FROM MEMORY TO ACTION: BENCHMARKING LONG-TERM AGENT MEMORY IN PERSONAL ASSISTANT ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Always-on personal assistants are an emerging application of agents powered by large language models (LLMs). Unlike conventional assistants, they operate over long-term, complex streams of user activities and agent interactions, accumulating information that is essential for tasks with ambiguous instructions, evolving user preferences, prior feedback, and reusable experience. This places substantial demands on agent memory, not only to form and maintain useful memories, but also to correctly apply them during downstream task execution. Existing benchmarks, however, either evaluate memory largely in isolation from end-task success or rely on simplified interaction histories that do not reflect the complexity of sustained daily use. As a result, there remains a gap in evaluating agent memory end to end, from memory formation to its eventual effect on task completion. To address this gap, we introduce a comprehensive memory benchmark comprising 100 tasks with fine-grained target-memory attribution, enabling separate diagnosis of *Formation, Availability, Application, and Outcome*. Evaluations across different LLMs show that the largest performance loss occurs after relevant memory becomes available but before it is translated into the intended action, indicating that memory utilization, rather than storage or retrieval alone, is a major bottleneck. Controlled memory-swap experiments further reveal that the utility of a generated memory depends strongly on the downstream executor that uses it. We hope these insights can guide future advances in agent memory for practical applications.

1 INTRODUCTION

Personal assistants operate over long periods and across applications, accumulating a history of conversations, preferences, habits, constraints, and unfinished tasks (Lin et al., 2026; Cheng et al., 2026). Some of this history may later shape a decision, while much of it is irrelevant. Yet the user requests that determine which parts of this history matter arrive only later.

This temporal gap creates a central challenge for long-term agent memory (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025; Li et al., 2026). The agent must form and maintain persistent memory as user activity accumulates, before it knows which information a later request will require (Chen et al., 2026; Liu et al., 2026c;b;a). Once a request arrives, the retained memories may shape how the agent interprets it and acts on it. Evaluating long-term memory therefore requires asking two questions: whether the information needed for the request was retained, and whether the retained memories helped the agent act correctly (Shen et al., 2026; He et al., 2026).

To answer these questions, we build an agent memory benchmark based on Claw-Anything (Lin et al., 2026). Before downstream tasks appear, agents review a long-term personal activity history in stages and form persistent memory. Afterward, they use these memories to complete multi-step personal-assistant tasks within the user's current digital environment, acting across multiple applications (Ye et al., 2026; Lin et al., 2026). The tasks cover diverse user needs and operations. Different tasks draw on different memories, each formed from the long-term activity preceding that task.

The benchmark contains 100 tasks and provides fine-grained mappings from target-memory annotations to specific task decisions and task-level scoring criteria. We compare w/o Memory with w/

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.

Memory to measure how persistent memory affects final task performance. The mappings then distinguish failures of Formation, Availability, and Application: the agent may fail to form task-relevant information as persistent memory, lack access to that memory during execution, or see it but fail to take the correct action.

Across multiple evaluated models, w/ Memory improves Pass Rate and average Task Score over w/o Memory. Most target information becomes available in the execution context, but only a subset is converted into correct actions. Controlled memory swaps further show that available memories differ in their ability to support downstream action under a fixed executor. Changing the memory source produces a larger Task Score gain with the Claude Opus 5 executor than with the Qwen3.8-27B executor, indicating that the effect of a memory source is executor-dependent. Memory-SFT-Qwen3.8-27B (ours), trained on both consolidation and execution trajectories, further improves end-to-end performance across the full memory pipeline. Controlled substitutions suggest that these gains come primarily from the executor.

In summary, our contributions are fourfold. 1) We build an agent memory benchmark based on Claw-Anything that tests whether agents can form memories from a mixed long-term personal activity history before downstream tasks appear and use them in cross-application action. 2) We provide 100 tasks with fine-grained annotations that connect target memories to executable task decisions and task-level scoring criteria. 3) We analyze the gaps among Formation, Availability, Application, and Outcome, and use fixed-executor memory swaps to study how downstream utility varies with memory source and executor. 4) We systematically evaluate multiple models and construct training trajectories that cover Formation and Application to study whether supervised fine-tuning can improve both stages.

2 Related Work

Agent memory systems. Agent memory systems differ in how they write, organize, consolidate, and retrieve persistent information (Park et al., 2023; Zhong et al., 2023; Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Li et al., 2025). We hold the agent scaffold and memory interface fixed and test whether formed memories support correct downstream action.

Long-horizon conversational and personal memory. LoCoMo, LongMemEval, and MemoryAgentBench evaluate retrieval, temporal reasoning, knowledge updates, and selective forgetting over long conversations or multi-session histories (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025), while LoCoMo-Plus studies the semantic gap between an implicit cue and information needed for a later response (Li et al., 2026). These benchmarks establish core evaluations for long-term personal memory; our focus is whether persistent memory later changes executable decisions and actions.

Memory operations and stage-wise diagnosis. HaluMem, AMemGym, Streaming Memory Benchmark, and WorldMemArena go beyond a single end-to-end score by exposing memory operations or lifecycle stages for diagnosis (Chen et al., 2026; Cheng et al., 2026; Liu et al., 2026c;a). We apply this perspective to cross-application tasks by separately analyzing Formation, Availability, and Application and linking each target-memory annotation to memory-bearing decisions, task-level scoring criteria, and final environment outcomes.

