show that models can miss relevant information, overweight distractors, become sensitive to evidence position, or lose reliability as tasks require longer induction and generation (Liu et al., 2023; Bai et al., 2023; Yen et al., 2024; Zhang et al., 2024; Yan et al., 2025; Wu et al., 2025). In multi-turn settings, the difficulty compounds: an agent must integrate partial instructions, recover from early assumptions, and preserve state through many turns. The ICLR 2026 multi-turn study shows that top LLMs degrade substantially when a task is delivered through conversation rather than a fully specified single prompt (Laban et al., 2026). Long-horizon agent evaluations and task-time-horizon measurements point in the same direction: as agents are asked to execute longer tasks, error accumulation and context management become central failure modes (Kwa et al., 2025; Sinha et al., 2025; Zheng et al., 2025). Agent memory systems that continually append retrieved facts therefore inherit the same risk at a larger scale.

This bottleneck matters most for durable memories. A user's persistent preferences, a project's stable constraints, or an agent's long-lived domain facts may be needed across many sessions. Re-retrieving those memories every time treats stable knowledge as transient context, forcing the model to rediscover and integrate it on every call. Parameterized memory aims to move this stable subset from the fragile context layer into a persistent learned layer.

A.2. Bottleneck 2: dynamic memory disrupts prefix caching

LLM serving systems accelerate repeated prompts by caching the key-value tensors for shared prefixes. This is especially valuable for agents, whose prompts often contain long role instructions, tool schemas, examples, and workflow-specific policies. Dynamic memory retrieval weakens this reuse: if each call injects a different set or order of memories, the reusable prefix shrinks and cache locality deteriorates. In multi-agent workflows, switching among agents further complicates reuse. Recent KV-cache and prefix-cache systems therefore focus on reuse detection, cache migration, offloading, reloading, eviction, and scheduling under long-context workloads (Qin et al., 2024; Yao et al., 2025; Wang et al., 2025; Liu et al., 2025; Jiang et al., 2026; Li et al., 2025; Bian et al., 2025; Yang et al., 2025; Pan et al., 2025). These systems show that cache reuse is a major lever for serving efficiency, but agent memory makes the reusable prefix itself dynamic.

Parameterizing stable memory can reduce this pressure. If recurring memories no longer have to be included as prompt text, the static prefix becomes more stable, retrieval payloads become smaller, and cache reuse becomes easier to preserve. Our experiments evaluate this stable-memory write path directly. A full agent system should additionally learn when recent, uncertain, infrequent, private, or rapidly changing information should remain retrieval-only, which we leave to future work.

A.3. Workload Accounting for Context Memory

Table 7 expands the simple workload model behind the commented workload figure below. The table is intended as an accounting device rather than a complete serving simulator. It isolates the prompt-side cost of repeatedly replaying durable memories and shows why shifting stable memory into adapters can keep the runtime prompt closer to a fixed budget.

Table 7. Simple workload accounting for a multi-round agent session. Let n_t be the prompt length and o_t be the output length at round t. If each round appends prior observations so that n_t grows approximately linearly with t, cumulative KV-cache demand follows $\sum_t (n_t + o_t)$, matching the $(1 + 2 + \cdots + T)k$ growth pattern when o_t has expectation k.

Quantity	Per-round form	Growth pressure
Prefill attention	$O(n_t^2)$	Attention matrix has $n_t \times n_t$ interactions.
Decode workload	$O(o_t n_t + o_t^2)$	Each generated token attends to the growing prefix.
KV-cache footprint	$O(n_t + o_t)$	Key/value states are retained for processed tokens.
Cumulative KV slots	$O(\sum_{t=1}^{T}(n_t + o_t))$	Becomes $O((1 + \cdots + T)k)$ under linear history growth.

B. LoRA Serving Discussion

PAM stores durable memories in parameter-efficient adapters rather than in separate full model copies. This choice is important for serving. Full-parameter fine-tuning produces a distinct dense checkpoint for each user, task, or memory scope, so a serving cluster must either host many large model replicas or repeatedly load and unload full weights. That design is difficult for both shared clusters and personal serving, where GPU memory is the limiting resource.

