IS ONE SCORE ENOUGH? RETHINKING THE EVALUATION OF SEQUENTIALLY EVOLVING LLM MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory plays a central role in enabling large language models (LLMs) to operate over sequential tasks by accumulating and reusing experience over time. However, existing evaluations of LLM memory mostly rely on aggregate metrics such as final hold-out accuracy or cumulative online performance. We argue that these metrics can be misleading: they collapse distinct memory behaviors into a single number and obscure critical failure modes such as forgetting and negative transfer. We introduce SEQMEM-EVAL to diagnose the behavior of sequentially evolving LLM memory. Drawing inspiration from continual learning, the framework targets a distinct test-time setting in which memory is external, prompt-mediated, and updated without changing model parameters. Beyond measuring whether the final memory state improves performance, SEQMEM-EVAL examines how memory states evolve, generalize, consolidate experience, and retain useful information during sequential inference. Specifically, it measures online utility, hold-out generalization, backward transfer, forgetting and efficiency, providing a finer-grained view of whether memory updates help current tasks, generalize to unseen tasks, improve past predictions, or degrade previously acquired knowledge. Through extensive experiments across diverse tasks and memory methods, we uncover several previously overlooked phenomena. In particular, we show that higher final or cumulative accuracy does not necessarily imply better memory quality: many methods exhibit strong performance gains while suffering from significant forgetting or negative transfers. Moreover, different memory designs exhibit distinct trade-offs between adaptability and stability, which are invisible under standard evaluation metrics. Our findings show that aggregate metrics systematically miss several recurring failure modes, suggesting that a multi-dimensional perspective is essential for understanding LLM memory.

1 INTRODUCTION

Large language models (LLMs) are increasingly equipped with external memory to evolve over sequential tasks (Xiang et al., 2026; Fang et al., 2025b; Wei et al., 2025), where models are expected to accumulate experience and adapt their behavior over time. Through continual interaction with tasks or environments, LLMs generate rich trajectories that contain not only successful solutions but also failed attempts, feedback signals, and intermediate reasoning traces (Zhao et al., 2024). Rather than serving as passive records of past interactions, these trajectories can provide valuable experience for refining future decisions, improving task strategies, and enabling test-time adaptation (Wei et al., 2025; Suzgun et al., 2025; Zhou et al., 2025). Such sequentially evolving systems are central to many emerging applications, including reasoning assistants (Ho et al., 2025), tool-use agents (Wang et al., 2025b), and interactive decision-making systems (Zheng et al., 2025; Agrawal et al., 2025), where performance depends not only on the current input but also on previously encountered tasks. Wei et al. (2025) formalize this setting by viewing memory as an evolving state that is *retrieved*, *synthesized*, and *updated* throughout a task sequence. This perspective marks an important step beyond static conversational recall, highlighting the role of memory in test-time adaptation and experience reuse.

Despite this progress, the evaluation of sequentially evolving LLM memory remains incomplete. Existing studies typically assess memory methods using aggregate performance metrics, such as final hold-out accuracy after memory construction (Zhao et al., 2024) or cumulative online accuracy

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.

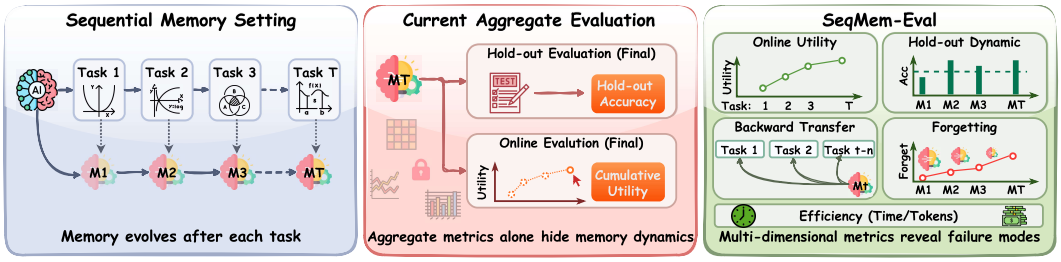


Figure 1: **SEQMEM-EVAL: Beyond aggregate evaluation of LLM memory.** Left: In sequential settings, an LLM processes a stream of tasks while maintaining an evolving memory state. Middle: Existing evaluations reduce memory performance to aggregate metrics, which collapses complex memory dynamics and hides important behaviors. Right: SEQMEM-EVAL decomposes memory quality into multiple dimensions, including online utility, hold-out generalization, backward transfer, forgetting, and efficiency, enabling fine-grained analysis of how memory evolves.

along the sequence (Wei et al., 2025; Suzgun et al., 2025). While useful, these metrics collapse the behavior of an evolving memory system into a single number. As a result, they do not reveal whether memory updates genuinely improve future behavior, whether later experiences help consolidate earlier ones, or whether the system forgets knowledge that was previously useful. In practice, similar final or average accuracy may mask fundamentally different learning dynamics: one method may steadily accumulate reusable knowledge, while another may exhibit oscillatory behavior or transient improvements followed by degradation. Thus, aggregate metrics can create an illusion of comparable memory quality, even when the underlying memory dynamics are substantially different.

In this paper, we propose SEQMEM-EVAL, a diagnostic evaluation framework for sequentially evolving LLM memory. While motivated by continual learning (Wu et al., 2022; Lopez-Paz & Ranzato, 2017), sequentially evolving LLM memory is a distinct test-time setting: the LLM remains fixed, and adaptation occurs through updates to an external textual memory that influences predictions via retrieval and context construction. Therefore, evaluation should focus not only on final task performance, but also on how these memory updates affect predictions throughout the sequence. SEQMEM-EVAL captures this behavior through five complementary dimensions: *online utility*, *hold-out generalization*, *backward transfer*, *forgetting*, and *efficiency*. Together, these diagnostics reveal whether memory updates are useful, transferable, stable, and cost-effective, rather than merely improving aggregate accuracy.

We conduct a systematic empirical study across diverse tasks, models, and representative memory methods under the SEQMEM-EVAL protocol. Our results show that standard aggregate metrics can be misleading: methods with strong final or online accuracy may still exhibit substantial forgetting, limited backward transfer, or weak generalization from accumulated experience. These findings suggest that current evaluation practices can overestimate the effectiveness of LLM memory and obscure important failure modes. Our contributions are summarized as follows:

- **Diagnostic evaluation framework.** We introduce SEQMEM-EVAL, a continual-learning-inspired framework for evaluating sequentially evolving LLM memory beyond aggregate accuracy.
- **Comprehensive empirical study.** We provide a systematic comparison of representative memory methods across diverse tasks and models under a unified sequential evaluation protocol.
- **Actionable findings for memory design.** We identify key failure modes, including forgetting, limited backward transfer, and weak generalization, offering design implications for more reliable memory-augmented LLMs.

2 SEQMEM-EVAL: A DIAGNOSTIC EVALUATION FRAMEWORK

We propose SEQMEM-EVAL, a diagnostic framework for evaluating sequentially evolving LLM memory. Unlike classical continual learning, this setting keeps LLMs fixed and changes behavior through external memory updates, retrieval, and prompt construction. Thus, evaluation should go beyond endpoint performance to measure whether memory is useful online, generalizes to unseen tasks, consolidates past experience, preserves acquired utility, and remains computationally efficient.

Table 1: Overview of SEQMEM-EVAL diagnostics. The metrics cover five complementary perspectives of sequential memory evaluation, with formal definitions provided in the following subsections.

Perspective	Metric	What it measures
Online Utility	OnlineAcc	Final cumulative online performance.
	PED	Peak-to-end degradation in the online trajectory.
	MER	Recovery from the worst online phase.
	$r_{\min}$	Timing of the lowest online performance.
Hold-out Generalization	HoldOutAcc	Final performance on unseen examples.
	$\mathrm{Trend}_{\mathrm{HO}}$	Direction of hold-out performance over memory evolution.
Backward Transfer	$\mathrm{BWT}(t)$	Effect of later memory updates on earlier examples.
	IV	Immediate reusability of the newly updated memory.
Forgetting	$\mathrm{F}(t)$	Loss of previously achieved capability over time.
Efficiency	Token Consumption	Total token cost of sequential evaluation.
	Runtime	End-to-end wall-clock time.

2.1 SEQUENTIAL MEMORY EVALUATION SETTING

Following Wei et al. (2025), we consider a sequentially evolving memory setting, where an LLM $\mathcal{L}$ interacts with a stream of tasks and maintains an external memory that is updated over time. Let $\mathcal{D} = \{(x_t, y_t)\}_{t=1}^{T}$ denote a task sequence, where x_t is the input at step t and y_t is the corresponding target. The memory state M_t is indexed before task t, after the first $t-1$ tasks. It may contain trajectories, examples, summaries, workflows, or reflections; M_1 is the initial state and M_{T+1} the final state. Given input x_t, the system retrieves or constructs a context C_t from the current memory state M_t, and the LLM produces a prediction $\hat{y}_t = \mathcal{L}(x_t, C_t)$. After prediction, the system may receive feedback f_t, such as correctness signals, execution results, or environment feedback. The memory is then updated by a method-specific update function:

$$M_{t+1} = \texttt{Update}(M_t, x_t, \hat{y}_t, f_t; \mathcal{L}), \tag{1}$$

where $\mathcal{L}$ is included because some memory methods use the LLM itself to generate, refine, compress, or reorganize memory entries. For methods that do not rely on LLM-based memory updates, this argument can be omitted, and `Update` reduces to a non-parametric operation, such as appending the current trajectory or updating a retrieval index.

This formulation abstracts a broad class of sequential memory methods, including retrieval-based memory, summarization-based memory, workflow memory, and reflection-based memory. We define diagnostic metrics over the evolving memory states $\{M_t\}_{t=1}^{T+1}$ to evaluate different aspects of memory behavior. Before introducing the formal definitions in the following subsections, Table 1 provides an overview of the SEQMEM-EVAL metrics and the memory behavior captured by each metric.

2.2 ONLINE UTILITY

Online utility reflects how well the model performs as it progressively updates its memory over the task sequence. Rather than reducing online performance to a single scalar, we treat it as a trajectory that captures how the model's behavior evolves over time.

Formally, let $\mathcal{D} = \{(x_\tau, y_\tau)\}_{\tau=1}^{T}$ denote the online task sequence. For each step τ, we record the per-step online performance and the cumulative online accuracy as

$$A(\tau) = \mathrm{Acc}(x_\tau; M_\tau), \quad \overline{A}(\tau) = \frac{1}{\tau}\sum_{i=1}^{\tau} A(i), \tag{2}$$

where $\mathrm{Acc}(x; M) \in \{0, 1\}$ indicates whether the model correctly answers x using memory state M.

We report the following online utility metrics:

• **Online Accuracy (OnlineAcc).** The final value of the cumulative online accuracy curve:

$$\mathrm{OnlineAcc} = \overline{A}(T). \tag{3}$$

Table 2: Qualitative interpretation of online trajectory diagnostics. MER and PED provide coarse indicators of desirable or undesirable online dynamics, while $r_{\min}$ further refines the interpretation by distinguishing different improvement patterns.

Trajectory pattern	MER	PED	$r_{\min}$	Preference (Interpretation)
Gradual improvement	High	Low / 0	Early	✓ Preferred (Effective accumulation)
Drop-then-recover	High	Low / 0	Middle	✓ Acceptable (Delayed recovery)
Early peak then degradation	Low / 0	High	Late	✗ Undesirable (Unstable evolution)
Rapid drop then stabilization	Low / 0	High	Late	✗ Undesirable (Persistent degradation)
Stable but non-improving	Low	Low	Arbitrary	● Mixed (Limited memory effect)
Highly fluctuating	High	High	Arbitrary	● Mixed (Volatile memory dynamics)

While OnlineAcc summarizes endpoint performance, it does not capture how the trajectory reaches that endpoint. We further derive three trajectory-level diagnostics from the cumulative curve.

• **Peak-to-End Drop (PED).** The drop from the best cumulative online performance to the final value:

$$\mathrm{PED} = \max_{\tau \in [1,T]} \overline{A}(\tau) - \overline{A}(T). \tag{4}$$

A larger PED indicates that the method reaches strong intermediate performance but later degrades.

• **Minimum-to-End Recovery (MER).** The recovery from the worst cumulative online performance to the final value:

$$\mathrm{MER} = \overline{A}(T) - \min_{\tau \in [1,T]} \overline{A}(\tau). \tag{5}$$

A larger MER indicates that the method improves after a low-performance phase.

• **Extremum Timing $r_{\min}$ [Optional].** The relative location of the minimum cumulative online accuracy:

$$r_{\min} = \frac{1}{T} \arg\min_{\tau \in [1,T]} \overline{A}(\tau). \tag{6}$$

This secondary diagnostic helps distinguish trajectory shapes with similar MER and PED, such as gradual improvement versus delayed recovery.

Overall, these diagnostics complement OnlineAcc by characterizing whether the online trajectory reflects improvement, recovery, degradation, or volatile memory dynamics. Table 2 summarizes the qualitative interpretation of these trajectory patterns. In particular, the two *Mixed* cases reflect different behaviors. Low MER and low PED indicate limited trajectory variation and may correspond to either consistently high or consistently low performance, which can be distinguished using OnlineAcc. In contrast, high MER and high PED indicate that substantial recovery and substantial degradation coexist, characterizing volatile memory dynamics.

2.3 Hold-out Generalization

Online utility measures performance within the observed task stream, but strong online performance does not necessarily imply that memory has learned reusable knowledge. To evaluate this aspect, we measure *hold-out generalization* on a fixed set of unseen tasks under different memory states.

Let $\mathcal{D}_{\text{test}} = \{x_i^{\text{test}}\}_{i=1}^{M}$ denote a hold-out test set that is never used for memory updates. After τ online tasks ($1 \le \tau \le T$), the corresponding hold-out checkpoint uses $M_{\tau+1}$.

• **Hold-out Accuracy (HoldOutAcc).** We define the hold-out accuracy trajectory and its final value as

$$H(\tau) = \frac{1}{M} \sum_{i=1}^{M} \mathrm{Acc}(x_i^{\text{test}}; M_{\tau+1}), \quad \mathrm{HoldOutAcc} = H(T). \tag{7}$$

The final value $\mathrm{HoldOutAcc} = H(T)$ evaluates M_{T+1}, after the full online sequence.

• **Hold-out Trend ($\mathrm{Trend}_{\mathrm{HO}}$).** To capture the overall direction of memory generalization, we compute the slope of the least-squares line fitted to the normalized hold-out trajectory:

$$\mathrm{Trend}_{\mathrm{HO}} = \mathrm{slope}\left(\{(\tau/T, H(\tau))\}_{\tau=1}^{T}\right). \tag{8}$$

A positive $\mathrm{Trend}_{\mathrm{HO}}$ indicates that hold-out performance generally improves as more experiences are incorporated into memory, while a flat or negative value suggests limited generalization benefit or possible degradation.

2.4 BACKWARD TRANSFER

Hold-out generalization evaluates whether memory improves performance on unseen tasks. We next consider the complementary question: how later memory updates affect previously encountered tasks. This is related to *backward transfer* in continual learning (Lopez-Paz & Ranzato, 2017), but here it is defined over evolving external memory states rather than model parameters.

• **Backward Transfer (BWT).** For an integer horizon $1 \le t \le T$, we define

$$\mathrm{BWT}(t) = \frac{1}{T-t+1} \sum_{\tau=1}^{T-t+1} \left[\mathrm{Acc}(x_\tau; M_{\tau+t}) - \mathrm{Acc}(x_\tau; M_\tau) \right], \tag{9}$$

where M_τ precedes task x_τ, and $M_{\tau+t}$ follows its update and the next $t-1$ updates. The average covers all $T-t+1$ valid starting tasks. A positive $\mathrm{BWT}(t)$ indicates that later memory updates improve performance on earlier tasks, a value close to zero suggests limited retrospective effect, and a negative value indicates negative backward transfer.

• **Immediate Validity (IV).** We define *Immediate Validity* as the one-step backward transfer:

$$\mathrm{IV} = \mathrm{BWT}(1) = \frac{1}{T} \sum_{\tau=1}^{T} \left[\mathrm{Acc}(x_\tau; M_{\tau+1}) - \mathrm{Acc}(x_\tau; M_\tau) \right]. \tag{10}$$

IV evaluates whether the memory update induced by the current task immediately makes that experience reusable. A positive IV suggests that the update effectively incorporates the new experience, while a value close to zero or below indicates that the update provides limited reusable benefit.