Memory-guided agent action. Mem2ActBench and MemoryArena evaluate whether long-term memory can guide downstream action, spanning grounded tool calls and explicitly interdependent multi-session task chains (Shen et al., 2026; He et al., 2026). Such evaluations often organize histories around particular evidence, user states, or task dependencies. In our setting, agents first consolidate mixed long-term personal activity across time and applications, before downstream tasks appear. We then test whether the needed information enters persistent memory and supports multi-step action.

Personal-assistant agent benchmarks. Claw-Eval evaluates autonomous agents in executable software environments with multi-step workflows, while Claw-Anything focuses on always-on personal assistants with long-horizon activity histories and cross-service state (Ye et al., 2026; Lin et al., 2026). Claw-Anything provides our long-horizon activity, cross-service state, and executable-task substrate; we add the memory layer, target-memory annotations, the w/o Memory versus w/ Memory protocol, and formation-to-action diagnostics.

3 TASK AND EVALUATION PROTOCOL

Long-running personal assistants accumulate histories of user conversations and activities. Their usefulness depends on retaining relevant information in persistent memory for later judgments and actions. To distinguish failures along this chain, we divide evaluation into a temporally ordered consolidation phase and a subsequent, independent task-execution phase. Hidden annotations connect historical evidence, target memories, and the task requirements they constrain.

3.1 LONG-TERM PERSONAL MEMORY TASKS

Each memory task comprises a long-term personal activity history, the current environment at task time, persistent memory formed by the agent, a subsequent user task, and an executable scoring specification. The history records timestamped conversations and activities, while the current environment reflects their lasting effects and defines what the agent can observe and do during task execution.

The subsequent task requires the agent to combine user-specific information from the history with objective facts in the current environment. Completing the task involves more than restating a past preference, constraint, or updated choice. The agent must determine whether that information applies to the available options and then use it to select a tool, choose its arguments, or produce an answer.

To analyze the role of historical information, the benchmark stores one or more target memories for each task. A target memory is an out-of-band annotation of the historical information on which the task is expected to depend; it is never provided to the evaluated agent. It is distinct from the persistent memory that the agent forms in free text or another representation. Evaluation does not require persistent memory to reproduce the annotation's schema or wording; it need only preserve the target information semantically. Availability and Application then assess whether that information reaches the agent during execution and is used correctly for a judgment, answer, or action.

3.2 FROM CONSOLIDATION TO TASK EXECUTION

During consolidation, the memory-formation phase, the agent processes several segments of long-term activity in chronological order, as illustrated in Figure 1. This resembles a practical assistant periodically reviewing recent conversations and activities and deciding what information is worth retaining as persistent memory before it knows which later request will need it. Each session contains only the history from the current period and previously saved persistent memory; it contains no future history, subsequent task, or target-memory annotation. Short-term context is independent across sessions, and only saved persistent memory carries over. The agent must therefore decide, without seeing a future request, what information to retain over the long term.

After processing the history, the agent enters an independent execution session. The executor receives the user request, current environment, and tools, but no longer has direct access to the raw long-term history. Under w/ Memory, it additionally receives the persistent memory produced during consolidation; under w/o Memory, it does not. Neither condition exposes target-memory annotations, evidence links, or the scoring specification.

The w/o Memory and w/ Memory conditions use the same model, task, environment, tools, execution budget, and scorer. The only difference relevant to the research question is whether persistent memory is provided to the executor. Memory Gain therefore measures the change in task outcome from introducing the model's own long-term memory while holding the task and execution conditions fixed.

3.3 TARGET-MEMORY ATTRIBUTION AND LIFECYCLE DIAGNOSIS

Comparing only complete-task outcomes cannot reveal whether a piece of historical information was lost during memory formation, was unavailable at execution time, or was visible but failed to affect the resulting action. The benchmark therefore records, for each target memory, its supporting historical evidence, the memory-bearing task requirement it supports, and the corresponding scoring

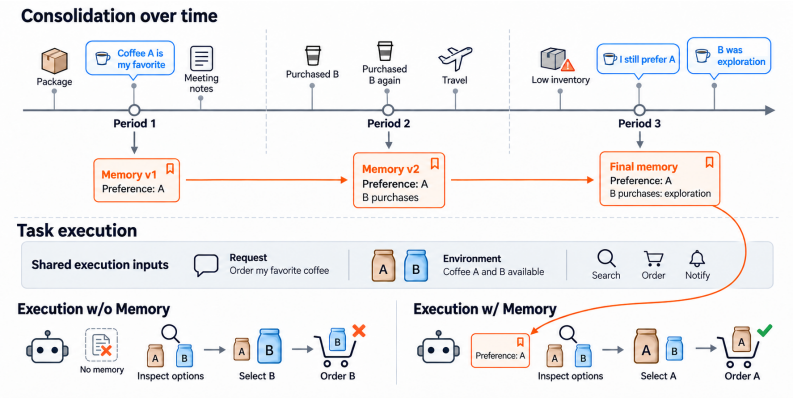

Figure 1: Memory formation and use in a personal-assistant task. In the memory-formation phase (consolidation), the agent reviews each chronological period—like summarizing completed conversations and activity before a later interaction—and updates its persistent memory. In this example, it records that coffee A is the user's preference and that purchases of B were exploratory. Later, in an independent memory-use phase (task execution), the agent receives the request "Order my favorite coffee," the currently available options, and tools, but not the raw history. Without persistent memory, the depicted agent selects B; with the saved memory, it selects and orders A.

check. These mappings are model-invisible, out-of-band annotations and are not rendered into the timeline, persistent memory, or task prompt.

