REWRITING IS FORGETTING: ZERO REWRITE CAPTURE AND BUDGETED READING FOR LONG HORIZON AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory is a core component of large language model agents, enabling them to store and retrieve information across sessions. We systematically study the common practice of rewriting conversations before storage in memory systems. We build a controlled framework with three history-length regimes (no truncation, truncation, and long horizon) and evaluate this practice under a unified protocol. We find that the value of memory is truncation-dependent. When the full dialogue history fits in the model context, memory mechanisms provide no additional quality gain. When history is truncated, memory systems that rewrite degrade substantially, while the no-memory baseline of full-context injection remains almost unaffected. Rewriting before storage also causes a net information loss: the median storage size of rewriting systems is four times to over an order of magnitude smaller that of verbatim storage. We term this irreversible loss the rewriting-is-forgetting effect. We therefore propose Plain Writing with Memory-Managed Lookup (PW-MLM): the write side stores raw messages verbatim, and the read side controls query cost through temporal resolution and fixed-budget scheduling. Across three QA benchmarks spanning different history lengths, PW-MLM uses only about one-third of the injected context under moderate truncation and matches the quality of full-context injection. In the long-horizon regime, it degrades far less than rewriting-based baselines, a result that delineates the applicability boundary of zero-rewrite, budgeted reading. Our findings offer testable principles for memory design: when the full history can be injected into the context, memory mechanisms are unnecessary; when it cannot, verbatim fidelity outperforms intelligent rewriting. `https://anonymous.4open.science/r/PW-MLM-E1AA/`.

1 INTRODUCTION

The long-horizon interaction capabilities of Large Language Model agents are constrained by fixed context windows, as continuously accumulating interaction histories often exceed their capacity (Xu et al., 2024; Zhang et al., 2025a). Memory mechanisms (Hu et al., 2026; Shutova et al., 2026; Zhong et al., 2024; Gutiérrez et al., 2024; Zhao et al., 2024), which preserve historical information across sessions and retrieve it for subsequent interactions, have become a core component of LLM agents, enabling them to retain user-specific knowledge and track dynamic information (Jiang et al., 2026; Wu et al., 2026b; Behrouz et al., 2026; Park et al., 2023; Wu et al., 2024; Maharana et al., 2024; Wang et al., 2024). Although existing memory systems employ diverse underlying data structures such as hierarchies, forgetting curves, knowledge graphs, or fact bases, they all adhere to the same convention when writing to the memory store. They first invoke an LLM to summarize, extract, consolidate, and forget raw interactions, thereby processing them into more condensed and clean memories (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2026). This practice of rewriting prior to storage has become the de facto standard in agent memory systems.

However, this de facto standard rests on an unexamined assumption. Most existing evaluations are conducted on relatively short interaction histories where memory mechanisms have not yet been subjected to capacity pressure. Consequently, the performance gap between rewriting and verbatim storage remains unobserved (Jiayang et al., 2026; Xu et al., 2022a;b). When evaluations shift to

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.

longer interaction histories, disparate backbone models, retrievers, and evaluator models across different systems make it impossible to attribute performance differences solely to the writing strategy itself (Wu et al., 2024; Chen et al., 2026). Even though prior work has proposed unified evaluation frameworks encompassing indexing, retrieval, and reading, the focus remains predominantly on optimizing retrieval and reading stages, leaving the rewriting convention at the writing stage as a default choice (Trivedi et al., 2023; Shinn et al., 2023). Meanwhile, sporadic evidence suggests that rewriting entails information loss. For instance, retrieval using conversational summaries as context yields high recall rates but fails to improve question answering accuracy (Yang et al., 2025; Abbasiantaeb et al., 2026). Collectively, these clues point to an unanswered question regarding whether rewriting at write time distills information or discards it. Unlike prior work focused on retrieval side optimization, this paper shifts the analytical focus to the writing strategy itself.

To answer this question, we conduct controlled experiments under a unified configuration, varying only the writing strategy across three history length regimes including no truncation, truncated, and long horizon. This design ensures that observed performance differences can be attributed directly to the writing strategy. Our experiments reveal three key phenomena. First, the value of memory mechanisms depends critically on the degree of history truncation. When the complete history is injected directly into the context, the quality benefits of various memory mechanisms vanish. Conversely, when the history exceeds model capacity, rewriting based memory systems collapse. Second, a storage audit of the memory store quantifies information loss during the writing phase, revealing that rewriting based systems discard the vast majority of original information during storage. Finally, in the long-horizon regime rewriting systems collapse while verbatim reading degrades far less, demonstrating that the bottleneck lies not in retrieval but in retention. These controlled comparisons indicate that what determines performance is whether information is preserved rather than how intelligently it is retrieved.

Based on these findings, we propose the principle of zero rewriting at write time and budgeted allocation at read time, and introduce Plain Writing with Memory Managed Lookup (PW-MLM). On the write side, user messages are stored verbatim with only timestamps appended without invoking any LLM processing, thereby eliminating information loss at the source. On the read side, retrieved memory entries undergo conflict resolution via a newer overrides older policy and are then filled chronologically into a fixed budget context window. We validate this design across three long horizon benchmarks comprising approximately 3,700 question answering instances. In the medium truncation regime, PW-MLM matches the performance of the full context injection baseline while using only one third of the query context length, outperforming rewriting based baselines by over 15 percentage points. In the long horizon regime, PW-MLM achieves quality comparable to full context injection at roughly one sixth of the query cost, with score degradation less than half that of sliding window systems. The main contributions of this paper are as follows:

- Through controlled comparisons, we demonstrate that LLM rewriting at write time results in net information loss rather than distillation, and quantify this phenomenon as the rewriting is forgetting effect via storage auditing.
- We propose a controlled evaluation protocol that couples history-length regime comparison with storage and query-cost auditing, enabling performance differences to be attributed specifically to the writing strategy under a unified configuration.
- We design and implement PW-MLM, adhering to the principles of zero rewriting at write time and budgeted reading, achieving full context injection quality at approximately one sixth of the query cost.

2 Related Work

2.1 Memory Systems and Writing Strategies

Early agent memory systems largely follow the retrieval-augmented generation paradigm (Shuster et al., 2021), storing raw messages verbatim in an external store and retrieving relevant fragments by semantic similarity in later conversations (Park et al., 2023). Although this zero-intervention strategy avoids the risk of information loss, it suffers from storage bloat and retrieval noise, and these pressures become particularly acute in long-horizon interactions (Packer et al., 2023; Zhang et al., 2025b). To mitigate these bottlenecks, prior work organizes raw interactions at write time,

for instance by synthesizing observations into high-level reflections and topic summaries, encoding interactions into evolvable structured notes (Xu et al., 2026), extracting conversations into atomic facts with conflict-aware updates (Chhikara et al., 2025), or granting the agent self-editing abilities to move and compress entries across hierarchical memories (Packer et al., 2023), so as to reduce context pressure and improve retrieval precision. On the read side, existing systems mostly rely on fixed top-K retrieval (Tan et al., 2025; Pan et al., 2025) or concatenation of the full history (Rasmussen et al.), without explicit budget control over query cost. Unlike retrieval-side optimization efforts (Trivedi et al., 2023; Shinn et al., 2023), these systems all share the assumption that LLM rewriting improves memory, and we question this assumption itself.