2.5 FORGETTING

BWT measures the net effect of memory updates on previously attempted tasks, but it does not capture whether a capability that was temporarily acquired is later lost. We therefore measure *forgetting*, which evaluates whether previously achieved performance is retained over time.

• **Forgetting.** For an integer horizon $1 \le t \le T$, we define forgetting as

$$\mathrm{F}(t) = \frac{1}{T-t+1} \sum_{\tau=1}^{T-t+1} \left[\max_{\tau' \in [\tau,\, \tau+t]} \mathrm{Acc}(x_\tau; M_{\tau'}) - \mathrm{Acc}(x_\tau; M_{\tau+t}) \right], \tag{11}$$

where the maximum includes the pre-update state M_τ, and the endpoint $M_{\tau+t}$ follows t updates, including the update induced by x_τ. A larger $\mathrm{F}(t)$ indicates stronger loss of previously acquired capability, while values close to zero suggest stable retention.

Computing $\mathrm{F}(t)$ exactly requires evaluating each past task under all intermediate memory states, which can be expensive for long sequences. In practice, we use a checkpoint-based approximation that reuses the evaluations performed for backward transfer. Let $\mathcal{T} \subseteq \{1, \ldots, T\}$ denote the evaluated horizons, including 1 for IV. For each valid starting task x_τ and $t \in \mathcal{T}$, we approximate

$$\max_{\tau' \in [\tau,\, \tau+t]} \mathrm{Acc}(x_\tau; M_{\tau'}) \approx \max_{t_i \in \{0\} \cup \mathcal{T},\, t_i \le t} \mathrm{Acc}(x_\tau; M_{\tau+t_i}). \tag{12}$$

The zero-lag value is the initial online score $A(\tau)$; including both lag 0 and the endpoint t retains both window boundaries and ensures nonnegative forgetting. Together, BWT and forgetting provide complementary views: BWT measures net retrospective transfer, whereas forgetting measures degradation from the best previously achieved performance.

2.6 EFFICIENCY

Memory methods may improve performance at the cost of longer contexts, more stored experiences, or additional LLM calls. We therefore evaluate efficiency using two metrics: **Token Consumption**, the total tokens used during sequential evaluation, including memory operations and intermediate outputs; and **Runtime**, the wall-clock time for the full task sequence, including LLM calls and memory-related overhead. Together, they quantify the computational cost of performance gains.

3 EXPERIMENTS AND ANALYSIS

We evaluate sequentially evolving LLM memory under the proposed SEQMEM-EVAL framework across diverse tasks, models, and representative memory methods. We provide the detailed experimental setup in Appendix C. Rather than only comparing final accuracy, we organize our analysis around the following research questions (RQs): **RQ1: Online Utility.** How does memory affect performance during sequential inference, and what online trajectory patterns emerge as memory evolves? **RQ2: Hold-out Generalization.** Does the evolving memory improve performance on unseen tasks, including in-distribution and out-of-distribution hold-out sets? **RQ3: Backward Transfer.** Do later memory updates help consolidate knowledge for previously encountered tasks, or do they induce negative transfer? **RQ4: Forgetting.** Does the memory retain capabilities achieved during the sequence, or does performance on earlier tasks degrade over time? **RQ5: Efficiency.** What are the efficiency trade-offs of different memory mechanisms in terms of computational cost? Additional metric formulations, redundancy checks, and sensitivity analyses are provided in Appendix B.7.

3.1 ANALYSIS OF ONLINE UTILITY (RQ1)

FINDING 1. Memory improves aggregate online accuracy, but the gains do not imply stable accumulation. Table 3 summarizes the standard aggregate view of online performance. Under this view, memory-augmented methods often appear beneficial: most methods improve over the memory-free baseline across datasets, and stronger memory mechanisms such as G-Memory and ExpeL-MT achieve clear gains across both LLM backbones. However, the cumulative trajectories in Figure 2 (top) reveal a more nuanced picture. Monotonic improvement is largely absent: across HumanEval and MMLU-Pro-Engineering, many methods either decline after an initially strong phase or rapidly drop and then plateau. This suggests that additional memory updates do not consistently translate into sustained experience accumulation. The complete online trajectories and the corresponding MiniMax-M2.7 results are provided in Appendix B.2.

Table 3: Average $\mathrm{OnlineAcc}$ (%). Avg-5 excludes ALFWorld; ALFWorld is reported separately due to method applicability.

Method	Qwen3-8B		MiniMax-M2.7	
	Avg-5	**ALF**	**Avg-5**	**ALF**
Memory-free	66.2	52.9	72.8	68.6
$\mathrm{ExpRec}_{k=10}$	68.3	–	75.0	–
$\mathrm{ExpRec}_{k=3}$	68.1	–	76.3	–
$\mathrm{ExpRAG}_{k=3}$	68.8	–	75.7	–
DC-RS	68.2	–	75.0	–
AWM	68.9	51.4	77.7	72.1
G-Memory	71.6	62.9	82.0	83.6
ExpeL-ST	70.4	59.3	78.2	70.0
ExpeL-MT	**80.0**	**79.3**	**86.5**	**88.6**

"–" denotes not applicable.

FINDING 2. Methods with similar online accuracy can exhibit substantially different memory dynamics. Figure 2 (bottom) plots the PED–MER diagnostics for the same datasets. The scatter plots show that methods with comparable final ONLINEACC can occupy different regions in the diagnostic space. On HumanEval, several methods have non-trivial PED but small MER, corresponding to early high performance followed by degradation. On ALFWorld, the separation is more pronounced: ExpeL-MT achieves high MER with relatively lower PED compared with other agentic methods, while AWM-Online suffers from large PED and near-zero MER, indicating severe peak-to-end degradation without recovery. We further analyze the extremum timing metric $r_{\min}$ in Appendix B.2.

Overall, **RQ1** shows that aggregate OnlineAcc provides only a coarse endpoint measure. Although memory methods often improve online performance on average, their trajectories reveal whether these gains are stable, recoverable, or transient. This supports the need to evaluate online utility as an evolving trajectory rather than a single aggregate score.

3.2 ANALYSIS OF HOLD-OUT GENERALIZATION (RQ2)

FINDING 3. Online gains do not reliably translate to hold-out generalization. Table 4 shows that final in-distribution (ID) hold-out generalization is often weak, even when memory methods improve online accuracy. Across several datasets, many methods only match or fall below the memory-free baseline on the final hold-out set. For example, on APIBench, most memory methods underperform the memory-free baseline in final hold-out accuracy, despite showing gains in aggregate online

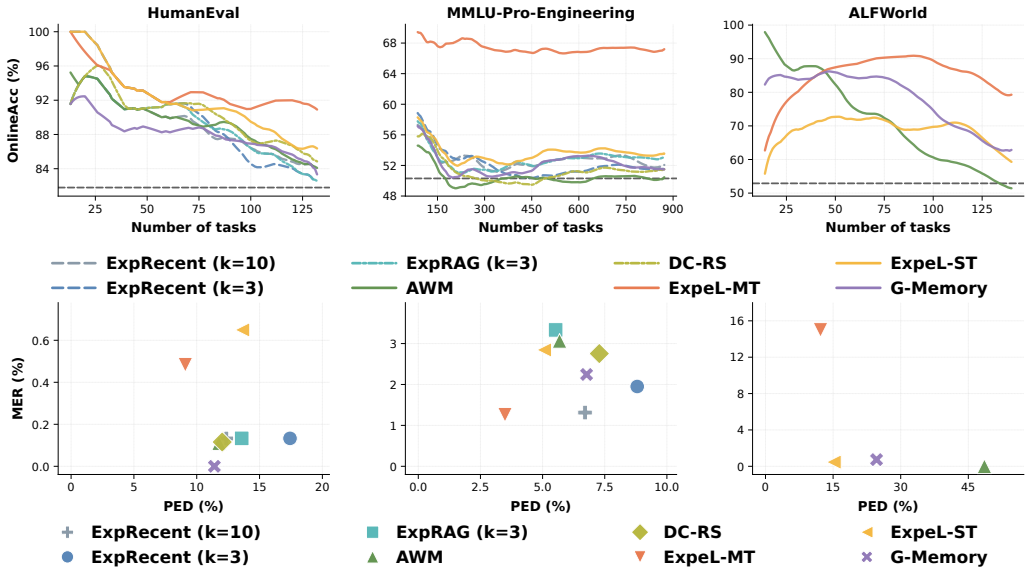


Figure 2: Online utility analysis for Qwen3-8B on three representative datasets. Top row: cumulative online accuracy over the task stream; the dashed horizontal line denotes the memory-free baseline. Bottom row: PED–MER diagnostics computed from the corresponding cumulative trajectories.

Table 4: In-distribution hold-out generalization on Qwen3-8B.We report the final $\mathrm{HoldOutAcc}$ using the final memory state and the hold-out trajectory trend $\mathrm{Trend}_{\mathrm{HO}}$. Higher values are better.

Method	HumanEval		MATH500		APIBench		MMLU-Eng		MMLU-Phys		ALFWorld	
	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$	HO-Acc	$\mathrm{Trend}_{\mathrm{HO}}$
Memory-free	75.0	–	67.5	–	67.8	–	50.0	–	69.0	–	75.0	–
$\mathrm{ExpRec}_{k=10}$	78.1	-0.001	**71.4**	**+0.030**	54.2	-0.111	53.1	+0.039	66.7	+0.004	–	–
$\mathrm{ExpRec}_{k=3}$	78.1	-0.017	70.4	-0.003	67.5	-0.145	55.2	+0.011	69.8	+0.028	–	–
ExpRAG	68.8	-0.107	67.5	-0.023	61.7	-0.014	53.1	-0.014	65.9	+0.004	–	–
DC-RS	75.0	**+0.068**	61.8	-0.010	67.5	+0.015	51.0	+0.006	65.9	-0.004	–	–
AWM	71.9	-0.012	67.5	+0.012	65.0	-0.053	53.1	**+0.089**	67.4	-0.008	58.3	+0.032
ExpeL-ST	**78.1**	-0.012	70.0	-0.014	66.7	-0.054	**58.3**	+0.066	65.9	-0.004	58.3	+0.088
ExpeL-MT	75.0	-0.005	67.5	+0.009	**73.3**	**+0.072**	54.2	-0.095	**72.1**	**+0.057**	62.5	**+0.220**
G-Memory	75.0	-0.034	68.6	-0.006	67.5	-0.064	57.3	+0.030	68.2	-0.002	**70.8**	-0.090

performance. This gap suggests that memory updates can improve performance on the observed stream without necessarily learning reusable knowledge that transfers to unseen examples.

Finding 4: Final hold-out accuracy can hide unstable generalization dynamics. Table 4 reports final $\mathrm{HoldOutAcc}$ together with $\mathrm{Trend}_{\mathrm{HO}}$. The former summarizes aggregate generalization under the final memory state, whereas the latter characterizes how hold-out performance evolves throughout memory construction. Thus, methods with similar final hold-out performance may still follow different memory-evolution trajectories. For example, on APIBench, ExpRecent ($k = 3$) and DC-RS have $\mathrm{Trend}_{\mathrm{HO}}$ values of -0.145 and $+0.015$, respectively, yet both achieve a final $\mathrm{HoldOutAcc}$ of 67.5%. This contrast highlights differences in generalization dynamics that are not captured by final hold-out accuracy alone. In Appendix B.4, we further analysis whether the observed trajectory differences between methods exceed the uncertainty induced by the finite hold-out set.

The OOD hold-out results in Appendix B.5 show a similar but stronger pattern, where cross-task generalization is substantially weaker than ID generalization and memory updates can even hurt performance compared with the memory-free baseline. Overall, **RQ2** shows that hold-out generalization should be analyzed through both endpoint performance and trajectory trends. These results highlight the need for memory methods to validate update quality, preserve useful intermediate states, and prevent harmful updates.

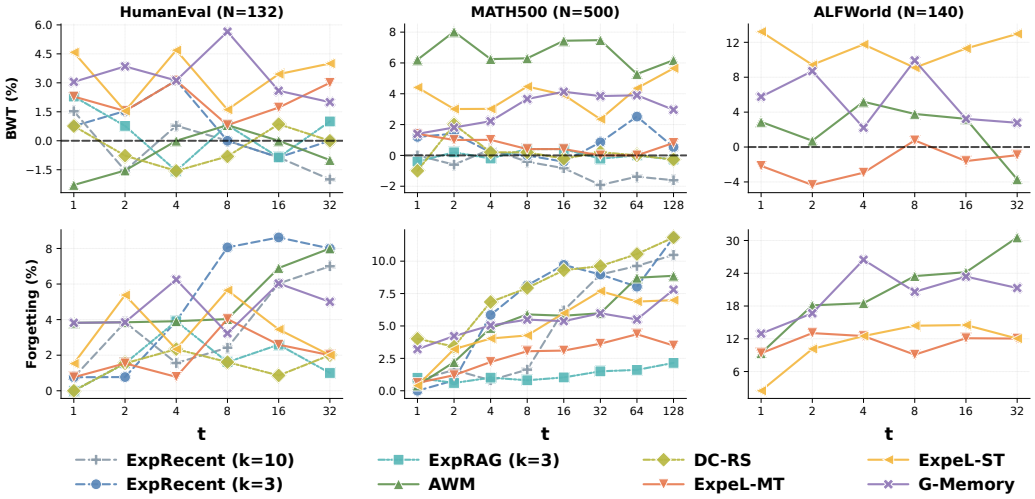


Figure 3: Backward transfer (top row) and forgetting (bottom row) for Qwen3-8B across different temporal horizons t. Higher BWT is better, while lower forgetting indicates better retention.

3.3 Analysis of Backward Transfer (RQ3)

Finding 5. Later memory updates provide short-term benefits, but do not reliably consolidate transferable knowledge. As in Figure 3 (top row), BWT is often positive at short horizons but fails to consistently strengthen as the horizon t grows, instead fluctuating or diminishing. This indicates that while recent memory updates effectively improve earlier tasks, later updates fail to reliably accumulate into increasingly useful retrospective knowledge. This overall pattern arises from different failure modes across memory designs. Retrieval-based methods reveal a recency–retention trade-off: ExpRecent can exhibit negative BWT once past tasks fall outside its recent window, while ExpRAG is more stable but often remains near zero, indicating limited consolidation beyond instance retrieval. More structured memory methods, such as G-Memory, sometimes achieve stronger positive BWT, but their curves are still highly non-monotonic. These results suggest that current memory methods can reuse nearby or directly relevant experiences, but often fail to transform longer streams of interaction into stable transferable knowledge.

Finding 6. Immediate validity is often weak even right after a memory update. The case $t = 1$ corresponds to immediate validity, $\mathrm{IV} = \mathrm{BWT}(1)$. It can be decomposed into the fraction of predictions repaired from incorrect to correct, $P(✗ \to ✓)$, minus the fraction that regress from correct to incorrect, $P(✓ \to ✗)$. Its maximum possible immediate improvement is the pre-update error rate, $\mathrm{IV}_{\max} = 1 - \mathrm{Acc}_{\mathrm{pre}}$. An oracle reference achieves this ceiling by storing the exact problem-solution pair upon feedback and retrieving it on re-evaluation. Across our evaluated configurations, the average attainable ceiling is 24.2%. In contrast, many existing methods achieve net immediate validity below 5%, with some even exhibiting negative values. Even the best configuration realizes only approximately 55% of its own ceiling. This large gap shows that current memory methods convert only a limited fraction of available feedback into immediately reusable capability.

Overall, **RQ3** reveals a key limitation of existing memory mechanisms: they can often absorb local feedback, but they do not reliably consolidate it into transferable knowledge that benefits earlier tasks over longer horizons. This highlights the need for memory updates that go beyond storage and perform more effective reflection, abstraction, and validation.

3.4 Analysis of Forgetting (RQ4)

Finding 7. Sequential memory updates often interfere with previously acquired utility. The bottom row of Figure 3 shows that forgetting generally increases with the horizon t. This indicates that performance previously achieved on earlier tasks is often not retained after additional memory updates. As more experiences are incorporated, memory states may overwrite useful information,

dilute earlier knowledge, or retrieve less relevant content for past tasks. The underlying interference mechanism is method-dependent due to diverse update and retention strategies. Taking ExpeL-MT as an example, its bounded insight pool exhibits a trade-off between rule eviction and retrieval effectiveness. We provide detailed eviction logs, traced failure cases, and a controlled capacity ablation in Appendix G.

