REMEMBER BEFORE YOU'RE ASKED: MEMDREAM FOR SELF-PROBING MEMORY EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory lets LLM-based agents stay coherent and personalized over long-horizon interactions. Stored facts can remain inaccessible when relevant evidence is fragmented, buried, or diluted during retrieval. Self-generated probing queries offer a direct signal for locating and repairing these latent failures. We ask whether a memory system can find and fix its own failures before they occur. We present MEMDREAM, a framework for *self-probing memory evolution*. During offline dream cycles, three specialized agents—Dreamer, Analyst, and Consolidator—probe, diagnose, and repair the memory graph before failures happen. A policy trained via Group Relative Policy Optimization learns which repair operations yield durable retrieval improvements, and a soft decay mechanism provides reversible forgetting driven by the same anticipatory signal. On LoCoMo and MemoryAgentBench (MAB), MEMDREAM improves answer F1 by 3.98 points on LoCoMo and achieves a 9.1-point higher overall score on MAB over the strongest evaluated baselines. We hope self-probing memory evolution will serve as a solid baseline for agents that maintain their own memory.

1 INTRODUCTION

LLM-based agents now perform well in interactive settings such as multi-step reasoning, tool use, and open-ended dialogue (Wang et al., 2024; Li et al., 2024; Schlegel et al., 2025). Over long-horizon interactions, an agent must accumulate, retain, and retrieve information across many turns as user goals and contexts shift (Maharana et al., 2024). Memory carries this information forward, and it governs whether responses stay coherent and behavior stays consistent over time. But it can be wrong, and an agent is only as reliable as the memory it can correct before a failure reaches the user (Zhang et al., 2025b; Pan et al., 2025).

Prior work on long-horizon memory for LLM agents falls into three lines. (1) *Structured memory architectures* organize information into hierarchical tiers or graphs, which enriches retrieval but fixes the organization in advance (Packer et al., 2024; Zhong et al., 2024; Lu et al., 2026; Kang et al., 2025). (2) *Multi-stage retrieval pipelines* combine sparse, dense, and reranking stages for precision, but leave the stored memory unchanged (Gutiérrez et al., 2025; Asai et al., 2024). (3) *Self-evolving memory methods* reflect over accumulated observations to synthesize higher-level memories (Park et al., 2023; Shinn et al., 2023), self-question stored content to reorganize it (Yang et al., 2026a), tune retrieval configurations from observed failures (Liu et al., 2026), or learn ADD / UPDATE / DELETE policies with reinforcement learning (Yan et al., 2025; Wang et al., 2025c; Yuan et al., 2025; Zhang et al., 2026; 2025a; Cai et al., 2025; Zhai et al., 2025). These advances moti-

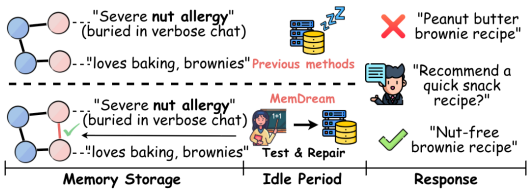

Figure 1: (a) Existing reactive systems store a user's nut allergy and baking preferences as disconnected nodes, missing the dependency between them. When queried for a recipe, the system recommends a nut-containing dish, a harmful answer the user must catch. (b) Through asynchronous background consolidation, MEMDREAM's Dream Cycle finds and repairs this missing link, preventing the unsafe recommendation before the query arrives.

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

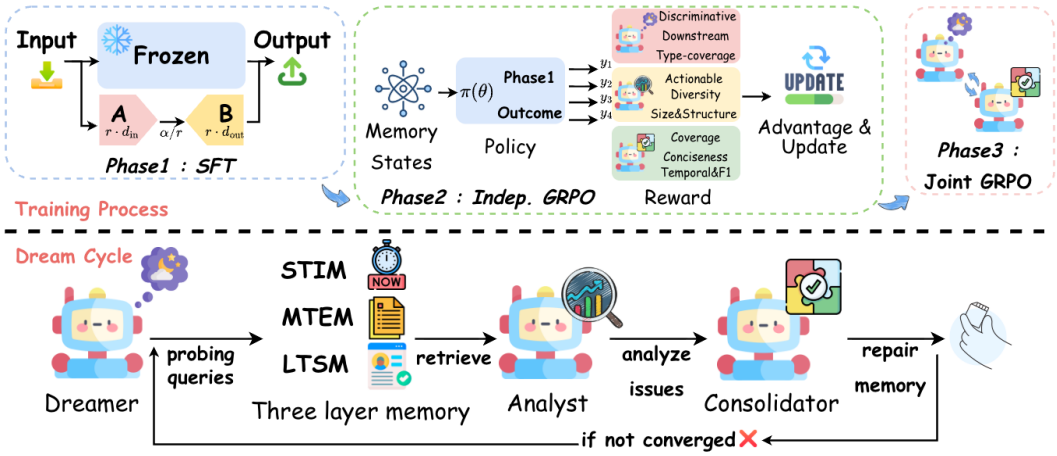

Figure 2: Overview of MEMDREAM. Three LoRA-specialized agents share one Llama-3.1-8B backbone: the Dreamer generates probing queries aimed at weak regions, the Analyst diagnoses failures along four axes (buried, scattered, redundant, noisy), and the Consolidator applies repairs (extract, merge, synthesize, prune). The cycle repeats until retrieval F1 converges. We first train the agents independently with GRPO, then jointly fine-tune the Dreamer and Consolidator (with the Analyst frozen) using end-to-end memory quality as the reward.

vate a complementary question: how can a memory system construct a repair signal that identifies information it stores but cannot reliably retrieve?

By the time a real query exposes a weakness, the user has already hit it. As interactions accumulate, facts fragment across disconnected nodes, links between sessions go missing, and entries grow stale. Self-generated probing questions can expose these defects before a user encounters them: the system tests whether relevant evidence is retrievable, diagnoses the failure, and repairs the affected region. Reversible forgetting retains demoted content for later queries (Wang et al., 2025a).

To address this, we present MEMDREAM, a framework for *self-probing memory evolution*. The key insight: a memory system should test itself. Inspired by memory consolidation during sleep (Kumaran et al., 2016), MEMDREAM enters offline dream cycles in which three agents work in turn: a Dreamer probes the memory graph with forward-looking queries, an Analyst diagnoses where retrieval breaks down, and a Consolidator repairs the graph before a real query can fail. A policy trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Guo et al., 2025) replaces hand-crafted heuristics, learning which repair operations yield durable retrieval gains. A soft decay mechanism adds reversible forgetting, driven by the same anticipatory signal. Together, these let MEMDREAM fix weak regions before a real query reaches them, rather than after. On LoCoMo and MemoryAgentBench (MAB), it improves answer F1 by 3.98 points on LoCoMo and overall score by 9.1 points on MAB over the strongest evaluated baselines, FluxMem and O-Mem, respectively. Our contributions are:

❶ **Multi-Agent Dream Cycle.** Three agents (Dreamer, Analyst, Consolidator) generate forward-looking probes that turn unobserved future queries into a repair training signal.

❷ **Learned Repair Policy.** GRPO trains the Consolidator on changes in retrieval success for the Dreamer's queries, replacing hand-tuned heuristics and assigning credit before user queries arrive.

❸ **Reversible Selective Forgetting.** The same anticipatory signal drives soft decay: nodes are demoted rather than deleted and restored when later cycles reveal renewed relevance.

2 RELATED WORK

Memory Architectures for LLM Agents. Memory designs for LLM agents have evolved from flat retrieval over vector stores (Lewis et al., 2020; Gao et al., 2023) to hierarchical and structured architectures (Edge et al., 2024; Sarthi et al., 2024): MemGPT virtualizes context across tiers (Packer

et al., 2024); MemoryBank adds Ebbinghaus-style strengthening and decay (Zhong et al., 2024); MemoryOS adopts OS-inspired storage and eviction (Kang et al., 2025); FluxMem selects memory structures by interaction characteristics (Lu et al., 2026); and MAGMA organizes memory as multi-graphs for relational retrieval (Jiang et al., 2026). These architectures shape *how memory is organized and retrieved*, but leave content maintenance to implicit or hand-crafted rules. MEMDREAM is complementary: it operates on top of such graphs, repairing their content through anticipatory dream cycles rather than changing the storage layer.

Self-Evolving and Policy-Based Memory. Adaptive memory systems learn update policies from task feedback (Yan et al., 2025; Wang et al., 2025c; Yuan et al., 2025; Zhang et al., 2026), reflect on stored experience (Park et al., 2023; Yang et al., 2026a), or consolidate memory between sessions (Anthropic, 2026). Recent methods target complementary objectives: Auto-Dreamer learns offline region rewriting through GRPO (Ye et al., 2026); TrustMem verifies memory transitions for coverage, preservation, and faithfulness (Yang et al., 2026b); and RecMem reduces construction costs through recurrence-triggered consolidation (Dai et al., 2026). MEMDREAM focuses on the repair signal: self-generated queries expose latent retrieval failures, whose diagnoses guide targeted repairs. Appendix M provides a detailed comparison.