2.2 The Evolution of Conversational Memory Evaluation

Evaluation of conversational memory for LLM agents has evolved from static full-context injection to incremental interaction. Early benchmarks inject the entire dialogue history into the model context at once and probe information extraction with multi-type question answering over single-hop, multi-hop, and temporal reasoning queries (Shaham et al., 2023; Bai et al., 2024). As dialogue scale grows, benchmarks begin to simulate realistic incremental interaction by feeding long conversations session by session to examine memory retention and updating (Shen et al., 2026; He et al.), or introduce long-term multi-session dialogues to test temporal sensitivity and temporal reasoning (Banerjee et al., 2026; Ge et al., 2025). Recent work further extends memory evaluation to precise retrieval, online adaptation during interaction (test-time learning), and dynamic suppression of irrelevant memories (Cao et al., 2026), and starts to examine the effect of history length on memory performance (Xie et al., 2026; Wu et al., 2026a). However, existing evaluations typically couple the writing strategy with specific backbone models and retrieval architectures and lack controlled comparisons across history length regimes, which makes it hard to attribute performance differences to the writing strategy itself. We address this gap by attributing performance differences to the writing strategy through cross-regime length comparisons under a unified configuration.

2.3 Information Fidelity of Large Language Models

Writing into the memory of an LLM agent is essentially a lossy compression of the raw conversation, and recent studies show that such information loss is not incidental but stems from inherent limitations of models in context utilization and information transformation (Maharana et al., 2024). In context utilization, models exhibit positional bias in information extraction, using information in the middle of the context markedly worse than information at the beginning or the end (Liu et al., 2024), and the inclusion of irrelevant content distracts attention and degrades generation quality (Shi et al., 2023). In information compression and rewriting, retrieval-based reduction cannot guarantee coverage of all task-critical information, and an incomplete context significantly harms the final generation quality (Zhang et al., 2024). When memory writing adopts a fixed segmentation granularity, overly fine cuts fragment semantics while overly coarse cuts inject substantial irrelevant noise, and both weaken retrieval and reasoning (Pan et al., 2025). If the system further consolidates or overwrites stored entries, the historical context required for temporal reasoning may be lost (Banerjee et al., 2026). These limitations jointly point to a question that remains empirically unanswered, namely whether rewriting at write time distills information or discards it. We answer this question through controlled comparative experiments.

3 Methodology

We model a long-horizon memory system as a quadruple $(\mathcal{C}, \mathcal{S}, \mathcal{R}, \mathcal{L})$: capture maps each new interaction ΔH_k to a memory operation, storage maintains the store $\mathcal{M}$, read builds an injected context $C(q)$ for query q, and lifecycle governs retention and eviction. Each entry $e = (x, s, t)$ holds content x, source tag $s \in \{\text{raw}, \text{fact}\}$, and timestamp t. Two policies parameterize the quadruple: a write policy π_w that updates $\mathcal{M}_k$ from ΔH_k, and a read policy π_r that constructs $C(q)$ under the budget $|C(q)|_t \le B$. The objective is

$$\max_{\pi_r} \ \mathbb{E}_q\big[\text{quality}(q, C(q))\big] \quad \text{s.t.} \quad |C(q)|_t \le B \tag{1}$$

where $\mathbb{E}_q$ is over all queries. Conventional systems place the scheduling intelligence in π_w, compressing via an LLM ($\pi_w(m) = f_{\text{LLM}}(m)$, $|\pi_w(m)| \ll |m|$) and thus irreversibly discarding raw

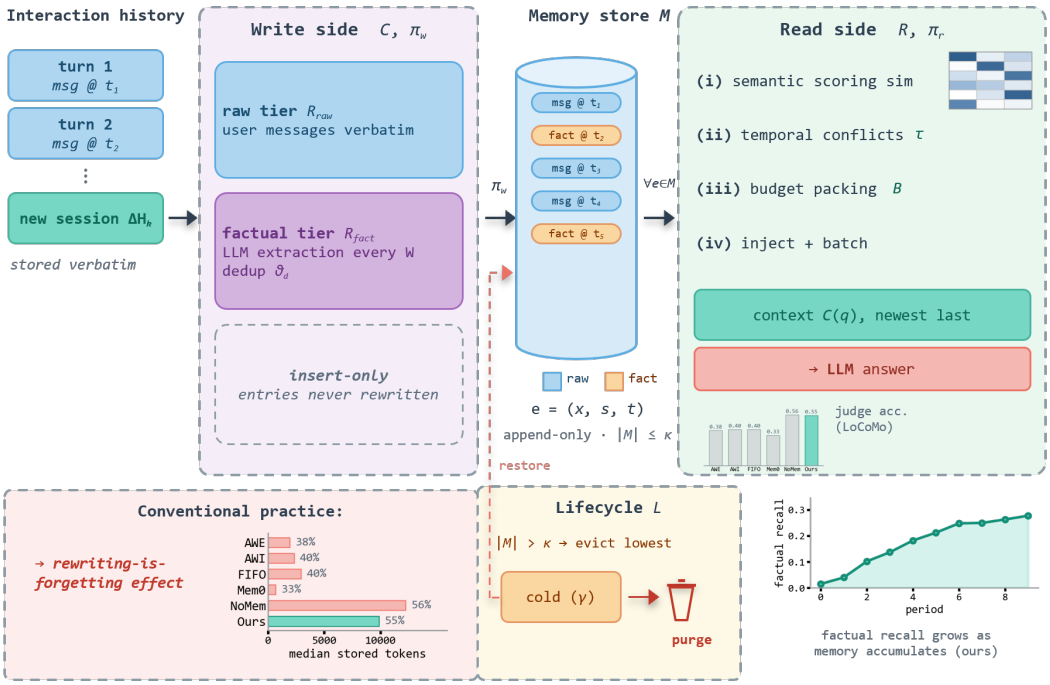


Figure 1: Overview of the PW-MLM framework. The system reverses conventional memory scheduling by deferring compression to read time, preserving raw information through incremental zero-rewrite capture.

information before any query arrives. PW-MLM reverses this division: Sec. 3.1 instantiates $\mathcal{C}$ and π_w with an incremental zero-rewrite mapping; Sec. 3.2 realizes $\mathcal{R}$ and π_r via full-store parsing and budgeted scheduling to solve Eq.(1); Sec. 3.3 reduces $\mathcal{L}$ to an eviction-only capacity valve (Fig. 1).

3.1 Write Side: Incremental Zero-Rewrite Capture

As an instantiation of the capture module $\mathcal{C}$ and write policy π_w, we design the capture mapping to be purely incremental: it only inserts new entries and never touches existing ones, thereby enforcing the zero-rewrite principle on the write side. The memory store evolves as,

$$\mathcal{M}_k = \mathcal{M}_{k-1} \cup \pi_w(\Delta H_k) \tag{2}$$

This stands in contrast to the rewrite mapping adopted by conventional systems, i.e., $\mathcal{M}_k = \rho(\mathcal{M}_{k-1} \cup \Delta H_k)$. The inserted content set comprises two tiers,

$$\pi_w(\Delta H_k) = \mathcal{T}_{\text{raw}} \cup \mathcal{T}_{\text{fact}} \tag{3}$$

where the raw tier $\mathcal{T}_{\text{raw}}$ stores user messages verbatim, and the factual tier $\mathcal{T}_{\text{fact}}$ is extracted by an LLM every W rounds (where W denotes the extraction window size). It is crucial to emphasize that factual extraction still invokes an LLM; the distinction from rewriting lies in the boundary of side effects. Since π_w performs only insertion operations, existing raw entries never disappear. In contrast, rewrite mappings subject information to irreversible transformations before it can compete for the token budget. If the extraction call fails, the system gracefully degrades to the raw tier, leaving fidelity unaffected. Furthermore, to suppress redundant expansion caused by repeatedly extracting the same facts, an entry x is not inserted if there already exists an entry with a cosine similarity higher than the deduplication threshold θ_d within the recent window. All hyperparameter values are detailed in Sec. 4.1.