Together with the BWT results, **RQ4** reveals a central tension in sequential LLM memory: methods can often incorporate recent feedback and sometimes improve past performance, but they struggle to preserve these gains over longer horizons. Future memory mechanisms should therefore not only add or summarize new experiences, but also preserve useful knowledge, validate updates, and prevent interference between old and new memory content.

3.5 Analysis of Efficiency (RQ5)

Finding 8. Stronger memory performance often comes with substantial token and runtime overhead. Figure 4 summarizes the efficiency–performance trade-off on Qwen3-8B. Most memory-augmented methods incur clear overhead over the memory-free baseline. Even lightweight retrieval methods such as ExpRecent and ExpRAG require about 4.3–$4.7\times$ more tokens while providing only modest gains. More complex methods often achieve stronger performance but at substantially higher cost: ExpeL-MT obtains the best Avg-5 accuracy with $7.4\times$ token usage and the largest runtime increase, while G-Memory uses around $13\times$ more tokens. DC-RS is especially costly, requiring about $3.2\times$ runtime and $36\times$ token usage, yet performs close to much cheaper methods.

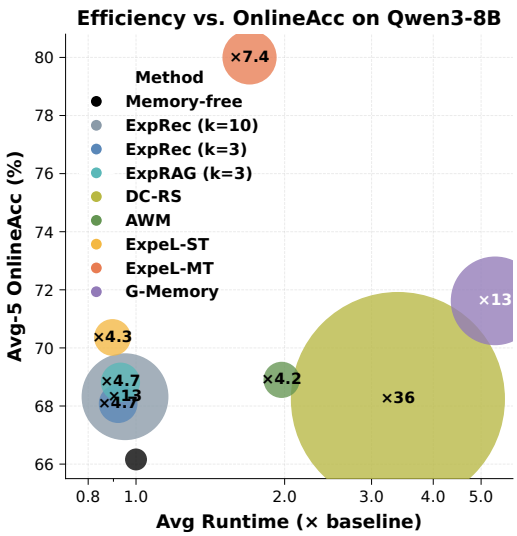


Figure 4: Efficiency–performance trade-off on Qwen3-8B. Bubble labels indicate total token usage normalized by the memory-free baseline.

Overall, RQ5 shows that memory effectiveness should be interpreted together with computational cost. Strong performance can reflect substantially higher token and latency budgets, rather than more efficient memory utilization. Detailed runtime and token breakdowns are provided in Appendix B.6.

4 Conclusion

Sequentially evolving LLM memory is an important step toward agents that can accumulate experience and adapt beyond independent task solving. However, existing evaluations often reduce memory quality to a single aggregate score. Our study showed that this view is incomplete: similar aggregate performance can hide substantially different memory dynamics, like unstable online trajectories. We introduced SeqMem-Eval to diagnose sequential memory behavior through five complementary dimensions: online utility, hold-out generalization, backward transfer, forgetting, and efficiency. Across diverse tasks and memory methods, our analysis showed that these diagnostics provide a more complete view of memory behavior than aggregate scores alone. The resulting observations also offer useful insights for future memory research, highlighting the need to design and evaluate memory mechanisms in terms of generalization, transfer, retention, and efficiency. Overall, SeqMem-Eval reframes LLM memory evaluation as the analysis of an evolving process rather than a final endpoint. We hope this perspective helps guide the development of more reliable, generalizable, and efficient memory-augmented LLM agents.

AI USE STATEMENT

We used generative AI tools to assist with manuscript writing, the implementation of some baseline methods, and literature search and review. The authors take full responsibility for the accuracy and integrity of the final manuscript, code, and reported results.

ETHICS STATEMENT

This work evaluates sequential LLM memory using existing, publicly available benchmarks. Memory-augmented systems may retain sensitive information or propagate biases in real-world applications. Our evaluation focuses on memory utility and dynamics and should not be interpreted as establishing the safety or fairness of these systems.

REPRODUCIBILITY STATEMENT

The evaluation protocol and metric definitions are provided in the main text. Dataset splits, model settings, retrieval configurations, and method-specific hyperparameters are detailed in Appendix C. Algorithmic descriptions and prompts are provided in Appendices E and F, respectively. Additional experimental results and robustness analyses are included in the appendix. Our code is available anonymously at `https://anonymous.4open.science/r/SeqMem-Eval/`.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-Jussa. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th international conference on computational linguistics*, pp. 6523–6541, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

Silin Chen, Shaoxin Lin, Yuling Shi, Heng Lian, Xiaodong Gu, Longfei Yun, Dong Chen, Lin Cao, Jiyang Liu, Nu Xia, et al. Swe-exp: Experience-driven software issue resolution. *arXiv preprint arXiv:2507.23361*, 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.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025a.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025b.

Erhu Feng, Wenbo Zhou, Zibin Liu, Le Chen, Yunpeng Dong, Cheng Zhang, Yisheng Zhao, Dong Du, Zhichao Hua, Yubin Xia, et al. Get experience from practice: Llm agents with record & replay. *arXiv preprint arXiv:2505.17716*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Matthew Ho, Chen Si, Zhaoxiang Feng, Fangxu Yu, Yichi Yang, Zhijian Liu, Zhiting Hu, and Lianhui Qin. Arcmemo: Abstract reasoning composition with lifelong llm memory. *arXiv preprint arXiv:2509.04439*, 2025.

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*, 114 (13):3521–3526, 2017.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, et al. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*, 2025.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, et al. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470, 2025.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

MiniMax. MiniMax M2.7: Early echoes of self-evolution. https://www.minimax.io/news/minimax-m27-en, 2026. Accessed: 2026-05-04.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis, 2023. URL https://arxiv.org/abs/2305.15334.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL https://arxiv.org/abs/2010.03768.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025. URL https://arxiv.org/abs/2504.07952.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jiayin Wang, Zhiqiang Guo, Weizhi Ma, and Min Zhang. How far can llms improve from experience? measuring test-time learning ability in llms with human comparison. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25688–25702, 2025a.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025b.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024a.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024c. URL https://arxiv.org/abs/2409.07429.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan Fang Li, Guilin Qi, and Gholamreza Haffari. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations 2022*. OpenReview, 2022.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Zhishang Xiang, Chengyi Yang, Zerui Chen, Zhimin Wei, Yunbo Tang, Zongpei Teng, Zexi Peng, Zongxia Li, Chengsong Huang, Yicheng He, et al. A systematic survey of self-evolving agents: From model-centric to environment-driven co-evolution. 2026.

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.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems, 2025. URL https://arxiv.org/abs/2506.07398.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*, 2023.

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.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

A Related Work

Sequentially evolving LLM memory and evaluation. Memory has become a central mechanism for enabling LLM agents to move beyond isolated inputs and adapt over sequential interactions (Xiang et al., 2026; Fang et al., 2025a; Madaan et al., 2023; Chhikara et al., 2025). Recent memory-augmented agents extract reusable information from prior trajectories, feedback, reflections, or workflows, and use such information to improve future reasoning and decision-making (Wang et al., 2025a; Chen et al., 2025; Fang et al., 2025b; Zhong et al., 2024; Xu et al., 2025). Representative approaches include reflection- or experience-based methods such as ExpeL (Zhao et al., 2024), dynamic memory construction methods such as Dynamic Cheatsheet (Suzgun et al., 2025), workflow-level memory methods such as Agent Workflow Memory (AWM) (Wang et al., 2024c), and retrieval- or structure-based memory systems such as G-Memory (Zhang et al., 2025) and Memento (Zhou et al., 2025). These methods differ in how they store, retrieve, and update memory, but share the goal of improving future behavior through test-time experience reuse (Tang et al., 2025; Feng et al., 2025; Ho et al., 2025). Recent benchmark efforts further formalize this setting: for example, Evo-Memory (Wei et al., 2025) converts static datasets into sequential task streams and evaluates agents whose memory is searched, synthesized, and evolved after each interaction. Broader works of self-evolving agents also frame memory and trajectory reuse as part of environment-centric self-evolution Xiang et al. (2026); Gao et al. (2025). However, existing evaluations still largely rely on aggregate metrics such as final hold-out accuracy, cumulative online accuracy, or average success rate (Ouyang et al., 2025; Wu et al., 2025). Such metrics are useful for comparing end performance, but they provide limited insight into memory dynamics, like whether a method retains useful information. Our work complements prior memory methods and benchmarks by focusing on diagnostic evaluation rather than proposing another memory architecture.

Continual learning evaluation and diagnostic metrics. Our evaluation perspective is inspired by continual learning, where models learn from a sequence of tasks while attempting to acquire new knowledge without forgetting old knowledge (Biesialska et al., 2020; Kirkpatrick et al., 2017; Wang et al., 2024a). Classical continual learning evaluation uses metrics such as backward transfer, forward transfer, and stability–plasticity trade-offs to characterize learning dynamics beyond final accuracy (Chaudhry et al., 2018; 2019; Lopez-Paz & Ranzato, 2017; Wu et al., 2022; Qi et al., 2023; Wang et al., 2023). Recent works on continual learning for LLMs further highlight the importance of updating large models with new knowledge and skills while preserving previous capabilities (Shi et al., 2025; Wu et al., 2024). Sequential LLM memory shares this temporal structure, but differs in a crucial way: most memory-augmented agents do not update model parameters, and instead rely on external, prompt-mediated, or retrieval-based memory states (Zheng et al., 2023; Liang et al., 2025; Li et al., 2025; Wei et al., 2025). Therefore, continual learning metrics cannot be directly reused

without adaptation. We adapt their diagnostic principles to sequential LLM memory by defining online utility, hold-out generalization, backward transfer, and forgetting over evolving memory states, enabling fine-grained analysis of memory behavior beyond aggregate performance.

B Additional Experimental Results

B.1 Overall Comparison

To provide a holistic view of method behavior, Figure 5 summarizes representative memory mechanisms across the five diagnostic dimensions of SeqMem-Eval. For each dimension, we normalize the underlying metrics, average them into a dimension-level score, and rank methods accordingly. The radar plot reports these relative ranks to provide a compact profile of method trade-offs.

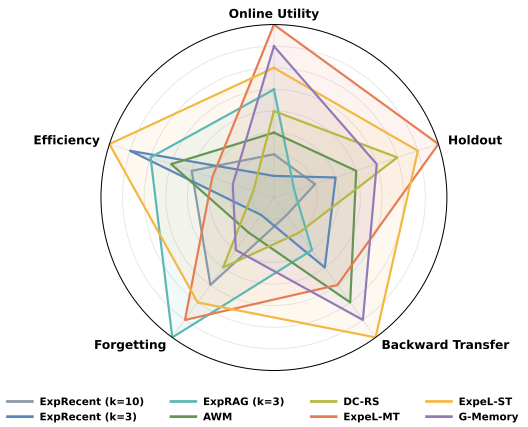


Figure 5: Overall radar comparison on Qwen3-8B across multiple dimensions. Larger radial values indicate better within-dimension rankings.

The radar profiles reveal that different algorithmic designs lead to distinct trade-offs. ExpeL-MT performs strongly on online utility and hold-out generalization, which is consistent with its use of multiple attempts and reflection-based experience extraction: repeated trials increase the chance of solving the current task, while distilled experiential rules can provide reusable signals for later tasks. However, the same design also introduces substantial generation and update overhead, explaining its weaker efficiency profile. ExpeL-ST removes the multi-try component and therefore offers a different trade-off: it retains some benefits of reflection-based memory while reducing the cost associated with repeated inference, suggesting that experience abstraction can be useful even without aggressive inference-time search.

Retrieval-based methods exhibit the opposite behavior. ExpRecent and ExpRAG are comparatively efficient because they mainly reuse stored examples through recent-window or similarity-based retrieval, without expensive memory synthesis. However, their weaker profiles on generalization and transfer suggest that instance-level retrieval alone provides limited abstraction: it can reuse nearby or similar experiences, but does not reliably consolidate them into higher-level knowledge that transfers across time or tasks. This observation is consistent with the BWT analysis, where retrieval-based methods show limited consolidation beyond local reuse.

Structured memory methods occupy an intermediate regime. G-Memory achieves a relatively balanced profile, especially on transfer- and retention-related dimensions, suggesting that organizing experiences into a more structured memory can help preserve and reuse information beyond raw instance retrieval. At the same time, its efficiency is constrained by graph construction, insight maintenance, and additional retrieval operations. DC-RS shows another form of trade-off. Its retrieve–update–solve pipeline can help the current task by reusing retrieved past history, but its fallback behavior may also weaken memory abstraction: when a new cheatsheet cannot be extracted, the retrieved history is used directly as the updated memory. This makes the memory useful as an ICL-style context for nearby tasks, but can also produce verbose final memories that are costly and less reliable for long-term generalization or retention.

Overall, this comparison reinforces the main message of SeqMem-Eval: memory quality is not determined by a single dominant score, but by how a memory mechanism balances adaptation, generalization, consolidation, retention, and cost. The profiles also suggest concrete design implications. Methods should move beyond raw instance storage toward mechanisms that abstract reusable experience, but such abstraction must be paired with validation and cost control; otherwise, stronger online performance may come primarily from additional inference or update budget rather than from more reliable memory evolution.

B.2 Online Utility Results

Table 5: Accuracy (%) across LLMs, tasks, and methods. Colored subscripts indicate changes relative to the corresponding memory-free baseline. Best results are bolded.

Model	Method	HumanEval	MATH500	APIBench	MMLU-Eng.	MMLU-Phys.	ALFWorld
Qwen3-8B	Memory-free	81.8	68.6	64.5	50.3	65.6	52.9
	ExpRec$_{k=10}$	84.1$_{\uparrow 2.3}$	73.6$_{\uparrow 5.0}$	64.3$_{\downarrow 0.2}$	52.1$_{\uparrow 1.8}$	67.5$_{\uparrow 1.9}$	–
	ExpRec$_{k=3}$	82.6$_{\uparrow 0.8}$	72.8$_{\uparrow 4.2}$	63.9$_{\downarrow 0.6}$	51.7$_{\uparrow 1.4}$	69.4$_{\uparrow 3.8}$	–
	ExpRAG$_{k=3}$	82.6$_{\uparrow 0.8}$	73.0$_{\uparrow 4.4}$	67.0$_{\uparrow 2.5}$	53.1$_{\uparrow 2.8}$	68.4$_{\uparrow 2.8}$	–
	DC-RS	84.8$_{\uparrow 3.0}$	68.0$_{\downarrow 0.6}$	71.4$_{\uparrow 6.9}$	51.5$_{\uparrow 1.2}$	65.5$_{\downarrow 0.1}$	–
	AWM	84.1$_{\uparrow 2.3}$	68.8$_{\uparrow 0.2}$	73.9$_{\uparrow 9.4}$	50.5$_{\uparrow 0.2}$	67.2$_{\uparrow 1.6}$	51.4$_{\downarrow 1.5}$
	G-Memory	83.3$_{\uparrow 1.5}$	73.6$_{\uparrow 5.0}$	80.8$_{\uparrow 16.3}$	51.5$_{\uparrow 1.2}$	68.9$_{\uparrow 3.3}$	62.9$_{\uparrow 10.0}$
	ExpeL-ST	86.4$_{\uparrow 4.6}$	73.4$_{\uparrow 4.8}$	70.6$_{\uparrow 6.1}$	53.6$_{\uparrow 3.3}$	67.8$_{\uparrow 2.2}$	59.3$_{\uparrow 6.4}$
	ExpeL-MT	**90.9**$_{\uparrow 9.1}$	**80.2**$_{\uparrow 11.6}$	**82.0**$_{\uparrow 17.5}$	**67.2**$_{\uparrow 16.9}$	**79.7**$_{\uparrow 14.1}$	**79.3**$_{\uparrow 26.4}$
MiniMax-M2.7	Memory-free	95.0	81.4	52.7	60.1	74.6	68.6
	ExpRec$_{k=10}$	95.5$_{\uparrow 0.5}$	82.4$_{\uparrow 1.0}$	50.3$_{\downarrow 2.4}$	64.2$_{\uparrow 4.1}$	82.5$_{\uparrow 7.9}$	–
	ExpRec$_{k=3}$	95.5$_{\uparrow 0.5}$	83.6$_{\uparrow 2.2}$	53.6$_{\uparrow 0.9}$	63.2$_{\uparrow 3.1}$	85.5$_{\uparrow 10.9}$	–
	ExpRAG$_{k=3}$	95.5$_{\uparrow 0.5}$	81.6$_{\uparrow 0.2}$	53.5$_{\uparrow 0.8}$	62.7$_{\uparrow 2.6}$	85.0$_{\uparrow 10.4}$	–
	DC-RS	91.7$_{\downarrow 3.3}$	75.6$_{\downarrow 5.8}$	62.0$_{\uparrow 9.3}$	63.3$_{\uparrow 3.2}$	82.3$_{\uparrow 7.7}$	–
	AWM	95.5$_{\uparrow 0.5}$	82.6$_{\uparrow 1.2}$	66.3$_{\uparrow 13.6}$	61.7$_{\uparrow 1.6}$	82.2$_{\uparrow 7.6}$	72.1$_{\uparrow 3.5}$
	G-Memory	97.0$_{\uparrow 2.0}$	83.6$_{\uparrow 2.2}$	73.5$_{\uparrow 20.8}$	72.2$_{\uparrow 12.1}$	83.9$_{\uparrow 9.3}$	83.6$_{\uparrow 15.0}$
	ExpeL-ST	96.2$_{\uparrow 1.2}$	83.0$_{\uparrow 1.6}$	65.1$_{\uparrow 12.4}$	62.0$_{\uparrow 1.9}$	84.7$_{\uparrow 10.1}$	70.0$_{\uparrow 1.4}$
	ExpeL-MT	**97.7**$_{\uparrow 2.7}$	**87.8**$_{\uparrow 6.4}$	**80.8**$_{\uparrow 28.1}$	**75.8**$_{\uparrow 15.7}$	**90.2**$_{\uparrow 15.6}$	**88.6**$_{\uparrow 20.0}$