LoRA-style memory adapters keep the base model frozen and shared while storing each memory specialization as a small set of low-rank weight deltas (Hu et al., 2021). Serving systems can therefore route a request to a shared base model and attach the relevant adapter at runtime. S-LoRA shows that many adapters can be stored in host memory and fetched into GPU memory with unified paging and batched LoRA kernels (Sheng et al., 2024). Punica uses specialized kernels and scheduling to serve multiple LoRA models while keeping only one copy of the base model on the GPU (Chen et al., 2024). dLoRA further studies dynamic merge/unmerge decisions and request-adapter migration for LoRA serving under skewed workloads (Wu et al., 2024). These systems support the infrastructure intuition behind REMO: parameter memory is more practical when the durable part of a user's memory can live in compact adapters selected per request, rather than in full model forks.

This does not remove the need for external memory. A production agent still needs a routing policy that decides which facts are stable enough for adapter writing and which facts should stay in retrieval because they are fresh, private, uncertain, or rapidly changing. Our LoCoMo experiments isolate the adapter-writing side of this design, and future real-agent evaluations should study adapter selection and volatile retrieval jointly.

C. Additional Experimental Details

C.1. Compute resources

Our experiments were run on a server with 8 NVIDIA RTX A6000 GPUs. End-to-end training for the reported models and ablations required approximately two weeks of wall-clock time in total. We report this number as an approximate project-level compute summary rather than a per-run benchmark, since individual runs varied by backbone, ablation setting, and data size.

C.2. Evaluation data leakage analysis

To verify that the performance gains of PAM are not attributable to memorization of evaluation examples, we conduct a data leakage audit between the augmented training corpus and the LoCoMo evaluation QA pairs.

We define a **leak** as a training instance that contains both the semantic content of an evaluation question and its corresponding answer. We consider two levels of matching:

- **Strict match**: both the exact question string and the exact answer string appear verbatim in a training instance.
- **Semantic match**: at least 80% of the content words in the evaluation question, together with the exact answer string,

appear in a training instance.

To reduce false positives caused by short or ambiguous responses, we exclude answers shorter than four characters from the analysis.

Across all ten LoCoMo splits, strict leakage is negligible: fewer than 0.01% of training instances exactly reproduce an evaluation QA pair. Under the more relaxed semantic criterion, approximately 1.2% of training instances overlap with evaluation QA content. Inspection indicates that most semantic overlaps originate from factual paraphrases or reformulations generated by our augmentation pipeline, rather than direct duplication of evaluation examples.

These results suggest that the improvements achieved by PAM are unlikely to stem from memorization of benchmark answers, and instead reflect improved long-context reasoning and retrieval capabilities learned during training.

D. Broader Impact

This work studies how stable agent memories can be internalized into parameter-efficient adapters rather than repeatedly replayed as prompt text. A potential positive impact is improved efficiency and robustness for long-horizon assistants: reducing repeated memory replay can lower serving overhead, improve answer consistency, and make persistent assistants more accessible in resource-constrained settings.

At the same time, stronger long-term memory can create risks. Better memory internalization may make agents more capable of retaining sensitive user preferences, personal facts, or long-lived interaction histories, which raises privacy and misuse concerns if such systems are deployed without clear retention policies, user controls, or access safeguards. More capable persistent agents could also be misused for manipulation, profiling, or surveillance if memory mechanisms are applied to sensitive domains without adequate oversight.

Our paper does not release a production memory system or user-specific deployed model. It focuses on benchmark-based evaluation of stable memory writing, and it explicitly leaves volatile-memory routing, privacy controls, and deployment safeguards to future work. We therefore view transparent memory policies, selective writing of stable facts, and user-governed deletion or auditing mechanisms as important directions for responsible follow-up work.

E. Data Synthesis Prompt Details

This appendix summarizes the concrete prompt design used in REMO. The main text describes the goals and taxonomy; here we document the generator-facing instructions, required output format, and cross-cutting constraints.

E.1. Universal prompt constraints

All prompt families share the same global constraints:

- **Definitive wording.** Generated text must use factual language and avoid speculative markers such as *might*, *could*, *perhaps*, or *seems*.
- **Exact entity preservation.** Prompts require the exact entity names from the source text and forbid entity swapping or replacing named entities with ambiguous pronouns.
- **Absolute time resolution.** Relative expressions such as "yesterday" or "last week" are resolved into absolute dates before generation. A dedicated temporal strategy later re-introduces relative references as an additional supervision signal.
- **Answer prominence.** The key factual answer must remain the most extractable part of the output instead of being buried in long subordinate clauses.
- **No vague summarization.** Specific items must remain explicit. For example, a list of activities must keep every activity rather than collapse into a phrase such as "various hobbies".
- **Self-contained outputs.** Each line must be independently understandable, with one fact statement per line and no extra explanation.

```
 CRITICAL CONSTRAINTS (apply to EVERY generated statement):
- Use ONLY definitive, factual language.  FORBIDDEN words:  implies, suggests,
 indicates, might, could, perhaps, possibly, may, seems, appears
- Each statement must contain the COMPLETE fact with ALL entities named explicitly
- Do NOT number outputs.  No "1.", "2.", etc.
- Time expressions MUST be absolute dates, NEVER relative expressions
- The key factual answer must be the most prominent, extractable information
- Use the EXACT entity name from the input.  NEVER swap entity names or replace them
 with pronouns when the source uses a name
- Each statement must be self-contained and independently understandable
- One statement per line.  No extra text, no explanations, no headers
- ANTI-VAGUE RULE: never replace a specific fact with a vague generalization; if a fact
 contains a list, ALL items must appear
```

E.2. Group I: Single-utterance enhancement

Forward Assertion (`forward_assertion`, 5 samples). Input: one utterance after entity-name preservation and time normalization. The prompt asks the generator to restate the fact in canonical subject→relation→object form. The subject must remain the sentence anchor, and the output consists of one declarative fact per line.

```
 You are a factual knowledge encoder.  Rewrite the input into clear, definitive
factual statements that maximize answer extractability.
Task:  Forward Assertion
Express the fact in canonical forward direction (Subject -> Relation -> Object).  The
subject entity must be the sentence anchor.
Method:  rephrase into semantically equivalent declarative statements; foreground
Subject-Relation-Object; vary syntax but keep the same factual core; do not weaken
specificity.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} distinct forward assertion statements.  Output ONLY the
statements, one per line.
```

Inverse Indexing (`inverse_indexing`, 5 samples). Input: one utterance. The object entity becomes the primary retrieval hook, while the subject and the complete fact must still appear explicitly. This prompt is designed to create a reverse access path for the same memory.

```
 You are a factual knowledge encoder specializing in reverse retrieval pathways.
Task:  Inverse Indexing
Re-express the fact by making the Object entity the primary access point for recalling
the Subject.
Method:  the object starts or anchors the sentence; the subject is introduced through
the object; the COMPLETE fact must still be present.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} distinct inverse statements.  Output ONLY the statements, one
per line.
```

Attribute-Centric (`attribute_centric`, 5 samples). Input: one utterance. The prompt rewrites the fact as a defining attribute or stable property of an entity, for example using forms such as "One attribute of X is that ..." The attribute value must remain explicit.

```
 You are a factual knowledge encoder specializing in entity-attribute representation.
Task:  Attribute-Centric Expression
Encode the fact as an intrinsic attribute, property, or defining characteristic of the
```

```
relevant entity.
Method: focus on one attribute of one entity; state the attribute value explicitly;
use forms such as "One attribute of X is that ..." or "A defining fact about X is
that ..."
{constraints}
INPUT TEXT: {text}
Generate {num_samples} distinct attribute-centric statements. Output ONLY the
statements, one per line.
```

Premise Injection (`premise_injection`, 5 samples). Input: one utterance. The generator writes the original fact as the main clause and appends a short, realistic consequence as a subordinate clause. The prompt explicitly states that the consequence must not overshadow the fact itself.

```
 You are a factual knowledge encoder specializing in causal knowledge structures.
Task: Premise Injection
Use the input fact as a logical premise that supports a plausible downstream
consequence.
Critical structure: the original fact must be the MAIN CLAUSE; the consequence must
be short and subordinate; do not let the consequence overshadow the original fact.
Method: "[Original fact], which ..." or "[Original fact], so ..."; the consequence
must be realistic and non-speculative; the original fact must remain fully
extractable.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} premise-injection statements. Output ONLY the statements, one
per line.
```

Consequence Extraction (`consequence_extraction`, 5 samples). Input: one utterance. The prompt starts from a concrete clue or observable signal and asks the generator to reconstruct the original fact as the most plausible explanation. The clue must be specific rather than generic.

```
 You are a factual knowledge encoder specializing in backward inference.
Task: Consequence Extraction
Start from a concrete, observable clue and infer the original fact as the most
plausible explanation.
Critical structure: the clue must be specific and concrete; the reconstructed fact
must be complete with all entities named; the clue must strongly point to the fact.
Method: "[Specific observation], which means [Complete fact]" or "[Concrete clue]
reveals that [Complete fact]".
{constraints}
INPUT TEXT: {text}
Generate {num_samples} consequence-extraction statements. Output ONLY the statements,
one per line.
```