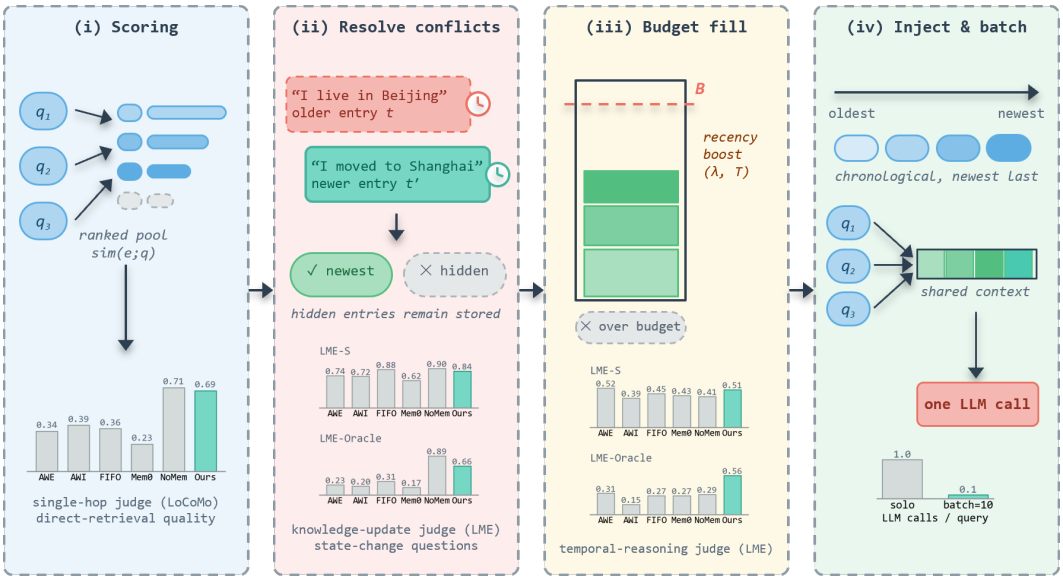


Figure 2: The four-stage read pipeline: (i) scoring and candidate pool construction, (ii) temporal conflict resolution, (iii) budget competition via rank-weighted knapsack, and (iv) context injection with batching.

3.2 Read Side: Full-Store Parsing and Budgeted Scheduling

The reading module $\mathcal{R}$ and the read policy π_r bear the full scheduling intelligence. Since the quality term in Eq. (1) is implicitly determined by the generative model and cannot be directly optimized, π_r adopts a computable ranking score as its proxy. It approximates this objective and constructs the context $C(q)$ under the budget constraint B through the four stages illustrated in Fig. 2.

(i) Scoring and Candidate Pool Construction. We score the entire memory store to construct a candidate pool,

$$sim(e; q) = \cos(f(x_e), f(q)) \tag{4}$$

where $f(\cdot)$ denotes the shared text embedding encoder that maps content into a cosine-comparable vector space. However, truncating based solely on similarity may discard entries that, despite differing in phrasing from the query, contain the current state required for the answer.

(ii) Temporal Conflict Resolution. We resolve temporal conflicts within the candidate pool. An entry e' is said to *dominate* e if and only if:

$$t_{e'} > t_e \wedge \cos(f(x_{e'}), f(x_e)) > \tau \tag{5}$$

where e' is a strictly updated state of the same topic with semantic similarity above threshold τ. When two entries share the same creation timestamp, ties are broken in favor of the raw tier, so the verbatim record is preferred as the surviving state. Conflict resolution retains only non-dominated entries:

$$\mathcal{M}_{res} = \{e \in \mathcal{M} \mid \nexists e' \text{ s.t. } e' \text{ dominates } e\} \tag{6}$$

This implies that dominated old states are logically hidden but physically preserved, while non-dominated entries always survive. Thus, no unique information is lost during conflict resolution: which entries are actually injected is decided by the budget competition in the next stage, whereas physical deletion is confined to the lifecycle module (Sec. 3.3).

(iii) Budget Competition. Surviving entries then compete for the budget using the following ranking score:

$$rank(e; q) = sim(e; q) + \lambda \cdot 2^{-\Delta t(e)/T} \tag{7}$$

where $\Delta t(e)$ is the entry's age, T is the total interaction span, and λ is a small weight. The exponential decay encodes the prior that "current state" queries are more likely to involve recent facts.

Budget filling solves the following rank-weighted knapsack problem via a greedy algorithm:

$$\max_{\mathcal{S}_q \subseteq \mathcal{M}_{res}} \sum_{e \in \mathcal{S}_q} rank(e; q) \quad \text{s.t.} \quad \sum_{e \in \mathcal{S}_q} |x_e|_t \leq B \tag{8}$$

(iv) Context Injection and Batching. Surviving entries are injected in chronological order (raw before factual at identical timestamps) under the instruction that the latest state prevails, so the model resolves old-versus-new states in tandem with the parsing step. Questions in the same (session, time-window) group share one LLM call: their parsed sets are unioned, re-resolved for temporal conflicts, and filled jointly under budget B; since the pooled context is a superset of every single-question retrieval before budgeting, batching loses no reachable evidence.

3.3 Lifecycle Module: Eviction-Only Capacity Control

The lifecycle module $\mathcal{L}$ assumes only a capacity-control function. When the storage size exceeds the preset capacity limit κ, the module is triggered to remove the $|\mathcal{M}| - \kappa$ entries with the lowest composite scores, restoring the storage size to κ; otherwise, the module remains inactive. The composite score of an entry is defined as,

$$\text{score}(e) = \omega_1 \, s_{\text{rec}}(e) + \omega_2 \, s_{\text{freq}}(e) + \omega_3 \, s_{\text{cons}}(e) + \omega_4 \, s_{\text{red}}(e), \tag{9}$$

where $s_{\text{rec}} = \delta^{\,k - a_e}$ scores recency (a_e: last access round; $\delta \in (0, 1)$: decay), s_{freq} log-normalizes the cumulative access count n_e so that hot entries cannot dominate, $s_{\text{cons}} = 1 - c_e / \max(1, n_e)$ penalizes entries flagged as superseded c_e times (the checker is disabled by default, making this term constant), and $s_{\text{red}} = 1 - \max_{e' \neq e} \cos(f(x_e), f(x_{e'}))$ down-weights redundant entries; $\omega_1, \ldots, \omega_4$ weight the four dimensions. Evicted entries move to a cold-storage buffer and are restored if a query within γ periods matches them at similarity no less than θ_r; otherwise they are permanently removed. The append-only property of π_w therefore holds even under capacity pressure: no surviving entry is ever rewritten, and the only irreversible operation is deletion after the observation window.

4 Experiments

4.1 Setup

We evaluate seven memory strategies in one unified framework, varying only the write/read policy so that any difference is attributable to the memory design itself. Strategies fall into three groups: *verbatim* (No Memory full-context injection; FIFO sliding window (Packer et al., 2023)), *rewriting* (AWI (Wang et al., 2024), AWE (Xu et al., 2024), Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2026)), and *PW-MLM* (ours), which writes verbatim and reads via temporal conflict resolution, deduplication, and fixed-budget filling. Evaluation spans three QA benchmarks (∼3,700 instances) covering three history-length tiers: AMemGym (no truncation) (Jiayang et al., 2026), LoCoMo (medium truncation) (Maharana et al., 2024), and LongMemEval (long horizon) (Wu et al., 2024). We report accuracy (multiple choice), Token-F1 (open-ended), LLM-judge score, query cost $|C(q)|_t$, and storage size. Backbones, frozen components, decoding settings, full benchmark statistics, and every hyperparameter are given in App. A.

4.2 Main Results: Comparison Across History Length Tiers

Table 1 presents the quality comparison across three history length tiers, while Table 2 details the corresponding query and storage costs. We summarize three key observations below.