Lifecycle analysis uses a task–target-memory observation as its unit. A task that depends on two target memories contributes two observations. The 100 tasks in our benchmark contain 143 observations, each of which is evaluated separately for Formation, Availability, and Application.

Formation assesses whether the target information is correctly written into the agent's final persistent memory. Availability is assessed only for Formed targets. It asks whether that formed information reaches the agent in the execution session, either in the system prompt or after a memory lookup. Application is assessed only for targets that are Available and checks whether the information is correctly realized as the corresponding judgment, answer, or action.

For an Available target, Application distinguishes four outcomes: the memory-supported requirement is satisfied and the complete task passes; the memory-supported requirement is satisfied but another requirement fails; the memory-supported requirement is missing; or the agent attempts the memory-supported requirement but makes an incorrect judgment or action. The first two outcomes count as Applied correctly, whereas the latter two are Application failures.

Application measures the local task requirement supported by each target memory, whereas Pass Rate and Task Score measure the complete task outcome. The complete grader also aggregates other scoring criteria. Correctly applying a target memory therefore does not necessarily imply that the complete task passes, and complete-task failure does not necessarily imply that the target memory was used incorrectly.

4 Benchmark Construction

Building a benchmark for long-term personal assistants requires histories that contain useful personal information before the future task is known, while ensuring that the task genuinely depends on that information. To support this setting, we first adapt the data-construction setup of Claw-Anything (Lin et al., 2026): we retain its digital environments and service interfaces, while adding long-term activity histories, target-memory annotations, and checks for genuine memory dependence. The adapted construction process jointly generates histories, task instances, and their evaluation specifications, followed by automated validation and human review. Figure 2 summarizes this construction flow.

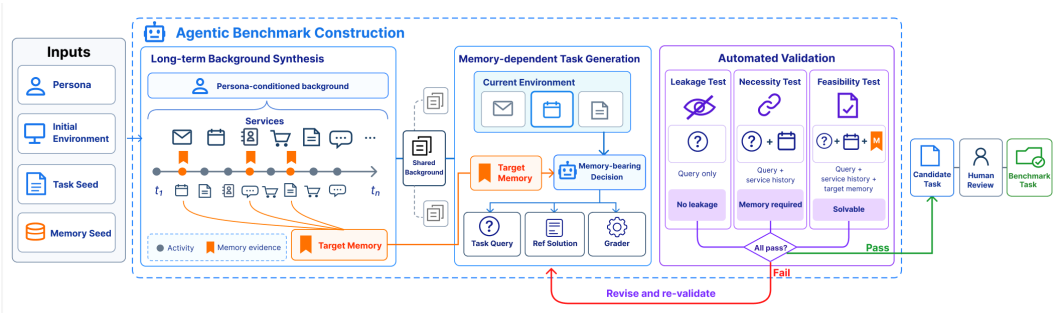

Figure 2: Overview of the agentic pipeline for constructing memory-dependent benchmark instances. The pipeline first synthesizes a shared long-term background. Task-specific branches then generate the current environment, task and scoring artifacts, and target-memory annotations. Leakage, necessity, and feasibility tests validate each candidate. Failed candidates are revised and re-validated. Candidates that pass automated validation undergo human review before becoming benchmark tasks.

We implement this adapted setup as an agentic generation pipeline: a generation agent plans content, produces artifacts, and interprets failures, while deterministic components enforce schemas, IDs, transactions, derived state, and validation. The process first constructs a shared long-term background and its baseline environment. It then creates task-specific branches from this shared state. Each branch generates the task request, reference solution, scoring specification, and other task-related artifacts, and may modify part of the current environment while preserving the rest of the shared environment.

On top of this adapted setup, we add three memory-specific capabilities. The generator creates personal information within long-term activity that can support later tasks, records hidden mappings among historical evidence, target memories, and task requirements, and uses context-controlled counterfactual probes to test prompt leakage, completion without long-term memory, and solvability when the correct memories are provided.

4.1 Constructing Memory-Dependent Tasks

Long-term background generation. For a persona, the generator creates conversations, activity records, cross-service state, and irrelevant noise across multiple periods. Target memories can be stated directly, reflected in repeated behavior such as recurring choices, or inferred by combining separate records that jointly reveal a preference or impose a constraint. These memories are interleaved with similar entities, outdated states, and unrelated activity in a continuing timeline shown to the evaluated agent as long-term history, rather than written as a short prompt centered on a future task.

Target-memory annotations and structure. The generator stores structured target-memory annotations outside the timeline. Only natural conversations and activity facts are rendered into the timeline shown to the evaluated agent during the history phase, so the agent cannot read the target memories directly. A memory brief controls the hidden structure along three independent axes: topology specifies the number of target memories and their relations, trace specifies how each memory is expressed across the history, and cue specifies whether the information needed to trigger retrieval appears in the user request or the current environment.

Task branching. Once the long-term background is frozen as an immutable checkpoint, the pipeline creates independent task branches from it. The generation agent declares the target memories on which a task actually depends and receives a seed task assigned by code. The former determines what user information the task must follow and how multiple pieces of information combine; the latter supplies the scenario and basic task structure. Both must be grounded in a natural current situation.