Comparative Bridging (`comparative_bridging`, 8 samples). Input: one utterance. The generator expresses the fact by contrasting it with a semantically similar or confusable entity. The compared entity must stay in the same semantic domain, and the contrast must make the correct attribution more salient.

```
 You are a factual knowledge encoder specializing in contrastive knowledge.
Task: Comparative Bridging
Express the fact by contrasting it with a similar or confusable entity within the same
semantic domain.
Method: use "unlike", "different from", "in contrast to", or "compared to"; the
compared entity must be meaningfully similar; the contrast must highlight the
distinguishing aspect tied to the original fact.
{constraints}
```

```
INPUT TEXT: {text}
Generate {num_samples} comparative statements.  Output ONLY the statements, one per
line.
```

Enumeration Drill (`enumeration_drill`, 5 samples). Input: one utterance that may contain multiple items. The prompt forces exhaustive listing of all items, using constructions such as "including", "namely", or "consisting of". It explicitly forbids collapsing the list into a category label.

```
 You are a factual knowledge encoder specializing in complete enumeration.
Task:  Enumeration Drill
When the fact contains multiple items, express it with EXHAUSTIVE enumeration and no
omissions.
Method:  identify ALL items; express them as a complete list using "including",
"namely", "specifically", or "consisting of"; never collapse a list into a vague
category label.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} enumeration statements.  Output ONLY the statements, one per
line.
```

Exact Answer Anchor (`exact_answer_anchor`, 5 samples). Input: one utterance. The exact answer span, such as a person, date, place, or list, is moved to the most prominent position in the sentence. This prompt maximizes answer extractability under lexical matching metrics.

```
 You are a factual knowledge encoder specializing in precise answer extraction.
Task:  Exact Answer Anchor
Encode the fact so that the most specific, concise answer is the anchor point of the
statement.
Method:  place the exact answer at the sentence anchor; use forms such as "The answer
is [exact]" or "[Exact answer] is what ..."; never generalize; if the fact contains
multiple items, list them all explicitly.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} exact-answer-anchored statements.  Output ONLY the statements,
one per line.
```

E.3. Group II: Session context enhancement

Cross-Turn Fact Composition (`cross_turn_composition`, 3 samples). Input: a local conversation window with at least two related utterances. The prompt asks the generator to compose a single coherent statement that preserves all explicit facts from multiple turns without hallucinating any new information.

```
 You are a factual knowledge encoder specializing in multi-turn fact integration.
Task:  Cross-Turn Fact Composition
Combine related facts from different conversation turns into a single coherent factual
statement.
Rules:  MUST integrate at least two utterances; all entities must be named explicitly;
no abstract summarization; preserve all facts exactly; no hallucination; include all
items in any list.
{constraints}
CONVERSATION CONTEXT: {context}
Generate {num_samples} cross-turn composition statements.  Output ONLY the statements,
one per line.
```

Structured Multi-Hop Chain (`multi_hop_chain`, 3 samples). Input: a local multi-turn window. The generator reorganizes the window into a verifiable chain such as condition → action → result or event → reaction → outcome. Each step must be grounded in the conversation.

```
 You are a factual knowledge encoder specializing in structured fact chaining.
Task:  Structured Multi-Hop Chain
Reorganize related facts from the conversation into a clear multi-step structure such
as condition -> action -> result or event -> reaction -> outcome.
Rules:  MUST use at least two utterances; each step must be a verifiable fact; do not
add speculative logic; name all entities explicitly; make the chain relation explicit.
{constraints}
CONVERSATION CONTEXT: {context}
Generate {num_samples} structured multi-hop statements.  Output ONLY the statements,
one per line.
```

Chronological Timeline (`chronological_timeline`, 3 samples). Input: a local multi-turn window with time information. The prompt asks for a time-ordered statement or short timeline that preserves the order of events using absolute dates computed from session timestamps.

```
 You are a factual knowledge encoder specializing in temporal ordering.
Task:  Chronological Timeline Expression
Express facts from the conversation in a time-ordered structure that preserves the
sequence of events.
Rules:  MUST integrate multiple turns; explicitly reflect order of events; use
absolute dates; include key actions, changes, or outcomes; no summarization or
abstraction.
{constraints}
CONVERSATION CONTEXT: {context}
Generate {num_samples} chronological statements.  Output ONLY the statements, one per
line.
```