Table 5 reports the full online accuracy breakdown underlying the aggregate results in the main text. Overall, memory improves performance across most settings, but the gains are highly task- and method-dependent. ExpeL-MT achieves the strongest performance on nearly all datasets and both LLM backbones, suggesting that iterative experience reuse and multi-trajectory refinement can substantially improve online prediction. However, these improvements are not uniform: lightweight retrieval-based baselines improve performance on some tasks while degrading on others, and several methods underperform the memory-free baseline on specific datasets despite improving aggregate accuracy. This heterogeneity further supports the main-text observation that aggregate online accuracy is informative as a coarse endpoint metric but insufficient for characterizing memory quality. In particular, strong average performance can be driven by gains on a subset of tasks, while degradations on others suggest that memory updates may introduce task-dependent interference or ineffective context construction.

Across both Qwen3-8B and MiniMax-M2.7, the full online trajectories in Figures 6 and 7 show that memory evolution is rarely monotonic. Many methods exhibit early improvements followed by gradual degradation or premature saturation, indicating that continued memory updates do not consistently yield cumulative gains. This pattern is particularly evident on HumanEval and MMLU-Pro-Engineering, where several methods peak early and then steadily deteriorate despite continued exposure to additional tasks. Retrieval-based baselines generally remain close to the memory-free baseline, suggesting limited long-term accumulation of transferable experience. In contrast, stronger memory methods such as ExpeL-MT and G-Memory maintain larger margins over the baseline on several datasets, although their trajectories remain highly task-dependent rather than uniformly stable.

The dataset-level breakdown further reveals substantial variation in memory dynamics across tasks and backbones. On APIBench and ALFWorld, methods diverge more clearly over time, suggesting that memory design plays a larger role in environments requiring procedural adaptation or multi-step interaction. However, even methods with strong final OnlineAcc often exhibit large intermediate fluctuations, delayed recovery, or late-stage degradation, indicating that endpoint accuracy alone obscures substantial instability during sequential inference. Moreover, identical memory mechanisms can produce qualitatively different trajectory patterns across LLM backbones. For example, some methods that remain relatively stable on MiniMax-M2.7 degrade substantially on Qwen3-8B, suggesting that effective memory utilization depends not only on the memory mechanism itself but also on the backbone's ability to incorporate retrieved or synthesized experience over long sequences.

Table 6 further analyzes online performance through trajectory-level diagnostics. Across both Qwen3-8B and MiniMax-M2.7, low PED and high MER do not consistently correspond to the methods with the highest final OnlineAcc, indicating that endpoint accuracy and trajectory stability capture

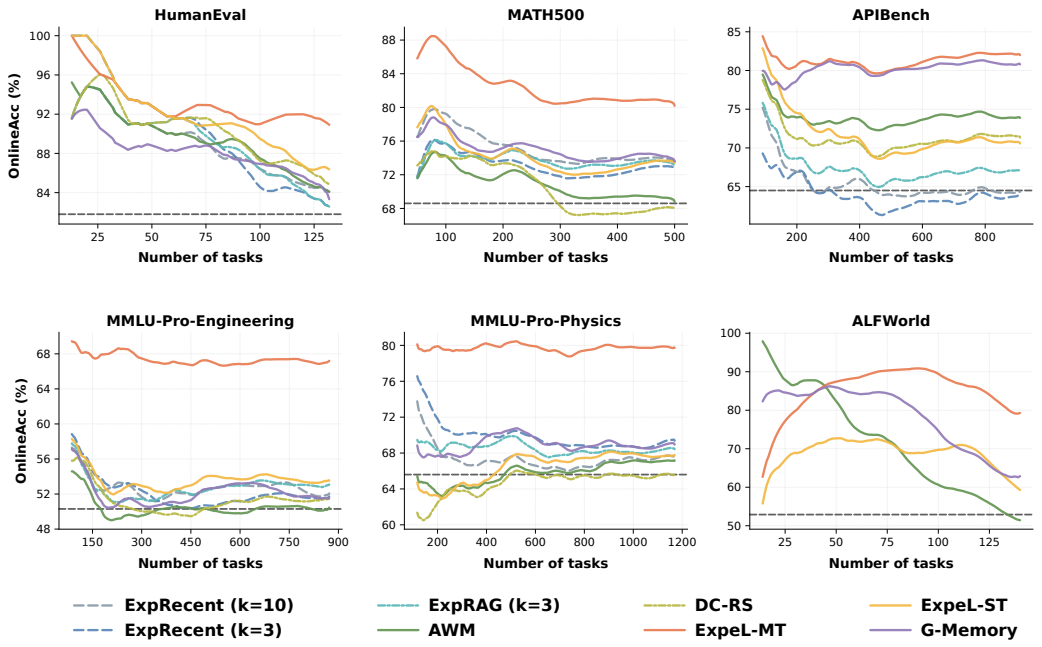


Figure 6: Online accuracy over sequential steps for the Qwen3-8B model.

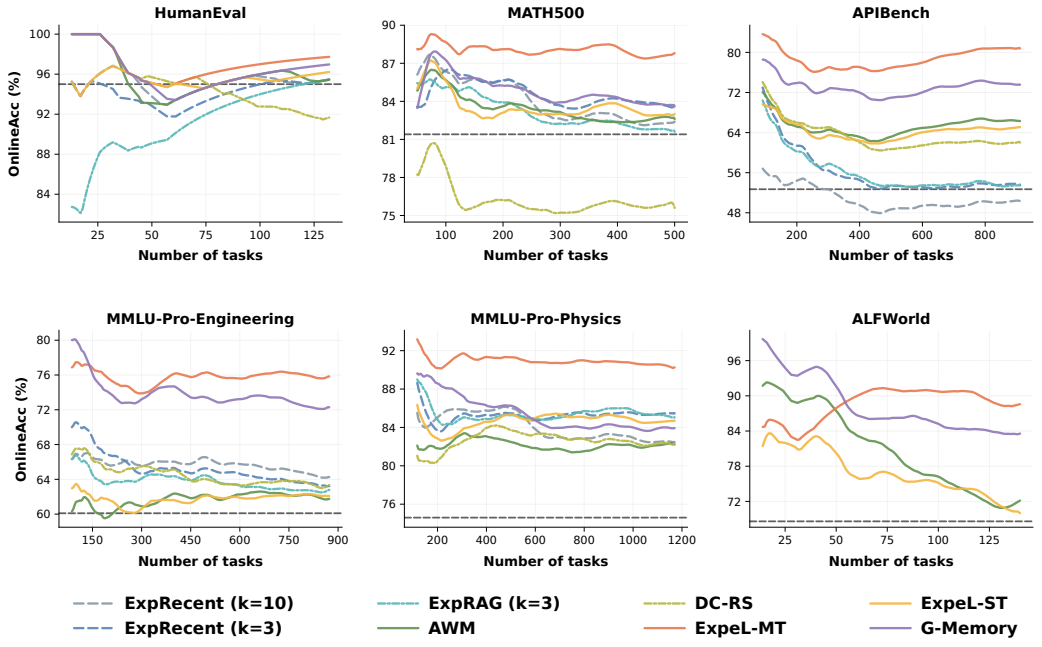


Figure 7: Online accuracy over sequential steps for the MiniMax-M2.7 model.

distinct aspects of memory behavior. For instance, although ExpeL-MT frequently achieves the best OnlineAcc, its PED/MER profile varies substantially across tasks, suggesting that strong final performance may still emerge from unstable or delayed recovery dynamics. In contrast, some retrieval-based methods exhibit relatively low PED but limited MER, corresponding to stable yet non-improving trajectories. The cross-backbone comparison further shows that identical memory mechanisms can induce different trajectory patterns under different LLMs, implying that online

Table 6: Online trajectory diagnostics on Qwen3-8B and MiniMax-M2.7. Higher MER and lower PED are generally preferred; $r_{\min}$ indicates the relative timing of the lowest online performance and should be interpreted together with MER and PED.

Method	HumanEval			MATH500			APIBench			MMLU-Eng.			MMLU-Phys.			ALFWorld		
	MER ↑	PED ↓	$r_{\min}$ ↓	MER ↑	PED ↓	$r_{\min}$ ↓	MER ↑	PED ↓	$r_{\min}$ ↓	MER ↑	PED ↓	$r_{\min}$ ↓	MER ↑	PED ↓	$r_{\min}$ ↓	MER ↑	PED ↓	$r_{\min}$ ↓
Backbone LLM: Qwen3-8B																		
ExpRec$_{k=10}$	0.001	0.121	0.99	0.007	0.084	0.66	0.019	0.113	0.28	0.013	0.067	0.38	0.018	0.060	0.61	–	–	–
ExpRec$_{k=3}$	0.001	0.174	0.99	0.018	**0.061**	0.62	0.034	0.070	0.50	0.020	0.088	0.50	0.011	0.076	0.89	–	–	–
ExpRAG$_{k=3}$	0.001	0.136	0.99	**0.030**	**0.061**	**0.10**	0.031	0.095	0.50	**0.033**	0.055	0.24	0.018	0.024	0.17	–	–	–
DC-RS	0.001	0.120	0.99	0.013	0.087	0.62	0.036	0.076	0.50	0.028	0.073	0.50	**0.072**	0.010	0.12	–	–	–
AWM	0.001	0.121	0.99	0.000	0.084	1.00	0.024	0.060	0.50	0.031	0.057	0.24	0.047	**0.003**	0.18	0.000	0.486	1.00
G-Memory	0.000	0.114	1.00	0.004	0.071	0.73	**0.045**	**0.008**	**0.11**	0.022	0.068	0.23	0.034	0.022	**0.10**	0.007	0.246	0.94
ExpeL-ST	**0.007**	0.136	0.90	0.022	0.084	0.62	0.030	0.118	0.50	0.028	0.051	0.24	0.057	**0.008**	0.12	0.005	0.152	0.12
ExpeL-MT	0.005	**0.091**	**0.71**	0.003	0.100	0.62	0.035	0.013	0.19	0.013	**0.035**	**0.16**	0.020	0.012	0.12	**0.150**	**0.122**	**0.10**
Backbone LLM: MiniMax-M2.7																		
ExpRec$_{k=10}$	0.033	0.019	0.39	0.006	0.070	0.87	0.033	0.089	0.50	0.009	0.052	**0.10**	0.002	0.039	0.92	–	–	–
ExpRec$_{k=3}$	0.046	0.010	0.42	**0.028**	0.047	**0.10**	0.018	0.186	0.50	0.001	0.103	0.95	0.027	0.035	0.17	–	–	–
ExpRAG$_{k=3}$	**0.185**	**0.000**	**0.10**	0.000	0.056	0.87	0.008	0.170	0.50	0.005	0.065	0.97	0.013	0.039	0.17	–	–	–
DC-RS	0.004	0.083	0.95	0.014	0.074	0.26	0.024	0.121	0.50	0.006	0.073	0.97	0.027	0.023	0.14	–	–	–
AWM	0.033	0.046	0.39	0.006	0.057	0.87	0.047	0.067	0.50	**0.040**	**0.026**	**0.10**	0.019	0.018	**0.11**	0.021	0.229	0.91
G-Memory	0.042	0.030	0.42	0.002	0.066	0.87	0.040	0.070	0.50	0.004	0.091	0.97	0.004	0.066	0.89	0.004	0.164	0.94
ExpeL-ST	0.039	0.012	**0.10**	0.014	0.064	0.32	0.040	0.065	0.50	0.028	0.034	0.30	**0.028**	**0.017**	0.18	0.000	0.170	1.00
ExpeL-MT	0.036	0.023	0.39	0.018	**0.037**	0.21	**0.057**	**0.041**	**0.28**	0.025	0.040	0.34	0.009	0.030	0.17	**0.086**	**0.034**	**0.18**

memory dynamics depend not only on the memory design itself but also on the backbone's ability to utilize retrieved or updated experience.

In addition to MER and PED, we analyze the extremum timing metric $r_{\min}$, which records where the lowest point of the cumulative online accuracy trajectory occurs. While MER and PED quantify the magnitude of recovery and degradation, they do not indicate when these events happen. This timing information is important for distinguishing qualitatively different memory dynamics. For example, a method with high MER and low PED may correspond either to gradual improvement from a weak initial memory state or to a delayed drop followed by recovery. The value of $r_{\min}$ helps separate these cases: a small $r_{\min}$ indicates that the worst performance occurs early and is followed by recovery, whereas a large $r_{\min}$ indicates that the method reaches its lowest performance late in the sequence, suggesting persistent degradation as memory evolves.

The results in Table 6 show that late minima are common across several tasks and methods. For instance, on Qwen3-8B, many methods on HumanEval have $r_{\min}$ close to 1.0, indicating that their cumulative online performance reaches its lowest point near the end of the sequence. This pattern is consistent with the online trajectories in Figure 6, where several methods gradually decline after an initially strong phase. Such behavior would be difficult to diagnose from the final OnlineAcc alone, since the endpoint only reports the final value without revealing whether the method steadily improved or progressively degraded.

In contrast, smaller $r_{\min}$ values often indicate recovery from an early low-performance phase. This is especially visible in ALFWorld, where stronger agentic memory methods tend to have lower $r_{\min}$ together with higher MER and lower PED. These trajectories suggest that the method initially struggles when memory is sparse, but later benefits from accumulated experience. Thus, $r_{\min}$ provides complementary evidence to MER and PED: MER measures the amount of recovery, PED measures the amount of late degradation from the peak, and $r_{\min}$ indicates whether the lowest-performance phase occurs early enough for the method to recover or late enough to signal persistent instability. Overall, the $r_{\min}$ analysis reinforces our main observation that online memory utility should be evaluated as a trajectory rather than a single endpoint. Two methods with similar OnlineAcc or similar MER/PED values can still differ substantially in when their failures occur. Therefore, extremum timing provides an additional diagnostic for identifying whether memory evolution reflects early-stage adaptation, delayed recovery, or late-stage degradation.