The current environment provides objective candidates, while target memories provide the basis for a personal choice. The generator jointly produces the task request, reference solution, answer sheet, and grader. Together, these artifacts specify what the task asks the agent to do, which memories it must use, and how successful execution is checked. Each branch has its own current environment and scoring artifacts and cannot write back to the shared background or leak information to other tasks.

4.2 Validation and Quality Control

Each candidate instance first undergoes deterministic structural checks. These checks verify schemas, IDs, entity references, and tool-call arguments, ensure that scoring criteria can be evaluated from the execution trace or final state, and confirm that each target memory supports a task requirement.

The pipeline then performs context-controlled validation with independent sidecar readers that do not share conversational state. A reader that receives only the task request checks whether the prompt directly reveals the answer. A second reader receives the task request and current environment but no target memories, and checks whether the task can be completed without reading or using the long-term history. A third reader receives the same current information together with the correct target memories, and checks whether the task admits an executable path that applies those memories.

These probes provide quality control during construction, but they are not the model-evaluation metrics defined in Section 3.3. The sidecar readers use the correct target memories stored by the benchmark, whereas the executor under the formal w/ Memory condition can use only the persistent memory formed by the model during consolidation. The probes therefore test whether a task depends on memory as designed and is solvable when the correct memory is given. Formation, Availability, and Application instead measure whether an evaluated model forms the needed memory, carries it into the task, and uses it to act.

The automated validation and probe results are returned to the generation agent, which uses them to revise the task artifacts or discard failed candidates. Any unresolved blocking finding prevents the instance from being completed. Instances that pass automated validation also undergo reference execution and human review. These steps confirm that the historical evidence supports the target memory, the task naturally depends on that information, and the grader rewards correct behavior while rejecting clear errors and bypass paths.

4.3 Benchmark Composition

The final benchmark contains 100 task instances and 143 task–target-memory observations across 13 personas in seven role domains, including technology and product, organizational and business operations, and education and academia. Each instance is paired with a 121-day personal activity history (163–269 activity records and approximately 257k–411k characters) and a current digital environment.

Each history contains at least 25 potentially memorable pieces of personal information, preferences, habits, constraints, or state changes, most irrelevant to a given task. Each task depends on one to four target memories: the agent must form them from the long-term history before execution and combine them with current objective facts to make the correct judgment, answer, or action. The target-memory annotations are out-of-band and define memory dependence for Formation, Availability, and Application attribution. In 86 tasks, success requires at least one specified tool operation, such as creating a calendar event, sending an email, or updating a record; across these tasks, agents must perform 128 required operations. The remaining 14 tasks are scored by a judgment or answer without a specified tool operation. Task branches have separate current environments and scoring artifacts and cannot write back to shared histories; they are isolated at execution, although shared-history tasks are not fully independent history samples.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate Qwen3.6-27B and Qwen3.8-27B Base (Qwen Team, 2026a;b), our fine-tuned Memory-SFT-Qwen3.8-27B (ours), and Claude Opus 5 (Anthropic, 2026) using the same OpenHarness agent scaffold (HKUDS Teams, 2026), tool interfaces, system prompt, and execution environment. We compare w/o Memory with w/ Memory and define Memory Gain as w/ Memory minus w/o Memory. Complete-task graders produce Pass Rate and continuous Task Score.

5.2 MAIN RESULTS

As shown in Table 1, w/ Memory improves Pass Rate by 26–42 percentage points and Task Score by 0.240–0.308 across all four models. This result shows a consistent end-to-end benefit of memory across the evaluated models and tasks. Among the three evaluated Qwen models, Memory-SFT-Qwen3.8-27B achieves the highest w/ Memory Pass Rate and Task Score, as well as the largest Memory Gain.

5.3 FROM MEMORY AVAILABILITY TO ACTIONABILITY

5.3.1 LIFECYCLE ANALYSIS

The 100 benchmark tasks contain 143 task–target-memory pairs, which serve as the unit of lifecycle analysis because one task may depend on multiple target memories. Table 2 reports how many target memories each model formed, had available in the effective execution context, and applied correctly.

For the three non-SFT models, 81.1%–85.3% of target memories are available in the execution context, whereas only 51.7%–68.5% are ultimately applied correctly. The loss from Availability to Application is therefore much larger than the loss from Formation to Availability. The main bottleneck arises after the memory has entered the effective context but before it produces the target behavior.

For target memories that were available to the agent during task execution, Table 3 separates four mutually exclusive and exhaustive outcomes. The first two indicate correct Application and differ only in whether the complete task passes. The final two indicate Application failures in which the target behavior is either omitted or performed incorrectly. Here, target behavior refers to the memory-bearing decision or action constrained by the target memory. Percentages use the target memories available to each agent as the denominator.