Finding 1: Memory mechanisms yield no gains in the no-truncation tier. When the complete history fits within the context window (AMemGym), full-context injection serves as the quality upper bound; no memory mechanism surpasses it (Table 1). FIFO achieves comparable performance via verbatim windowing, whereas rewriting-based systems incur a 4–13 judge-point deficit due to information loss during the write phase. PW-MLM matches rewriting systems in quality at this tier but reduces query cost to merely $1/13$ of full-context injection (Table 2). Its marginal gap with FIFO stems from read-side budget truncation (B) rather than write-side loss, as the full history is

Table 1: Question-answering quality across three history length tiers.

Backbone	Strategy	AMemGym		LoCoMo		LME-S		LME-Oracle	
		Acc	Judge	F1	Judge	F1	Judge	F1	Judge
qwen3.7-plus	No Memory	56.7	66.0	28.4	55.7	43.6	65.7	42.0	61.2
	FIFO	55.4	65.2	10.5	39.7	43.6	66.4	8.1	25.2
	AWI	51.4	62.0	11.0	39.7	29.6	49.9	5.7	17.2
	AWE	51.1	61.8	8.4	38.5	40.3	63.5	9.6	22.8
	Mem0	40.1	53.0	4.2	33.3	43.6	64.6	6.0	22.0
	PW-MLM (Ours)	**51.4**	**62.0**	**27.4**	**54.9**	**42.6**	**64.5**	**29.6**	**48.8**
GLM-5.3-Flash	A-Mem	50.8	60.8	24.2	61.5	41.8	84.6	42.2	78.0
	PW-MLM (Ours)	**54.3**	**64.4**	**21.7**	**59.0**	**34.7**	**72.1**	**24.8**	**53.2**

Table 2: Query cost (Q-Tok) and storage audit (median store size in tokens).

Backbone	Strategy	AMemGym		LoCoMo		LME-S		LME-Oracle	
		Q-Tok	Store	Q-Tok	Store	Q-Tok	Store	Q-Tok	Store
qwen3.7-plus	No Memory	5,054	5,637	12,160	12,221	4,731	5,495	119,761	122,408
	FIFO	5,078	8,071	3,012	2,944	5,051	5,401	5,454	5,630
	AWI	117	1,078	111	2,328	93	135	54	$0^{\dagger}$
	AWE	461	443	326	1,949	293	295	398	498
	Mem0	684	834	564	689	584	596	656	846
	PW-MLM (Ours)	**397**	**7,455**	**4,047**	**9,856**	**736**	**807**	**3,909**	**13,998**
GLM-5.3-Flash	A-Mem	22,604	43,626	3,697	19,931	6,722	6,573	15,007	123,472
	PW-MLM (Ours)	**394**	**13,815**	**4,064**	**10,265**	**1,415**	**1,586**	**3,911**	**15,190**

inherently preservable. This corroborates our first introductory finding: when history is not truncated, memory mechanisms provide no additional quality gain over full-context injection, and any write-side rewriting constitutes pure information loss.

Finding 2: Rewriting systems collapse under medium truncation, while verbatim reading matches full context at $1/3$ the cost. Once injecting the full history becomes infeasible (LoCoMo), the two system paradigms diverge significantly. FIFO and all three rewriting systems fall 16–22 judge points below the full-context baseline, whereas PW-MLM remains on par with full context while injecting only one-third of the tokens. Furthermore, PW-MLM outperforms the strongest rewriting baseline by over 15 judge points (Tables 1 and 2). Storage audits reveal the source of this asymmetry: rewriting systems compress their final stores by up to $18\times$ compared to verbatim storage, effectively discarding evidence required for multi-hop and temporal reasoning before any query arrives.

Finding 3: Retention over retrieval determines survival in the long-horizon tier. On LongMemEval-S, PW-MLM trails full context and FIFO by merely 1–2 points while costing approximately one-sixth of the full-context query budget. The oracle segmentation, with haystacks reaching $\sim$120k tokens, pushes this divergence to its extreme: FIFO and rewriting systems collapse entirely, whereas PW-MLM degrades by less than half the margin of FIFO, at only 3.3% of the full-context query cost (Tables 1 and 2). Notably, PW-MLM surpasses full-context injection on multi-session and temporal reasoning questions. This suggests that budgeted reading with conflict resolution effectively suppresses interference noise that becomes unmanageable when entire haystacks are injected verbatim.

4.3 Ablation Study and Budget Analysis

To isolate the sources of performance, we conduct component-wise ablations of PW-MLM on AMemGym and LoCoMo, alongside a sweep over the read budget B (Table 3 and Fig. 4; the full sweep is deferred to Table 10). The ablated components include: the verbatim layer ($\mathcal{T}_{\text{raw}}$), the fact layer ($\mathcal{T}_{\text{fact}}$), temporal conflict resolution ($\tau \to \infty$), the recency prior ($\lambda = 0$), and batch scheduling.

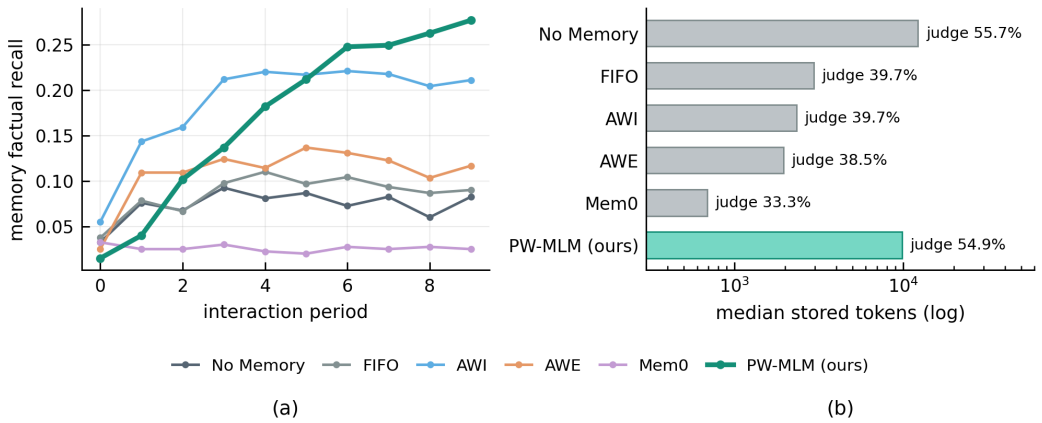


Figure 3: (a) State retention over the horizon: per-period factual recall of the current user states on AMemGym across the ten-period horizon. Only verbatim storage keeps accumulating retained states (PW-MLM rises from 0.02 to 0.28), whereas rewriting-based stores plateau early and Mem0 remains near zero. (b) Storage audit on LoCoMo: median final store size (log scale) with the corresponding judge score; rewriting shrinks the store by up to the $18\times$ quantified in Sec. 4.2 while simultaneously losing quality.

Table 3: Component ablation of PW-MLM. Bold indicates the full model.

Configuration	**AMemGym**		**LoCoMo**	
	F1	Judge	F1	Judge
PW-MLM (Full)	**57.9**	**62.0**	**27.4**	**54.9**
w/o Verbatim Layer	40.6	49.8	8.7	35.6
w/o Fact Layer	55.1	59.7	24.2	50.4
w/o Temporal Conflict Resolution	54.3	58.4	23.1	49.1
w/o Recency Prior	55.6	59.9	25.0	51.3
w/o Batch Scheduling	56.4	60.6	26.1	53.2

The verbatim layer is the primary driver of performance. Removing the verbatim layer while retaining only the fact layer effectively degrades the write side to pure LLM-based summary extraction. This causes the Judge score to drop from 62.0 to 49.8 on AMemGym and from 54.9 to 35.6 on LoCoMo—a decline far exceeding that of any other single component. Notably, the resulting performance matches that of rewriting-based systems on LoCoMo (Table 1), confirming that without the verbatim layer, PW-MLM degenerates into a standard rewriting system and reproduces the "rewriting-is-forgetting" collapse. Removing each of the remaining four components causes only a 1–5 point decrease, with consistent trends across both benchmarks (Fig. 4(a)). The contribution ranking aligns with our design philosophy of prioritizing retention over reading.