Selective Forgetting. Continual learning has recognized the need to forget selectively (Kirkpatrick et al., 2017; Parisi et al., 2019; Shin et al., 2017); in LLMs, this appears as weight-level unlearning (Jin et al., 2024; Das et al., 2024), fine-grained selective forgetting (Wang et al., 2025a), and energy-based continual learning (Li et al., 2020). Closest to our setting, FSFM combines passive decay, active deletion, safety-triggered removal, and RL-based forgetting in one hybrid pipeline (Gu et al., 2026). All of these treat forgetting as a standalone operation with its own rules or scores. Our *augment_decay* instead runs on the *same* anticipatory signal as consolidation: a node decays only when it is predicted to hurt future retrieval, and returns when a later dream cycle finds it relevant. Forgetting is thus reversible within the dream cycle, rather than a separate, permanent step.

3 METHODOLOGY

MEMDREAM tests its memory and repairs its weaknesses through asynchronous background dream cycles (Figure 2). The agent continues answering queries using its current conversational context and existing memory, while recent interactions remain available in context during consolidation. Online interaction does not wait for a dream cycle to complete. It has two parts: a hierarchical memory store and a three-agent dream cycle that probes and repairs it. We define the problem and memory architecture, then describe the dream cycle and GRPO-based policy learning.

3.1 PROBLEM FORMALIZATION

We consider a long-horizon setting where an agent converses with a user over many turns. Let $\mathcal{H}_t = \{p_1, \ldots, p_t\}$ be the dialogue history up to turn t, where each page $p_i = (u_i, a_i)$ is a user utterance paired with an agent response. Memory is a hierarchical store $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with three tiers: short-term interaction memory (STIM) buffers recent pages with LRU eviction, mid-term episodic memory (MTEM) groups pages into sessions with session-level metadata, and long-term semantic memory (LTSM) holds compressed knowledge entries. A node $v \in \mathcal{V}$ is a memory page, carrying a content embedding, an entity set, access statistics, and a retrieval weight $w_v \in [0, 1]$. An edge $e \in \mathcal{E}$ connects nodes that share entities, belong to the same session, or were merged during consolidation. At turn t, given a user query q_t, the agent retrieves relevant context $\mathcal{C}_t = \text{Retrieve}(q_t, \mathcal{G})$ and generates a response $y_t = \text{Generate}(q_t, \mathcal{C}_t)$. Retrieval uses reciprocal rank fusion (RRF) over dense and sparse retrievers, weighted by each node's retrieval weight:

$$s(v, q) = \left(\sum_{i \in \{\text{dense, BM25}\}} \frac{1}{k + \text{rank}_i(v, q)} \right) \cdot w_v, \tag{1}$$

where $\text{rank}_i(v, q)$ is the rank of node v under retriever i, k is a smoothing constant, and $w_v \in [0, 1]$ is the node's retrieval weight (modulated by the dream cycle). Top-ranked nodes after cross-encoder reranking form the context $\mathcal{C}_t$. The memory is then updated: $\mathcal{G}_{t+1} = \text{Update}(\mathcal{G}_t, p_t)$.

Left to accumulate passively, $\mathcal{G}$ develops three failure modes: **fragmentation** (related information scattered across disconnected nodes), **redundancy** (overlapping content in multiple nodes), and **staleness** (outdated information conflicting with recent interactions). These compound over time, lowering retrieval precision. We formalize memory evolution as learning a policy $\pi : \mathcal{G} \mapsto \mathcal{G}'$ that maximizes expected retrieval success over a distribution of likely future queries:

$$\mathbb{E}_{q\sim\mathcal{Q}}\left[\text{Success}(q,\ \text{Retrieve}(q,\ \mathcal{G}'))\right], \tag{2}$$

where $\mathcal{Q}$ denotes a distribution of plausible future queries, not a corpus of past observed failures. Since $\mathcal{Q}$ is unknown, the dream cycle (§3.2) learns to draw samples from it, converting an unobservable distribution into a tractable training signal.

3.2 Multi-Agent Dream Cycle

The dream cycle periodically probes the memory graph for weaknesses, diagnoses their causes, and repairs them before real queries expose them. Three agents divide this work over one LLM backbone, each with its own LoRA adapter: the Dreamer probes, the Analyst diagnoses, and the Consolidator repairs. The shared backbone keeps all three lightweight, and the separate adapters let each learn a distinct role. Adapter and quantization settings appear in the experimental setup.

Dreamer. Generates probing queries that target weak regions of the memory graph. The Dreamer first computes a weakness score for each node:

$$\begin{aligned}\phi(v) = \lambda_1\Big(1 - \frac{\deg(v)}{\max_u \deg(u)}\Big) \\ + \lambda_2 \cdot \max_{u\in\mathcal{N}(v)} \cos(\mathbf{e}_v, \mathbf{e}_u) \\ + \lambda_3 \cdot \frac{t_{\text{now}} - t_v}{T},\end{aligned} \tag{3}$$

where the three terms capture connectivity deficit (isolated nodes), redundancy (high similarity to a neighbor), and staleness (time since last access), respectively. We select nodes with $\phi(v)$ above a threshold as targets. Because $\phi(v)$ scores nodes individually, we then pair each target $v^\star$ with a *bridge partner*: the unlinked node most often co-retrieved with $v^\star$ across local probe queries $\mathcal{Q}(v^\star)$,

$$u^\star = \underset{u:\,(u,v^\star)\notin\mathcal{E}}{\arg\max}\ \mathbb{E}_{q\sim\mathcal{Q}(v^\star)}[\mathbf{1}[\{u, v^\star\} \subseteq \text{Retrieve}(q, \mathcal{G})]], \tag{4}$$

where $\text{Retrieve}(q, \mathcal{G})$ is the retrieved set and $\mathcal{E}$ the edge set. Pairing by co-retrievability rather than embedding similarity links nodes that recur together yet lack an edge, and the Dreamer then constructs a query that integrates $\{v^\star, u^\star\}$. For instance, if separate nodes store a user's symptoms and medication history with no connecting edge, the Dreamer generates a query requiring both. We pair each probe with an *expected answer* copied verbatim from a target node's stored content rather than generated by the model, so the repair signal stays grounded in existing memory and cannot chase a hallucinated target. We train the Dreamer with SFT on (memory state, effective query) pairs, then GRPO with retrieval success as reward.

Analyst. Receives each dream query, its retrieval results, and the expected answer, then classifies the failure along two axes: whether it lies within a single node or across nodes, and whether it costs recall or precision. The four cells are **buried** (intra-node recall: a fact is present but hidden in verbose text), **scattered** (inter-node recall: evidence split across unlinked nodes), **redundant** (inter-node precision: overlapping content across nodes), and **noisy** (intra-node precision: low-value nodes such as greetings or filler). Each cell maps to one repair the Consolidator can execute: extract, synthesize, merge, or prune. For each failure the Analyst emits the target node ids and the prescribed operation,

Table 1: Repair operation usage over the full evaluation.

Category (2×2 cell)	Repair	Count
buried (intra-node, recall)	extract	176
scattered (inter-node, recall)	synthesize	301
redundant (inter-node, precision)	merge	59
noisy (intra-node, precision)	prune	301
Total		837

ordering them worst-first so the most damaging defects are repaired first. Because the axes enumerate how retrieval can fail rather than list hand-picked symptoms, the taxonomy is complete over these two axes and extends by refining a cell into subcases without changing the Dreamer, the reward, or the training pipeline. Across the evaluation all four cells are exercised (Table 1), so diagnosis does not collapse onto one category.

Consolidator. Executes the repairs the Analyst prescribes. Extract turns a verbose page into structured nodes. Synthesize generates summary nodes with provenance links. Merge combines similar nodes, preserving edge connectivity. Prune applies a soft decay rather than hard deletion:

$$w_v \leftarrow \gamma \cdot w_v, \quad \gamma = 0.2, \tag{5}$$

where γ is a retention factor that keeps a γ fraction of the node's weight: $\gamma{=}0$ deletes the node outright (the w/o Decay ablation), while γ near 1 leaves it almost unchanged. Decay lowers the node's retrieval ranking but keeps it accessible (Eq. 1), so pruning never loses information. If a later cycle shows renewed relevance, the weight is restored:

$$w_v \leftarrow \min(1.0,\ w_v + \eta \cdot \text{success_count}_v). \tag{6}$$

Each dream round processes a different partition of memory nodes until convergence:

$$\frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} \text{F1}(q, \mathcal{G}) \geq \tau \quad \text{or} \quad t \geq T_{\max}, \tag{7}$$

where $\mathcal{D}$ denotes dream queries, τ the convergence threshold, and $T_{\max}$ the round limit.

3.3 GRPO-Based Agent Training

We train the three dream-cycle agents via Group Relative Policy Optimization (GRPO). Each agent optimizes its own reward: the Dreamer for generating discriminative probes, the Analyst for accurate diagnosis, and the Consolidator for repairs that improve retrieval. The agents share the Llama-3.1-8B backbone through separate LoRA adapters.