E.4. Group III: QA-format enhancement

Rhetorical QA Statement (`rhetorical_qa`, 5 samples). Input: one utterance. The prompt embeds the fact inside a natural-language question-answer framing such as "When asked about . . . , X confirmed that . . . " It is intentionally written as plain text rather than instruction tuning data.

```
 You are a factual knowledge encoder.  Embed facts into natural-language
question-answer patterns while staying in pretrain text format.
Task:  Rhetorical QA Statement
Express the fact using a rhetorical question-answer structure embedded in natural
language.
Patterns:  "When asked about ...", "The question of ...  is answered by ...", "If
someone asks ...", "[Entity] can confirm that ..."
Rules:  the factual answer must be the most prominent information; relate the question
naturally to the fact; keep all entity names and details exact; do not generate
standalone Q: A: pairs.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} rhetorical QA statements.  Output ONLY the statements, one per
line.
```

Direct QA as Factual Text (`direct_qa_text`, 5 samples). Input: one utterance. The output format is a direct QA pair: `Question: [question] Answer: [concise answer]`. The answer is restricted to a short span so the model learns a tight question→answer mapping.

```
 You are a factual knowledge encoder.  Express facts as direct QA pairs embedded in
pretrain text.
Task:  Direct QA as Factual Text
Format:  Question:  [question] Answer:  [concise answer, 1-10 words]
Rules:  answers must be concise; questions must uniquely identify the answer; use
diverse question types; do not add explanation beyond the QA pair; time answers must
be absolute dates; never generalize specific facts.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} direct QA pairs.  Output ONLY the QA pairs, one per line.
```

Fact Verification Triple (`fact_verification`, 5 samples). Input: one utterance. The generator produces a three-part structure: factual statement, verification question, and concise confirmed answer. This prompt trains the model to associate a memory with its own verification query.

```
 You are a factual knowledge encoder.  Encode facts with built-in verification.
Task:  Fact Verification Triple
Express each fact as:  (1) a definitive factual statement, (2) a verification question,
and (3) a short confirmed answer.
Format:  [Statement].  Verify:  [question]?  [Answer].
Rules:  the statement must be complete; the verification question must target the key
fact; the answer must be concise; all entity names and time values must stay exact.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} fact verification triples.  Output ONLY the triples, one per
line.
```

Dual-Format Time QA (`dual_format_time_qa`, 5 samples). Input: one time-anchored utterance plus its reference date. The prompt requires two QA forms for the same event: one with the exact absolute date and one with a relative reference such as "the Friday before [date]." The two answers must denote the same time point.

```
 You are a factual knowledge encoder specializing in temporal information.
Task:  Dual-Format Time QA
For each time-anchored fact, generate TWO QA pairs:  one with the exact absolute date
and one with a relative-reference format.
Format:  Question:  [question] Answer:  [absolute date] OR Question:  [question]
Answer:  the [weekday/week/weekend] before [reference date]
Rules:  MUST generate both formats; both answers must denote the same time point; use
the closest known reference date; keep entity names exact.
{constraints}
INPUT TEXT: {text}
Generate {num_samples} dual-format time QA pairs.  Output the QA pairs, one per line.
```

E.5. Group IV: Full-session extraction

Entity Profile Extraction (`entity_profile`, 3 samples). Input: the full conversation and a target entity. The prompt extracts all facts associated with that entity into a structured fact profile. It requires exhaustive coverage over attributes, events, preferences, possessions, and time-anchored facts.

```
 You are a factual knowledge encoder.  Extract ALL facts about a specific entity and
organize them into a structured profile with EXHAUSTIVE enumeration.
Task:  Entity Profile Extraction
Format:  [Entity name]'s fact profile:  - [Fact 1] - [Fact 2] ...
Rules:  include ONLY facts belonging to the specified entity; use absolute dates; be
exhaustive across identity, relationships, activities, events, possessions, and
preferences; include all items explicitly; include image-derived facts when present.
```

```
{constraints}
CONVERSATION CONTEXT: {context}
ENTITY TO PROFILE: {entity}
Generate the complete entity profile.  Output the profile as a single block.
```