Quality saturates at a budget of ∼4k tokens. Performance increases monotonically with B but plateaus after $B = 3{,}900$. Increasing the budget from 3.9k to 8k yields merely 0.3–0.4 additional Judge points on both benchmarks while doubling the query cost. Conversely, reducing B to 1k incurs a substantial loss of approximately 10–13 points. This demonstrates that the default budget lies precisely at the knee of the quality–cost curve, capturing the vast majority of retrievable evidence without unnecessary overhead.

4.4 Analysis and Discussion

Storage Audit: Quantifying "Rewriting Is Forgetting." Table 2 audits the write phase, revealing that rewriting-based systems retain $\leq 0.7\%$ of long histories (e.g., LME-Oracle), whereas PW-MLM stores grow synchronously with raw history (Fig. 3(b)). This pre-query information loss is

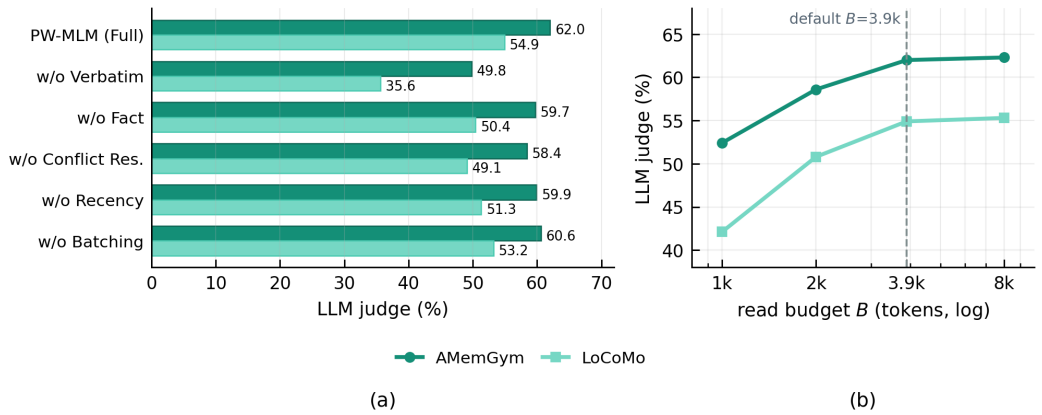


Figure 4: (a) Component ablation of PW-MLM (LLM judge): removing the verbatim layer causes by far the largest drop (12.2 and 19.3 points on AMemGym and LoCoMo), degrading PW-MLM into a standard rewriting system, while each remaining component contributes 1–5 points. (b) Budget sweep saturation: quality saturates at the default $B = 3{,}900$; doubling the budget adds only 0.3–0.4 judge points.

irrecoverable by read-side intelligence, substantiating our core claim: the long-horizon bottleneck lies in *retention*, not retrieval.

Backbone Robustness and the Positioning of A-Mem. Under the `GLM-5.3-Flash` backbone, PW-MLM's cost advantages persist ($1/3$ to $1/9$ query cost of full-context). A-Mem achieves the highest LongMemEval judge score in this panel, but its 32-message chunking effectively functions as near-verbatim storage, injecting $\sim 4\times$ more tokens than PW-MLM. Thus, A-Mem's success aligns with our *retention* thesis, while genuine rewriting baselines collapse. On AMemGym, an interrupted A-Mem run (24/40 personas) shows no significant accuracy difference from PW-MLM on the same subset, yet incurs $56.6\times$ higher query costs.

Applicability Boundary and Cost–Quality Positioning. PW-MLM's residual gap with full-context injection concentrates on assistant-side queries (14.5 vs. 79.7 F1), as our write side currently limits verbatim capture to the user side. This delineates our applicability boundary: zero-rewrite capture must cover all queried dialogue roles. On the cost–quality plane, PW-MLM consistently occupies the Pareto frontier, trailing full-context quality by merely 1–12 points at $1/3$ to $1/30$ of the cost. Ultimately, while short histories favor direct full-context injection, verbatim writing with budgeted reading strictly dominates the "rewrite-then-retrieve" convention for all long-horizon workloads.

5 Conclusion

In this work, we uncover the *rewriting-is-forgetting* effect in LLM agent memory: compressing interactions prior to storage causes irreversible information loss that bottlenecks long-horizon performance. To overcome this, we propose PW-MLM, shifting the paradigm from write-side abstraction to read-side budgeted scheduling. By preserving verbatim history and resolving temporal conflicts under a strict token budget, PW-MLM matches full-context injection quality at a fraction of the cost, substantially outperforming rewriting baselines. Our findings demonstrate that preserving raw fidelity is vastly superior to premature lossy compression. Future work will explore asymmetric capture granularities to further improve assistant-side retention.

AI Use Statement

In this work, we used generative AI tools for: (1) generating the AMemGym benchmark via `qwen3.7-plus`; (2) translating manuscript portions from Chinese to English. We did not use

AI for research ideation, method design, or result interpretation. AI-assisted language polishing was also used as a recommended-disclosure task. All AI-generated data and translations were manually verified by both authors for accuracy and consistency. We take full responsibility for the final content of this work.

Ethics Statement

This work involves no human subjects, sensitive personal data, or harmful content. All experiments use public benchmarks or synthetic data without real user information. The proposed method poses no dual-use risks or negative societal impacts and complies with the ICLR Code of Ethics.

Reproducibility Statement

Full source code, configs, evaluation scripts, and reproduction instructions are available at the anonymous GitHub repository linked in the abstract. All hyperparameters and implementation details are documented in the repository and Appendices A–B.

References

Zahra Abbasiantaeb, Simon Lupart, and Mohammad Aliannejadi. Generating multi-aspect queries for conversational search. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8201–8217, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. Apex-mem: Agentic semi-structured memory with temporal reasoning for long-term conversational ai. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16470–16489, 2026.

Ali Behrouz, Farnoosh Hashemi, Adel Javanmard, and Vahab Mirrokni. Language models need sleep: Learning to self-modify and consolidate memories. *arXiv preprint arXiv:2606.03979*, 2026.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. *Advances in Neural Information Processing Systems*, 38:115487–115510, 2026.

Zhikai Chen, Jialiang Gu, Junyu Yin, Xianxuan Long, Shenglai Zeng, Xiaoze Liu, Kai Guo, Keren Zhou, and Jiliang Tang. Exploring cross-scenario generality of agentic memory systems: Diagnostics and a strong baseline. *arXiv preprint arXiv:2606.04315*, 2026.

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.

Yubin Ge, Salvatore Romeo, Jason Cai, Raphael Shu, Yassine Benajiba, Monica Sunkara, and Yi Zhang. Tremu: Towards neuro-symbolic temporal reasoning for llm-agents with memory in multi-session dialogues. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18974–18988, 2025.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, et al. Benchmarking agent memory in interdependent multi-session agentic tasks. In *Forty-third International Conference on Machine Learning*.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, volume 2026, pp. 156259–156291, 2026.

Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, et al. Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations. *arXiv preprint arXiv:2602.19320*, 2026.

Cheng Jiayang, Dongyu Ru, Lin Qiu, Yiyang Li, Xuezhi Cao, Yangqiu Song, and Xunliang Cai. Amemgym: Interactive memory benchmarking for assistants in long-horizon conversations. *arXiv preprint arXiv:2603.01966*, 2026.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

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, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. On memory construction and retrieval for personalized conversational agents. *arXiv preprint arXiv:2502.05589*, 2025.