3.3.1 Per-Agent Rewards

Each agent optimizes its own fixed-weight reward, a sum of interpretable terms:

$$r_{\text{D}} = w_{\text{disc}}\, r_{\text{disc}} + w_{\text{down}}\, r_{\text{down}} + w_{\text{cov}}\, r_{\text{cov}}, \tag{8}$$

$$r_{\text{A}} = w_{\text{act}}\, r_{\text{act}} + w_{\text{div}}\, r_{\text{div}} + w_{\text{size}}\, r_{\text{size}} + w_{\text{struct}}\, r_{\text{struct}}, \tag{9}$$

$$r_{\text{C}} = w_{\text{qa}}\, r_{\text{qa}} + w_{\text{cov}}\, r_{\text{cov}} + w_{\text{temp}}\, r_{\text{temp}} + w_{\text{concise}}\, r_{\text{concise}}. \tag{10}$$

The **Dreamer** terms favor probes that are discriminative, expose downstream failures, and cover diverse query types: r_{disc} rewards queries that separate weak from strong memory regions, r_{down} measures downstream QA impact, and r_{cov} measures coverage over query types. The **Analyst** terms favor diagnoses that are actionable, varied in repair type, appropriately sized, and well-structured: r_{act} is the fraction of repairs that reference valid target nodes, r_{div} rewards using distinct repair types, r_{size} peaks at a moderate number of repairs per cycle, and r_{struct} rewards well-formed output. The **Consolidator** reward is operation-specific. For an extract operation (r_{C}), r_{qa} is QA-F1 on affected queries, r_{cov} measures retained coverage, r_{temp} rewards preserved temporal cues, and r_{concise} penalizes bloat. Merge and synthesize use analogous rewards that trade QA-F1 and coverage against deduplication and grounding. All weights are fixed constants, set a priori from each term's role (primary objective versus regularizer) rather than tuned on the evaluation benchmarks. Their values are given in the appendix. For each training instance, we sample N rollouts from the current policy and normalize their advantages within the group:

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{N} \sum_{i=1}^{N} A_i \cdot \log \pi_\theta(a_i \mid s),$$
$$A_i = \frac{r_i - \bar{r}}{\sigma_r + \epsilon}, \tag{11}$$

where $\bar{r}$ and σ_r are the group mean and standard deviation of rewards, and π_θ is the LoRA-adapted agent policy. Only rollouts above $\bar{r}$ receive positive gradient. This yields a harder curriculum.

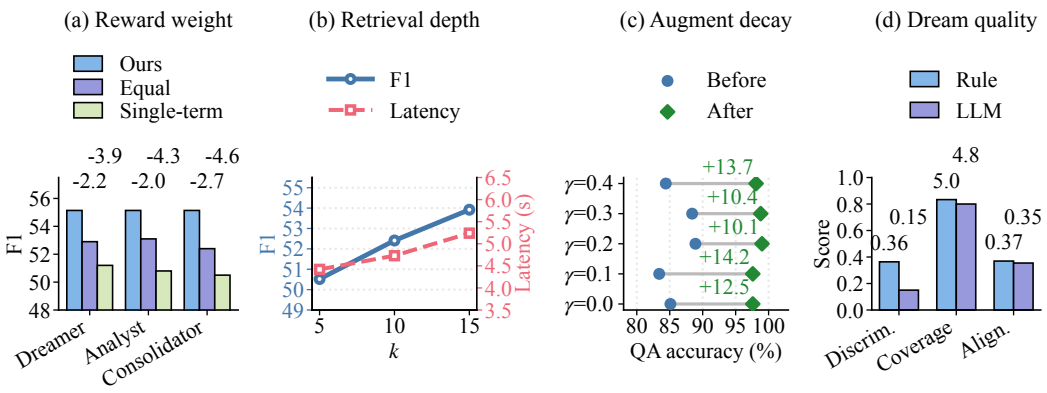

Figure 3: Sensitivity analysis. (a) Per-agent GRPO weights outperform equal weighting and single-term ablations. (b) Increasing retrieval depth k improves F1 with diminishing returns and linear latency growth. (c) Dream cycles improve accuracy at every decay factor γ. (d) Rule-based probes lead on discrimination and alignment, with similar coverage.

3.3.2 Training Procedure

Training proceeds in three phases on MSC dialogue data with pre-generated dream-augmented queries. First, the Dreamer, Analyst, and Consolidator are pre-trained via SFT on teacher data (GPT-4.1 outputs). Second, each agent undergoes independent GRPO with its own reward (Eqs. 8–10). Third, Dreamer and Consolidator are jointly fine-tuned while Analyst remains frozen. The joint reward measures memory improvement:

$$r_{\text{joint}} = \frac{1}{|\mathcal{D}_{\text{new}}|} \sum_{q \in \mathcal{D}_{\text{new}}} \left[\text{F1}(q, \mathcal{G}') - \text{F1}(q, \mathcal{G}) \right], \tag{12}$$

where $\mathcal{D}_{\text{new}}$ is a fresh set of Dreamer-generated queries (not seen during independent training), $\mathcal{G}$ is the memory before consolidation, and $\mathcal{G}'$ is the memory after. This rewards the Dreamer for queries that expose fixable weaknesses, and the Consolidator for repairs that durably improve retrieval. This joint reward replaces the per-agent reward in Eq. 11, with gradients flowing only to the Dreamer and Consolidator adapters. The agents are trained with learning rate 1×10^{-3}, group size N=4, and exploration rate ϵ=0.2 that decays over training. Full hyperparameters are provided in Appendix D. These training stages are completed before deployment. During deployment, model parameters remain fixed, and dream cycles update only the external memory.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on two benchmarks. **LoCoMo** (Maharana et al., 2024) contains 10 long-horizon conversations and 1,986 human-written QA pairs (about 199 per conversation). Following the standard protocol, we evaluate 1,540 multi-hop, temporal, open-ended, and single-hop questions, excluding the adversarial category. **MemoryAgentBench (MAB)** (Hu et al., 2026) covers four memory competencies: Accurate Retrieval (AR), Test-Time Learning (TTL), Long-Range Understanding (LRU), and Selective Forgetting (SF), across 9 sub-tasks including RULER QA, EventQA, LongMemEval, ICL classification, Detective QA, InfBench summarization, and FactConsolidation.

Baselines. We compare against ten memory systems: **LangMem** (LangChain, 2025): a flat memory with embedding-based retrieval. **Mem0** (Chhikara et al., 2025): a flat retrieval-based system with incremental extraction and deduplication. **ZEP** (Rasmussen et al., 2025): a time-aware knowledge graph memory with entity-level timestamping. **MemoryOS** (Kang et al., 2025): an OS-inspired memory with tiered storage and eviction policies. **A-Mem** (Xu et al., 2025): a structured memory using evolving linked notes with self-reflection. **O-Mem** (Wang et al., 2025b): a graph-based memory with explicit temporal ordering and fact-update mechanisms. **MEMOS** (Li et al., 2025): a multi-granularity memory with redundancy filtering. **MemR**^{3} (Du et al., 2025): a closed-loop controller

Method	Cat1: Multi-hop			Cat2: Temporal			Cat3: Open			Cat4: Single-hop			**Average**		
	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L
LangMem	41.11	32.09	37.71	53.67	46.22	47.37	33.38	27.26	24.55	51.13	44.22	48.37	44.82	37.45	39.50
Mem0	28.84	20.22	27.45	24.21	18.75	23.23	23.59	19.09	23.13	32.74	27.58	33.02	27.35	21.41	26.71
ZEP	8.74	5.98	6.79	7.40	4.70	6.07	7.64	3.46	5.18	7.91	3.80	6.26	7.92	4.49	6.08
MemoryOS	43.03	34.09	39.75	44.43	32.03	42.64	28.36	23.24	27.83	49.31	42.61	49.84	41.28	32.99	40.02
A-Mem	32.41	24.28	31.26	37.23	33.02	39.61	16.16	14.64	15.19	40.94	35.54	42.32	31.69	26.87	32.10
MEMOS	20.27	13.35	15.75	14.86	7.86	11.58	11.87	6.85	9.32	18.96	11.42	17.37	16.49	9.87	13.51
MemR^3	18.65	11.17	14.37	25.39	15.90	22.63	11.56	5.77	8.96	24.35	12.72	22.00	19.99	11.39	16.99
O-Mem	42.64	34.08	32.78	57.48	**49.76**	48.88	30.58	25.69	21.11	54.89	48.98	44.26	46.40	39.63	36.76
HippoRAG 2	28.11	18.92	27.68	10.35	6.92	9.82	24.91	21.10	24.48	44.82	37.25	44.37	27.05	21.05	26.59
FluxMem	48.56	39.90	44.05	56.67	41.88	54.89	**37.30**	29.63	**36.76**	62.12	53.52	62.33	51.16	41.23	49.51
MemDream	**53.64**	**43.88**	**47.50**	**66.05**	49.07	**63.43**	35.89	**30.69**	35.43	**64.97**	**57.66**	**65.12**	**55.14**	**45.33**	**52.87**
Improv.	+5.08	+3.98	+3.45	+8.57	-0.69	+8.54	-1.41	+1.06	-1.33	+2.85	+4.14	+2.79	+3.98	+4.10	+3.36

Table 2: LoCoMo results (GPT-4.1 reader). Best in bold; second-best underlined.