Temporal Event Registry (`temporal_registry`, 1 sample). Input: the full conversation. The generator enumerates all events with temporal anchors in chronological order, using absolute dates and, when available, a matched relative reference in parentheses.

```
 You are a factual knowledge encoder specializing in temporal information extraction.
Task:  Temporal Event Registry
Extract ALL time-anchored events from the conversation and organize them into a
chronological registry with absolute dates.
Format:  Event registry:  - [Absolute date]:  [Entity] [event with details]
Rules:  convert all relative times into absolute dates; order chronologically; include
every event with any temporal anchor; omit events without time information; include
relative-reference form in parentheses when available.
{constraints}
CONVERSATION CONTEXT: {context}
Generate the complete temporal event registry.  Output the registry as a single block.
```

Topic-Clustered Fact Sheet (`topic_fact_sheet`, 5 samples). Input: the full conversation. The prompt groups facts into semantic topics such as activities, family, art, travel, or work. Facts may appear in multiple topics if that improves retrieval by semantic domain.

```
 You are a factual knowledge encoder specializing in thematic knowledge organization.
Task:  Topic-Clustered Fact Sheet
Organize facts from the conversation into thematic clusters.
Format:  Topic:  [topic name] - [Fact 1] - [Fact 2] ...
Rules:  group by natural semantic topics; each fact must name the entity explicitly;
use absolute dates; a fact may appear under multiple topics; preserve all factual
detail; each topic cluster must contain at least two facts.
{constraints}
CONVERSATION CONTEXT: {context}
Generate a topic-clustered fact sheet covering the major topics in the conversation.
Output the complete fact sheet.
```

Entity Boundary Guard (`entity_boundary_guard`, 5 samples). Input: the full conversation. For each confusable fact, the generator writes both the correct attribution and an explicit denial for the wrong entity. This creates hard negative examples for entity-swap errors.

```
 You are a factual knowledge encoder specializing in entity-disambiguation training.
Task:  Entity Boundary Guard
For each fact, generate statements that assert the correct entity attribution and
explicitly deny the same fact for the confusable entity.
Method patterns:  "[Entity A] [fact].  [Entity B] did NOT ...", "Question:  Is [fact]
true for [Entity B]? Answer:  No ...", "Unlike [Entity B], [Entity A] ..."
Rules:  generate both the positive assertion and the negative denial; name the wrong
entity explicitly; cover all facts where entity confusion is possible; include visual
facts when present.
{constraints}
CONVERSATION CONTEXT: {context}
Generate {num_samples} entity boundary guard statements.  Output ONLY the statements,
one per line.
```

Contrastive Entity Disambiguation (`contrastive_entity_disambiguation`, 5 samples). Input: the full conversation. The prompt creates side-by-side comparisons for the same topic domain, listing both entities' facts together. If one entity has no known fact in a domain, the prompt requires that absence to be stated explicitly.

```
 You are a factual knowledge encoder specializing in preventing entity confusion.
Task:  Contrastive Entity Disambiguation
Create side-by-side comparisons of both entities' facts within the same semantic
domain.
Format:  [Domain]:  [Entity A] vs [Entity B] - [Entity A]: [fact] - [Entity B]: [fact]
Rules:  list both entities' facts explicitly for each domain; if one entity has no
fact in a domain, state that explicitly; use absolute dates; include visual facts; be
exhaustive within each domain.
{constraints}
CONVERSATION CONTEXT: {context}
Generate {num_samples} contrastive entity disambiguation entries.  Output each entry
as a block.
```

E.6. Post-processing and filtering

Prompting alone is not sufficient, so we apply lightweight automatic filtering before adding a sample to the final corpus. A candidate generation is removed if it (i) uses speculative or hedged language, (ii) contains unresolved relative time, (iii) replaces specific facts with vague summaries, or (iv) drops required entities or list items. This filter is especially important for preserving exact-answer supervision and preventing entity drift.

E.7. Design summary

The full prompt suite contains 20 strategies. Their roles are complementary:

- Group I creates multiple local lexical access paths for a single fact.
- Group II teaches the model to compose facts across turns and across time.
- Group III aligns stored memories with downstream question-answer retrieval.
- Group IV constructs high-coverage entity-, timeline-, and topic-level summaries for durable memory storage.

Together, these prompts turn the same conversation into a dense supervision graph. A fact can be recalled directly, inversely, contrastively, temporally, and through QA-style access, which is precisely the behavior we want from parameterized agent memory.