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, 2025. *URL https://arxiv.org/abs/2501.13956*.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. Zeroscrolls: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 7977–7989, 2023.

Ye Shen, Dun Pei, Yiqiu Guo, Junying Wang, Yijin Guo, Zicheng Zhang, Qi Jia, Jun Zhou, and Guangtao Zhai. Evolmem: A cognitive-driven benchmark for multi-session dialogue memory. *arXiv preprint arXiv:2601.03543*, 2026.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International conference on machine learning*, pp. 31210–31227. PMLR, 2023.

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.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3784–3803, 2021.

Alina Shutova, Alexandra Olenina, Ivan Vinogradov, and Anton Sinitsin. Evaluating memory structure in llm agents. *arXiv preprint arXiv:2602.11243*, 2026.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Di Wu, Zixiang Ji, Asmi Kawatkar, Bryan Kwan, Jia-Chen Gu, Nanyun Peng, and Kai-Wei Chang. Longmemeval-v2: Evaluating long-term agent memory toward experienced colleagues. *arXiv preprint arXiv:2605.12493*, 2026a.

Shengguang Wu, Hao Zhu, Yuhui Zhang, Xiaohan Wang, and Serena Yeung-Levy. Automem: Automated learning of memory as a cognitive skill. *arXiv preprint arXiv:2607.01224*, 2026b.

Wenya Xie, Shengming Zhou, Zelin Li, Pouya Parsa, Shuang Zhou, Xinheng Ding, Chinmay Arvind, Guanchu Wang, Vladimir Braverman, Ali Payani, et al. Dynamicmem: A long-horizon memory benchmark in real-world settings. *arXiv preprint arXiv:2606.22877*, 2026.

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

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets long context large language models. In *International Conference on Learning Representations*, volume 2024, pp. 49569–49584, 2024.

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

Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2639–2650, 2022b.

Seunghan Yang, Juntae Lee, Jihwan Bang, Kyuhong Shim, Minsoo Kim, and Simyung Chang. Learning contextual retrieval for robust conversational search. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12002–12014, 2025.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *International Conference on Learning Representations*, volume 2025, pp. 101932–101948, 2025a.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

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.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

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.

A Implementation Details

Framework and frozen components. As noted in Sec. 4.1, all seven strategies share one framework and differ only in their write/read policies. Concretely, the embedding encoder (`qwen3-embedding`), the LLM judge (`qwen3.7-plus`), and the simulated user (`qwen3.7-flash`) are frozen across all runs; decoding uses temperature 0 and seed 42. Memory operations of every baseline (extraction, consolidation, retrieval scoring) use the same backbone as the answering policy of that run.

Backbones and compared strategies. The agent backbone is `qwen3.7-plus`, with `GLM-5.3-Flash` as a second backbone for robustness; A-Mem is run with its official `GLM-5.3-Flash` backbone. The seven strategies fall into three groups. *Verbatim*: No Memory (full-context injection) and FIFO sliding window (Packer et al., 2023). *Rewriting*: AWI (in-context memory list) (Wang et al., 2024), AWE (external vector store) (Xu et al., 2024), Mem0 (atomic facts) (Chhikara et al., 2025), and A-Mem (chunked notes with evolution) (Xu et al., 2026). ***PW-MLM*** (ours) writes verbatim with fact extraction every W turns and reads via temporal conflict resolution (τ), deduplication (θ_d), and budget filling (B); the lifecycle acts only as a capacity valve (κ, δ, γ, θ_r). All numeric values are listed in Table 4.

Hyperparameters. Table 4 lists the complete schedule. All values are fixed across the three history-length tiers; in particular, the read budget B, the conflict threshold τ, and the lifecycle parameters are never tuned per benchmark.

Table 4: Full hyperparameter schedule of PW-MLM. All values are fixed across the three history-length tiers; no per-benchmark tuning is performed.

Symbol / value	Role
$W = 6$	fact-extraction window (turns)
$\tau = 0.78$	conflict-resolution similarity threshold
$\theta_d = 0.92$ (window 60)	write-side deduplication threshold
top-$k = 30$	candidate pool size
$B = 3{,}900$ / 800	read budget for QA / interaction contexts
$\lambda = 0.05$	recency weight in the ranking score
$\kappa = 400$	store capacity (entries)
$\delta = 0.98$	recency decay of the lifecycle score
$\omega = (.15, .45, .20, .20)$	weights of rec./freq./cons./red. scores
$\gamma = 3$, $\theta_r = 0.60$	cold-storage observation cycles, rescue threshold

Benchmark statistics. AMemGym provides 40 personas over 10 evolution periods; question-answering probes are collected on the last three periods (1,215 multiple-choice instances), while per-period state retention is measured on all ten (Fig. 3(a)). LoCoMo contributes 1,986 questions (single-hop 841, adversarial 446, temporal 321, multi-hop 282, open-domain 96). LongMemEval-S contributes 500 questions; its oracle segmentation yields haystacks of roughly 120k tokens.

Cost accounting. Query cost (Q-Tok) is the average number of tokens injected into the answering call per question; for batched strategies this cost is amortized over the questions sharing the call. Storage size is the median token count of the final memory store per persona.

B Prompts

We list verbatim every prompt used by PW-MLM and by the two agentic-write baselines. Placeholders in curly braces are filled at run time.

PW-MLM write prompt (fact tier).

```
You are a fact extraction system for a personal assistant's memory.
```

```
Given this conversation, extract ALL factual information that should
be remembered long-term | both about the user AND information the
assistant stated (recommendations, names, schedules, bookings,
advice).

## Conversation:
{conversation}

## Rules:
- Extract concrete facts, preferences, events, relationships, plans
- Include facts the assistant stated or committed to (e.g.
recommendations the assistant made, schedules the assistant confirmed,
names the assistant provided)
- Each fact should be self-contained (understandable without context)
- Do NOT extract greetings or small talk
- Do NOT extract information that is already obvious/common knowledge

## Output (strict JSON):
{{"facts": ["fact 1", "fact 2", ...]}}
```

PW-MLM temporal-consistency prompt.

```
<!-- memlife/prompts/pw_consistency.txt -->
You are a memory consistency checker.

## Candidate memories (pre-filtered by semantic similarity):
{memories}

## Rules:
1. Find groups of memories that are CONTRADICTORY (same
attribute/topic, different values).
2. For each contradiction, keep only the NEWEST entry (highest turn
number);
   list the outdated/superseded entry IDs in "remove".
3. Also remove exact duplicates, keeping the oldest one.
4. If there are no conflicts or duplicates, return an empty "remove"
list.

## Output (strict JSON):
{{"remove": ["mem_id_1", "mem_id_2", ...]}}
```

PW-MLM consolidation prompt (lifecycle scoring).

```
<!-- memlife/prompts/pw_consolidation.txt -->
You are a memory consolidation system.

## New interaction traces:
{traces}

## Existing memories:
{existing_memories}

## Rules:
1. ADD (default): each trace with new info → one separate new entry.
   Do NOT combine unrelated traces into one entry, and do NOT
   summarize
   away specifics (names, numbers, dates, places must survive
   verbatim).
2. MERGE: ONLY when a trace clearly updates ONE existing memory about
the
   same narrow fact → rewrite that entry; merged_ids must contain
   exactly
```

```
   that one memory id (never more).
3. ABSTRACT: 2+ traces independently repeat the same pattern → extract
   the general rule, but keep the specific instances too (as ADD
   entries).
4. STATE: ONLY when a trace EXPLICITLY contradicts or supersedes an
   existing memory (user changed job, diet, relationship, ...) → write
   the CURRENT state as a new entry and list exactly ONE superseded
   memory id in merged_ids. Do not use STATE for mere elaboration.
5. Each entry: self-contained, < 80 words.
6. When in doubt, prefer ADD over MERGE/STATE | losing a fact is worse
   than a duplicate.

## Output (strict JSON):
{{"entries": [{{"action": "merge|add|abstract|state", "content":
"...", "merged_ids": []}}]}}
```