Method	Cat1: Multi-hop			Cat2: Temporal			Cat3: Open			Cat4: Single-hop			**Average**		
	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L
Mem0	9.96	6.25	7.69	10.73	5.99	8.86	7.23	5.08	6.00	18.40	11.21	16.76	11.58	7.13	9.83
ZEP	15.04	11.56	2.89	3.49	2.68	2.78	**26.67**	18.44	2.57	30.15	17.15	3.37	18.84	12.46	2.90
MemoryOS	13.60	10.57	24.91	22.18	18.68	29.85	9.11	8.21	17.60	24.81	21.09	38.14	17.43	14.64	27.63
A-Mem	20.46	14.37	21.53	25.33	22.01	28.45	15.62	13.72	14.95	33.45	28.85	37.07	23.72	19.74	25.50
MEMOS	33.00	24.19	32.40	42.16	34.66	43.26	17.97	12.79	16.15	41.91	33.01	48.89	33.76	26.16	35.18
MemR^3	9.27	4.52	5.92	7.47	2.76	4.48	6.50	2.72	3.73	12.07	4.41	8.33	8.83	3.60	5.62
O-Mem	35.62	27.27	32.78	49.77	37.21	48.88	21.41	15.97	21.11	44.08	38.26	44.26	37.72	29.68	36.76
HippoRAG 2	23.23	**29.48**	21.47	7.55	9.77	7.46	19.77	**27.67**	19.30	34.85	35.79	34.35	21.35	25.68	20.65
FluxMem	25.61	29.22	23.85	30.54	32.17	29.93	18.26	24.03	17.39	37.58	38.19	36.43	28.00	30.90	26.90
MemDream	**39.00**	25.09	**34.85**	**52.70**	**37.57**	**50.64**	22.48	25.50	**22.17**	**52.02**	**42.79**	**52.34**	**41.55**	**32.74**	**40.00**
Improv.	+3.38	-4.39	+2.07	+2.93	+0.36	+1.76	-4.19	-2.17	+1.06	+7.94	+4.53	+3.45	+3.83	+1.84	+3.24

Table 3: LoCoMo results (Llama-3.1-8B reader). Best in bold; second-best underlined.

alternating retrieval, reflection, and reranking for evidence coverage. **HippoRAG 2** (Gutiérrez et al., 2025): a hippocampus-inspired knowledge-graph retrieval framework. **FluxMem** (Lu et al., 2026): an adaptive multi-structure memory that learns to select among memory organizations per interaction. All systems use the same reader LLM (GPT-4.1, temperature 0).

Implementation Details. MemDream uses a three-layer memory hierarchy (STIM capacity 4, MTEM up to 2000 sessions, LTSM up to 100 entries). The dream cycle deploys three LoRA-adapted agents on Llama-3.1-8B-Instruct with 8-bit quantization (rank r=16, α=32, applied to the attention projections; $\sim$39.3M trainable parameters total). All training and inference run on a single A100 (80GB). Retrieval fuses BM25 and dense encoding (BAAI/bge-m3) via reciprocal rank fusion, followed by cross-encoder reranking (bge-reranker-v2-m3). The agents are trained via GRPO with group size N=4 on MSC dialogue data. On LoCoMo, we use chunk size 4096 with the full dream consolidation pipeline. On MAB, we use chunk size 512 for AR tasks (GPT-4.1 reader) and SF tasks, chunk size 4096 for TTL, LRU, and AR tasks with the Llama-8B reader.

Evaluation Metrics. On LoCoMo, we report F1, BLEU-1 (B-1), and ROUGE-L (R-L) per category; Average equally weights the four categories. On MAB, we follow the original protocol: substring accuracy for AR, SF, TTL, and Detective QA, and ROUGE-L for summarization. We report results with deterministic decoding (temperature 0), following the LoCoMo and MAB protocols. The gains are stable under bootstrap confidence intervals and repeated runs; full per-category intervals are given in the appendix.

Method	AR				TTL	LRU		SF		Overall
	SH-QA	MH-QA	LME	EventQA	MCC	Summ.	Det QA	FC-SH	FC-MH	
A-Mem	24.0	38.0	6.0	68.0	29.0	15.6	52.1	8.0	2.0	27.0
HippoRAG 2	30.0	46.0	7.7	69.6	54.0	16.2	47.9	18.0	6.0	32.8
MemoryOS	48.0	45.0	9.0	70.6	27.0	16.9	**73.2**	11.0	5.0	34.0
MEMOS	32.0	49.0	6.0	66.2	55.0	16.1	49.3	8.0	1.0	31.4
MemR^3	27.0	40.0	6.7	66.6	47.0	16.1	49.3	16.0	4.0	30.3
O-Mem	62.0	2.0	26.7	76.0	85.0	**18.2**	69.0	56.0	11.0	45.1
MemDream	**67.0**	**56.0**	**28.7**	**81.6**	**91.0**	17.4	71.8	**61.0**	**13.3**	**54.2**

Table 4: Performance on MAB (GPT-4.1 reader). Metrics follow the MAB protocol. Best in bold, second-best underlined. We compare only memory agent methods under identical settings.

Method	AR				TTL	LRU		SF		Overall
	SH-QA	MH-QA	LME	EventQA	MCC	Summ.	Det QA	FC-SH	FC-MH	
FluxMem	19.0	26.0	6.7	33.3	80.0	15.5	16.7	28.0	5.0	25.6
HippoRAG 2	10.0	21.0	6.0	31.8	40.0	14.4	53.5	22.0	**7.0**	22.9
MemoryOS	19.0	8.0	4.3	31.0	13.0	8.2	47.9	2.0	1.0	14.9
MEMOS	3.0	24.0	5.0	32.0	45.0	25.5	**59.2**	24.0	1.0	24.3
MemR^3	10.0	22.0	5.7	38.0	43.0	**27.1**	40.8	17.0	3.0	23.0
O-Mem	20.6	22.0	13.3	27.3	36.0	11.4	33.3	**45.0**	6.0	23.9
MemDream	**43.0**	**35.0**	**20.0**	**39.0**	**84.0**	26.3	56.0	29.0	2.0	**37.1**

Table 5: MAB results (Llama-3.1-8B; metrics as in Table 4). Best in bold; second-best underlined.

Variant	Cat1: Multi-hop			Cat2: Temporal			Cat3: Open			Cat4: Single-hop			**Average**		
	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L	F1	B-1	R-L
MemDream (full)	**53.64**	**43.88**	**47.50**	**66.05**	**49.07**	**63.43**	**35.89**	**30.69**	**35.43**	**64.97**	**57.66**	**65.12**	**55.14**	**45.33**	**52.87**
w/o Dreamer	49.81	40.22	44.17	62.43	46.51	59.87	34.72	28.93	33.61	61.38	54.12	61.29	52.09	42.45	49.74
w/o Analyst	51.47	42.15	45.82	63.89	47.64	61.05	35.11	29.48	34.26	63.25	55.71	63.48	53.43	43.75	51.15
w/o Consolidator GRPO	50.93	41.67	45.21	63.18	47.12	60.54	34.95	29.17	33.89	62.84	55.28	62.91	52.98	43.31	50.64
w/o Decay	39.44	27.07	35.82	52.96	38.80	51.22	17.35	12.51	17.89	54.41	44.67	54.70	41.04	30.76	39.91

Table 6: Ablation study. Best results in each column are shown in bold.

4.2 Overall Performance

Tables 2, 3, 4, and 5 report results on LoCoMo and MAB.

LoCoMo. With both GPT-4.1 and Llama readers (Tables 2 and 3), MemDream reaches the highest average F1, B-1, and R-L. With GPT-4.1, it gains 3.98 average F1 over FluxMem and 8.57 temporal F1 over O-Mem, the strongest baselines for the respective metrics. It trails FluxMem on open-ended questions by 1.41 F1; these favor broad context over targeted probes.

MAB. Across both readers (Tables 4 and 5), MemDream reaches the highest overall score, 54.2 with GPT-4.1 and 37.1 with Llama, and leads on 7 of 9 tasks with GPT-4.1. It improves on Accurate Retrieval (SH-QA 67.0, MH-QA 56.0, LME 28.7, EventQA 81.6) and Test-Time Learning (91.0), where the dream cycle links memories across sessions. Two tasks favor other systems: MemoryOS leads Detective QA (73.2 vs 71.8), where hierarchical indexing fits long narratives, and O-Mem leads Summarization (18.2 vs 17.4), where the dream cycle trades verbatim coverage for factual precision and loses ROUGE overlap. With the Llama reader, MemDream leads the second-best (FluxMem 25.6) by 11.5, with the largest gains on Accurate Retrieval and TTL.

4.3 Ablation Study

We test four dream-cycle ablations on LoCoMo (GPT-4.1 reader), removing or degrading one component at a time: (1) **w/o Dreamer**: no probing queries; the system keeps the initial memory state without dream-guided maintenance. (2) **w/o Analyst**: the Consolidator gets no targeted diagnosis and extracts uniformly from all nodes above a length threshold. (3) **w/o Consolidator GRPO**: the Consolidator keeps its SFT-initialized weights, without GRPO fine-tuning. (4) **w/o Decay** (γ=0): prune permanently removes nodes instead of applying soft decay. Table 6 reports the results.

Results. The largest effect comes from augment_decay. Removing it (w/o Decay) drops average F1 from 55.14 to 41.04 (-14.10), and by 18.54 F1 on open-ended questions. Hard pruning discards nodes that later sessions need, so reversible forgetting is what keeps long-horizon memory intact.