B.3 ID hold-out test results

Figure 8 and 9 present the full ID hold-out trajectories over sequential memory updates. Consistent with the main-text analysis, hold-out generalization rarely improves monotonically as memory evolves. Across many datasets, trajectories fluctuate substantially around the memory-free baseline, and later memory states often fail to outperform earlier ones. This pattern is particularly pronounced on APIBench and MMLU-Pro-Engineering, where several methods exhibit transient improvements

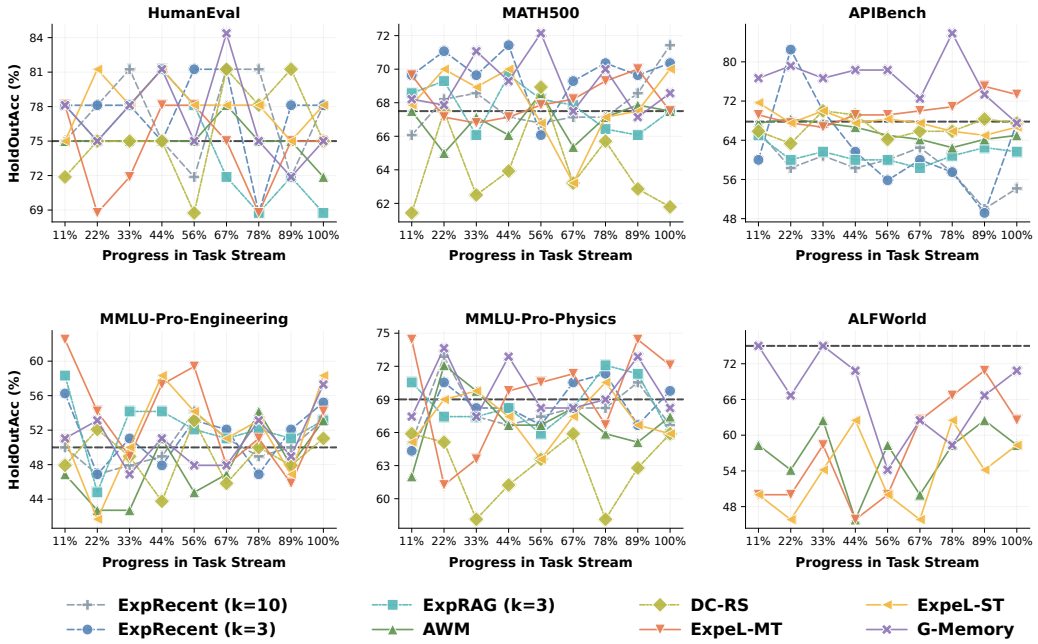


Figure 8: In-distribution (ID) hold-out accuracy over sequential steps for the Qwen3-8B model.

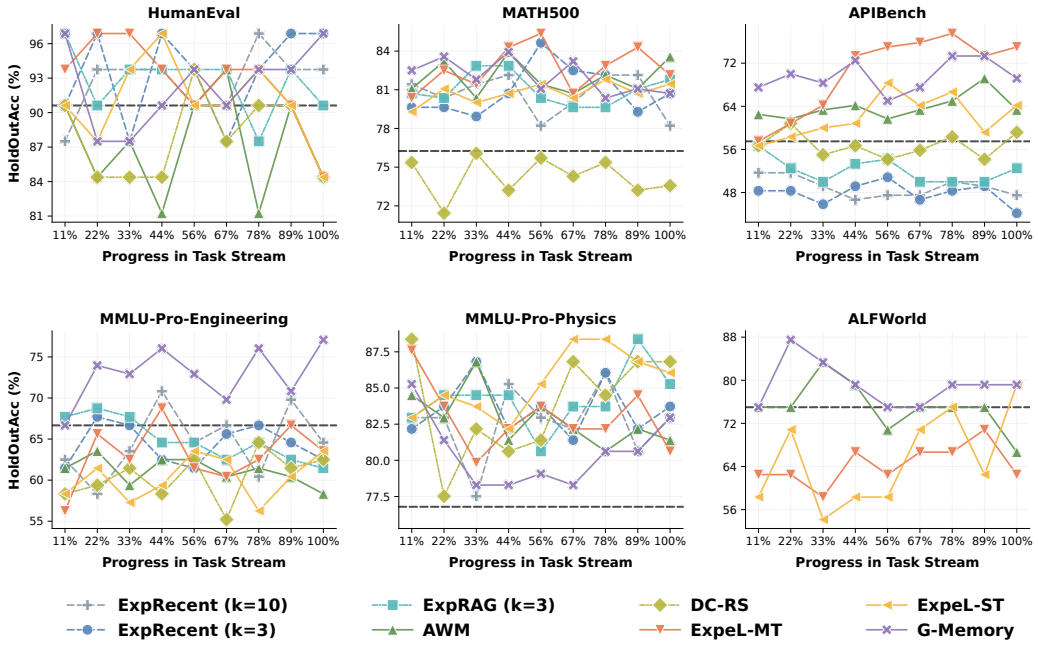


Figure 9: In-distribution (ID) Hold-out accuracy over memory steps for the MiniMax-M2.7 model.

followed by degradation, suggesting that memory updates may overfit to recently observed interactions rather than accumulate transferable knowledge. In multiple cases, peak hold-out performance occurs at intermediate stages rather than in the final memory state, indicating that useful information can later be diluted, overwritten, or negatively affected by subsequent updates.

The trajectory-level analysis also reveals clear differences across memory mechanisms. Retrieval-based methods generally produce relatively stable, low-variance trajectories close to the baseline, implying limited long-term knowledge accumulation despite reduced instability. In contrast, stronger memory systems such as ExpeL-MT and G-Memory often achieve higher peak hold-out performance but exhibit highly non-stationary trajectories with substantial oscillations across memory steps. This trade-off suggests that aggressive memory synthesis and refinement may improve adaptability while reducing memory stability. Moreover, identical methods can exhibit qualitatively different hold-out dynamics across LLM backbones, indicating that memory generalization depends not only on stored experience but also on the backbone's ability to consistently exploit evolving memory states. Overall, these results further support the central claim of RQ2: final hold-out accuracy alone obscures important temporal properties of memory generalization, including instability, reversibility, and sensitivity to later updates.

B.4 Paired Bootstrap Analysis of Hold-out Trajectories

Paired bootstrap procedure. The point estimates in Table 4 indicate that methods with similar final $\mathrm{HoldOutAcc}$ can have different hold-out trends. To quantify the uncertainty of these differences, we perform paired comparisons between methods evaluated with the same backbone on the same task and, therefore, on the same held-out problems.

For every method pair (a, b), we jointly resample their shared held-out problems with replacement. The same sampled problem indices are used for both methods and across all memory checkpoints, preserving both the method pairing and the dependence of repeated evaluations of the same problem over the trajectory. We use 10,000 paired-bootstrap replicates.

For bootstrap replicate r, we first recompute the difference in final hold-out accuracy:

$$\Delta H^{(r)} = H_a^{(r)}(T) - H_b^{(r)}(T), \tag{13}$$

where $H_m^{(r)}(T)$ is the final hold-out accuracy of method m in replicate r. We then recompute the hold-out accuracy at every memory checkpoint, refit the OLS trajectory slope for each method, and calculate

$$\Delta S^{(r)} = \mathrm{Trend}_{\mathrm{HO},a}^{(r)} - \mathrm{Trend}_{\mathrm{HO},b}^{(r)}. \tag{14}$$

We call a pair *endpoint-unresolved* when the paired 95% confidence interval for ΔH contains zero. This terminology indicates that the available hold-out results do not distinguish the methods at the endpoint; it does not constitute an equivalence test. Among endpoint-unresolved pairs, a trajectory difference is considered statistically distinguishable when the paired 95% confidence interval for ΔS excludes zero.

Aggregate paired-comparison results. Among 292 eligible method pairs, 197 pairs (67%) are unresolved by their final $\mathrm{HoldOutAcc}$. Among these 197 endpoint-unresolved pairs, 23 have opposite-signed hold-out trends and a paired 95% confidence interval for ΔS that excludes zero. Paired trajectory analysis distinguishes the methods' memory-evolution patterns in these 23 pairs, even though their endpoint differences remain unresolved using the available hold-out problems.

We additionally consider a cost-restricted comparison in which the token usage and runtime of the two methods must each be within a factor of 1.5. Under this restriction, 38 endpoint-unresolved pairs remain, of which 4 have opposite-signed trends and a paired confidence interval for ΔS that excludes zero. In addition, 4 endpoint-unresolved pairs, including 2 under the cost restriction, have trends with the same sign but statistically distinguishable magnitudes.

Representative comparison. Table 7 presents two representative comparisons. ExpeL-ST and G-Memory both achieve a final $\mathrm{HoldOutAcc}$ of 0.792 with MiniMax-M2.7 on ALFWorld. The paired confidence interval for their endpoint difference contains zero, but the confidence interval for their trend difference excludes zero.

Table 7: Representative paired-bootstrap comparisons for MiniMax-M2.7 on ALFWorld. Each confidence interval is computed from 10,000 paired-bootstrap replicates over the shared held-out problems. Both method pairs satisfy the 1.5× token-and-runtime cost restriction.

Method comparison	$\Delta\mathrm{HoldOutAcc}$ [95% CI]	$\Delta\mathrm{Trend}_{\mathrm{HO}}$ [95% CI]
ExpeL-ST − G-Memory	0.000 $[-0.167,\ 0.167]$	0.204 $[0.044,\ 0.383]$
ExpeL-ST − AWM	0.125 $[-0.042,\ 0.292]$	0.251 $[0.019,\ 0.520]$

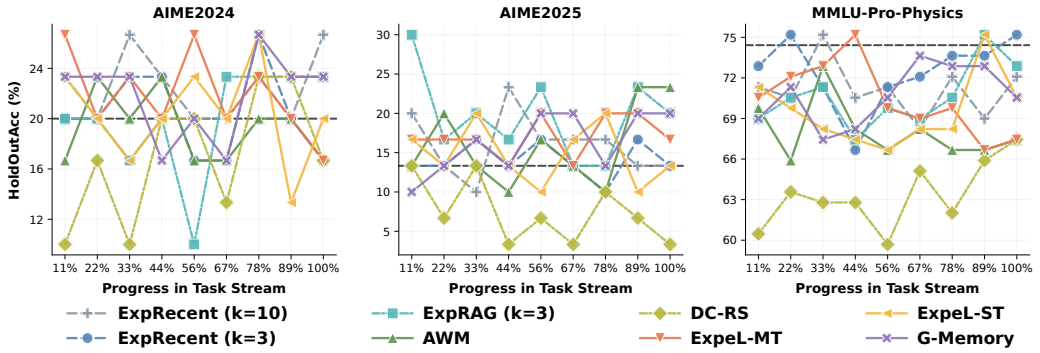


Figure 10: Out-of-distribution (OOD) hold-out accuracy over sequential steps for the Qwen3-8B model, evaluated on unseen tasks not encountered during the online updating process.

The point estimates help explain this difference. ExpeL-ST reaches its highest hold-out performance at the final checkpoint, resulting in $\mathrm{PED}_{\mathrm{HO}} = 0.000$ and $\mathrm{Trend}_{\mathrm{HO}} = 0.169$. In contrast, G-Memory reaches a peak hold-out accuracy of 0.875 after approximately one fifth of the stream and subsequently loses 0.083 by the final checkpoint, resulting in $\mathrm{PED}_{\mathrm{HO}} = 0.083$ and $\mathrm{Trend}_{\mathrm{HO}} = -0.034$. Therefore, the same final $\mathrm{HoldOutAcc}$ can correspond to either continued improvement toward the endpoint or earlier improvement followed by degradation.

The comparison between ExpeL-ST and AWM shows a similar distinction: their final hold-out difference is unresolved by the paired confidence interval, whereas their trajectory difference is distinguishable. These results do not provide a context-independent ranking of the methods. Instead, they show that endpoint accuracy and trajectory behavior capture complementary aspects of memory quality, and that paired inference can identify trajectory differences that are hidden by final $\mathrm{HoldOutAcc}$.

B.5 OOD HOLD-OUT TEST RESULTS ON QWEN3-8B

We evaluate OOD generalization by transferring memory states learned from the MATH500 stream to AIME2024, AIME2025, and MMLU-Pro-Physics, with results shown in Figure 10. Compared with the ID setting, OOD trajectories exhibit weaker and less stable gains, indicating limited transferability of learned memory beyond the training distribution.

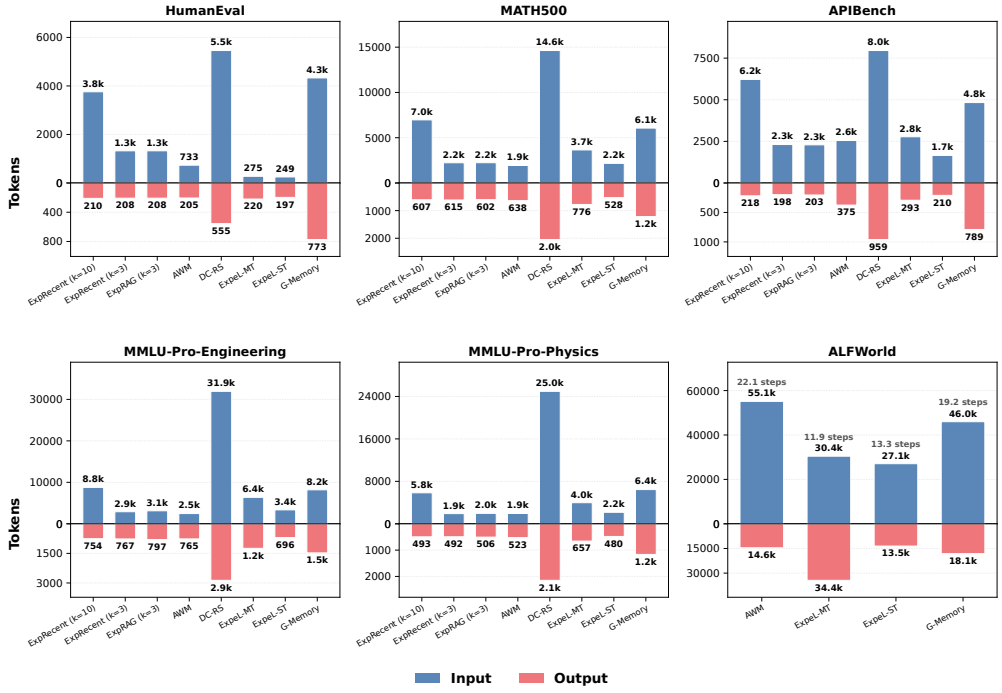
On AIME2024 and AIME2025, memory provides only marginal improvements over the memory-free baseline, likely due to partial overlap in mathematical reasoning patterns. However, these gains are inconsistent across methods and memory stages, suggesting that the learned memory captures limited transferable structure. The degradation is more pronounced on MMLU-Pro-Physics, where performance remains consistently below the baseline, even for methods that construct higher-level synthesized insights. This behavior indicates weak cross-domain transfer and limited generalization of memory beyond the source domain.

Across all OOD datasets, hold-out trajectories exhibit substantial fluctuations, and later memory states frequently fail to outperform earlier ones. Although this instability mirrors the behavior observed under ID evaluation, it becomes more pronounced under distribution shift, highlighting the lack of stable and transferable memory evolution.

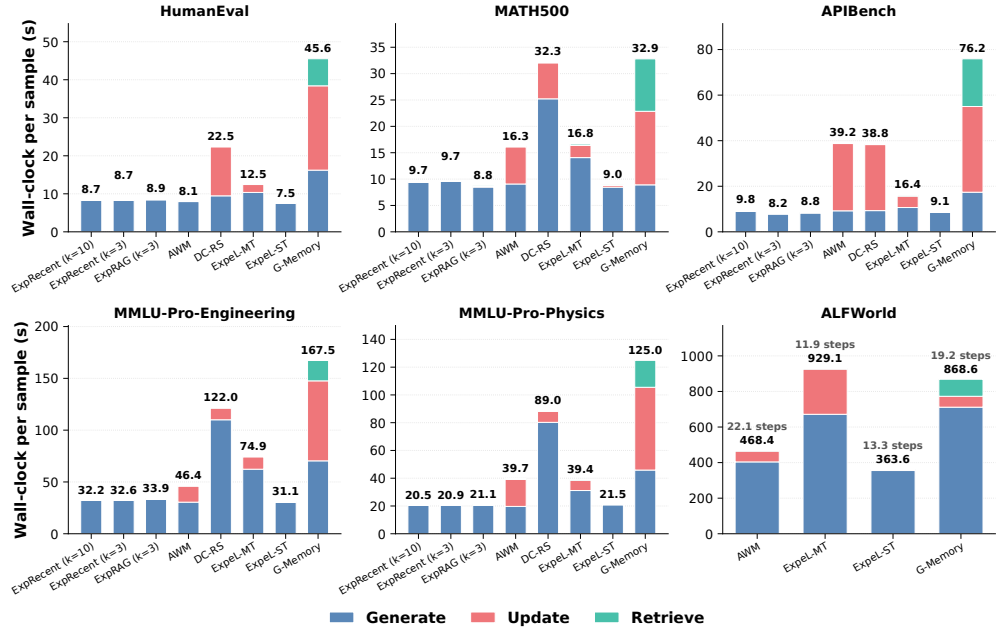
B.6 Efficiency Results

Figures 11 and 12 present detailed runtime and token-consumption breakdowns across datasets and memory mechanisms for Qwen3-8B and MiniMax-M2.7. Consistent with the main-text analysis, runtime is dominated by generation, while token overhead primarily originates from memory construction and update operations. The breakdown further shows that different memory mechanisms induce distinct efficiency profiles. Retrieval-based methods such as ExpRecent and ExpRAG mainly increase input tokens through longer retrieved contexts while incurring relatively small update overhead, resulting in moderate and stable computational growth. In contrast, methods based on synthesized or structured memory introduce substantially larger update costs. DC-RS exhibits particularly high token consumption because each update requires long structured prompts for cheatsheet synthesis, causing update tokens to dominate total usage on several datasets. G-Memory shows a different overhead pattern: although its token growth is smaller than that of DC-RS, runtime increases substantially due to graph construction, insight maintenance, and additional retrieval operations during inference.