PW-MLM snapshot prompt (disabled by default).

```
You maintain the memory of a personal assistant. Below are the recent
conversation traces from the period that just ended, plus the existing
memory entries.

## Task:
Write ONE short entry per topic whose CURRENT state is stated or
clearly
implied by the traces. Typical topics: diet, sleep, health, job,
residence,
household, relationships, activities, hobbies, goals, finances,
preferences.

## Rules:
- Each entry: "<Topic>: <current state in the user's own terms>", one
line,
  self-contained, under 30 words.
- Only include topics explicitly mentioned in the traces. Skip the
rest.
- If the state changed during the traces, record the LATEST state.
- Preserve concrete details (numbers, names, places) verbatim.
- Do NOT invent or infer states that were not mentioned.

## Existing memories:
{existing_memories}

## Recent traces:
{traces}

## Output (strict JSON):
{{"snapshots": ["Diet: ...", "Sleep: ..."]}}
```

PW-MLM batched answering prompt.

```
You answer questions about a user based STRICTLY on the user
information provided.

User information (chronological order):
{memories}

Answer EVERY question below, choosing exactly one option per question.
Entries appear in chronological order; when two entries conflict,
trust
the most recent one | the answer must reflect the user's CURRENT
state.
Before finalizing, re-check each answer against the most recent entry
on
```

```
that topic and correct it if an earlier state was used.

{questions}

Reply with strict JSON only, no extra text:
{{"answers": [{{"id": 1, "answer": "C"}}, {{"id": 2, "answer":
"A"}}]}}
```

LLM judge prompt.

```
You are a strict evaluator judging whether an assistant's answer is
semantically correct.

Question:
{question}

Options:
{options}

Assistant's answer:
{agent_answer}

Ground-truth answer value:
{golden_answer}

Judge whether the assistant's answer conveys the same factual meaning
as the ground-truth value. Paraphrasing, different wording, or extra
detail is acceptable as long as the core fact matches. Ignore
stylistic differences.

Scoring rubric:
- 5: Semantically equivalent to the ground truth
- 4: Mostly correct, minor wording difference, no factual deviation
- 3: Partially correct or too vague to be sure
- 2: Related topic but factually different
- 1: Clearly wrong
- 0: No meaningful answer / unrelated

Output JSON only:
{{"reason": "...", "score": <0-5>}}
```

AWE baseline write prompt.

```
You are a fact extraction system for a personal assistant's memory.

Given this conversation, extract ALL factual information about the
user that should be remembered long-term.

## Conversation:
{conversation}

## Rules:
- Extract concrete facts, preferences, events, relationships, plans
- Each fact should be self-contained (understandable without context)
- Do NOT extract greetings, small talk, or assistant responses
- Do NOT extract information that is already obvious/common knowledge

## Output (strict JSON):
{{"facts": ["fact 1", "fact 2", ...]}}
```

AWI baseline update prompt.

Table 5: AMemGym question-answering accuracy (%) per evaluated period. QA probing is performed on the last three periods of the horizon, where accumulated state changes make personalization hardest; per-period state retention over the full horizon is shown in Fig. 3(a).

Strategy	P7	P8	P9	Mean
No Memory	56.0	59.0	55.1	56.7
FIFO	56.8	56.5	52.8	55.4
AWI	51.9	54.3	47.9	51.4
AWE	52.8	52.3	48.1	51.1
Mem0	37.0	42.2	41.0	40.1
PW-MLM (Ours)	**54.1**	**50.6**	**49.4**	**51.4**

Table 6: LoCoMo LLM-judge score (%) by question category.

Strategy	Multi-hop	Temporal	Open-dom.	Single-hop	Advers.
No Memory	63.8	31.3	52.3	71.3	39.4
FIFO	41.8	24.5	38.1	36.4	55.7
AWI	39.1	24.9	40.6	39.3	51.3
AWE	33.7	23.1	38.5	34.2	60.6
Mem0	32.5	19.4	38.3	23.1	62.2
PW-MLM (Ours)	**63.0**	**32.8**	**48.1**	**68.7**	**41.3**

```
You are a memory extraction system. Given a conversation snippet and
existing memories, extract/update structured memory entries.

## Conversation:
{conversation}

## Existing memories:
{existing}

## Rules:
1. Extract factual information about the user (preferences, events,
relationships, plans).
2. If info updates an existing memory (same label), output it with
action "update".
3. If info is new, output with action "add".
4. Each entry: {{"label": "category", "value": "specific fact"}}

## Output (strict JSON):
{{"memories": [{{"label": "...", "value": "...", "action":
"add|update"}}]}}
```

C Additional Results

This section reports the full per-category and per-type breakdowns behind Table 1, the per-period question-answering accuracy on AMemGym, the second-backbone cost audit, and the read-budget sweep that the main text summarizes in Sec. 4.3.

Reading the per-type breakdowns. The breakdowns localize each system's failure mode. On LongMemEval-S, where most evidence still lies inside the reachable context, the no-memory baseline leads the state-change and single-session types; rewriting systems instead collapse on the oracle segmentation (Table 8), where evidence must survive a 120k-token haystack. PW-MLM's residual gap to full-context injection concentrates on single-session-assistant questions, consistent with the user-side capture boundary discussed in Sec. 4.4. On LoCoMo (Table 6), rewriting systems lose

Table 7: LongMemEval-S LLM-judge score (%) by question type.

Strategy	SS-User	SS-Asst	SS-Pref	Multi	Know-U	Temp-R
No Memory	98.3	97.5	56.7	47.5	89.7	41.2
FIFO	98.0	95.7	61.3	47.4	87.7	45.3
AWI	82.0	27.5	50.0	40.3	71.8	39.1
AWE	85.1	30.0	65.3	70.7	73.8	52.5
Mem0	99.1	76.4	69.3	63.6	62.3	42.6
PW-MLM (Ours)	**87.4**	**29.3**	**64.7**	**69.6**	**84.1**	**50.5**

Table 8: LongMemEval-Oracle LLM-judge score (%) by question type.

Strategy	SS-User	SS-Asst	SS-Pref	Multi	Know-U	Temp-R
No Memory	88.0	100.0	20.0	52.9	88.6	29.1
FIFO	20.0	20.0	30.0	24.7	31.4	27.3
AWI	30.0	6.7	20.0	11.8	20.0	14.5
AWE	26.0	13.3	30.0	16.5	22.9	30.9
Mem0	18.0	20.0	40.0	21.2	17.1	27.3
PW-MLM (Ours)	**46.0**	**80.0**	**40.0**	**34.1**	**65.7**	**56.4**

most on the multi-hop and temporal categories, exactly the question types that require evidence preserved across sessions.

D Additional Analysis

Fig. 5 makes the write-side information loss explicit as a function of the horizon: the rewriting stores of AWE and Mem0 saturate at a few hundred tokens within two periods and never grow afterwards, whereas the verbatim store of PW-MLM grows linearly with the interaction history. Fig. 6 complements Tables 1 and 2 on the cost–quality plane: in every tier, no baseline dominates PW-MLM in both quality and query cost simultaneously, and the distance to the full-context baseline shrinks relative to its cost advantage as histories lengthen.

E Case Studies

We inspect the final store of one AMemGym persona (persona 000, period 9; 273 entries: 54 raw, 219 factual) to illustrate the three mechanisms of PW-MLM.