Among the three agents, the Dreamer matters most: removing it drops F1 by 3.05, leaving the consolidation cycle with no self-generated signal to target. Removing the Analyst drops F1 by 1.71, as the Consolidator then extracts uniformly, without diagnosis. Removing GRPO drops F1 by 2.16; the SFT-only Consolidator still repairs, but without a learned policy for choosing operations. The full system leads on every category and metric, so the three agents provide complementary value.

Offline computational cost. The approximate timing profile in Appendix K assigns 2.1, 1.5, and 3.2 minutes per round to the Dreamer, Analyst, and Consolidator, respectively, with another 1.4 minutes for retrieval and verification. A round takes about 9 minutes including model and adapter loading; a full cycle is reported at about 22 minutes for 2–3 rounds. These are offline maintenance costs, separate from online query answering.

4.4 Sensitivity Analysis

We examine four key hyperparameters (Figure 3).

Reward weighting (panel a). We compare our per-agent reward weights against two variants that keep the same reward terms but reweight them: *equal*, which splits each agent's weight uniformly across its terms, and *single-term*, which places all weight on each agent's primary term. Our weighting gives the best F1 for all three agents, indicating that the per-agent terms, though set a priori rather than tuned on the benchmarks, are a consistent choice.

Retrieval depth k (panel b). Increasing retrieval depth k improves F1, with linear latency growth and diminishing gains per second. We use k=5 to balance accuracy and inference cost.

Augment decay γ (panel c). Across all tested γ values, the dream cycle consistently improves accuracy over the no-dream baseline. The optimal decay factor is γ=0.2. A larger γ (approaching 1) reduces the weight only slightly, so superseded nodes retain too much low-utility content and dilute retrieval precision; a smaller γ (approaching 0) decays nodes too aggressively, approaching the failure mode observed in the w/o Decay ablation (γ=0).

Dream quality (panel d). Rule-based probes have higher discriminative power (0.36 vs. 0.15) and alignment (0.37 vs. 0.35), with similar query-type coverage. We retain the LLM Dreamer as a trainable query generator and use rule-based probes as a diagnostic reference.

5 Conclusion

We presented MemDream, a framework that uses prospective self-probing to identify and repair memory weaknesses before real interactions. Its Dreamer, Analyst, and Consolidator learn through GRPO, while soft decay makes forgetting reversible. Across reader models on LoCoMo and MemoryAgentBench, MemDream improves over memory baselines, with the largest gains in temporal reasoning and cross-session integration, supporting proactive maintenance of agent memory.

AI USE STATEMENT

Generative AI tools assisted with language polishing. The authors take responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies memory maintenance using existing dialogue datasets and memory benchmarks. Applications involving personal conversations should provide user control over memory retention, access, and deletion.

REPRODUCIBILITY STATEMENT

Sections 3.1–3.3 specify the memory architecture, dream cycle, reward functions, and training procedure. Section 4.1 describes the models, retrieval components, and evaluation settings. Appendices A–D provide dataset configurations, baseline descriptions, agent prompts, and training hyperparameters. Appendices F and K describe the statistical analysis and computational costs.

REFERENCES

Anthropic. Dreaming: Asynchronous memory consolidation for Claude agents. Product documentation, `https://docs.anthropic.com/en/docs/agents-and-tools/memory/dreaming`, 2026. Announced May 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. FLEX: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

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

Zijie Dai, Shiyuan Deng, Sheng Guan, Yizhou Tian, Xin Yao, Xiao Yan, and James Cheng. RecMem: Recurrence-based memory consolidation for efficient and effective long-running LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32353–32376. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1619. URL `https://aclanthology.org/2026.findings-acl.1619/`.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Soham Dan, et al. Larimar: Large language models with episodic memory control. *arXiv preprint arXiv:2403.11901*, 2024.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. MemR^3: Memory retrieval via reflective reasoning for LLM agents. *arXiv preprint arXiv:2512.20237*, 2025.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Yunfan Gao et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Yingjie Gu, Wenjian Xiong, Liqiang Wang, Pengcheng Ren, Chao Li, Xiaojing Zhang, Yijuan Guo, Qi Sun, Jingyao Ma, and Shidang Shi. FSFM: A biologically-inspired framework for selective forgetting of agent memory. *arXiv preprint arXiv:2604.20300*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bernal Jiménez Gutiérrez, Yu Shu, Weihong Qi, Sanping Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions, 2026. URL https://arxiv.org/abs/2507.05257.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. *arXiv preprint arXiv:2601.03236*, 2026.

Xisen Jin et al. Machine unlearning of pre-trained large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025. arXiv:2506.06326.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2017.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534, 2016.

LangChain. LangMem: Long-term memory for agents. https://blog.langchain.com/langmem-sdk-launch/, 2025. Accessed: 2026-07-29.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Shuang Li, Yilun Du, Jan-Willem van de Meent, and Igor Mordatch. Energy-based models for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Evolve-Mem: Self-evolving memory architecture via AutoResearch for LLM agents. *arXiv preprint arXiv:2605.13941*, 2026.

Mingfei Lu, Mengjia Wu, Feng Liu, Jiawei Xu, Weikai Li, Haoyang Wang, Zhengdong Hu, Ying Ding, Yizhou Sun, Jie Lu, and Yi Zhang. Choosing how to remember: Adaptive memory structures for LLM agents. *arXiv preprint*, 2026.

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

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024.

Katja Schlegel, Nils R Sommer, and Marcello Mortillaro. Large language models are proficient in solving and creating emotional intelligence tests. *Communications Psychology*, 3(1):80, 2025.

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

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Lingzhi Wang et al. Selective Forgetting: Advancing machine unlearning techniques and benchmarks in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2025a.

Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, et al. O-mem: Omni memory system for personalized, long horizon, self-evolving agents. *arXiv e-prints*, pp. arXiv–2511, 2025b.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025c.

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

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

Chengyuan Yang, Zequn Sun, Wei Wei, and Wei Hu. Beyond static summarization: Proactive memory extraction for llm agents. *arXiv preprint arXiv:2601.04463*, 2026a.

Tianyu Yang, Sudipta Paul, Vijay Srinivasan, Vivek Kulkarni, and Srinivas Chappidi. TRUSTMEM: Learning trustworthy memory consolidation for LLM agents with long-term memory. *arXiv preprint arXiv:2606.25161*, 2026b. URL https://arxiv.org/abs/2606.25161.

Chongrui Ye, Yuxiang Liu, Yu Wang, Haofei Yu, Yining Zhao, Ge Liu, Julian McAuley, and Jiaxuan You. Auto-dreamer: Learning offline memory consolidation for language agents. *arXiv preprint arXiv:2605.20616*, 2026. URL https://arxiv.org/abs/2605.20616.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. MemSearcher: Training LLMs to reason, search and manage memory via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.02805*, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. AgentEvolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025b.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

APPENDIX CONTENTS

A Dataset Details 15

B Baseline Descriptions 15

C Dream Cycle Prompts 16

D Training Details 17

E Additional Experimental Results 18

E.1 Dream Cycle Rounds Comparison 18

E.2 Dream Quality Detailed Analysis 18

E.3 Training Signal Comparison 18

F Statistical Significance of Results 19

G Multi-Round Consolidation Stability 19

H Dream Query Groundedness 20

I Case Study 21

J Repair Quality Audit 21

K Computational Cost Analysis 22

L Comparison with Claude Dreaming 22

M Comparison with Recent Memory Consolidation Methods 23

A Dataset Details

LoCoMo. LoCoMo (Maharana et al., 2024) comprises 10 long-horizon multi-session conversations between a user and an agent, with an average of 300 turns and 9,000 tokens per conversation. Each conversation is accompanied by an average of about 199 human-written questions. Following the standard protocol, we evaluate on four reasoning categories, excluding the adversarial category: **Cat1: Multi-hop** (requiring integration of facts from multiple sessions), **Cat2: Temporal** (requiring reasoning over time-ordered events), **Cat3: Open-ended** (requiring broad contextual understanding), and **Cat4: Single-hop** (requiring retrieval of a single specific fact). In total, the released benchmark contains 1,986 QA pairs, of which the 1,540 questions in these four categories constitute our evaluation set. We use full conversations as memory input, with 4096-token chunks.

MemoryAgentBench (MAB). MAB evaluates four memory competencies across 9 sub-tasks:

- **Accurate Retrieval (AR)**: RULER QA1 (single-hop, 197K context), RULER QA2 (multi-hop, 197K), LongMemEval (cross-session), and EventQA (temporal).
- **Test-Time Learning (TTL)**: Multi-class Classification (MCC) with in-context labels.
- **Long-Range Understanding (LRU)**: InfBench Summarization and Detective QA.
- **Selective Forgetting (SF)**: FactConsolidation single-hop (FC-SH) and multi-hop (FC-MH), testing whether the agent correctly updates contradictory facts.

Each task provides interaction pages that the memory system must process and store, followed by evaluation queries. We use chunk size 512 for AR and SF tasks (following the original protocol) and chunk size 4096 for TTL and LRU tasks.