The dataset-level breakdown further indicates that efficiency depends strongly on task characteristics. On reasoning-intensive datasets such as MMLU-Pro-Engineering and MMLU-Pro-Physics, runtime increases disproportionately even under moderate token growth, suggesting that latency is affected not only by context length but also by more complex inference trajectories. ALFWorld exhibits the most distinct behavior: methods with iterative interaction or retrieval mechanisms incur extremely large runtime despite comparatively moderate token usage, indicating that multi-step environment interaction amplifies latency beyond pure language generation cost. In addition, identical memory mechanisms can exhibit substantially different efficiency profiles across LLM backbones. MiniMax-M2.7 generally produces higher runtime despite similar or lower token counts, implying that memory overhead is jointly determined by both the external memory algorithm and the backbone-specific inference characteristics. Overall, these results show that aggregate token usage alone is insufficient to characterize computational efficiency, as memory update complexity, retrieval structure, and interaction dynamics contribute to end-to-end cost differently.

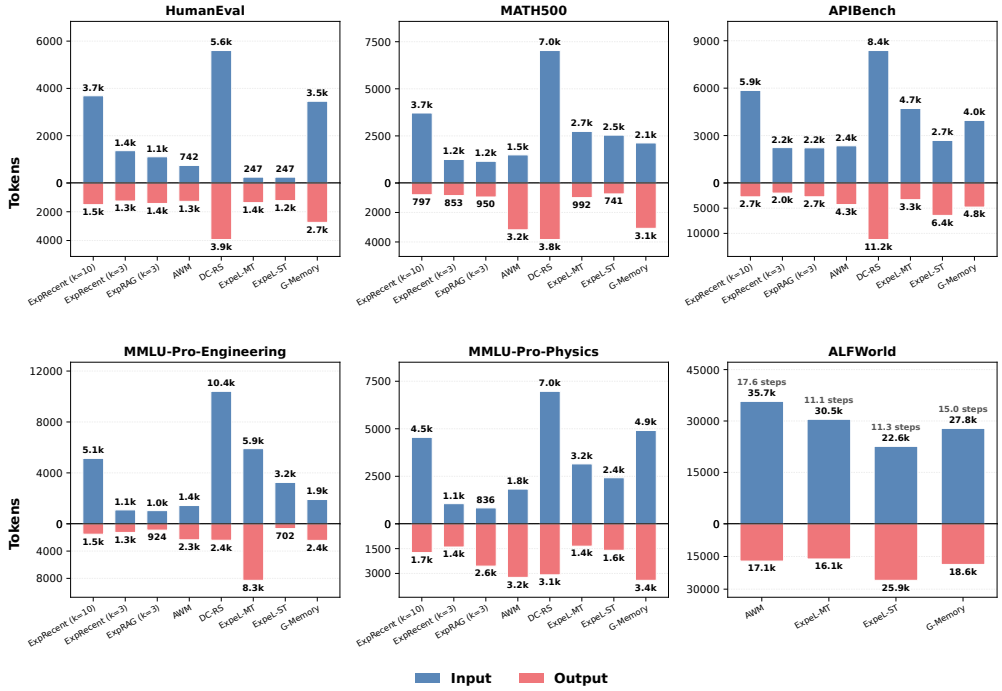


(a) Token consumption.

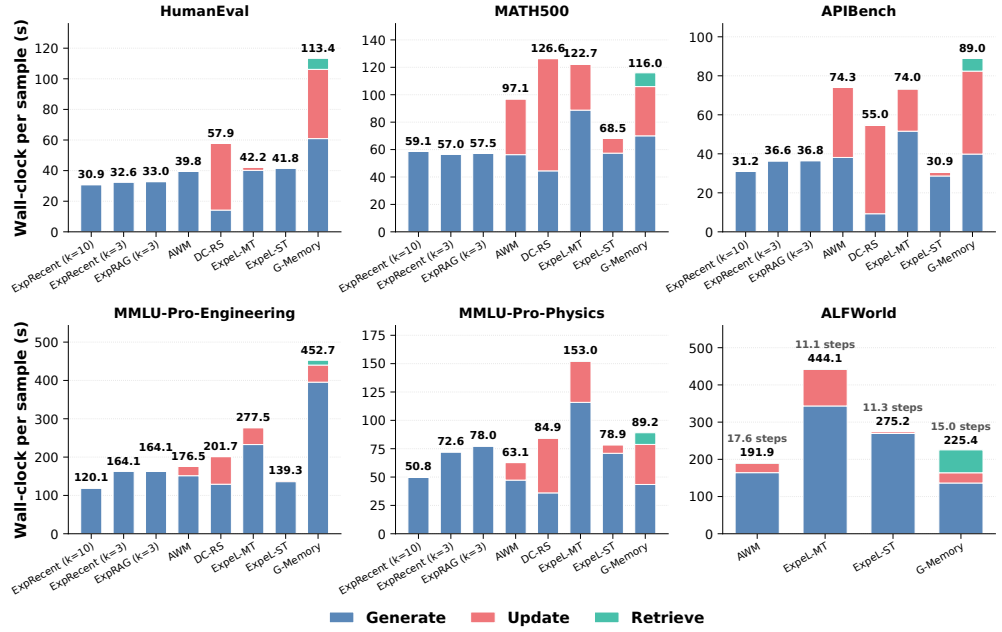


(b) Runtime.

Figure 11: Efficiency analysis of token consumption and runtime for the Qwen3-8B model.



(a) Token consumption.



(b) Runtime.

Figure 12: Efficiency analysis of token consumption and runtime for the MiniMax-M2.7 model.

B.7 Validation and Robustness Analyses

We conduct additional analyses to examine whether the proposed diagnostics provide redundant information, whether their conclusions are robust to alternative metric formulations, and how sensitive the reported results are to task order and retrieval configuration. These experiments are intended as targeted robustness checks rather than a complete sweep over every possible evaluation condition.

B.7.1 Cross-Metric Correlation and Redundancy

We first examine whether the diagnostic metrics capture distinct aspects of memory behavior or largely duplicate one another. The analysis covers 88 available backbone–task–method cells organized into 12 backbone–task blocks, corresponding to six tasks and two backbones. Because backward transfer and forgetting require repeated re-inference, they are available for only three of these blocks.

To avoid conflating between-task performance differences with within-condition method behavior, we rank methods within each backbone–task block, normalize the ranks, and pool the normalized ranks across blocks. For each metric pair, we compute the rank correlation over the cells for which both metrics are available. Confidence intervals are obtained using a block bootstrap. We use $|\rho| \geq 0.90$ as a prespecified heuristic threshold for potential redundancy, corresponding to $\rho^2 \geq 0.81$ and a variance inflation factor above approximately five.

Table 8: The five largest cross-metric rank correlations among the 45 evaluated metric pairs. Confidence intervals are obtained by block bootstrap over backbone–task conditions.

Metric pair	ρ	95% CI
$\mathrm{OnlineAcc}$–$\mathrm{HoldOutAcc}$	$+0.64$	$[+0.54, +0.75]$
MER–$r_{\min}$	$+0.60$	$[+0.42, +0.76]$
MER–PED	$+0.53$	$[+0.29, +0.74]$
$\mathrm{HoldOutAcc}$–$\mathrm{Trend}_{\mathrm{HO}}$	$+0.49$	$[+0.34, +0.62]$
PED–$r_{\min}$	$+0.48$	$[+0.33, +0.63]$

As shown in Table 8, the largest observed correlation is 0.64, and none of the 45 metric pairs reaches the prespecified $|\rho| \geq 0.90$ redundancy threshold. On the subset for which forgetting is available, its correlation with PED is $+0.37$. The diagnostics are therefore related, as expected, but are not interchangeable: endpoint performance, trajectory shape, transfer, and retention retain distinct empirical information.

B.7.2 Robustness to Alternative Metric Formulations

We test two metric-design choices that may be sensitive to individual checkpoints. First, the main experiments estimate $\mathrm{Trend}_{\mathrm{HO}}$ using an ordinary least-squares slope. We refit every hold-out trajectory using the Theil–Sen estimator, which uses the median of pairwise slopes and is less sensitive to anomalous checkpoints.

The Theil–Sen estimates agree strongly with the least-squares estimates, with $\rho = 0.96$ over the pooled metric values and $\rho = 0.92$ over the within-condition method ranks. The two estimators also identify the same worst-trending method in all 12 evaluated backbone–task conditions. Differences arise primarily when the estimated slope is close to zero. This result supports the overall robustness of $\mathrm{Trend}_{\mathrm{HO}}$, while also motivating our decision not to assign strong qualitative meaning to the bare sign of a near-zero trend.

Second, the main forgetting metric uses the best previously achieved performance as its reference. Because a maximum can be sensitive to a transient success, we also compute a mean-reference degradation variant using average prior performance as the reference. As expected, its absolute degradation values are smaller, but its method ordering remains similar to that of the original max-reference metric, with $\rho = 0.90$.

We retain the max-reference definition because it directly measures the loss of a capability that was demonstrably achieved at an earlier memory state. The mean-reference result nevertheless shows that the main comparative pattern is not driven solely by isolated performance peaks.

B.7.3 SENSITIVITY TO TASK ORDER

Trajectory- and transfer-based metrics can depend on the ordering of the task stream. A complete multi-order sweep over the full evaluation grid is computationally prohibitive: one sweep on a single backbone requires more than 400 hours of sequential model execution, and steps within a memory stream cannot be parallelized because task $t + 1$ must use the memory state produced by task t.

We therefore conduct a targeted order-sensitivity study using the Qwen3-8B/HumanEval stream and three representative methods. We evaluate five task orders: the submitted order (O_0), two random permutations (O_1 and O_2), and two difficulty-ordered stress tests (O_3 and O_4), constructed according to whether the memory-free backbone solves each problem. The difficulty-ordered streams are structured stress tests rather than bounds over all possible permutations.

Table 9: Online and hold-out accuracy across five orders of the Qwen3-8B/HumanEval stream. O_0 is the submitted order, with accuracies taken from Tables 5 and 4. O_1–O_2 are random permutations, and O_3–O_4 are difficulty-ordered stress tests. Mean and sample SD are computed across the five displayed values.

Metric	Method	O_0	O_1	O_2	O_3	O_4	Mean	SD
OnlineAcc	ExpRecent ($k = 3$)	.826	.854	.854	.854	.862	.850	.014
OnlineAcc	ExpRAG ($k = 3$)	.826	.823	.854	.808	.846	.831	.019
OnlineAcc	ExpeL-MT	.909	.946	.938	.938	.938	.934	.014
HoldOutAcc	ExpRecent ($k = 3$)	.781	.781	.750	.813	.844	.794	.036
HoldOutAcc	ExpRAG ($k = 3$)	.688	.656	.750	.750	.719	.713	.041
HoldOutAcc	ExpeL-MT	.750	.813	.813	.781	.813	.794	.028

Table 10: Mean and standard deviation of the diagnostic metrics over the five Qwen3-8B/HumanEval task orders.

Method	PED	MER	$\mathrm{Trend}_{\mathrm{HO}}$	BWT@8	F@32
ExpRecent ($k = 3$)	$.089 \pm .072$	$.153 \pm .281$	$+.007 \pm .034$	$-.002 \pm .007$	$+.055 \pm .026$
ExpRAG ($k = 3$)	$.097 \pm .071$	$.140 \pm .283$	$-.089 \pm .032$	$+.012 \pm .011$	$+.006 \pm .009$
ExpeL-MT	$.032 \pm .035$	$.104 \pm .136$	$-.037 \pm .012$	$+.026 \pm .009$	$+.016 \pm .009$

ExpeL-MT achieves the highest OnlineAcc under all five orders, while ExpRecent matches or outperforms ExpRAG. The hold-out comparison is less decisive. ExpRAG ranks last under every order, whereas ExpRecent and ExpeL-MT differ by at most one problem on the 32-problem hold-out set. Because one problem corresponds to approximately 3.1 percentage points, their alternating nominal ordering should be treated as unresolved at the resolution of this hold-out set rather than as a meaningful rank reversal.

The diagnostic metrics exhibit different levels of order sensitivity. PED and MER vary most strongly because they are computed from prefixes of the cumulative online trajectory and are inherently order-conditioned. $\mathrm{Trend}_{\mathrm{HO}}$ is more stable when it is clearly separated from zero: it remains negative under all five orders for ExpRAG and ExpeL-MT, whereas ExpRecent remains near zero and its sign varies. We therefore interpret the latter as no clear trend rather than assigning meaning to its nominal sign.

For backward transfer, ExpeL-MT is the only method with positive BWT@8 under all five orders. ExpRecent remains near zero, while ExpRAG exhibits a smaller and less stable effect. For forgetting, ExpRecent has the largest F@32 on average, although its magnitude varies across orders.

This targeted study does not establish complete order invariance. Instead, it stratifies the conclusions: ExpeL-MT leads in OnlineAcc across the tested orders; the leading HoldOutAcc comparison is unresolved because of hold-out-set granularity; transfer and hold-out trends are more stable when sufficiently separated from zero; and prefix-based trajectory metrics should be interpreted as conditional on the task order.

Table 11: Sensitivity to retrieval top-k and embedding-model choice in the representative Qwen3-8B/HumanEval setting. The control uses Qwen3-Embedding-0.6B with $k = 3$.

Method	Embedding model	Params.	k	OnlineAcc	PED	MER	HoldOutAcc	$\mathrm{Trend}_{\mathrm{HO}}$
ExpRAG	`Qwen3-Embedding-0.6B`	600M	1	.800	.158	.004	.750	$-.094$
ExpRAG	`Qwen3-Embedding-0.6B`	600M	3	.808	.154	.003	.750	$-.086$
ExpRAG	`Qwen3-Embedding-0.6B`	600M	10	.846	.115	.011	.781	$-.092$
ExpRAG	`embeddinggemma-300m`	300M	3	.815	.122	.000	.750	$-.064$
ExpRAG	`all-MiniLM-L6-v2`	22M	3	.838	.085	.004	.750	$-.105$
ExpeL-MT	`Qwen3-Embedding-0.6B`	600M	3	.923	.077	.019	.813	$-.045$
ExpeL-MT	`embeddinggemma-300m`	300M	3	.938	.062	.013	.813	$-.017$
ExpeL-MT	`all-MiniLM-L6-v2`	22M	3	.938	.062	.013	.813	$-.011$

B.7.4 Sensitivity to Retrieval Configuration

We further examine sensitivity to retrieval top-k and embedding-model choice in the representative Qwen3-8B/HumanEval setting. Table 11 reports a separately run sensitivity study; its control configurations were rerun independently of the main experiments. We vary k for ExpRAG, for which retrieval directly determines the memory context, and replace the embedding encoder for both ExpRAG and ExpeL-MT, which use the same similarity-based retrieval interface. The retrieval index is rebuilt independently for each encoder. The three tested encoders range from 22M to 600M parameters.