Across the four models, 30.3%–37.1% of target memories available to the agent are applied correctly but still do not produce a complete task pass because other requirements remain unmet. Qwen3.8-27B Base and Qwen3.6-27B have similar correct-Application rates among these target memories (64.5% versus 63.8%), but their failure profiles differ: Qwen3.8-27B Base more often omits the target behavior, whereas Qwen3.6-27B more often performs it incorrectly. For Qwen3.8-27B Base, these omissions included redundant confirmation and stopping after a recommendation or draft instead of completing an explicitly authorized action. Opus 5 has the highest correct-Application rate among target memories available to the agent (80.3%) and the lowest rate of omitted target behavior

Table 1: End-to-end results under w/o Memory and w/ Memory. Memory Gain is computed as w/ Memory minus w/o Memory. The italicized row reports the change from Qwen3.8-27B Base to Memory-SFT-Qwen3.8-27B (ours).

	w/o Memory		w/ Memory		Memory Gain	
Model	Pass Rate	Task Score	Pass Rate	Task Score	Pass Rate	Task Score
Qwen3.6-27B	12.0%	0.2998	38.0%	0.5401	+26.0 pp	+0.2403
Qwen3.8-27B Base	13.0%	0.3012	44.0%	0.5529	+31.0 pp	+0.2517
Memory-SFT-Qwen3.8-27B (ours)	13.0%	0.3465	55.0%	0.6502	+42.0 pp	+0.3037
Gain over Qwen3.8-27B Base	*0.0 pp*	*+0.0453*	*+11.0 pp*	*+0.0973*	*+11.0 pp*	*+0.0520*
Opus 5	21.0%	0.4027	58.0%	0.7110	+37.0 pp	+0.3083

Table 2: Formation, Availability, and Application over 143 task–target-memory pairs. Parentheses show percentages of all 143 pairs.

Model	Formed	Available	Target memory applied correctly
Qwen3.6-27B	120 (83.9%)	116 (81.1%)	74 (51.7%)
Qwen3.8-27B Base	129 (90.2%)	121 (84.6%)	78 (54.5%)
Memory-SFT-Qwen3.8-27B (ours)	132 (92.3%)	126 (88.1%)	94 (65.7%)
Opus 5	130 (90.9%)	122 (85.3%)	98 (68.5%)

Table 3: Outcomes for target memories available to the agent during task execution. Target behavior is the memory-bearing decision or action constrained by the target memory. The four columns are mutually exclusive and exhaustive within each model; percentages use the target memories available to that agent as the denominator.

Model	Task passed	Other requirements unmet	Target behavior omitted	Target behavior incorrect
Qwen3.6-27B	31 (26.7%)	43 (37.1%)	7 (6.0%)	35 (30.2%)
Qwen3.8-27B Base	38 (31.4%)	40 (33.1%)	20 (16.5%)	23 (19.0%)
Memory-SFT-Qwen3.8-27B (ours)	53 (42.1%)	41 (32.5%)	8 (6.3%)	24 (19.0%)
Opus 5	61 (50.0%)	37 (30.3%)	5 (4.1%)	19 (15.6%)

(4.1%). Nevertheless, 30.3% of its available target memories are applied correctly without producing a complete task pass.

5.3.2 Controlled Memory Swaps

Binary Formation and Availability measures do not capture whether a memory can support downstream execution. We therefore independently vary the memory source and executor to separate their downstream effects.

Table 4 shows both directions of the component substitution. Replacing Qwen-generated Memory with Opus-generated Memory changes Pass Rate by 1 percentage point and Task Score by 0.0176 under the Qwen3.8-27B Base executor, compared with 9 percentage points and 0.0988 under the Opus 5 executor. Replacing the Qwen3.8-27B Base executor with the Opus 5 executor produces larger point-estimate gains: 6 percentage points and 0.0598 with Qwen-generated Memory, and 14 percentage points and 0.1410 with Opus-generated Memory. In this comparison, the downstream usefulness of generated memory depends on the agent using it, rather than on the memory source alone.

5.4 Effects of Supervised Fine-Tuning

The training data cover both consolidation and execution, with 213 execution trajectories and 279 fully recorded consolidation phases. As shown in Table 1, Memory-SFT-Qwen3.8-27B improves w/ Memory Pass Rate from 44.0% to 55.0% and Task Score from 0.5529 to 0.6502 relative to Qwen3.8-27B Base. Its Task Score increase is larger under w/ Memory than under w/o Memory, expanding the Task Score Memory Gain from 0.2517 to 0.3037. Table 2 further shows that Memory-SFT-Qwen3.8-27B increases correct Application from 54.5% to 65.7%. For target memories available to the agent, Table 3 shows that omitted target behavior decreases from 16.5% to 6.3%, while incorrect target behavior remains at 19.0%. The primary behavioral change after training is therefore a reduction in failures to carry an explicitly authorized task through to action, rather than a reduction in incorrect behavior after action begins.

To attribute the training improvement, Table 5 independently varies the memory source and executor before and after training.

With the Qwen3.8-27B Base executor fixed, replacing Qwen-generated Memory with SFT-generated Memory changes Pass Rate by -1 percentage point and increases Task Score by 0.0024. With the Memory-SFT-Qwen3.8-27B executor fixed, the same replacement increases Pass Rate by 9 percentage points and Task Score by 0.0530. Conversely, with Qwen-generated Memory fixed, replacing

Table 4: Controlled substitutions between Qwen3.8-27B Base and Opus 5 components. Absolute cells report Pass Rate (Task Score). Δ Memory replaces Qwen-generated Memory with Opus-generated Memory under a fixed executor, whereas Δ Executor replaces the Qwen3.8-27B Base executor with the Opus 5 executor under a fixed memory source.

Executor	Qwen-generated Memory	Opus-generated Memory	Δ Memory
Qwen3.8-27B Base executor	44% (0.5529)	45% (0.5705)	+1 pp (+0.0176)
Opus 5 executor	50% (0.6127)	59% (0.7115)	+9 pp (+0.0988)
Δ Executor	+6 pp (+0.0598)	+14 pp (+0.1410)	–

Table 5: Controlled substitutions between Qwen3.8-27B Base and Memory-SFT-Qwen3.8-27B components. Cells report Pass Rate (Task Score); deltas replace the Base component with its Memory-SFT counterpart while holding the other component fixed.