Zero-rewrite capture. The raw tier keeps the user message verbatim, with only a timestamp appended:

> `[raw, t=6]` "Hey, can you help me figure out how to adjust my grocery list this week? I want to make sure I'm keeping my sodium in check while still fueling up for my long run this weekend."

The fact tier adds a compact abstraction of the same turn, which coexists with the raw entry instead of replacing it:

> `[fact, t=6]` "The user is training for a long run scheduled for the weekend."

Temporal conflict resolution hides, never deletes. Three factual entries track the same topic across the horizon:

> `[fact, t=6]` "The user is training for a long run scheduled for the weekend."
> `[fact, t=12]` "The user is trying to get back into a running routine despite their ankle injury."

Table 9: Second-backbone (`GLM-5.3-Flash`) quality and cost audit on the external benchmarks. Judge in %; Q-Tok in tokens.

Strategy	**LoCoMo**		**LME-S**		**LME-Oracle**	
	Judge	Q-Tok	Judge	Q-Tok	Judge	Q-Tok
A-Mem	61.5	3,697	84.6	6,722	78.0	15,007
PW-MLM (Ours)	**59.0**	**4,064**	**72.1**	**1,415**	**53.2**	**3,911**

Table 10: Read budget B sweep (Token-F1 / Judge). $B = 3{,}900$ (bold) is the default configuration in the main text. Quality saturates beyond this point, indicating that budgeted reading captures most retrievable evidence at ∼4k tokens.

Budget B	**AMemGym**		**LoCoMo**	
	F1	Judge	F1	Judge
1k	48.2	52.4	19.3	42.1
2k	55.0	58.6	25.1	50.8
3.9k (Default)	**57.9**	**62.0**	**27.4**	**54.9**
8k	58.1	62.3	27.8	55.3

`[fact, t=36]` "The user is training for a half-marathon and runs almost every morning."

For a query issued at $t = 36$, the resolver marks the first two entries as dominated (same topic, similarity above τ, older timestamp) and injects only the newest state; all three entries nevertheless remain physically stored and can be restored to later queries that target the earlier states.

Batched reading. The ten evaluation questions of each (persona, period) group share a single answering call over one pooled context (batch median 10 questions, amortized ∼390 tokens per question versus a 3,900-token budget), which is the source of the query-cost ratios in Table 2.

F LIMITATIONS

First, the raw tier currently captures user-side messages verbatim while the assistant side is covered by the fact tier; assistant-side queries therefore remain the main quality gap (the 14.5 vs. 79.7 F1 gap quantified in Sec. 4.4), and extending zero-rewrite capture to both sides is the direct next step. Second, on AMemGym the question-answering probes are collected on the last three periods for cost reasons; retention over the full horizon is instead measured with the per-period factual-recall metric (Fig. 3(a)). Third, open-ended benchmarks rely on an LLM judge; wherever multiple-choice accuracy is available (AMemGym) we verify that both metrics agree on the strategy ordering. Finally, all runs use a single seed; the reported differences between PW-MLM and the strongest baselines exceed the per-period variance shown in Table 5, but a multi-seed study remains future work.

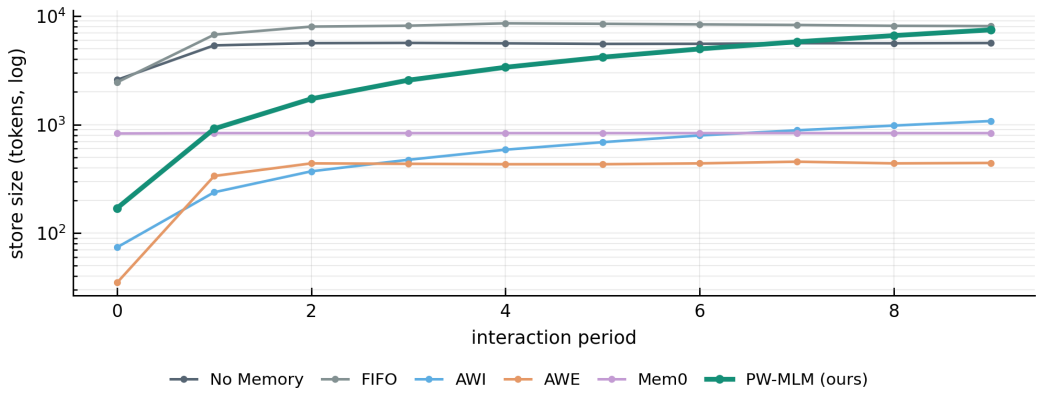


Figure 5: Store growth per interaction period on AMemGym (log scale). Verbatim stores (PW-MLM, No Memory, FIFO) track the accumulated history, while rewriting-based stores saturate within the first two periods: their write policy discards information before any query arrives, which is the mechanism behind the rewriting-is-forgetting effect.

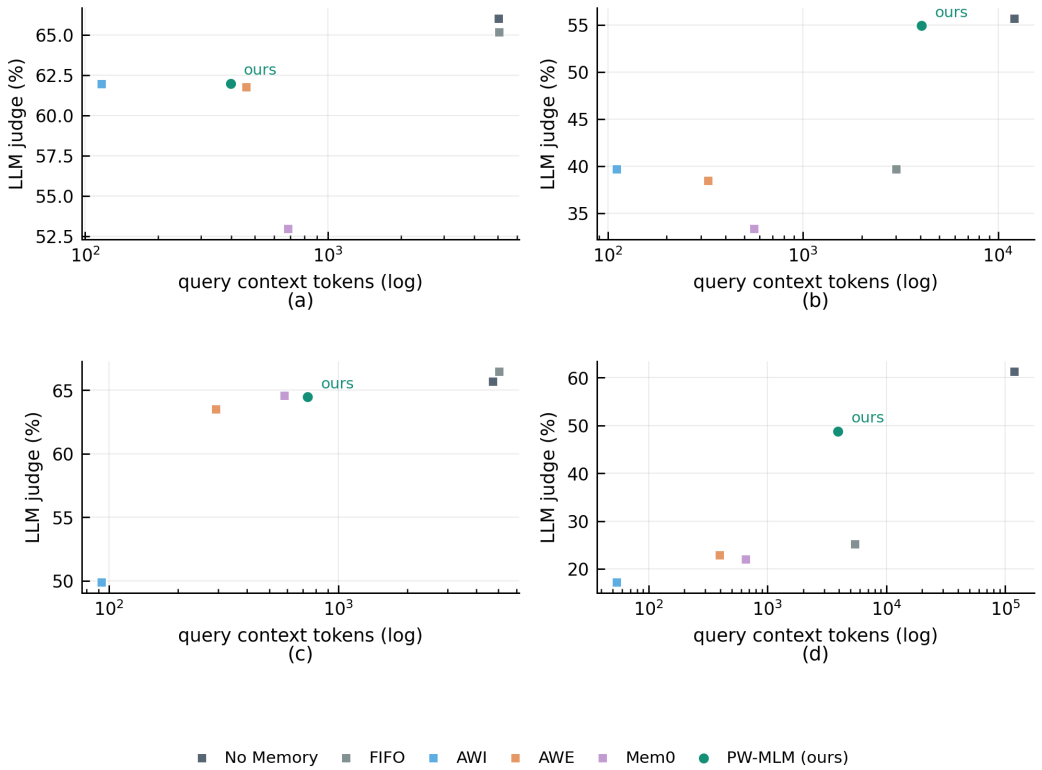


Figure 6: (a) AMemGym, (b) LoCoMo, (c) LME-S, (d) LME-Oracle: cost–quality positioning per history-length tier; LLM judge against injected query tokens (log scale). PW-MLM lies on the Pareto frontier of every tier, matching the quality of full-context injection at a fraction of its query cost; the advantage widens with the history length.