B Baseline Descriptions

We compare against 10 representative memory systems:

- **LangMem** (LangChain, 2025): Extracts key-value memories from conversations and retrieves via embedding similarity. Uses a flat memory store without structure.
- **Mem0** (Chhikara et al., 2025): Maintains a user memory layer with incremental extraction and deduplication. Retrieves via hybrid search (embedding + keyword).
- **ZEP** (Rasmussen et al., 2025): Builds a temporally-aware knowledge graph from conversations with entity extraction and edge timestamping. Retrieves via graph traversal.
- **MemoryOS** (Kang et al., 2025): Adopts an operating-system metaphor with storage tiers and eviction policies. Uses session-level metadata for retrieval routing.
- **A-Mem** (Xu et al., 2025): Maintains an evolving network of notes via reflection and linking. Each note is periodically updated through self-reflection.
- **O-Mem** (Wang et al., 2025b): Organizes memory as a structured graph with explicit temporal ordering and fact-update mechanisms. Specialized for contradictory information.
- **MEMOS** (Li et al., 2025): Multi-granularity memory with both fine-grained facts and coarse-grained summaries, managed by multiple specialized agents.
- **MemR^3** (Du et al., 2025): A closed-loop retrieval controller that alternates retrieve, reflect, and answer actions, using an evidence-gap tracker to decide when to stop.
- **HippoRAG 2** (Gutiérrez et al., 2025): Hippocampus-inspired memory indexing with pattern separation and pattern completion. Uses an entity-indexed knowledge graph.
- **FluxMem** (Lu et al., 2026): Adaptive memory with a learned selector for linear, graph, or hierarchical structures based on conversation features, plus BMM-based fusion.

All baselines use identical chunking, the same reader LLM (GPT-4.1 or Llama-3.1-8B depending on the experiment), and temperature 0 for reproducibility. Memory uses identical input pages.

C DREAM CYCLE PROMPTS

We provide the core prompts used for each agent in the dream cycle.

```
You are a memory probing agent. Given the current memory graph state,
generate queries that test whether the memory
can support plausible future information needs.

Focus on:
1. Regions with low connectivity
(isolated nodes that should be linked)
2. High redundancy areas (similar nodes that could cause confusion)
3. Temporal gaps (sessions with no recent
access that may contain stale info)
4. Cross-session dependencies (facts from
different sessions that must be integrated)

For each weak region, generate a natural question whose correct
answer requires information from that region. The question should be
realistic (something a user might plausibly ask in a future session).

Output format:
- query: <the probing question>
- expected_answer: <what the correct answer should contain>
- target_nodes: <node IDs being tested>
- weakness_type: <buried|scattered|redundant|noisy>
```

```
You are a memory diagnosis agent. Given a probing query, retrieved
results, and the expected answer, diagnose
why retrieval failed or partially succeeded.

Diagnosis categories:
- BURIED: important facts hidden in verbose conversational
text, not extractable by current retrieval
- SCATTERED: related
information split across multiple disconnected nodes
- REDUNDANT: overlapping content in multiple nodes causing dilution
- NOISY: low-value nodes (greetings, filler) ranked too high

For each diagnosis, prescribe ONE operation:
- extract: create structured node from verbose page
- merge: combine similar/related nodes
- synthesize: generate summary node with provenance links
- prune: apply soft decay to low-value node

Output format:
- diagnosis: <buried|scattered|redundant|noisy>
- confidence: <0.0-1.0>
- affected_nodes: <list of node IDs>
- prescribed_operation: <extract|merge|synthesize|prune>
- rationale: <brief explanation>
```

```
You are a memory consolidation agent. Given a diagnosis and
prescribed operation, execute the repair on the memory graph.

Operations:
1. EXTRACT: From a verbose memory page, extract structured key facts
as a new node. Preserve entity mentions and temporal markers.
2. MERGE: Given two or more related nodes, produce a single unified
```

```
node that preserves all information. Maintain edge connectivity.
3. SYNTHESIZE: Given a cluster of nodes, generate a summary node.
Add provenance edges to source nodes.
4. PRUNE (decay): Reduce node weight by factor
gamma=0.2. Do NOT delete the node.

Constraints:
- Preserve all factual
content (no information loss during merge/synthesize)
- Maintain temporal ordering of events
- Update edge connections after operations
- Log all changes for reversibility

Output the modified memory state in JSON format.
```

D TRAINING DETAILS

LoRA Configuration. All three agents use PEFT LoRA adapters on Llama-3.1-8B-Instruct with 8-bit quantization (bitsandbytes). Configuration details are in Table 7.

Parameter	Value
Base model	Llama-3.1-8B-Instruct
Quantization	8-bit (bitsandbytes)
LoRA rank r	16
LoRA alpha α	32
Target modules	q_proj, k_proj, v_proj, o_proj
LoRA dropout	0.05
Trainable params (per agent)	$\sim$13.1M
Total trainable params	$\sim$39.3M

Table 7: LoRA adapter configuration.

SFT Pre-training. Each agent is first trained via supervised fine-tuning on teacher-generated data. The Dreamer is trained on (memory state, effective probing query) pairs generated by GPT-4.1. The Analyst is trained on (query, retrieval result, ground-truth diagnosis) triples. The Consolidator is trained on (diagnosis, correct operation output) pairs. SFT uses learning rate 2×10^{-4}, batch size 4, and 3 epochs with cosine scheduling.

GRPO Training. After SFT, each agent undergoes GRPO fine-tuning with the settings in Table 8.

Parameter	Value
Group size N	4
Learning rate	1×10^{-3}
Exploration rate ϵ	0.2 (decaying)
Training data	MSC dialogue (5 sessions)
Dream queries per conversation	8–12
Training steps (independent)	500 per agent
Training steps (joint)	200
Gradient accumulation	4
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Weight decay	0.01
Max gradient norm	1.0

Table 8: GRPO training hyperparameters.

Joint Training. After independent GRPO, the Dreamer and Consolidator are jointly fine-tuned while the Analyst remains frozen. The joint reward is ΔF1: the change in average retrieval F1 on dream queries after applying the Consolidator's operations. The Dreamer is rewarded for generating queries that expose fixable weaknesses, and the Consolidator for producing repairs that durably improve retrieval. Joint training uses a shared optimizer with alternating updates (Dreamer for 1 step, Consolidator for 1 step) via `set_adapter()`.

Hardware. Training is conducted on a single NVIDIA A100 (80GB). SFT takes $\sim$2 hours per agent. Independent GRPO takes $\sim$4 hours per agent. Joint GRPO takes $\sim$3 hours. Total training time is $\sim$21 hours. One consolidation round on $\sim$200 nodes takes $\sim$9 minutes (Appendix K).

E Additional Experimental Results

E.1 Dream Cycle Rounds Comparison

We vary the number of dream cycle rounds (0, 1, 3, 5, 8) on a subset of 3 LoCoMo conversations to study convergence behavior. Table 9 reports macro-averaged F1, B-1, and R-L.

Rounds	F1	B-1	R-L
0 (no dream)	48.72	38.15	46.83
1	51.36	40.62	49.58
3	53.41	42.87	51.92
5	**53.89**	**43.21**	**52.35**
8	53.52	42.94	51.87

Table 9: Effect of dream cycle rounds on LoCoMo.

On this subset, performance improves from 0 to 5 rounds (+5.17 F1), then declines at 8 rounds due to over-consolidation (excessive merging reduces node diversity). This supports the early-stopping criterion (Eq. 7): in practice, convergence is typically reached within 2–3 rounds on the full 10-conversation evaluation, as gains per round rapidly diminish after critical weaknesses are repaired.

E.2 Dream Quality Detailed Analysis

Table 10 compares rule-based and LLM Dreamer query generation across all 10 LoCoMo conversations on three metrics: discriminative power (fraction of dreams that expose retrieval failures), type coverage (number of distinct query types generated), and downstream alignment.

Dreamer Variant	Disc.	Cov.	Align.
Rule-Based	**0.36**	**5.0**	**0.37**
LLM Dreamer	0.15	4.8	0.35

Table 10: Dream quality comparison.

Rule-based probes achieve higher discriminative power (0.36 vs. 0.15) and slightly higher downstream alignment (0.37 vs. 0.35), while query-type coverage is similar (5.0 vs. 4.8). We use the LLM Dreamer as the trainable query generator, with rule-based probes as a complementary diagnostic reference.

E.3 Training Signal Comparison

Table 11 compares GRPO training signals by average reward on held-out conversations.

The oracle signal (training on actual future queries) provides an upper bound (0.262). Dream-guided training with top-30% query selection achieves the highest non-oracle reward (0.241), outperforming the no-dream baseline (0.198) by 0.043. This confirms that the Dreamer generates queries that provide a stronger and more aligned training signal than passive accumulation alone. Top-30% outperforms top-50% (0.232), showing that selective dream-query filtering improves signal quality.

Signal Source	Mean Reward	Std
Baseline (no dream)	0.198	0.031
Dream-guided (top-50%)	0.232	0.035
Dream-guided (top-30%)	0.241	0.033
Oracle (real queries)	0.262	0.038

Table 11: Training signal comparison .