Executor	Qwen-generated Memory	SFT-generated Memory	Δ Memory
Base executor	44% (0.5529)	43% (0.5553)	−1 pp (+0.0024)
Memory-SFT executor	46% (0.5972)	55% (0.6502)	+9 pp (+0.0530)
Δ Executor	+2 pp (+0.0443)	+12 pp (+0.0948)	–

the Qwen3.8-27B Base executor with the Memory-SFT-Qwen3.8-27B executor increases Pass Rate by 2 percentage points and Task Score by 0.0443. With SFT-generated Memory fixed, the same executor replacement increases Pass Rate by 12 percentage points and Task Score by 0.0948. Averaged across the two memory sources, the Memory-SFT-Qwen3.8-27B executor improves Task Score by 0.0696. These controlled substitutions therefore provide clearer evidence for improved task execution than for improved memory generation.

We additionally compare the jointly trained model with a model trained only on the same 213 execution trajectories. Because the joint setting additionally includes consolidation trajectories, this comparison is not data-matched. The jointly trained model has a lower w/o Memory Task Score than the execution-only model (0.3465 versus 0.3967), but a higher w/ Memory Task Score (0.6502 versus 0.6062), yielding a larger Memory Gain (0.3037 versus 0.2095).

6 Conclusion

We evaluate long-term agent memory by asking whether information retained from personal activity histories supports correct cross-application decisions and actions in later personal-assistant tasks. Across all four evaluated models, w/ Memory improves complete-task performance over w/o Memory, showing that persistent memory provides measurable end-to-end value. However, Availability does not guarantee Application: the largest loss occurs after target information reaches the effective execution context but before it produces the intended memory-bearing decision or action. Controlled memory swaps show that changing the memory source yields different gains under different executors. Memory-SFT improves performance across the full pipeline, with controlled substitutions suggesting that its gains come more from the executor than from the memory source. These findings suggest that long-term agent memory should be evaluated not only by what is written, stored, or retrieved, but also by whether retained information leads to correct downstream action.

Reproducibility statement

The task protocol, information visibility, target-memory attribution, and evaluation metrics are specified in Section 3. Section 4 describes the benchmark construction pipeline and validation procedures, and Section 5 reports the evaluated models, execution conditions, training setup, and outcome metrics. The appendix will provide the remaining construction, evaluation, and training details needed to reproduce the reported results.

AI use statement

We used generative AI tools to generate synthetic benchmark histories, task artifacts, and target-memory annotations through the construction pipeline described in Section 4, and to assist with

English writing and editing of this manuscript. Generated benchmark candidates were checked through the automated validation and human-review procedures described in Section 4. All AI-assisted manuscript text was reviewed and revised by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, 2026. Official model release.

Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. HaluMem: Evaluating hallucinations in memory systems of agents, 2026.

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

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

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks, 2026.

HKUDS Teams. OpenHarness: Open agent harness with a built-in personal agent–ohmo! https://github.com/HKUDS/OpenHarness, 2026. GitHub repository.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions, 2025.

Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. LoCoMo-Plus: Beyond-factual cognitive memory evaluation framework for LLM agents, 2026.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhenren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system, 2025.

Yusong Lin, Xinyuan Liang, Haiyang Wang, Qipeng Gu, Siqi Cheng, Jiangui Chen, Shuzhe Wu, Feiyang Pan, Lue Fan, Sanyuan Zhao, and Dandan Tu. Claw-anything: Benchmarking always-on personal assistants with broader access to user's digital world, 2026.

Chengzhi Liu, Yuzhe Yang, Sophia Xiao Pu, Yepeng Liu, Lin Long, Yichen Guo, Nuo Chen, Zhaotian Weng, Elena Kochkina, Simerjot Kaur, Charese Smiley, Xiaomo Liu, James Zou, Sheng Liu, Yuheng Bu, Songyou Peng, and Xin Eric Wang. WorldMemArena: Evaluating multimodal agent memory through action–world interaction, 2026a.

Guanming Liu, Yuqi Ren, Hansu Gu, Peng Zhang, Weihang Wang, Jiahao Liu, Ning Gu, and Tun Lu. StreamMemBench: Streaming evaluation of agent memory for future-oriented assistance, 2026b.

Guanming Liu, Haoran Yin, Tianchen Li, Sikuan Yan, Hongru Wang, Baian Chen, and Xiaoteng Ma. Streaming memory benchmark: Stage-level diagnosis with evidence dependency control, 2026c. URL https://openreview.net/forum?id=i1gkKNMX0K.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023.

Joon Sung Park, Joseph O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model. https://qwen.ai/blog?id=qwen3.6-27b, 2026a. Official model release.

Qwen Team. Qwen3.8-27B. https://huggingface.co/Qwen/Qwen3.8-27B, 2026b. Official model card.

Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2ActBench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8173–8190, 2026. doi: 10.18653/v1/2026.acl-long.370. URL https://aclanthology.org/2026.acl-long.370/.

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents, 2025.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-Eval: Toward trustworthy evaluation of autonomous agents, 2026.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 29733–29735, 2025. doi: 10.1609/aaai.v39i28.35383.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory, 2023.