Retrieval top-k affects the magnitude of ExpRAG's performance but does not change its qualitative trajectory diagnosis. Increasing k from 1 to 10 improves $\mathrm{OnlineAcc}$ from 0.800 to 0.846 and reduces PED from 0.158 to 0.115. However, MER remains close to zero and $\mathrm{Trend}_{\mathrm{HO}}$ remains negative under all three values of k.

Embedding-model choice has a smaller effect in this experiment. Across the three encoders, $\mathrm{OnlineAcc}$ varies by 3.0 percentage points for ExpRAG and 1.5 percentage points for ExpeL-MT, while ExpeL-MT's $\mathrm{HoldOutAcc}$ remains unchanged at 0.813. The negative hold-out trend is also qualitatively consistent across all tested encoders.

The relative comparison between the two representative methods is stable under the tested retrieval settings: every ExpeL-MT configuration achieves higher $\mathrm{OnlineAcc}$ and lower PED than every ExpRAG configuration. Nevertheless, these results constitute a targeted sensitivity study on one backbone, task, and stream order rather than evidence that retrieval configuration is universally unimportant. Moreover, increasing k expands the retrieved context and may increase token cost, so retrieval sensitivity should be interpreted jointly with efficiency.

C Detailed Experimental Setup

C.1 Experimental Setup

Datasets and Models. We evaluate on a diverse set of benchmarks spanning programming, mathematical reasoning, factual and domain-specific reasoning, tool use, and embodied interaction. For code generation, we use **HumanEval** (Chen et al., 2021); for mathematical reasoning, we adopt **MATH500** (Hendrycks et al., 2021); for factual and domain-specific reasoning, we include **MMLU-Pro** (Wang et al., 2024b), focusing on advanced subjects such as engineering and physics; for tool-use and API grounding, we evaluate on **APIBench** (Patil et al., 2023); and for long-horizon goal-oriented interaction, we include **ALFWorld** (Shridhar et al., 2021). Together, these benchmarks cover both single-turn and multi-step settings, enabling evaluation of sequential memory across diverse forms of reasoning and interaction. We conduct experiments using two representative large language models: Qwen3-8B (Team, 2025) and MiniMax-M2.7 (MiniMax, 2026).

Retrieval setup. For all methods involving retrieval, we standardize the pipeline by using Qwen3-0.6B-embedding (Team, 2025) as the embedding model. Dense retrieval is performed by computing cosine similarity using the Sentence Transformers library Reimers & Gurevych (2019).

Qwen3-8B model configuration. We use the official recommended temperature setting of 0.7 to balance generation quality and diversity. For HumanEval, MATH500, APIBench, and MMLU-Pro, reasoning is disabled, and the maximum number of generated tokens is set to 2048 for all methods. For ALFWorld, due to its higher task complexity, we enable reasoning and increase the maximum number of generated tokens to 8192 to allow for extended reasoning steps. All experiments are conducted on a single A100-80GB GPU.

MiniMax-M2.7 model configuration. We access the MiniMax model via the OpenRouter API. For all tasks and methods, we use a unified configuration with temperature set to 1.0, reasoning level set to low, and the maximum number of generated tokens set to 16384.

Dataset configuration. For HumanEval and MMLU-Pro, which lack dedicated training splits, we construct ID hold-out sets from the end of their respective test sets: 32 examples for HumanEval (approximately 20%), and 96 and 129 examples for MMLU-Pro-Engineering and MMLU-Pro-Physics, respectively (approximately 10% each). For APIBench, MATH500, and ALFWorld, the hold-out sets are sampled from the training data. The sampling is performed proportionally across task categories to preserve the original distribution, while ensuring that each category contains at least one task. For APIBench, we only use the HF subset, as the TF and TH subsets are relatively small and are nearly saturated by most models. The sizes of the sequential test streams and ID hold-out sets are summarized in Table 12.

Table 12: Dataset statistics, including total size and ID hold-out size.

Dataset	Sequential test	Hold-out
HumanEval	132	32
MATH500	500	280
MMLU-Pro-Engineering	873	96
MMLU-Pro-Physics	1170	129
APIBench-HF	911	120
ALFWorld	140	24

Sequential Evaluation Protocol. All datasets are evaluated under a unified sequential protocol. Each example is processed once in sequence, and the memory state is incrementally updated after each interaction. To ensure fair comparison, all methods are evaluated using the same retrieval budget. Each dataset is evaluated under a fixed task order shared by all methods. This design ensures that different memory mechanisms are compared under the same sequence of experiences. We emphasize, however, that task ordering is an intrinsic factor in sequential memory evaluation: different orders may expose different forms of adaptation, interference, and forgetting. Therefore, our reported results should be interpreted as diagnostics under controlled task streams rather than order-invariant estimates of memory quality. This protocol enables consistent measurement of online utility, hold-out generalization, backward transfer, forgetting, and efficiency across tasks and memory methods.

C.2 METHODS

We evaluate a representative set of sequentially evolving LLM memory methods to systematically analyze memory behaviors under a unified sequential setting.

(1) **Memory-free baseline.** The memory-free baseline solves each task using only the underlying LLM without persistent memory. It serves as a reference point for measuring whether external memory provides benefits beyond the base model.

(2) **Instance-level experience retrieval.** We include ExpRecent$_{k=10}$, ExpRecent$_{k=3}$, and ExpRAG$_{k=3}$ as lightweight baselines for raw experience reuse (Wei et al., 2025). ExpRecent$_k$ conditions on the most recent k tasks, while ExpRAG$_k$ retrieves the top-k most similar past tasks based on embedding similarity to the current input. This comparison separates recency-based memory from relevance-based retrieval and evaluates whether instance-level experiences alone are sufficient for sequential adaptation without additional memory abstraction or refinement.

(3) **Structured and evolving memory.** We evaluate Dynamic Cheatsheet Retrieval-and-Synthesis variant(DC-RS) (Suzgun et al., 2025), Agent Workflow Memory (AWM) (Wang et al., 2024c), and G-Memory (Zhang et al., 2025). These methods go beyond raw experience retrieval by organizing memory into higher-level structures such as cheatsheets, workflows, or hierarchical memory graphs. They allow us to examine whether structured memory representations improve transfer, retention, and generalization during sequential inference.

(4) **Experiential reflection.** We include ExpeL (Zhao et al., 2024), which derives reusable insights from successful and failed trajectories through reflection. Since the original formulation is not directly designed for our sequential evaluation protocol, we refactor it into a sequential memory setting. We evaluate its multi-try version as ExpeL-MT, following the original strategy of allowing multiple attempts per task, and additionally implement a single-try variant, ExpeL-ST, for controlled comparison. This comparison helps distinguish gains from experiential memory from those introduced by additional inference-time attempts. Detailed algorithmic descriptions and implementation details for all methods are provided in Appendix E.

Method configuration. Method configurations are summarized in Table 13.

Table 13: Method-specific configurations used in sequential experiments.

Method	Configuration
ExpRecent	`top-k=3`,`top-k=10`
ExpRAG	`top-k=3`
AWM	`top-k=3`, `induce-steps=1`
DC-RS	`top-k=3`
ExpeL-MT	`top-k=3`, `max-tries=3`, `batch-update-size=8`, `max-num-rules=20`
ExpeL-ST	`top-k=3`, `batch-update-size=8`, `max-num-rules=20`
G-Memory	`successful-topk=2`, `failed-topk=1`, `insights-topk=10`

D PRACTICAL USE OF SEQMEM-EVAL

SEQMEM-EVAL is not intended to collapse memory quality back into a single universal score. Different deployment settings may prioritize different aspects of memory behavior, such as online adaptation, hold-out generalization, retention, or computational cost. For example, an interactive agent may prioritize online utility and latency, whereas a long-term reasoning assistant may place more emphasis on hold-out generalization and forgetting.

As a practical guideline for method comparison, we suggest a two-stage procedure. First, identify methods that are Pareto competitive with respect to primary deployment constraints, such as $\mathrm{OnlineAcc}$, $\mathrm{HoldOutAcc}$, and efficiency. Second, among these candidates, use the diagnostic metrics to inspect memory dynamics: prefer methods with positive $\mathrm{Trend}_{\mathrm{HO}}$, non-negative BWT at short and medium horizons, low forgetting, and low PED. This procedure preserves the diagnostic nature of the framework while providing a concrete way to compare methods without reducing memory quality to a single score.

E ALGORITHMS

The pseudocode below uses method-specific internal notation, whose subscripts need not match the evaluation memory index. In the evaluation notation, M_i denotes the complete persistent state immediately before task i, and M_{i+1} denotes the state after completing that task. Internal component updates (e.g., to $\mathcal{M}_i$ or $\mathcal{H}_i$) within a task do not introduce additional evaluation checkpoints.

Algorithm 1 Dynamic Cheatsheet Retrieval-and-Synthesis (DC-RS)

Require: Sequence of tasks $\mathcal{T} = \{T_i\}_{i=1}^{N}$, large language model $\mathcal{L}$, retrival budget K, generator template P_{gen}, curator template P_{cur}

Initialize: Cheatsheet $\mathcal{M}_0 \leftarrow \emptyset$, history $\mathcal{H}_0 \leftarrow \emptyset$
for Task $T_i = (x_i, y_i) \in \mathcal{T}$ **do**
  if $|\mathcal{H}_{i-1}| > 0$ **then**
    Retrieve Top-K similar past cases C_{retr} from $\mathcal{H}_{i-1}$ ▷ Search
  else
    $\mathcal{P} \leftarrow \emptyset$
  end if
  Update cheatsheet $\mathcal{M}_i \leftarrow \mathcal{L}(P_{cur},\ C_{retr},\ x_i,\ \mathcal{M}_{i-1})$ ▷ Evolve
  Answer $\hat{y}_i \leftarrow \mathcal{L}(P_{gen},\ x_i,\ \mathcal{M}_i)$ ▷ Synthesis
  Update history $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \cup \{x_i, \hat{y}_i\}$ ▷ Evolve
end for

Algorithm 2 Agent Workflow Memory (AWM)

Require: Sequence of tasks $\mathcal{T} = \{T_i\}_{i=1}^{N}$, large language model $\mathcal{L}$, induction prompt P_{in}, one-shot prompt P_{one}

Initialize: base memory $\mathcal{M}_0 \leftarrow \emptyset$
for Task $T_i = (x_i, y_i) \in \mathcal{T}$ **do**
  Answer $\hat{y}_i \leftarrow \mathcal{L}(x_i, \mathcal{M}_{i-1})$ and get experience $e_i \leftarrow (x_i, \hat{y}_i)$ ▷ Synthesis
  $success \leftarrow \text{Evaluator}(e_i)$
  if $success = 1$ **then**
    $w_i \leftarrow \mathcal{L}(P_{in} \parallel P_{one} \parallel e_i \parallel$ `"## Summary Workflows"`$)$ ▷ Evolve
    $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup w_i$ ▷ Evolve
  else
    $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1}$
  end if
end for

Algorithm 3 ExpeL-ST

Require: Sequence of tasks $\mathcal{T} = \{T_i\}_{i=1}^{N}$, large language model $\mathcal{L}$, insight extraction prompt P_{in}, retrival budget K, step size L

Initialize: Experience pool $\mathcal{B} \leftarrow F_{\text{MANUAL}}$ (seed few-shot demos if available), recent success trajectories $\mathcal{S} \leftarrow \emptyset$, insight set $\hat{\iota} \leftarrow \emptyset$
for Task $T_i = (x_i, y_i) \in \mathcal{T}$ **do**
  Retrieve Top-K similar success cases F_{SIM} from $\mathcal{B}$ ▷ Search
  Answer $\hat{y}_i \leftarrow \mathcal{L}(x_i, F_{\text{SIM}}, \hat{\iota})$ ▷ Synthesis
  $success \leftarrow \text{Evaluator}(x_i, y_i, \hat{y}_i)$
  if $success = 1$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x_i, \hat{y}_i)\}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x_i, \hat{y}_i)\}$ ▷ Evolve
    if $|\mathcal{S}| = L$ **then**
      $\hat{\iota} \leftarrow \mathcal{L}(P_{in}, \mathcal{S}, \hat{\iota})$ ▷ Evolve
      $\mathcal{S} \leftarrow \emptyset$
    end if
  end if
end for

Algorithm 4 ExpeL-MT

Require: Sequence of tasks $\mathcal{T} = \{T_i\}_{i=1}^{N}$, large language model $\mathcal{L}$, self-reflection model $\mathcal{L}_{\text{REFLECT}}$, insight prompt P_{in}, retrieval budget K, step size L, max tries Z

Initialize: Experience pool $\mathcal{B} \leftarrow F_{\text{MANUAL}}$ (seed demos if available), recent success set $\mathcal{S} \leftarrow \emptyset$, insight set $\hat{\iota} \leftarrow \emptyset$
for Task $T_i = (x_i, y_i) \in \mathcal{T}$ **do**
  Retrieve Top-K similar success cases F_{SIM} from $\mathcal{B}$ ▷ Search
  $\nu \leftarrow$ " "
  $\tau^{\text{SUCC}} \leftarrow \emptyset$, $\tau^{\text{FAIL}} \leftarrow \emptyset$
  for $z = 1$ **to** Z **do**
    $\hat{y}_i^{(z)} \leftarrow \mathcal{L}(x_i, F_{\text{SIM}}, \hat{\iota}, \nu)$ ▷ Synthesis
    $success^{(z)} \leftarrow \text{Evaluator}(x_i, y_i, \hat{y}_i^{(z)})$
    if $success^{(z)} = 1$ **then**
      $\tau^{\text{SUCC}} \leftarrow (x_i, \hat{y}_i^{(z)})$
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tau^{\text{SUCC}}\}$ ▷ Evolve
      break
    else
      $\tau^{\text{FAIL}} \leftarrow (x_i, \hat{y}_i^{(z)})$
      $\nu \leftarrow \textsc{Concat}(\nu,\ \mathcal{L}_{\text{REFLECT}}(\tau^{\text{FAIL}}))$ ▷ Reflect
    end if
  end for
  if $\tau^{\text{SUCC}} \neq \emptyset$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\tau^{\text{SUCC}}\}$
  end if
  if $\tau^{\text{SUCC}} \neq \emptyset$ **and** $\tau^{\text{FAIL}} \neq \emptyset$ **then**
    $\hat{\iota} \leftarrow \mathcal{L}(P_{in}, \tau^{\text{SUCC}}, \tau^{\text{FAIL}}, \hat{\iota})$ ▷ Pair Update
  end if
  if $|\mathcal{S}| = L$ **then**
    $\hat{\iota} \leftarrow \mathcal{L}(P_{in}, \mathcal{S}, \hat{\iota})$ ▷ Batch Update
    $\mathcal{S} \leftarrow \emptyset$
  end if
end for

Algorithm 5 G-Memory

Require: Task stream $\mathcal{T}$, MAS $\mathcal{G}$, query graph $\mathcal{G}_{\text{QRY}} = (\mathcal{Q}, \mathcal{E}_{\text{QRY}})$ with query nodes $\mathcal{Q} = \{q_i\}$, insight graph $\mathcal{G}_{\text{INS}} = (\mathcal{I}, \mathcal{E}_{\text{INS}})$, budgets K, M

for $(Q_i, y_i) \in \mathcal{T}$ **do**
  $\mathcal{Q}^S \leftarrow \textsc{Retrieve}(Q_i, \mathcal{Q}, K)$ ▷ Search
  $\widetilde{\mathcal{Q}}^S \leftarrow \textsc{Expand}(\mathcal{Q}^S, \mathcal{G}_{\text{QRY}})$ ▷ Search
  $\mathcal{I}^S \leftarrow \Pi_{Q \to I}(\widetilde{\mathcal{Q}}^S, \mathcal{G}_{\text{INS}})$ ▷ Search
  $\mathcal{Q}^R \leftarrow \textsc{Select}(\widetilde{\mathcal{Q}}^S, M)$
  $\widehat{\mathcal{G}}_{\text{INTER}} \leftarrow \textsc{Sparsify}(\mathcal{Q}^R, Q_i)$
  for $C_j \in \mathcal{G}$ **do**
    $Mem_j \leftarrow \Phi(\mathcal{I}^S, \widehat{\mathcal{G}}_{\text{INTER}}, Role_j, Q_i)$
  end for
  $\hat{y}_i, \tau_i \leftarrow \mathcal{G}(Q_i)$ ▷ Synthesis
  $\Psi_i \leftarrow \text{Evaluator}(\hat{y}_i, y_i)$
  $\mathcal{G}_{\text{INTER}}^{(Q_i)} \leftarrow \textsc{Trace}(\tau_i)$
  $\mathcal{G}_{\text{QRY}} \leftarrow \textsc{UpdateQueryGraph}(\mathcal{G}_{\text{QRY}}, Q_i, \Psi_i, \mathcal{G}_{\text{INTER}}^{(Q_i)}, \mathcal{Q}^R, \mathcal{I}^S)$ ▷ Evolve
  $\mathcal{G}_{\text{INS}} \leftarrow \textsc{UpdateInsightGraph}(\mathcal{G}_{\text{INS}}, \mathcal{I}^S, \mathcal{G}_{\text{INTER}}^{(Q_i)}, \Psi_i)$ ▷ Evolve
end for