Method (reader)	n	Convs	Macro F1	95% CI (conv.)	Micro F1
MemDream (GPT-4.1)	1377	9	54.92	[51.06, 58.34]	61.31
MemDream (Llama-8B)	1529	10	39.55	[36.44, 42.32]	47.55
MemDream (Haiku-4.5)	1529	10	42.06	[38.50, 45.34]	47.21
Mem0 (GPT-4.1)	1529	10	27.35	[21.88, 32.88]	29.67
MemR3 (GPT-4o)	1529	10	19.99	[19.04, 20.87]	22.71
Mem0 (Llama-8B)	1529	10	11.57	[10.18, 12.77]	14.52
MemR3 (Llama-8B)	1529	10	8.83	[8.38, 9.33]	10.24
O-Mem (Llama-8B)	1371	9	37.80	[29.33, 45.51]	42.44

Table 12: Average F1 with conservative conversation-level 95% bootstrap confidence intervals (10,000 replicates), computed from each method's per-question score files.

F Statistical Significance of Results

To quantify the reliability of the LoCoMo results, we compute bootstrap confidence intervals and paired significance tests over per-question token F1 scores (B=10,000 replicates). Because questions within one conversation share a memory state and are not independent, we use a conservative conversation-level (cluster) bootstrap that resamples whole conversations. Averages are recomputed from each method's per-question score files under one consistent aggregation protocol; for the GPT-4.1 reader these cover 9 of 10 conversations, whose outputs were available at analysis time.

Table 13 reports paired comparisons against the baselines for which per-question outputs are available. Pairing on (conversation, question, gold answer) cancels per-question difficulty and yields tight intervals. Against Mem0 and MemR3 under both readers the improvement is large, significant at the conservative conversation level ($p < 0.001$), and positive in every conversation. Against O-Mem the question-level micro-F1 gain ($+5.18$, $p < 0.001$) is driven by two conversations, so we do not claim conversation-level significance there. For baselines without local per-question outputs (e.g. FluxMem), Table 2 compares point estimates; paired tests are deferred to the code release.

G Multi-Round Consolidation Stability

Because the memory graph is compact ($\sim$300 nodes), a concern is whether repeated dream cycles *accumulate* hallucination, *over-merge* nodes, or *lose* rare facts. We stress-test this by running K consecutive dream cycles ($K \in \{1, 2, 3, 5, 8, 12, 16, 20\}$) on each LoCoMo conversation with the GPT-4.1 reader, tracking graph size, hallucination rate on synthesized nodes, entity fidelity (fraction of named atoms grounded in source), rare-fact retention, and QA F1. Figure 4 averages 9 dialogues.

Even pushed to 20 cycles (roughly $10\times$ the 2–3 cycles used in practice), the graph neither collapses nor explodes (node count $308 \to 331$, $+7\%$), the hallucination rate does not accumulate ($0.266 \to 0.247$), and entity fidelity is flat to slightly rising ($0.781 \to 0.824$), confirming that consolidation invents no new named entities over time. Rare-fact retention and QA F1 decline by roughly 12–13% over 20 cycles, with no catastrophic drop. In the operating regime (2–3 cycles, where the cycle converges) all metrics are stable (e.g. K=3: rare retention 0.960, QA F1 0.517). Reversible decay and retrieval-success rewards thus stabilize the cycle rather than progressively degrade it.

Comparison	n	ΔMacro	95% CI (conv.)	p_{cluster}	Convs Δ>0
MemDream vs Mem0 (GPT-4.1)	1377	+26.18	[21.23, 31.59]	< 0.001	9/9
MemDream vs MemR3 (GPT-4o)	1377	+34.96	[31.91, 37.58]	< 0.001	9/9
MemDream vs Mem0 (Llama-8B)	1529	+27.97	[24.85, 31.40]	< 0.001	10/10
MemDream vs MemR3 (Llama-8B)	1529	+30.72	[27.93, 33.12]	< 0.001	10/10
MemDream vs O-Mem (Llama-8B)	1371	+1.84	[-4.90, 10.16]	0.71	3/9

Table 13: Paired comparisons (MemDream − baseline). "Convs Δ>0": count of conversation wins.

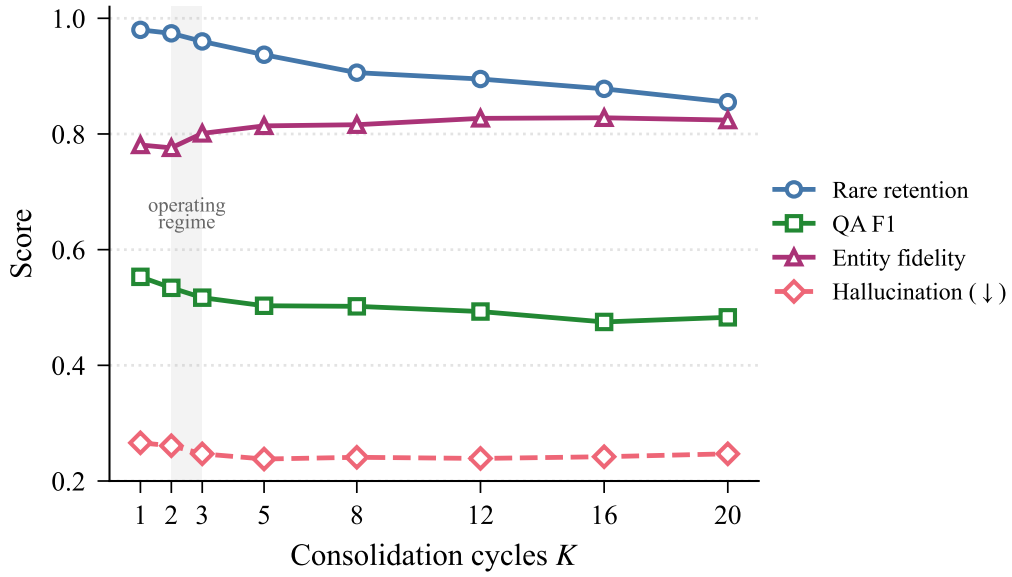

Figure 4: Stability over K consecutive consolidation cycles (mean over 9 LoCoMo conversations, GPT-4.1 reader). Rare retention, entity fidelity, and QA F1 are higher-is-better; hallucination is lower-is-better. The shaded band marks the operating regime (2–3 cycles). Omitted for readability: node count grows only 308 → 331 (+7%) and retrieval F1 stays flat (0.147 → 0.139) throughout.

H Dream Query Groundedness

A concern with self-generated probes is that the Dreamer might optimize memory toward *hallucinated* answers, or that its probes might be paraphrases of the evaluation questions (peeking at the test set). We address both directly.

Answers are grounded by construction. The Dreamer's training objective concerns only the probing *question*; each probe's expected answer is copied verbatim from the stored source memory rather than generated. Running the actual dream generator over all 10 LoCoMo conversations, every expected answer is an exact substring of the source memory: 110/110 (100%) under the default setting (11 probes per conversation), and 4150/4150 (100%) when sampling is scaled to 5 seeds and ∼415 probes per conversation, with zero exceptions. The grounding the reviewer asks for therefore holds by construction, not by chance.

Probes are not evaluation questions. We also measured lexical overlap between each dream probe and its nearest LoCoMo evaluation question in the same conversation (exact match, token Jaccard, ROUGE-L, TF-IDF cosine). There are zero exact matches, and the fraction of probes whose ROUGE-L against any same-conversation evaluation question exceeds 0.8 is $0/50$ for both the LLM and rule-based Dreamer (maximum ROUGE-L 0.571). Thus, convergence is not measured on paraphrased test questions. We analyze each conversation's first five stored probes.

I CASE STUDY

We illustrate one complete dream cycle iteration using the nut allergy scenario from Figure 1.

Memory State (before dream cycle). The memory graph contains 52 nodes accumulated over 9 sessions. Two relevant nodes are disconnected:

- **Node 7** (Session 1): "User mentioned having a severe nut allergy, carries an EpiPen, and avoids tree nut products."
- **Node 34** (Session 5): "User loves baking, brownies and cookies. Prefers recipes with chocolate and caramel."

No edge connects these nodes. The allergy information is buried in a conversational turn about general health, while the baking preference is stored as a standalone hobby node.

Dreamer Output. The Dreamer computes weakness scores (Eq. 3) and identifies this region as high-risk (Node 7 has low connectivity, $\phi(v)$=0.81). It generates:

Query: "What should I keep in mind when recommending baking recipes for this user?"
Expected answer: Should mention nut allergy and avoid recipes with almond flour, peanut butter, or walnut toppings.
Target nodes: [7, 34]
Weakness type: scattered

Retrieval Result. The retrieval pipeline returns Node 34 (rank 1, baking preferences) but not Node 7 (rank 11, below the top-5 cutoff). The allergy constraint is missed entirely.

Analyst Diagnosis.

Diagnosis: SCATTERED
Confidence: 0.95
Affected nodes: [7, 34]
Prescribed operation: MERGE
Rationale: Safety-critical constraint (allergy, Node 7) and activity preference (baking, Node 34) are causally linked but stored in disconnected sessions with no shared entities.