A Evaluation Protocol Details

A.1 Phased Memory Consolidation

A task's long-term activity history consists of 41–45 chronologically ordered timeline files, divided into five consecutive consolidation phases. The first four phases contain nine timeline files each, and the fifth contains the remaining 5–9 files. Every phase uses a fresh conversational context. The model can read only the timeline files for the current phase and the persistent memory saved from previous phases; timeline files from other periods and service logs spanning the complete history are unavailable.

A consolidation session provides neither the subsequent task request nor application tools. The model can only use file tools to read the current phase's history and update the OpenHarness persistent-memory directory (HKUDS Teams, 2026). Each phase runs for at most 40 agent turns, with the simulated current date set to the end date of the last timeline file in that phase. At the end of the phase, the conversational context is discarded, while persistent memory remains available to the next phase.

This protocol ensures that earlier history can affect later phases only through persistent memory. The model cannot reread the complete raw history during task execution. The resulting persistent memory therefore reflects both whether information was ever written and whether it was retained through subsequent updates.

A.2 Independent Task Execution

After the five consolidation phases, the model performs the subsequent task in a fresh execution session. The simulated date for every execution session is June 15, 2026, and each session runs for at most 100 agent turns. The execution session provides the task request, current digital environment, and application tools, but not the raw timeline or historical service logs.

The w/ Memory and w/o Memory conditions use the same OpenHarness prompt configuration, including the same project-memory and relevant-memories blocks. Under w/ Memory, the native memory directory contains the persistent memory produced during consolidation. Under w/o Memory, execution begins with an empty memory directory, so the corresponding blocks contain no historical memory. The two conditions also use the same model checkpoint, agent scaffold, task request, current environment, tool interfaces, execution budget, and grader.

All formal evaluations of Qwen-family models use no-thinking mode. Table A1 summarizes the information available to the model in each stage.

Table A1: Information available during consolidation and task execution.

Information or capability	Consolidation	Execution: w/o Memory	Execution: w/ Memory
Current phase's timeline	Available	Unavailable	Unavailable
Raw history from other phases	Unavailable	Unavailable	Unavailable
Persistent memory	Read/write	Not provided	Final state provided
Subsequent task request	Unavailable	Available	Available
Current digital environment	Inaccessible through application tools	Accessible	Accessible
Application tools	Not provided	Provided	Provided
Target-memory annotations, evidence links, and scoring specification	Unavailable	Unavailable	Unavailable

B Agentic Benchmark Construction Details

B.1 Event-by-Event Generation of Long-Term Histories

For each persona, the pipeline first determines the history's time span, available services and their data schemas, scale constraints for service records (fixtures) and activities, and the memory structure to cover. The service and scale constraints determine the required coverage of the long-term environment. The memory brief specifies the number and relations of target information items, how they are expressed in the history, and whether the subsequent task obtains its retrieval cue from the user request or the current environment.

The long-term history grows incrementally, one event at a time. In each round, the pipeline selects a background seed compatible with the persona. A generation agent then instantiates it as a concrete event consistent with that persona and their existing experiences. One event may produce records in multiple services, corresponding activity logs, and natural conversations. For example, a work arrangement may leave an email, a calendar event, and a todo record.

The generation agent proposes the event content, while deterministic components check data formats, record IDs, temporal order, and cross-service references and commit only changes that pass these checks. Task-relevant personal information is generated in the same process as ordinary activity, similar entities, and irrelevant records. The resulting timeline remains a continuous personal activity history rather than a prompt written around one future task.

When an event expresses personal information that may affect a future decision, the pipeline also stores an out-of-band target-memory annotation. This annotation never enters the timeline; the evaluated model sees only natural conversations and activity facts. Background generation continues until the history satisfies the prescribed time span, service and activity coverage, fixture scale, and memory-structure requirements. The result is then saved as an immutable checkpoint that contains no future task request.

B.2 Task Branching, Validation, and Revision

A frozen checkpoint can provide the shared background for multiple sibling task branches, each of which starts from an independent copy. The generation agent first declares the target memories indispensable to the task from the currently valid memory candidates, after which code assigns a seed task compatible with the current branch. The target memories specify the user-specific information that the task must follow, while the seed task supplies the scenario and basic task structure.

The generation agent jointly produces changes to the current environment, the task query, reference solution, answer sheet, and grader so that the task intent, executable path, and scoring conditions remain aligned within the same branch. Deterministic components then check entity references, tools and arguments, scoring criteria, and target-memory linkage before compiling a passing draft into a runnable task.

After the structural checks pass, the pipeline runs the context-controlled probes described in Section 4.2. Each probe uses an independent context, and its result is returned to the generation agent as structured feedback that identifies the failure location and supporting evidence. The agent may modify only the activities, environment, task, or scoring artifacts that remain mutable within the current branch; a task branch cannot modify the shared checkpoint. Any revision that changes a probe input invalidates the previous result and triggers re-validation. An unresolved blocking error prevents the task from being completed.