F PROMPTS

AIME2024 / AIME2025 Prompt

```
Solve the following AIME problem.  The final answer is a non-negative
integer between 0 and 999.
Think step by step.  Keep the reasoning concise but complete.  At
the very end, output ONLY the final integer inside a LaTeX box, e.g.,
\boxed{42}.  Do not wrap anything except the integer answer inside
\boxed{}.
Problem:  {question}
```

MATH500 Prompt

```
Solve the following math problem.
Please do your derivation in LaTeX as much as possible.  Keep the
reasoning concise but complete.  At the end, put only the final answer
in the LAST \boxed{...}.  If the final answer is a fraction, use
LaTeX fraction form like \frac{a}{b} (not decimal approximation unless
required).
Problem:  {question}
```

MMLU-Pro Prompt

```
Answer the following multiple-choice {subject} question.  Exactly one
option is correct.
Think step by step.  Keep the reasoning concise but complete.  At the
very end, output ONLY the letter of the correct option inside a LaTeX
box, e.g., \boxed{C}.  Do not wrap anything except the single letter
A-J inside \boxed{}.
Question:  {question}
Options:  {options}
```

HumanEval Prompt

````
Complete the following Python function.  Keep the given signature
and docstring unchanged; only fill in the body (you may add helper
functions above it if needed).  Your function must be named exactly
{entry_point}.
```python
{question}
```
You may reason briefly first, but only the code will be executed
against hidden unit tests.  Put your FINAL complete function inside
a single fenced block:
```python
# include the full def {entry_point}(...)  here, plus any helpers it
depends on
```
Do not output any other code fence.  No commentary after the code
block.
ANSWER:
````

ALFWorld Prompt

```
Imagine you are an intelligent agent in a household environment,
and your goal is to perform actions to complete the task.  At the
beginning of the interaction, you will be given a detailed description
of the environment and the goal.
For each turn, think briefly and then output your next action.
The available actions are:
     • go to {recep}
```

- take {obj} from {recep}
- move {obj} to {recep}
- open {recep}
- close {recep}
- use {obj}
- clean {obj} with {recep}
- heat {obj} with {recep}
- cool {obj} with {recep}

where {obj} and {recep} correspond to objects and receptacles.
After each turn, the environment will provide feedback. If the feedback is "Nothing happened", the previous action is invalid and you should try a different action.
Your response must follow this format:

- Thought:
- Action:

APIBench Prompt

You are a helpful API writer who can write APIs based on requirements.
{question}
Write a Python program in 1 to 2 lines to call API in {framework}.
The answer should follow the format: <<<domain>>> $DOMAIN, <<<api_call>>>: $API_CALL, <<<api_provider>>>: $API_PROVIDER, <<<explanation>>>: $EXPLANATION, <<<code>>>: $CODE. Here are the requirements:

1. $DOMAIN should be inferred from the task description.
2. $API_CALL should have only one line of code that calls the API.
3. $API_PROVIDER should be the programming framework used.
4. $EXPLANATION should be a step-by-step explanation.
5. $CODE is the Python code.
6. Do not repeat the format in your answer.

G Case Study And Capacity Ablation: Useful Memory State Destroyed by Subsequent Updates

This case study isolates a representative failure mode in sequentially evolving memory systems: a memory state that is initially useful gradually becomes corrupted or overwritten by later updates, eventually reducing downstream performance despite continued exposure to in-distribution data. We analyze ExpeL-MT on the MMLU-Pro-Engineering benchmark using Qwen3-8B as the backbone model, and focus on a single hold-out question on which the model initially answers correctly and progressively fails as the memory evolves. The full reconstruction is based on intermediate memory snapshots, retrieved trajectories, and checkpoint-level evaluation logs.

Hold-out evaluation is conducted with `max-tries=1` so that each checkpoint reflects the quality of the memory state itself rather than additional retry-and-reflect behavior.

G.1 The Hold-out Question

Question 87:

A $1\frac{1}{2}$-in schedule-40 steam pipe is laid in the atmosphere where the temperature is 50 °F. The steam inside it is saturated at 100 psia. Consider the pipe to be a grey body and uninsulated. The coefficient of heat transfer by natural convection from the outside surface is 2.0

Btu/(hr·ft^{2}·°R). Calculate the amount of steam condensed per hour per unit length of pipe.
A. 0.65 B. 0.50 **C. 0.70** D. 0.55 E. 0.40
F. 0.80 G. 0.75 H. 0.90 I. 1.00 **J. 0.60**

The question belongs to the **Heat Transfer / Thermodynamics** category. The standard solution accounts for both natural convection and thermal radiation from the pipe surface. At 100 psia, saturated steam has $T_s \approx 327.8°\text{F}$ and $h_{fg} \approx 880$ Btu/lbm. Using the 1.5-in schedule-40 outside diameter $D_o \approx 1.90$ in $= 0.158$ ft, the outside area per foot is

$$A = \pi D_o L \approx \pi \cdot 0.158 \cdot 1 = 0.497 \text{ ft}^2/\text{ft}.$$

The convective heat loss is

$$\dot{Q}_{\text{conv}} = hA(T_s - T_\infty) = 2.0 \cdot 0.497 \cdot (327.8 - 50) \approx 276 \text{ Btu}/(\text{hr} \cdot \text{ft}).$$

Because the pipe is grey-body and uninsulated, radiation must also be included:

$$\dot{Q}_{\text{rad}} = \epsilon\sigma A(T_s^4 - T_\infty^4).$$

Using a typical grey-body emissivity for an oxidized metal surface, the radiative loss is on the same order as convection, yielding total heat loss about

$$\dot{Q}_{\text{tot}} \approx 530 \text{ Btu}/(\text{hr} \cdot \text{ft}).$$

Therefore,

$$\dot{m} = \frac{\dot{Q}_{\text{tot}}}{h_{fg}} \approx \frac{530}{880} \approx 0.60 \text{ lb}/(\text{hr} \cdot \text{ft}).$$

Thus, the standard answer is $\boxed{\text{J}}$.

G.2 Memory Evolution and Prediction Transition

We reconstruct the full 20-slot ExpeL-MT insight pool from intermediate memory snapshots to analyze how thermo-related rules evolve throughout the sequential stream. A rule is considered thermo-related if it contains terms such as *steam, condensation, enthalpy, heat transfer, latent heat,* or h_{fg}. Representative early-stage rules include:

- *"Calculate heat transferred using final and initial enthalpy, considering temperature changes, specific heat capacities, and molecular weight for unit conversion."*
- *"Calculate exit temperature of a fluid in a heat exchanger..."*
- *"Calculate initial quality..."*

During the early phase of the stream, which primarily contains Thermodynamics and Heat Transfer questions, the bounded insight pool gradually accumulates thermo-related rules. However, after the stream transitions to electrical engineering and communication topics, newly inserted rules progressively overwrite the earlier thermo-related insights.

ExpeL-MT relies on two memory sources during inference: (1) retrieved top-k trajectories from the experience pool, and (2) an evolving insight pool produced through reflection and summarization. To identify the source of failure, we reconstruct both components across intermediate memory stages. Table 14 summarizes the prediction trajectory together with the number of thermo-related rules retained in memory. Prediction correctness strongly correlates with thermo-rule retention: the model remains mostly correct while thermo-related insights are preserved, but becomes consistently unstable once these rules disappear after $K = 582$.

Importantly, retrieval quality does not degrade over time. Both early and late checkpoints retrieve highly relevant steam-condensation trajectories, and the retrieval similarity score increases from

$$0.678 \rightarrow 0.753.$$

Thus, the retrieved top-k trajectories remain semantically relevant throughout the stream, indicating that the downstream degradation cannot be attributed to retrieval failure.

Table 14: Prediction trajectory of the hold-out thermodynamics question under evolving ExpeL-MT memory states.

Checkpoint K	Pred.	Correct?	# Thermo Rules	Model Behavior
97	J	✓	5	Uses correct thermo constants and computes $\dot{m} \approx 0.60$, mapped to J.
194	D	✗	6	Mis-applies LMTD and obtains an incorrect temperature difference.
291	J	✓	4	Same reasoning path as $K = 97$.
388	J	✓	3	Correct thermo grounding preserved.
485	J	✓	1	Final checkpoint with stable thermo-context grounding.
582	A	✗	0	Thermo-related insights disappear; latent heat value drifts and $\dot{m}$ shifts downward.
679	E	✗	0	Saturation temperature also drifts after thermo-context removal.
776	C	✗	0	Partial recovery of thermo constants but incorrect option selection.
873	B	✗	0	Prediction remains unstable after complete thermo-context overwrite.

Instead, the failure originates from corruption within the evolving insight pool. As thermo-related insights are progressively replaced by unrelated electrical-engineering rules, the model loses the thermo-specific contextual anchors required for stable reasoning. The transition approximately coincides with the disappearance of thermo-related rules from the bounded insight pool. By the final checkpoint, all 20 insight slots are occupied by unrelated rules involving SCR currents, signal-processing formulas, torque equations, capacitance relations, and Reynolds-number similarity constraints. These observations suggest that the degradation is primarily caused by destructive memory updates rather than insufficient retrieval quality or backbone reasoning capability.

G.3 Failure Mechanism and Implication

The single-item trajectory closely mirrors the aggregate hold-out behavior across all 96 Engineering questions: strong early improvement is followed by gradual erosion and eventual regression toward the no-memory baseline. The case study therefore reflects a broader structural failure mode rather than an isolated prediction error.

This degradation emerges from the interaction between bounded memory capacity and sequential subject-localized updates. In ExpeL-MT, the insight pool is capped at:

$$\texttt{max-num-rules} = 20.$$

Once the pool becomes saturated, newly generated rules implicitly replace or edit earlier entries. However, the eviction process does not estimate downstream utility, subject importance, or future dependency. As a result, memory retention becomes dominated by recency rather than long-term usefulness.

The effect is amplified by the subject-blocked ordering of the online stream. Neighboring tasks frequently belong to the same subject category, causing memory updates to become highly concentrated within local domains. During the early phase of the stream, thermo-related rules accumulate and stabilize prediction behavior on the hold-out Heat Transfer question. However, after the stream transitions into electrical-engineering and communication topics, newly inserted rules progressively overwrite the earlier thermo-related insights.

Importantly, the failure is not caused by retrieval degradation. Retrieved top-k trajectories remain highly relevant throughout the stream, and retrieval similarity even increases over time. Instead, the

Table 15: Controlled capacity ablation for ExpeL-MT. "Erased subdomains" reports the number of subdomains that contributed at least one rule during the stream but have no surviving rule at the end. The capacity of 20 matches the submitted setting and was evaluated in a separate run for this ablation.

`max-num-rules`	OnlineAcc	PED	Bound reached	Rules evicted	Erased subdomains	HoldOutAcc	$\mathrm{Trend}_{\mathrm{HO}}$
5	0.708	0.078	Task 16	19	2/6	0.688	$+0.000$
20	0.692	0.093	Task 61	28	1/9	0.677	-0.064
40	0.669	0.152	Task 188	11	0/7	0.677	-0.081

degradation originates from corruption within the bounded insight pool itself. Once thermo-related rules disappear from memory, the model loses the thermo-specific contextual anchors required for stable reasoning, leading to drifting constants, unstable intermediate calculations, and eventually persistent prediction failure.

Consequently, memory updates become competitive rather than accumulative: later subjects overwrite previously useful subject-specific context instead of consolidating it. Together, these dynamics produce a clear form of:

catastrophic forgetting through bounded memory eviction

G.4 Controlled Capacity Ablation

The preceding case study identifies bounded insight retention as one concrete mechanism through which later ExpeL-MT updates can destroy previously useful information. To test whether this observation is merely an artifact of the submitted capacity setting, we conduct a controlled capacity intervention by varying `max-num-rules` over $\{5, 20, 40\}$. This capacity ablation was conducted separately from the main experiments, including a rerun of the default capacity of 20. We keep ExpeL-MT's original ADD, EDIT, REMOVE, and AGREE operations and its strength-based truncation mechanism unchanged. The intervention therefore isolates the effect of insight-pool capacity without redesigning the underlying memory method.

Table 15 confirms that capacity affects retention. Increasing `max-num-rules` delays saturation from task 16 to task 188 and reduces complete subdomain loss from two of six represented subdomains to zero of seven. This supports the case-study mechanism: when the insight pool is small, subsequent updates are more likely to remove all surviving rules associated with an earlier subdomain.

However, increasing capacity does not consistently improve performance. As the capacity increases from 5 to 40, $\mathrm{OnlineAcc}$ decreases from 0.708 to 0.669, while PED increases from 0.078 to 0.152. Final $\mathrm{HoldOutAcc}$ also does not improve, and $\mathrm{Trend}_{\mathrm{HO}}$ becomes more negative. Thus, simply retaining more rules reduces information loss but does not guarantee that the larger memory can be retrieved and used effectively throughout the stream.

Taken together, the traced failure case and the controlled intervention show that bounded capacity is a contributing mechanism for forgetting in ExpeL-MT, but capacity alone does not explain or solve the problem. The results instead reveal a trade-off among retention, effective memory use, and trajectory stability. This analysis is specific to one representative bounded-memory method; systems with different update, compression, retrieval, and retention mechanisms may exhibit different causes of forgetting.

H Limitations

Task-order sensitivity. A limitation of our current study is that most of the main experiments are conducted under a fixed task order for each dataset. This choice enables controlled comparison across memory methods, but it does not fully characterize uncertainty induced by different sequential orders. In principle, SeqMem-Eval can be applied to multiple randomized task streams and report confidence intervals for the metrics. In practice, this is computationally expensive because each task order requires rerunning the full sequential memory process, including retrieval, memory updates, hold-out evaluation at intermediate checkpoints, and retrospective evaluation for BWT and forgetting. For LLM-based memory methods such as reflection- or synthesis-based approaches, this cost is

further amplified by additional API calls and generated tokens. We therefore leave large-scale multi-order uncertainty estimation as an important direction for future benchmark extensions.

Scope of memory mechanisms. Our evaluation focuses primarily on prompt-based sequentially evolving memory, where the base LLM remains fixed and memory is updated through retrieval, summarization, reflection, workflows, or other external text-based mechanisms. This scope matches many current memory-augmented agent systems and allows us to compare diverse methods under a unified test-time protocol. However, another emerging direction is training-based evolution, where experience is used not only to update an external memory state but also to improve policies, skills, or model behavior through learning. For example, recent work such as SkillRL Xia et al. (2026) studies recursive skill-augmented reinforcement learning, where a skill library co-evolves with the agent's policy during RL optimization. Such training-based approaches may exhibit different memory dynamics, including longer-term policy adaptation, different forms of transfer, and different efficiency trade-offs. While SEQMEM-EVAL's diagnostic perspective may still be useful for analyzing these systems, our current experiments do not directly evaluate training-based evolving agents. Extending the framework to jointly assess external memory evolution and parameter- or policy-level adaptation is an important direction for future work.

Evaluation cost of diagnostic metrics. Although SEQMEM-EVAL provides a more fine-grained view of memory behavior, some of its metrics require additional evaluation beyond standard online accuracy. For example, hold-out generalization requires evaluating intermediate memory states on unseen tasks, while BWT and forgetting require revisiting previously encountered tasks under later memory states. These retrospective evaluations increase the total number of LLM calls, especially for long task streams or methods with expensive memory updates. In this work, we mitigate this cost by evaluating BWT and forgetting over selected temporal horizons and by sampling memory checkpoints for hold-out evaluation. Future work could explore more efficient approximations, such as adaptive checkpoint selection, smaller diagnostic probes, or surrogate estimators for memory quality.