Consolidator Action. The Consolidator merges Nodes 7 and 34 into a new Node 53:

Node 53: "User loves baking (brownies, cookies, chocolate and caramel flavors). CRITICAL: User has a severe nut allergy (carries EpiPen). All recipe recommendations must avoid tree nuts, almond flour, peanut butter, and walnut-based ingredients."
Edges: 53 $\leftrightarrow$ 7 (provenance), 53 $\leftrightarrow$ 34 (provenance)
Weight: $w_{53} = 1.0$

Verification. After repair, re-querying with the same probe returns Node 53 at rank 1. The retrieval F1 for this query improves from 0.28 to 0.94. When Session 9 arrives and the user asks "Recommend me a quick snack recipe," the system retrieves the merged node and avoids nut-containing recipes.

J REPAIR QUALITY AUDIT

To move beyond a single case study, we audit every repair action the Consolidator takes across all 10 LoCoMo conversations. Of 837 repair actions (176 augment, 301 synthesize, 301 prune, 59 merge), none of the 59 merges introduces an unsupported number or date (0/59 fabricated merges). Classifying each action by its effect on answerable content, 5 actions add correct information (information gain) and 9 ($\sim$1%) cause information loss; the remainder are content-preserving. Of the nodes the Consolidator synthesizes, 82.1% are used in at least one correct answer, and 90.7% of correct answers that touch a synthesized node rely on it, indicating their utility.

K Computational Cost Analysis

Table 14 gives the approximate timing profile for the 8-bit configuration described in Appendix D. Loading is itemized separately from the five processing stages. The listed stage times sum to 8.2 minutes per round; including the 53-second startup gives approximately 9 minutes. The full-cycle estimate corresponds to 2–3 rounds, rather than a single round.

Component	Time	GPU Mem.
Load base model (8-bit)	45s	9.2 GB
Load 3 LoRA adapters	8s	+1.8 GB
Dreamer: generate queries	2.1 min	12.4 GB
Retrieval (BM25 + dense)	0.8 min	3.2 GB
Analyst: diagnose	1.5 min	12.4 GB
Consolidator: execute	3.2 min	12.4 GB
Verification (re-retrieval)	0.6 min	3.2 GB
Total (1 round)	**∼9 min**	**17.1 GB**
Typical convergence	2–3 rounds	–
Full cycle (avg.)	∼22 min	17.1 GB

Table 14: Approximate offline timing and GPU memory profile. Per-stage times describe one round; the final row reports the full-cycle estimate.

Online and offline costs. Dream cycles run asynchronously alongside online interactions. Query answering uses the current conversational context and existing memory, without waiting for background consolidation to finish. Recent interactions remain accessible in context while long-term memory is consolidated. Training is performed before deployment (Appendix D); Table 14 describes background maintenance costs. These costs are accounted for separately from online query latency, together with the frequency of dream cycles over the interaction stream.

When to trigger dream cycles. In deployment, dream cycles are triggered when: (1) more than 20 pages have been added since the last cycle, (2) retrieval confidence drops below a threshold, or (3) a time interval has elapsed. The triggering policy is configurable and not part of the trained system.

Recorded calls and tokens. Table 15 reports a separate profiling run over all 10 LoCoMo conversations with local Llama-3.1-8B. This run supplied 11 rule-generated probes per conversation, bypassing LLM Dreamer generation, and used 4–8 fixed partition rounds (mean 6.7). Counters started after model loading, initial memory construction, and probe generation. The initial stores contained 3,062 nodes in total (306.2 per conversation). Input and output tokens were summed across shared-model calls; wall time also includes non-generation work within the cycle.

Quantity	Mean per conversation	Per initial node
LLM calls	60.9	0.199
Input tokens	33,588.4	109.69
Output tokens	13,004.5	42.47
Total tokens	46,592.9	152.16
Model generation time (s)	486.41	1.59
Cycle wall time (s)	2,208.84	7.21

Table 15: Recorded overhead for the rule-probe configuration. Per-node values divide totals by 3,062 initial nodes. Model loading, initial memory construction, and LLM Dreamer generation are excluded; these measurements use a different configuration from Table 14.

L Comparison with Claude Dreaming

Anthropic recently introduced a "Dreaming" feature for Claude-based agents (Anthropic, 2026), which performs offline memory consolidation between user sessions. While the high-level motivation

(maintaining memory quality through background consolidation) overlaps with MEMDREAM, the two systems differ in signal source, operation logic, and optimization.

Signal source. Claude Dreaming reviews *already-observed* session data: it scans past interactions for repeated patterns (e.g., recurring API errors, style preferences), detects duplicates, and identifies contradictions. The signal is entirely backward-looking. MEMDREAM instead generates its own signal via the Dreamer agent, which synthesizes *hypothetical future queries* targeting weak memory regions. This enables discovery of latent weaknesses (e.g., disconnected nodes across sessions) that would remain invisible to any system that only reviews past data.

Operation logic. Claude Dreaming applies rule-based operations: deduplication (merge identical entries), conflict resolution (keep the more recent fact), and pattern extraction (distill repeated observations into a summary). These are deterministic transformations driven by surface-level similarity and recency. MEMDREAM's Consolidator selects among four operations (extract, merge, synthesize, prune) based on the Analyst's diagnosis, and the selection policy is trained via GRPO to maximize retrieval success on dream queries. This enables context-dependent decisions: the same nodes may be merged or kept separate depending on predicted future queries.

Self-testing. Claude Dreaming does not test whether its consolidation improves retrieval for future queries. It performs cleanup and moves on. MEMDREAM explicitly verifies each repair: after the Consolidator acts, the Dreamer's queries are re-executed against the updated memory, and the cycle continues until retrieval F1 converges (Eq. 7). This closed-loop verification prevents harmful consolidation (e.g., merging nodes that should remain separate).

Reversibility. Claude Dreaming's deletions and merges are permanent (no documented recovery mechanism). MEMDREAM's augment_decay provides soft forgetting: nodes are demoted rather than deleted, and can be restored if subsequent cycles reveal renewed relevance.

Summary. We characterize Claude Dreaming as *offline reactive consolidation*: proactive in timing but reactive in signal and mechanical in operation. MEMDREAM is *anticipatory self-probing*: it generates its own training signal, learns which operations to apply, verifies their effect, and preserves reversibility. They span rule-based maintenance and learned, self-supervised memory evolution.

M COMPARISON WITH RECENT MEMORY CONSOLIDATION METHODS

Auto-Dreamer, TrustMem, and RecMem address complementary aspects of memory consolidation: downstream utility, update trustworthiness, and construction efficiency. Table 16 contrasts their mechanisms with MemDream's explicit retrieval-testing loop.

Method	Signal or criterion	Memory update
Auto-Dreamer	Task performance and counterfactual memory utility during training	Rewrite a selected region into a compact replacement set
TrustMem	Transition-level coverage, preservation, and faithfulness	Learn trustworthy updates from preferences over candidate transitions
RecMem	Recurrence of semantically similar interactions	Trigger consolidation and refine fine-grained semantic facts
MemDream	Failures exposed by self-generated retrieval probes	Diagnose failures, apply targeted repairs, and re-evaluate retrieval

Table 16: Signals and update mechanisms in recent memory consolidation methods.

Auto-Dreamer: learned region rewriting. Auto-Dreamer (Ye et al., 2026) separates online acquisition from offline consolidation. Its consolidator inspects a selected memory region and provenance-linked trajectories, then synthesizes a replacement set. GRPO rewards downstream task performance and counterfactual utility estimated by masking memories. At deployment, working

regions include recently written and recently retrieved entries. MemDream instead selects repair targets through an explicit loop of query generation, retrieval, and failure diagnosis. Both methods can update memory before subsequent tasks; MemDream's distinction lies in constructing localized retrieval failure signals. Auto-Dreamer preserves historical evidence by retaining source trajectories.

TrustMem: trustworthy memory transitions. TrustMem (Yang et al., 2026b) evaluates memory transitions for coverage of incoming information, preservation of existing information, and faithfulness to evidence. Preferences over candidate updates guide reinforcement learning. MemDream tests whether stored evidence is accessible under generated questions and uses the resulting failures to select repairs. Transition verification and retrieval probing address complementary properties: an update may faithfully preserve a fact while leaving it difficult to retrieve. MemDream's soft decay preserves demoted nodes; newly synthesized content still requires factual verification.

RecMem: recurrence-triggered consolidation. RecMem (Dai et al., 2026) stores incoming interactions in a subconscious layer and uses lightweight embeddings for retrieval. Sustained semantic recurrence triggers LLM-based extraction of episodic and semantic memory; semantic refinement recovers fine-grained facts omitted during extraction. Its recurrence criterion allocates consolidation effort to repeated content. MemDream allocates repair effort according to failures observed on generated retrieval probes, including queries that require evidence from separate nodes. Its design therefore targets retrieval accessibility through a different signal from recurrence frequency.

Evaluation settings. Auto-Dreamer evaluates task experience consolidation on ScienceWorld, ALFWorld, and WebArena. TrustMem includes MemoryAgentBench, HaluMem, and the Mem-α validation set; RecMem includes LoCoMo. MemDream evaluates retrieval-oriented long-term memory on LoCoMo and MemoryAgentBench. The overlap with TrustMem and RecMem makes reader models, memory budgets, and evaluation protocols relevant to empirical comparisons. This comparison addresses mechanisms, without ranking the three methods under matched settings.