C Key Prompts and Model-Visible Inputs

C.1 Consolidation Prompt

Every consolidation phase uses the same prompt template, changing only the persistent-memory path, the date range for that phase, and the timeline-file paths. Timeline contents are not inserted directly into the prompt; the model must read each file with a file tool. The complete template appears below, with double braces marking fields replaced at runtime.

```
# Recent activity

You are the user's long-running personal assistant,
building up a durable memory of their life as time goes on.
You keep that memory in a file at {{MEMORY_PATH}},
which carries across sessions.

Time has moved on.
Here is what the user has been up to lately,
over {{DATE_RANGE}} - open each file
with the file-read tool to see it: {{WEEK_LOG_PATHS}}
```

```
Whether any of this is worth remembering,
and what to note, is up to you.
If something here matters going forward,
update {{MEMORY_PATH}} as you see fit
(read it first if it already exists, so you build on it rather than overwrite it).
If nothing here is worth keeping, leave your memory unchanged and stop.
```

The template does not disclose the future task or identify which historical information corresponds to target memories. The model must decide what is worth retaining over the long term and, as it reads each new phase, whether to add to, correct, or preserve its existing memory.

C.2 Memory Input in the Execution Prompt

The execution session uses OpenHarness's shared personal-assistant system prompt and the task request, without adding a task-specific memory instruction containing the correct answer. Both conditions retain the OpenHarness memory prompt blocks. Under w/ Memory, these blocks expose the native memory content produced during consolidation; under w/o Memory, the corresponding directory is empty. Target-memory annotations, answer sheets, graders, and lifecycle labels never enter the model input under either condition.

D Representative Application Case

Task `ECOM01_T01_place_the_q3_candle_reorder_with_firesid` asks the agent to complete Fireside's third-quarter candle reorder. The task depends on a user-specific replenishment policy established in the long-term history: order 60 units of each of two scents, for a total of 120 units. Other requirements include selecting the correct products, meeting the deadline, explaining the low-stock basis, and citing the source of the quote. The memory-controlled order quantity is therefore only one part of the complete task.

Under w/ Memory, Opus 5 created an order for 120 units and satisfied the other requirements, so it passed the complete task. Qwen3.8-27B Base received the relevant memory but only presented candidate options and asked the user what to do next; it never created the order. This example shows that having memory in the execution context does not guarantee that the agent will commit to the action required by the task. Qwen3.6-27B did create an order, but entered 200 units rather than 120. It initiated the correct type of action but incorrectly realized the parameter controlled by memory.

This within-task comparison separates two Application failures. Qwen3.8-27B Base never performed the target action, whereas Qwen3.6-27B performed it with the wrong parameter. The comparison also shows why merely checking whether a model mentions the relevant memory is insufficient for assessing downstream utility: evaluation must determine whether memory changes the actual judgment or tool call.

E Training Details

E.1 Training Data

The final training set contains 492 samples: 213 execution trajectories and 279 fully recorded consolidation phases. The execution data cover 193 training tasks. The 279 consolidation phases come from 98 complete consolidation-to-execution trajectories and therefore do not represent 279 mutually independent task trajectories. The training split contains 22 personas, while the evaluation split contains 13 different personas; the two persona sets do not overlap.

Joint training uses all 213 execution trajectories and 279 consolidation phases. The execution-only comparison uses the same 213 execution trajectories but excludes consolidation data. The two settings therefore share identical execution supervision, while the joint setting receives additional consolidation supervision. Because their sample counts and optimization steps differ, this comparison is not a strict data-matched ablation.

E.2 Optimization Configuration

Both checkpoints start from Qwen3.8-27B (Qwen Team, 2026b) and undergo full-parameter fine-tuning with ms-swift 4.5.2 (Zhao et al., 2025), bf16 precision, and the `qwen3_8` template. During preprocessing, a non-thinking prefix is added to each training sample.

Table E1: Optimization configuration for the fine-tuned checkpoints.

Configuration	Setting
Epochs	10
Learning rate	`1e-5`
Minimum learning rate	`1e-6`
Schedule	Cosine decay
Warmup ratio	`0.05`
Weight decay	`0.1`
Global batch size	16
Maximum sequence length	120,000 tokens
Overlength handling	Delete
Precision	bf16
MTP layers / loss factor	`1 / 0.1`

Joint training runs for 136 optimization steps, while execution-only training runs for 41 optimization steps.

F Limitations and Social Impact

F.1 Limitations

First, the benchmark uses synthetic personas, simulated services, and controlled tasks rather than long-term digital activity from real users. This design enables precise control over histories, environments, and scoring conditions, but it cannot capture permission changes, interface failures, user corrections, or cross-device differences in real products.

Second, the benchmark covers multiple personas, service domains, task types, memory counts, evidence expressions, and trigger mechanisms, but the ranges and combinations of these dimensions remain limited and uneven. The current results therefore do not represent all user populations, personal-assistant settings, or long-term memory structures.

Third, multiple tasks may branch from the same long-term background. Task branches have independent current environments and execution trajectories, but tasks sharing a history are not fully independent history samples.

Fourth, the results reflect the joint influence of the model, agent scaffold, system prompt, and memory interface. Alternative retrieval methods, context-management strategies, or tool planners may change the relative bottlenecks among Formation, Availability, and Application. The results therefore do not establish a fixed ranking of all long-term memory systems.

F.2 Social Impact

Long-term memory can help personal assistants follow user preferences, constraints, and unfinished tasks in later interactions, reducing repeated explanations and the cost of coordination across applications. At the same time, long-term memory may contain sensitive information about locations, health, finances, contacts, and schedules. Incorrect, outdated, or decontextualized memories may also repeatedly influence later actions.

Real-world deployment requires explicit purpose limitations, access controls, and data-retention policies, together with mechanisms that allow users to inspect, correct, and delete persistent memories. High-risk actions also require reconfirmation of critical constraints before execution. This work uses synthetic personas and simulated services and therefore does not collect long-term digital footprints from real users.